National College of Ireland

BSc in Computing

2015/2016

Glen Ward

X12436692

X12436692@student.ncirl.ie OR gmw6593@gmail.com

New Age of the Dead

Technical Report

Fourth Year Project

Table of Contents

EXECUTIVE SUMMARY

With the explosion of gaming onto the market in the last 20 years, game development has become task heavy and must be succeeded in order to remain competitive. This is part of the reason why I chose gaming and multimedia as my specialization stream. New Age of the Dead is a first person zombie shooter game which is told from the perspective of the main character, this depicts how the world is changing for the worse and gives the user the true fear of what living in a zombie apocalypse would be like. This game will offer a mass of different difficulty settings along with great functionality and a truly frightening environment which you will not want to take your eyes off. The functionality would include health for player and regen packs which would be discovered along the way, AI on enemies which the zombie would patrol throughout the environment and attack the player when they see them. The AI boss would be done significantly different as these would be more complex as the vision of the AI would be enhanced, have a better patrol system, and do more damage to the player when they attack. These would also have health attached. Other functionalities would include, inventory system, unlockable doors

through finding hidden keys, pick up notes which would help the player understand the background story, these could include hints for them to escape, and also this contains cinematics of the level and the environment which they find themselves in. There would also be jump scares to add a more spooky effect to the environment. Also this will include a multiplayer system which will have the same functionality highlighted above but each user can create a LAN server and play with their friends in the same game mode from different locations.

I also plan to use User testing and testing in general which I feel would greatly help drive the direction of this game development and ensure that potential consumers will have the best possible experience when playing this game.

1. INTRODUCTION

1.1 BACKGROUND

The history of horror genre is a vast and perhaps foolhardy thing to tackle. No matter how hard you try, there are myths, storytelling, films and games that will somewhat make you come out of your seat. There is a recognizable patterns that happens again and again, where there is a story, an event occurs which causes jump scares throughout the process. Like, movies which capture the imagination of the audience and can transform into bank, filmmakers flock to the cash like vampires and making money. Filmmakers can then spend the money to create sequels and imitators, all in an aim to make profit and create a name for themselves. In regards to horror games, this cannot be discussed without speaking about Amnesia: The Dark Descent which received two major awards and multiple nominations, and Slender Man, which are two of the most successful horror games in history. Amnesia is a survival horror video game by Frictional Games. This game features a protagonist exploring a dark and foreboding castle, while avoiding monsters and other obstructions. Slenderman is a fictional supernatural character which contains the Stories of the Slender Man commonly feature him stalking, abducting or traumatizing people, particularly children. The Slender Man is not confined to a single narrative, but appears in many disparate works of fiction, mostly composed online. Fiction relating to the Slender Man encompasses many media, including literature, art and video series such as Marble Hornets. Outside of online fiction.

However, the games which inspired me into creating a zombie first person shooter game is Dying Light, Left for Dead and Walking Dead – Telltale. These games all have a unique feature which I loved and would like to incorporate them all into one project.

Dying Light	Left for Dead	Walking Dead – Telltale
Inventory System with crafting weapons	First person shooter, Multiplayer	Great background story

1.2 AIMS

I aim to make a third/first person zombie shooter game which would be for the age range of 16+. What makes my game unique is that it would have a completely different story compared to any other zombie apocalypse game available in the market at the moment. And not only that I have combined the best features from the most popular games and put them all into one package which includes multiplayer. One main part of my game like any other on the market is the single player mode, this story gets told from the perspective of one individual where he wakes up one morning to see his whole family is missing and all that is remaining is the blood stains that they left. When the user selects to play the single player mode they will be given an option of whether to compete in the tutorial level, if they select yes then they will be given an introduction into the controls, background and how the game works. This level will be in the day time as it is an introduction into the environment and just the start of the story. When the user completes the goal this will bring them to the main level. Whereas if they select no then they will be thrown straight into the main level where their battle for survival begins. Another type of game mode which is incorporated is multiplayer. This uses LAN where a user can create a game server to host it or they can join the server by searching for available games to become the client. The host can set a maximum number of players so it is restricted to a certain number available in the game. The purpose of this is to race against the clock to find the keys before the other players and escape before they do. There is only one of every item so the players have to be quick! Other functionality that will be included in my project is Health for the player and AI, the player will also find health regen packs throughout the map. This will also include cinematics before and after every level to increase the players knowledge of the background story, and clips on the zombies and boss's that they are facing. Also there will be an inventory system where the player can equip/drop weapons and also use bits that they find around the area to craft other weapons. There will be 4 different versions of AI incorporated. The AI from the zombie and the boss's perspective will be significantly different, the bosses will be quicker, be able to receive a heat map of your locations which you have been, also be stronger, has more health and does more damage. The player will be able to defend themselves using fire weapons which they will find throughout the level, and using melee attacks also. The player will find letters throughout the level that will give them hints for which weapons

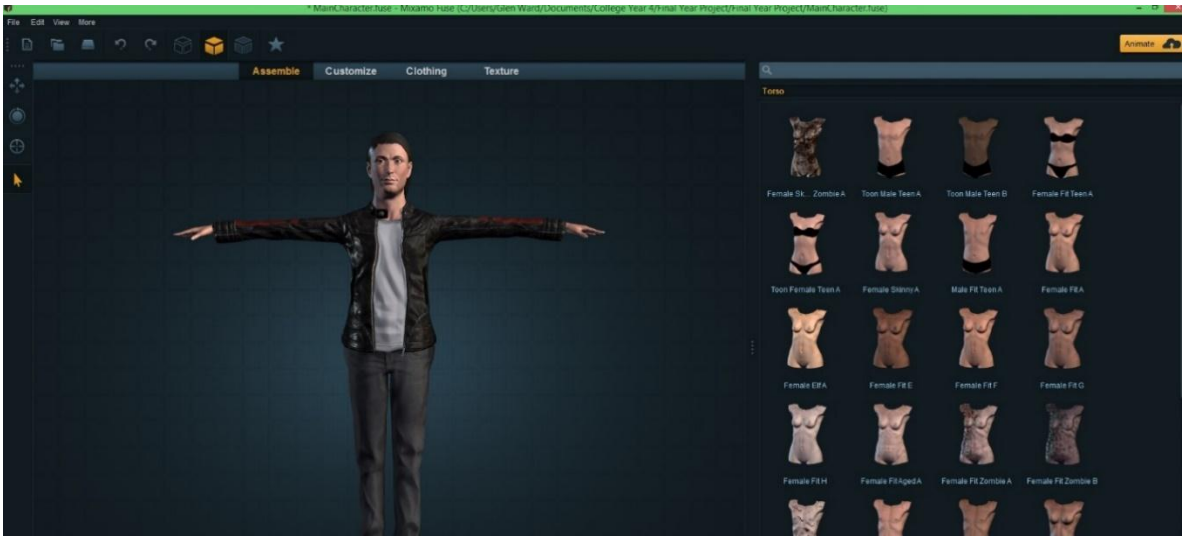
they use, the background story and clues of how to escape. These letters can be picked up in a radius, so if they stand close to it the user can simply hit “R” to pick up the letter and read it. They can put the letter down but hitting “R” again or walking outside the trigger zone. These letters can only be found inside locked doors, which the player will have to find keys throughout the level to open them, each key is specific to only one door. When the key is used once it is destroyed. There will also be jump scares which will display images of what is going on and causing the world to turn in the direction that it is going. These jump scares will also include dead bodies falling from the ceiling as you progress through the game at unexpected moments. The player will be against the clock in the main game mode and multiplayer where they have to escape within 10 minutes, the longer you spend the harder the game becomes as zombies will spawn in the game every three minutes.

1.3 TECHNOLOGIES

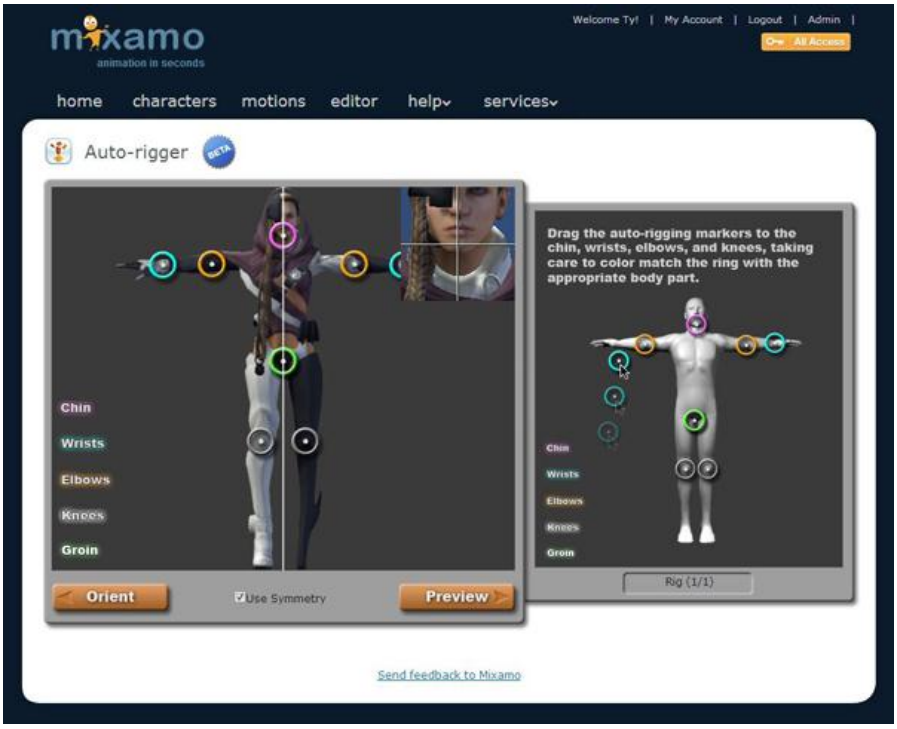
1.3.1 MIXAMO - CHARACTER DESIGN:

Character design has been one of the most fundamental and enjoyable events of my project to date. The first step was I installed Mixamo fuse. Fuse Character Creator is a 3D computer graphics software developed by Mixamo that enables users to create 3D characters. Its main novelty is the ability to import and integrate user generated content into the character creator.

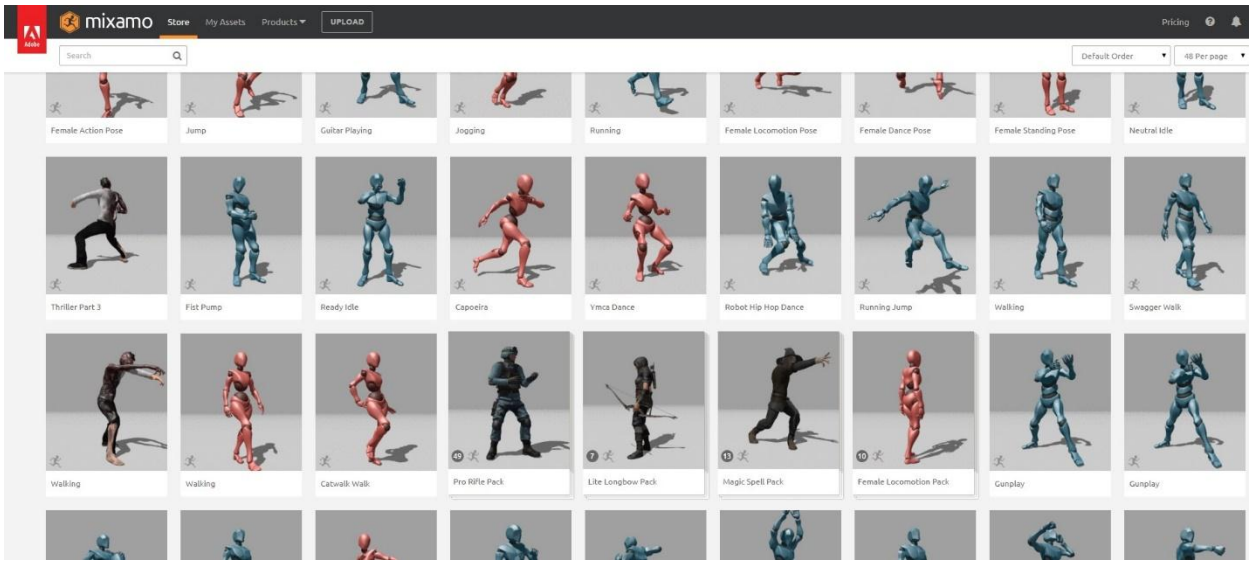
As you see in the image below a user can start by creating the head for their character, then move on to the body, arms and legs. Once this is complete you can now select the clothes you wish your character to have. You can mix and match different kinds of heads and bodies if you wish also. After this is complete you need to export your character to an fbx file. This is so you can easily import this character to mixamo.com where you can auto rig your character and apply their skeleton.



As you can see in the image above, to auto rig the character you need to set the placeholder to particular bones of the human body, so one goes on each hand, shoulder, groin, head, knees and feet. This is so the skeleton gets assigned properly. After this now we can chose the animations we would like our character to have, so we could chose, walk, run, death, shooting etc.



If the animations in the selections isn't to my liking I can simply make my own using blender and Maya. Once this is complete I can import the animations and character to Unreal Engine and apply my new skeleton my character got assigned and change the mesh of my character blueprint class to the new character and then this character is the new main character of my game.

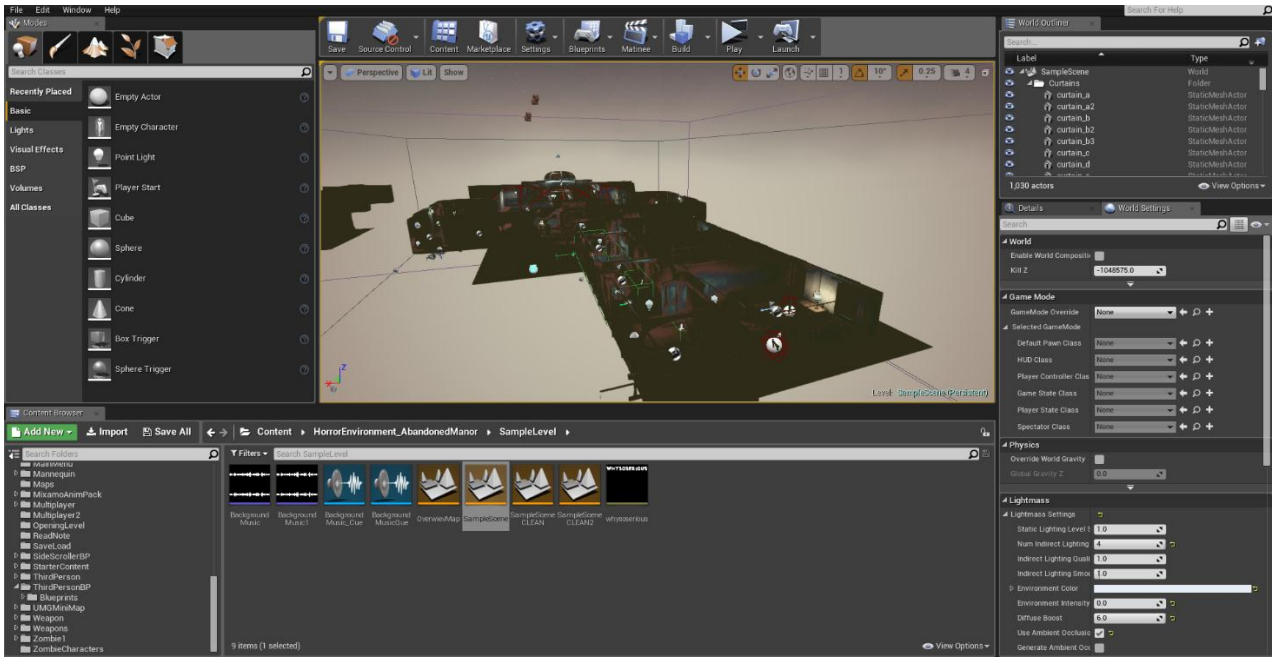


1.3.2 UNREAL ENGINE – SOFTWARE TO DEVELOP GAME:

Unreal Engine is the main platform for my game as it is a complete suite of game development tools made by game developers, for game developers. From 2D mobile games to console blockbusters and VR, Unreal Engine 4 gives you everything you need to start, ship, grow and stand out from the crowd. One of my favourite horror games is "Silent Hill", Djoexe has been working these past months on a recreation of Silent Hill P.T. in Unreal Engine 4, and released a demo a couple of months ago of a fly through video from it. Based on the awesome demo on PS4, he recreated this beautiful environment on the Unreal Engine 4.

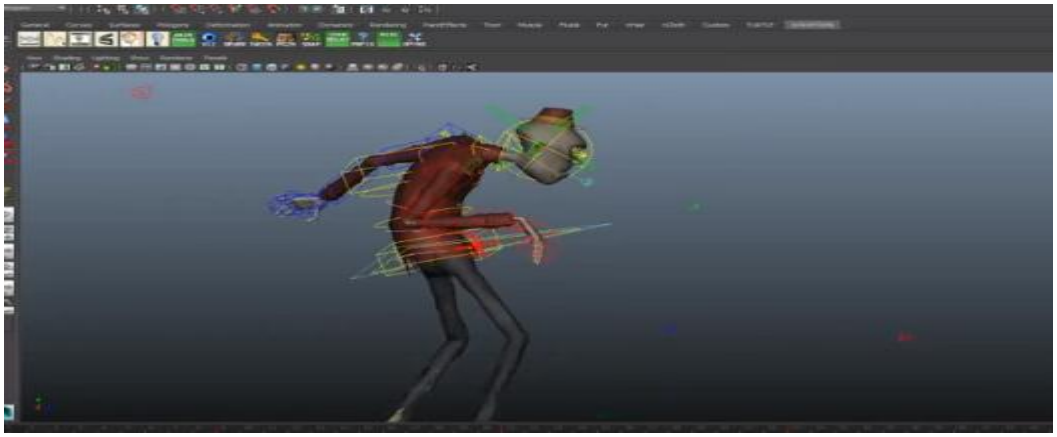
I took hundreds of screenshot while playing the game to get my references as well as playing the game and observing the proportion and mood. This video is for a portfolio purpose and not a commercial goal at all. Due to this fantastic recreated environment and aspiring story line, this was one of the main factors of why I chose this as my main platform.

Using Unreal Engine 4's potential I was able to create an abandoned manor which is going to serve as the main level of my game. This software is going to enable me to create all the functionality in my game, like inventory, player health, AI patrol, quickness, different levels of AI, AI health, guns, inventory, jump scares, unlockable doors with keys, cinematics and damage from external sources.



1.3.3 AUTODESK MAYA - CHARACTER DESIGN:

Autodesk Maya is a piece of software which allows users to create animations, so therefore I used this to create many animations for the zombies in my game. I found this to be very difficult at the start but eventually I found this to get easier and was able to create multiple animations which you can see throughout my game. Firstly I had to set the timeline of frames I wish to have, then select I needed to select my character and go to animate less than set key. Now I've set my first key frame. This key frame has all its values at 0 because the sphere hasn't been moved. Then I need to edit these key frames to move the character along, this is along the Z, Y and X axis. Now I need to keep editing and testing this over and over to ensure everything is set and the animation works accordingly.



2 REQUIREMENTS

- The game must have a start-up menu before the game can begin, this will also have options where the user can set the resolution, change the difficulty of the game and change settings such as texture and shading quality and exit

- The user must have an option of whether to play the tutorial mission or not. The tutorial mission will give the user and insight into the controls and storyline

- The user can start multiplayer by creating their own server or joining their friends who has created one already

- It must have a landscape which the characters can exist in the game

- The four versions of AI must have knowledge of the map and to also be able to check different spots of the level in search of the user. All without the AI getting confused. If the AI spots the player they need to be able to attack.

- The landscapes must be realistic, all levels will need sound effects, blood, pipes, corridors, and correct lighting to scare the user etc.

- A set of characters that will be playable

- A set of characters that will act as my AI

- The player character should be able to equip items, pick up items, search through inventory system, shoot and drop all weapons

- The landscapes must be realistic and also be able for the characters and AI to navigate through every bit without confusion

- Have a background story line available so the user can engage with the game even more

- The user should be able to kill all enemies throughout the game and collect an item that the AI leaves to add to their enemy killed count which will add to their score

- Midpoint pause throughout the game where the user can save and load their progress while also being able to exit the game

- The player can chose whether they want to play with keyboard and mouse or gamepad controller

- The user can set screen resolutions, shadow quality, texture quality and volume under the options in the main menu

- The user can pick up keys which unlock the correct door

- The user will get notifications throughout the level so they know what to do next

- A mini-map displayed in the corner of the screen so the user can see where the enemies location at all times while also being able to see their own location

- After the user completes the game they can submit their score to the leaderboard, this is based on how many enemies they killed and how quickly they completed the game

2.1 FUNCTIONAL REQUIREMENTS

An active internet connection is required to access this game as this will be hosted for any user to use. Other than that the user simply needs a computer, keyboard, single mouse and then a controller is optional.

Stages of the game:

Start:

When the user starts the game, they will be presented with an opening cinematic. After this closes the user will now chose their character that they will spend the game playing as.

Main Flow:

The game will now open and the user must keep alive until they can find a way to escape the environment

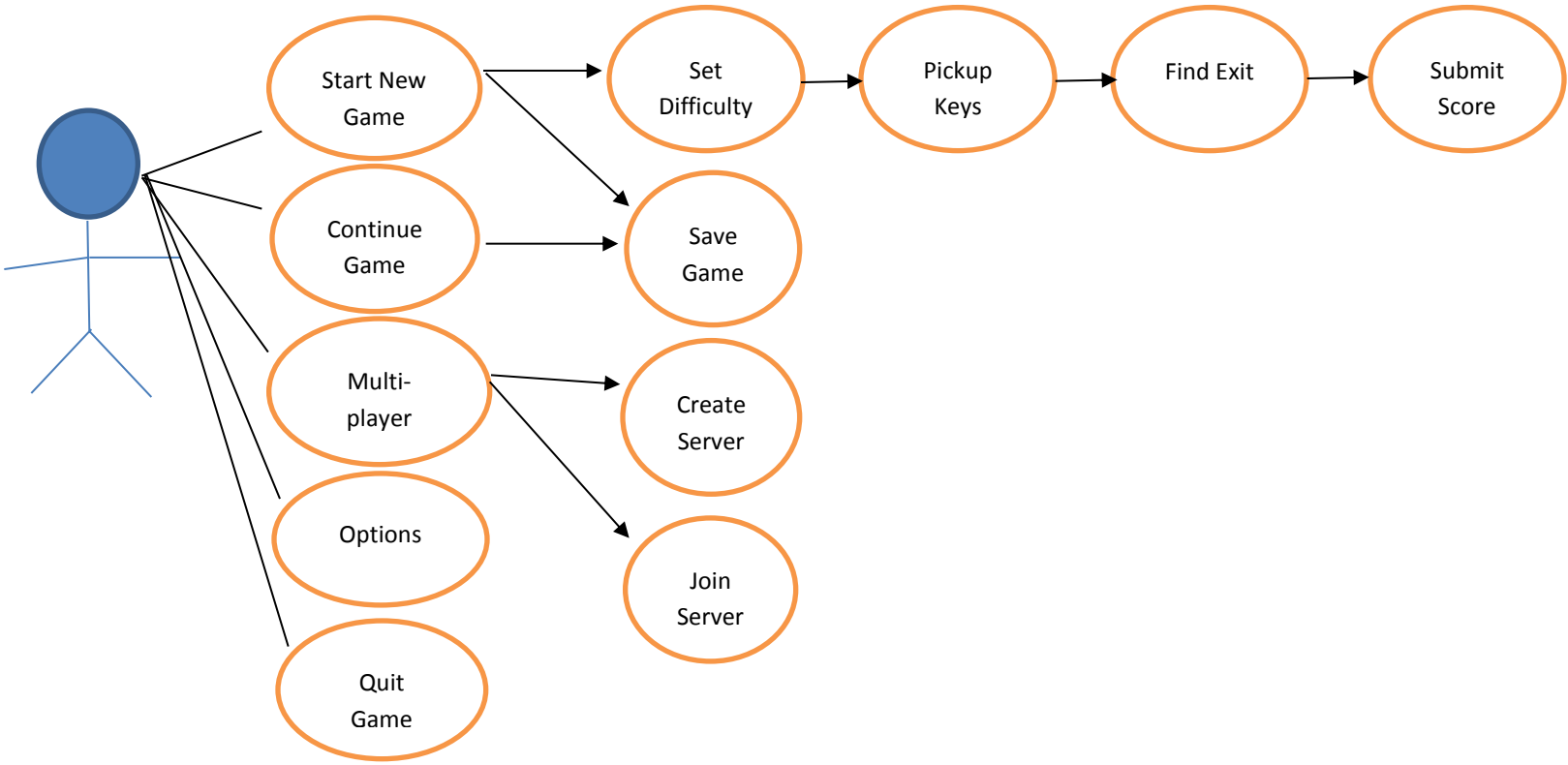
Exceptional Flow:

When the player dies they will be presented with the leaderboard where they submit their score.

Termination:

- You have died
- You escaped the environment
- Restart the game

2.1.1 USE CASE DIAGRAM



2.1.2 REQUIREMENT 1 <START GAME>

DESCRIPTION & PRIORITY

The player starts a new game. This requirement is essential to the operating of the system

USE CASE

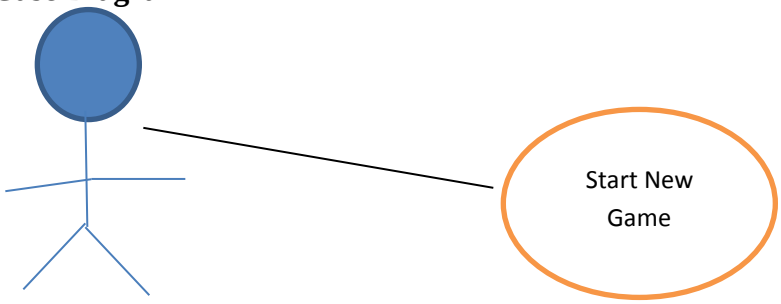
Scope

The scope of this use case is to allow the player to start a new game

Description

This use case describes the process by which a player starts a new game

Use Case Diagram



Flow Description

Precondition

The system is in initialisation mode

Activation

This use case starts when a player starts a new game

Main flow

1. The system identifies the player
2. The player starts a new game
3. The system loads up the first level

Exceptional flow

E1: Exit Game

Termination

The system presents the game to the user

Post condition

The system goes into a wait state

2.1.3 REQUIREMENT 2 <CONTINUE GAME>

DESCRIPTION & PRIORITY

The player saves the game. This use cause is really important in allowing the user to continue their progress.

USE CASE

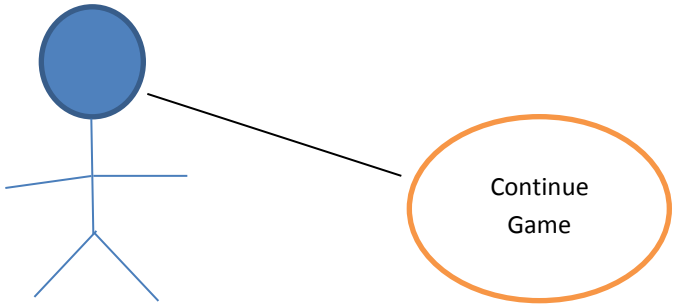
Scope

The scope of this use case is to allow the user to continue the game from where they left off.

Description

This use case describes the process by which a player loads the game

Use Case Diagram



Flow Description

Precondition

The system is in initialisation mode

Activation

This use case starts when a player loads the game

Main flow

1. The system identifies the last point where the user saved the game
2. The user loads the game
3. The system presents the player with the point of the game of where they left off

Exceptional flow

E1: Exit Game

Termination

The system presents the next screen to the user

Post condition

The system goes into a wait state

2.1.4 REQUIREMENT 3 <SAVE GAME>

DESCRIPTION & PRIORITY

The player saves their game. The requirement is really important for my game so the user doesn't have to start the game from the beginning every time

USE CASE

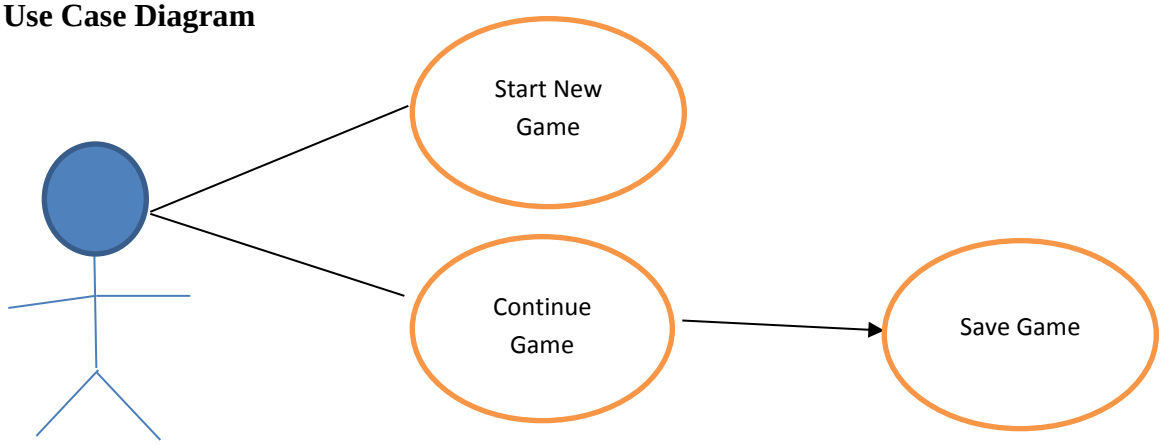
Scope

The scope of this use case is to allow the user to save their progress

Description

This use case describes the process by which a player saves the game

Use Case Diagram



Flow Description

Precondition

The system is in initialization mode

Activation

This use case starts when a player saves the game

Main flow

1. The system identifies the point where the user is on the map
2. The user saves the game
3. The system stores the save game data

Exceptional flow

E1: Exit Game

Termination

The system presents the next screen to the user

Post condition

The system goes into a wait state

2.1.5 REQUIREMENT 4 <SET DIFFICULTY>

DESCRIPTION & PRIORITY

The player sets the difficulty of the game. This use case is really important in allowing the user to be challenged through their progress of the game.

USE CASE

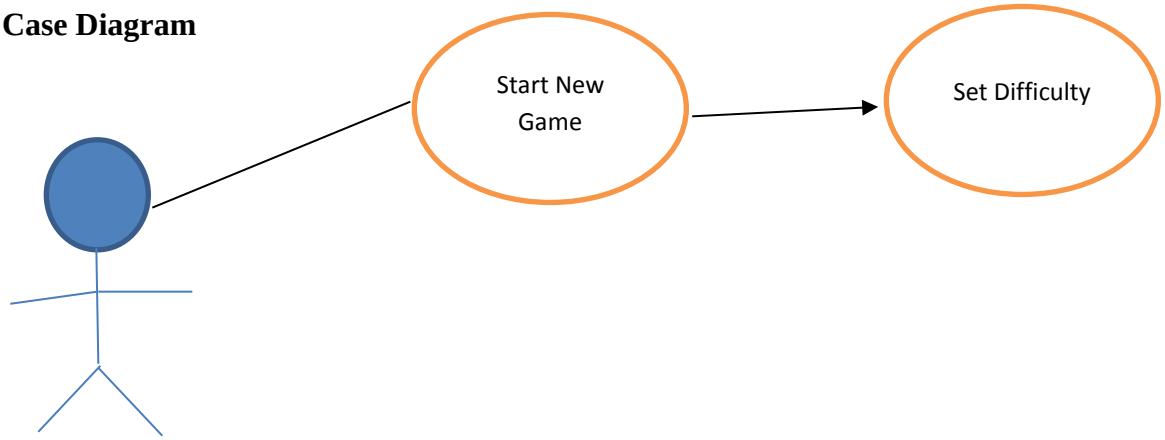
Scope

The scope of this use case is to allow the user to set the difficulty of the game

Description

This use case describes the process by which a player sets the difficulty of the game

Use Case Diagram



Flow Description

Precondition

The system is in initialisation mode

Activation

This use case starts when a player sets the difficulty of the game

Main flow

1. The system identifies the difficulty that the user has chosen
2. The user sets the difficulty of the game
3. The system presents the user with the difficulty that they have chosen

Exceptional flow

E1: Exit Game

Termination

The system presents the next screen to the user

Post condition

The system goes into a wait state

2.1.6 REQUIREMENT 5 <MULTIPLAYER – CREATE SERVER>

DESCRIPTION & PRIORITY

The player starts multiplayer session. This requirement is important as it allows the user to play the game with more than one player.

USE CASE

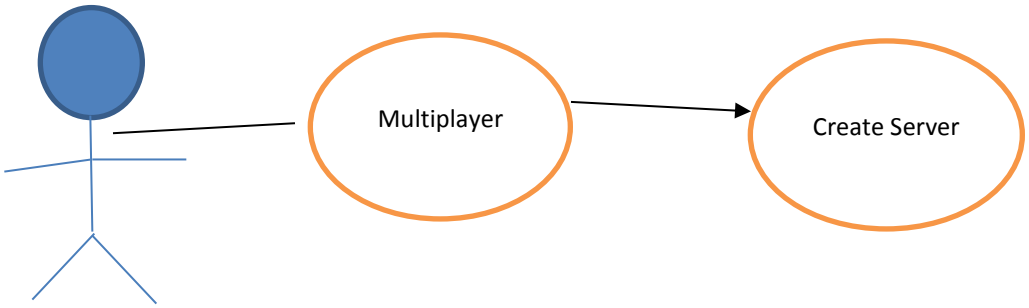
Scope

The scope of this use case is to allow the user to play multiplayer in the game

Description

This use case describes the process by which a player creates a server to play the game

Use Case Diagram



Flow Description

Precondition

The system is in initialisation mode

Activation

This use case starts when a player creates a server in multiplayer in the game

Main flow

1. The system identifies the player
2. The player creates a server
3. The system loads up the level

Termination

E1: Exit Game

Post condition

The system goes into a wait state

2.1.7 REQUIREMENT 6 <MULTIPLAYER – JOIN SERVER>

DESCRIPTION & PRIORITY

The player joins multiplayer session. This requirement is important as it allows the user to play the game with more than one player.

USE CASE

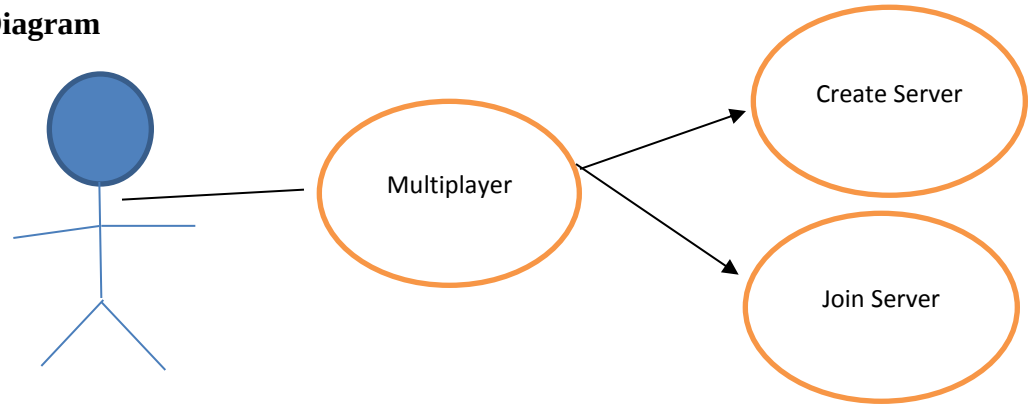
Scope

The scope of this use case is to allow the user to play multiplayer in the game

Description

This use case describes the process by which a player joins a server to play the game

Use Case Diagram



Flow Description

Precondition

The system is in initialisation mode

Activation

This use case starts when a player joins a server in multiplayer in the game

Main flow

1) The system identifies the player
2) The player joins a server to play
3) The system loads up the level

Termination

E1: Exit Game

Post condition

The system is off

2.1.8 REQUIREMENT 7 <PICKUP KEYS>

DESCRIPTION & PRIORITY

The player can pick up keys. This requirement is important as it allows the user to pick up the vital keys needed to escape.

USE CASE

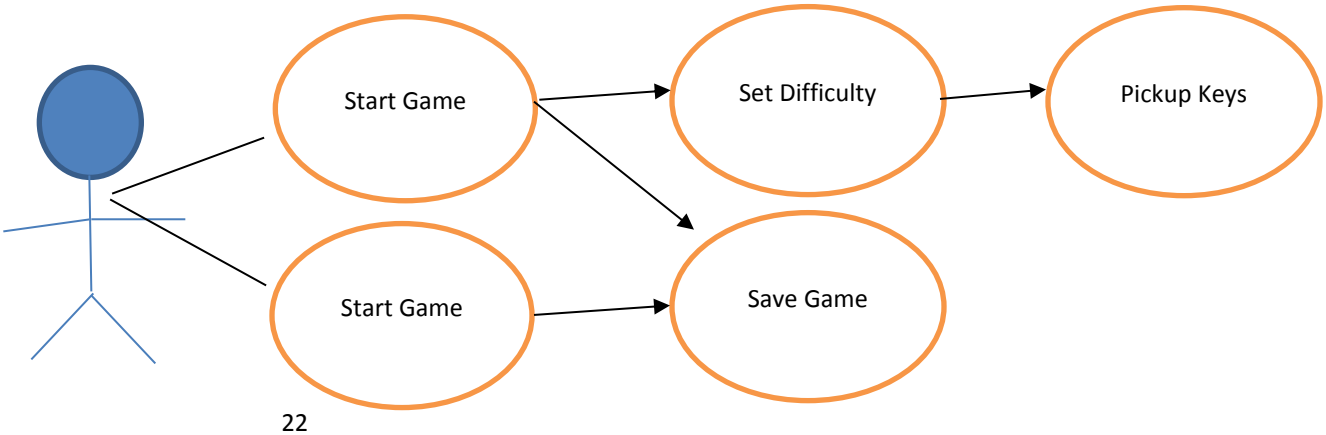
Scope

The scope of this use case is to allow the user in the game to pick up keys

Description

This use case describes the process by which a player picks up the keys

Use Case Diagram



Flow Description

Precondition

The system is in initialisation mode

Activation

This use case starts when a player picks up a key in the game

Main flow

1) The system identifies the player
2) The player picks up a key
3) The system stores this in the inventory

Termination

E1: Exit Game

Post condition

The system goes into a wait state

2.1.9 REQUIREMENT 8 <FIND EXIT>

DESCRIPTION & PRIORITY

The player needs to find the exit to complete the game. This requirement is important as it is the purpose of the game.

USE CASE

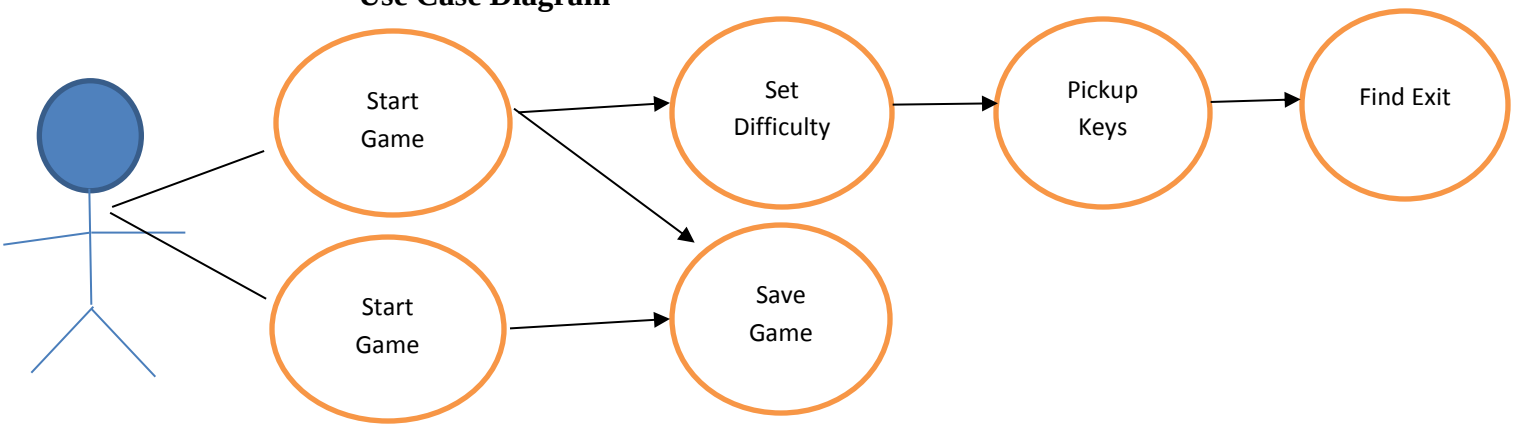
Scope

The scope of this use case is to allow the user to complete the game

Description

This use case describes the process by which a player finds the exit

Use Case Diagram



Flow Description

Precondition

The system is in initialisation mode

Activation

This use case starts when a player find the exit

Main flow

1) The system identifies the player
2) The player finds the exit
3) The system brings the user to submit their score

Termination

E1: Exit Game

Post condition

The system goes into a wait state

2.1.10 REQUIREMENT 9 <SUBMIT SCORE>

DESCRIPTION & PRIORITY

The player submits their score to appear on the leaderboard. This requirement is important as it shows how well they have done compared to other players.

USE CASE

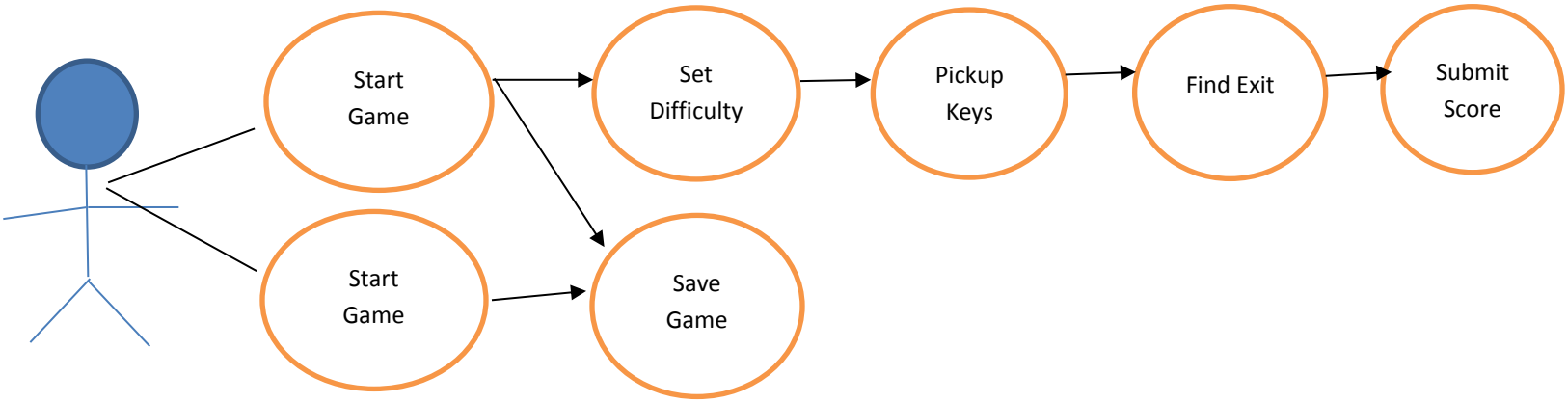
Scope

The scope of this use case is to allow the user to submit their score

Description

This use case describes the process by which a player submits their score

Use Case Diagram



Flow Description

Precondition

The system is in initialisation mode

Activation

This use case starts when a player submits their score

Main flow

1) The system identifies the player
2) The player submits their score
3) The system stores their score in a database

Termination

E1: Exit Game

Post condition

The system is off

2.1.11 REQUIREMENT 10 <EXIT>

DESCRIPTION & PRIORITY

The player quits the game. This requirement is important as it allows the user to exit the application.

USE CASE

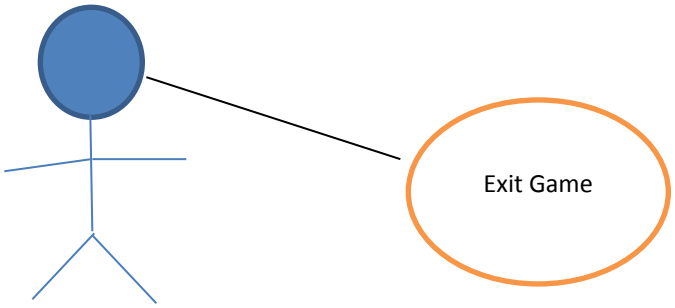
Scope

The scope of this use case is to allow the exit the game

Description

This use case describes the process by which a player quits the game

Use Case Diagram



Flow Description

Precondition

The system is in initialisation mode

Activation

This use case starts when a player sets the difficulty of the game

Main flow

1) The system identifies the identifies the player
2) The player quits the game
3) The system exits the application

Termination

The system presents the next screen to the user

Post condition

The system is off

2.2 NON FUNCTIONAL REQUIREMENTS

2.2.1 AVAILABILITY REQUIREMENT

Option 1: Deployed Website and Social Media formats

- This would be published and have a digital download ready for users. The website could be promoted on social media accounts as this is one of the most popular and best ways of advertising for any product prior to the official launch.
- As I would have control over the distribution due to this process, this would depend on a highly successful social media campaign to spread the word of the game
- This would also grant control over pricing and payment methods

Option 2: Digital Distributors

- Using digital distributors is industry standard for publishing games which have independent game developers. There are a countless number of platforms that can be used but the one which stands out from the rest is Steams Greenlight system.
- Allows users of steam to have the final say of what goes to the market and gets placed on the steam store itself
- Developers can publish beta versions of their game on the store to allow users to test their game and provide feedback on bug reports, ratings etc.
- Limited control over pricing

2.2.2 SECURITY REQUIREMENT

The security requirement for my game is that the user needs an active internet connection to download the game, therefore users simply have to ensure that they have the efficient security privileges to access and allow them to install and run the appropriate .exe file.

2.2.3 RELIABILITY REQUIREMENT

This provides a pretty straight forward way to install and run the game, once the .exe file is run the steam settings take control.

2.2.4 MAINTAINABILITY REQUIREMENT

As the game is supported through the official site and social media accounts where users post the bugs and errors. Maintainability is provided as I can provide bug fixes and updated versions of the game which fix these issues. These update patches and versions can be set automatically so the user doesn't have to go out of their way to have these applied.

2.2.5 EXTENDIBILITY REQUIREMENT

After the game is published for pc platform, dlc can be introduced with the intrusion of levels, goals, characters and enemies.

2.2.6 REUSABILITY REQUIREMENT

The assets, animations and materials that will be created throughout my game can be used in other games I create.

2.2.7 USER REQUIREMENT

The objects of this game is to enable the player to feel scared while playing my game while having an enjoyable experience. The player should also complete tasks, gain better items, and kill zombies to enable them closer to escaping the terrifying environment all while following the storyline based in the game. The user should have the ability to start a new game, save their progress at any time and then can continue from where they left off if they wish.

In order the play this game the user must have:

- A computer
- A keyboard
- An internet connection in order to access the game
- A controller is optional for the user

2.2.8 ENVIRONMENTAL REQUIREMENT

The game must run in a stable working environment in order to achieve maximum usability. The game was designed to operate on windows platform. As the game was developed using windows 8.1 and 10 means it will run flawlessly off this platform. This also runs perfectly on windows 7 as checked during testing.

2.2.9 USABILITY REQUIREMENT

The games menus should be easily visible, have clear descriptions about what they do and require minimal interaction from the user in order to complete a specific task. The following usability requirements are also present:

Understandability:

- The interface elements should be easy to understand and navigate through
- The goal of the game should be easily understood

Learnability:

- Help and tutorials should be implemented within the game in order to help the player to achieve basic tasks
- The game should be easy to learn

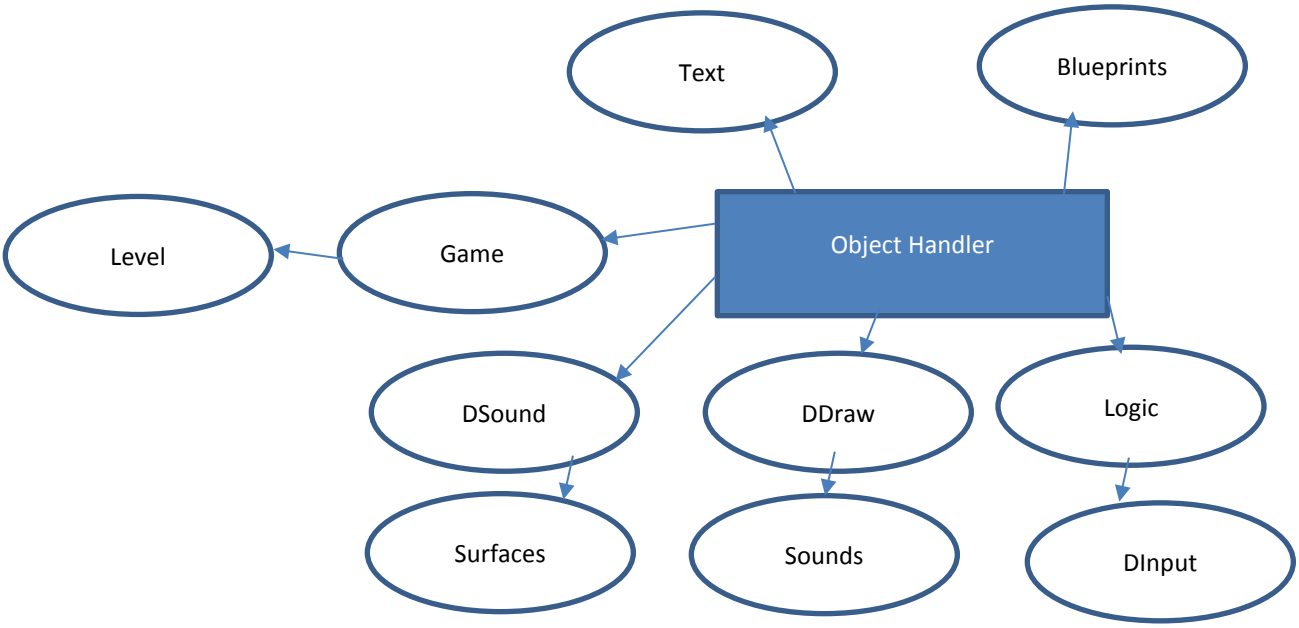
Operability:

- All interface actions should be consistent and lead to expected results
- The system will be customisable to meet specific user needs

Attractiveness:

The UI elements and layout of the background, menus, characters etc. should all be appealing to the user.

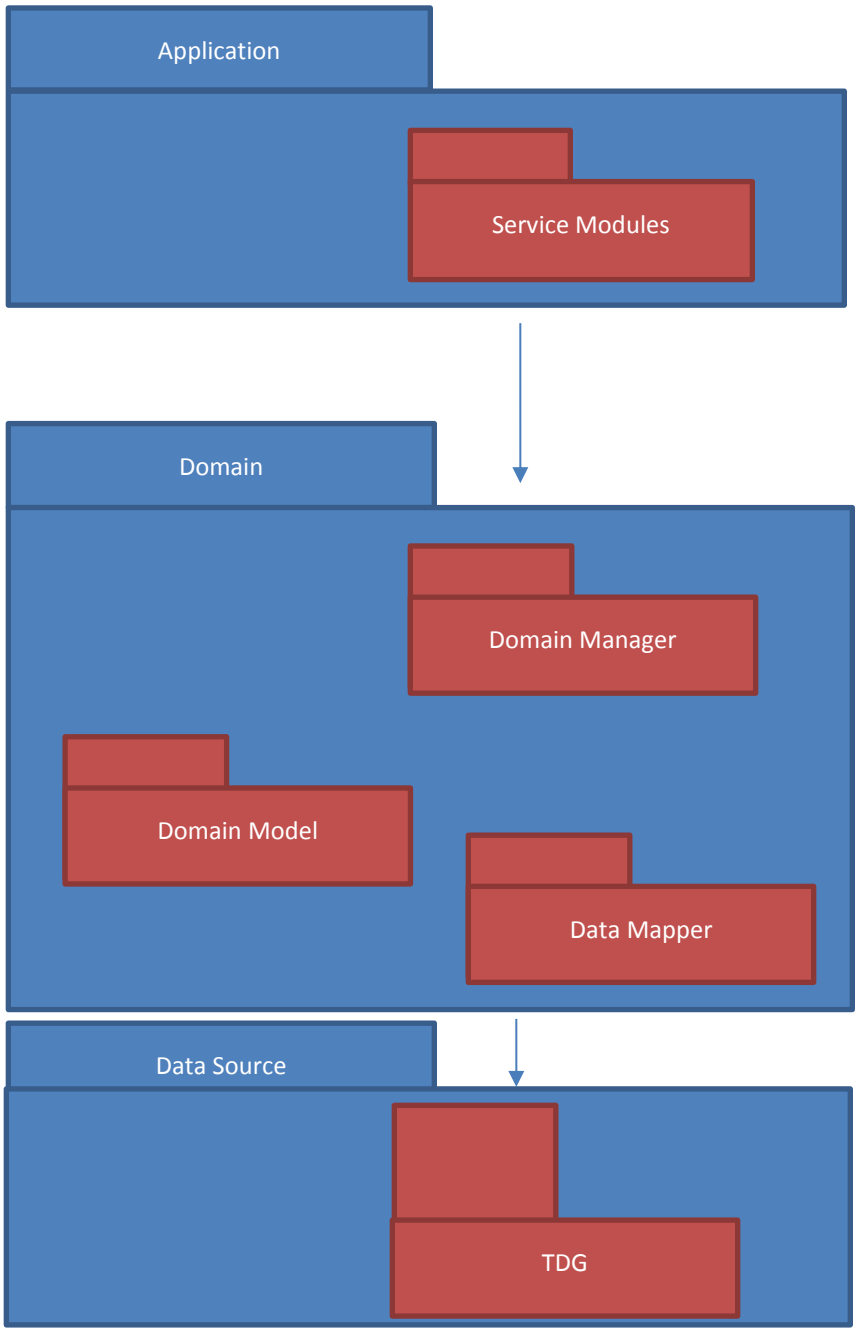
3 DESIGN AND ARCHITECTURE



It is important to note that my game architecture breaks up the blueprint system into a number of categories and classes, each allows the each other to communicate with the game objects and object handlers directly. Without this breakdown, a significant amount of data which is not needed must be passed back and forth through the Object Handler to pars down the blueprint information. With the current design, the Object Handler can easily retrieve blueprint information after receiving data requests from the other Handlers, with a minimum amount of parsing.

4. LOGICAL VIEW

The logical view of New Age of the Dead is separated into 3 main layers. Application layer, the Domain layer and the Data Source layer. These 3 main layers are structured following a 3 layer architecture convention, the presentation layer, the domain layer, and the data source layer.



4.1 APPLICATION LAYER:

The application layer in this implementation modifies the traditional Front Controller with the template view organization with different blueprint components, developed with the software of Unreal Engine. The front controller now handles request by referencing domain objects to and from the user to the game. The benefits of this approach allow a more robust and richer GUI implementation as the application can behave with characteristics of a native desktop game, additionally as most of the data can get cached locally on the first access subsequent access result in less stress to the server and a more lively experience for the user.

4.2 DOMAIN LAYER:

The domain package encloses sub-packages used to describe the New Age of the Dead game and the logic rules that control and govern the way information is processed. The sub-packages are the domain model, the data mapper, the unit of work, the identity map and the virtual proxy layer. The domain model sub-package contains a class for each of the major business entities, so like the users, roles, events and content. It also contains a layer Super type which is Domain Object to make use of the generic Unit of work and identity map.

4.3 DATA SOURCE LAYER:

The data source package contains the classes that define the technical services infrastructure used to store persistent data. The package includes a class source file for each class present in the data mapper sub-package. These data source objects are used by mapper objects to access data directly to store or retrieve information.

5. HARDWARE ARCHITECTURE

The only hardware needed for this project to operate is the standard requirements of having a set of components. The main component that we will need is the GPU, if this isn't powerful enough then we would need to acquire a different hardware as the game would not even start up. Also I would recommend a minimum of 2GB graphics card so you would be able to run the graphics on standard settings, otherwise the games frame rate would make the project run a small bit slow.

Overall the resources I need are:

- Mouse
- Keyboard
- Game Controller(Optional)

- Screen/monitor
- Power supply
- Processor: PII 200 MHz
- System Memory: 32 MB RAM
- Video Memory: 4 MB VRAM

6. SOFTWARE ARCHITECTURE

Software architecture is the process of defining a structured solution that meets all of the technical and operational requirements, while optimizing common quality attributes such as performance, security, and manageability. It involves a series of decisions based on a wide range of factors, and each of these decisions can have considerable impact on the quality, performance, maintainability, and overall success of the application.

The software I plan on using in my game is:

6.1 UNREAL ENGINE:

Unreal Engine is a game engine developed by Epic Games, first showcased in the 1998 first-person shooter game Unreal. Although primarily developed for first-person shooters, it has been successfully used in a variety of other genres, including stealth, MMORPGs, and other RPGs. With its code written in C++, the Unreal Engine features a high degree of portability and is a tool used by many game developers today.

The current release is Unreal Engine 4, designed for Microsoft's DirectX 11 and 12 this is for Microsoft Windows, Xbox One, Windows RT and OpenGL for OS X, Linux, PlayStation 4, iOS, Android, Ouya and Windows XP and JavaScript/WebGL for HTML5 Web browsers.

I'm using the latest version of Unreal Engine 4.10, which has been released with hundreds of updates, including 53 contributions from the talented community via GitHub. Us as developers can expect improved stability, lots of fixes, and updates to nearly every supported platform, including the new VR rendering optimizations they made while developing Bullet Train. This will be the main software I use to create my game.

6.2 AUTODESK MAYA:

Maya is an animation and modelling program used to create three-dimensional, full-motion effects. Maya incorporates the natural laws of physics to control the behaviour of virtual objects in computer animation. Maya can produce videos that

are more life-like than has been possible with less sophisticated programs. Versions are available for both IBM-compatible and Macintosh operating systems.

Until the development of Maya realistic rendering of certain natural effects, such as smoke blowing in a breeze, the rotation of clouds and dust in a tornado, or the sag and movement of clothing caused by gravity, was difficult or impossible to achieve. In addition to simulating the movements of objects and particles, Maya makes it possible to portray emotions in animated characters by enhancing facial expressions and the realism of body language. Some technical people at animation studios have begun taking courses in Maya. The program has been used in numerous movies, including Twister and Stuart Little. So this software is perfect for me to use in order to create all the animations I need for my game, and also help creates the effects I wish to use in my game.

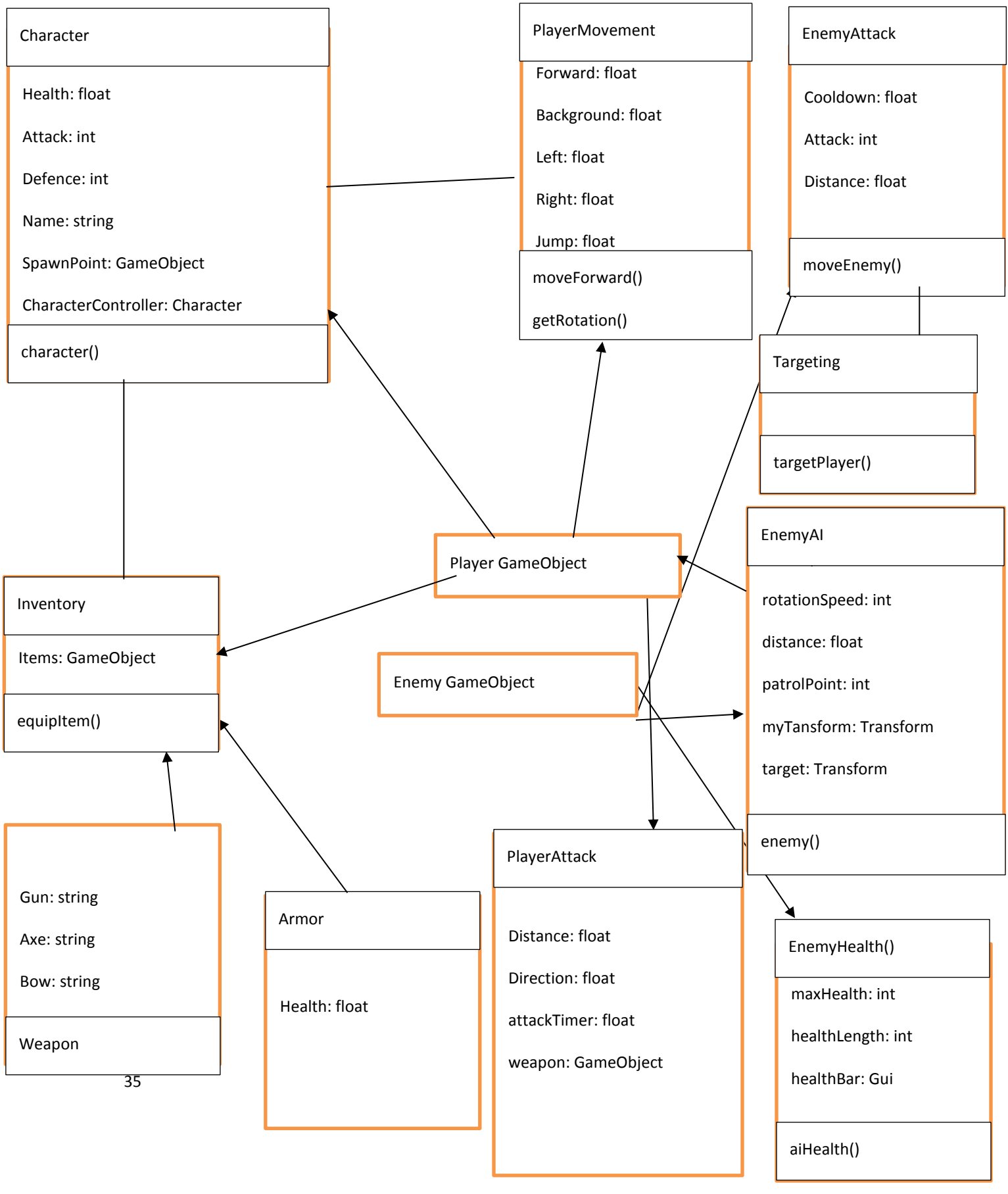
6.3 BLENDER:

Blender is a free and open source 3D creation suite. It supports the entirety of the 3D pipeline for modelling, rigging, animation, simulation, rendering, compositing and motion tracking, so this is perfect for my game creation. If I wish I could adopt this software, create my character and rig this character with a custom skeleton. Some users would employ Blender's API for Python scripting to customize the application and write specialized tools, often these are included in Blender's future releases. Blender is well suited to individuals and small studios who benefit from its unified pipeline and responsive development process.

6.4 MIXAMO:

Mixamo is an online platform that enables developers and artists to let their imagination run free and be even more ambitious about their games, films and other 3D projects. Traditionally extremely time consuming, expensive and complicated, achieving high quality results with 3D character art used to be reserved for large studios with expensive equipment and armies of modellers, riggers and animator but Mixamo allows all this and has all these features incorporated. Also with Mixamo, they have another software feature called Mixamo Fuse, this allows any user to rig their characters in the space of 30 seconds, this gives the user a default skeleton instead of them having to create their own, this is highly popular in the gaming industry as it saves time and money.

7. GAME ARCHITECTURE



I feel the class diagram is well suited to the architecture I want to apply to my game. The character select is the first class that the player will interact with as this is the first action in my game. Several variables will be stored with the values that they have chosen. The player will now start playing the game and now the movement class will be called. Here the player will be able to move in any direction that they wish like, left, right, forward, backward while also being able to rotate. While the player is moving they may come across AI zombie that will want to attack the player, in order for this to happen the enemy must target the player. While the enemy wants to attack the player, a number of classes will be called, such as enemyAttack, enemyAI, and also EnemyHealth. These will call a number of variables and will store the data which is being taken in while the AI is trying to attack the player. If the player wants to attack back this will also call a number of classes, like Inventory, Weapon, Armor and PlayerAttack. This will also store the data which is being processed.

If the player kills the enemy the player may roam and find a weapon that may want to equip, after they equip the weapon the item will be placed in an inventory. In this inventory will be all the weapons that the player has in order to fight the enemies. They will sorted from new to old.

8. PERFORMANCE

The performance of a game depends on a number of things, like the computer you have, the spec of your computer, graphics card etc. Like for example you may have to optimize your software in order to gain full power of your computer, this can be done by opening your task manager and figuring out which programs are constantly running in the background of your machine. Another way to improve the performance of a game running on PC would be to upgrade your graphics card, a graphics card is a printed circuit board that controls the output to a display screen. So this would enhance the display of the game, so you may be able to play any game of full high settings with this installed and also with a high resolution. The user also has an option to lower the quality of the settings in unreal engine by going into the settings and here there is four options available, low, medium, high and epic. Epic settings would be for a very powerful machine with a top range graphics card. Most computers available would either be on medium or high. These settings are perfect for any game to play on and wont effect the user’s experience of the gameplay at all.

9. IMPLEMENTATION

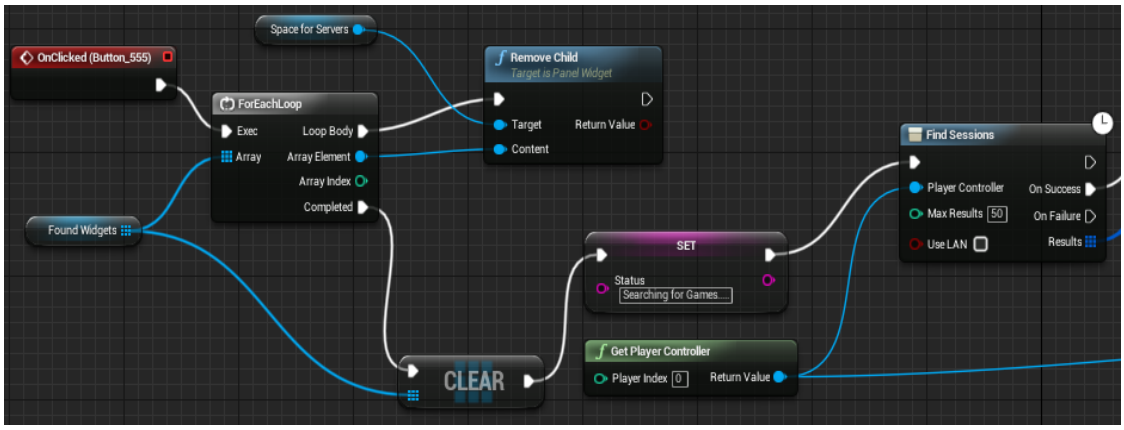
9.0 MULTIPLAYER:

This is the multiplayer to my game where the user can create servers to play their friends from two separate locations, and their friend can then join their server to play together. This will be a race to the finish as there is only one of everything so if you don't find the correct weapons in time you are defenceless to the zombies attacking you, but mainly there is only one key to the exit door!

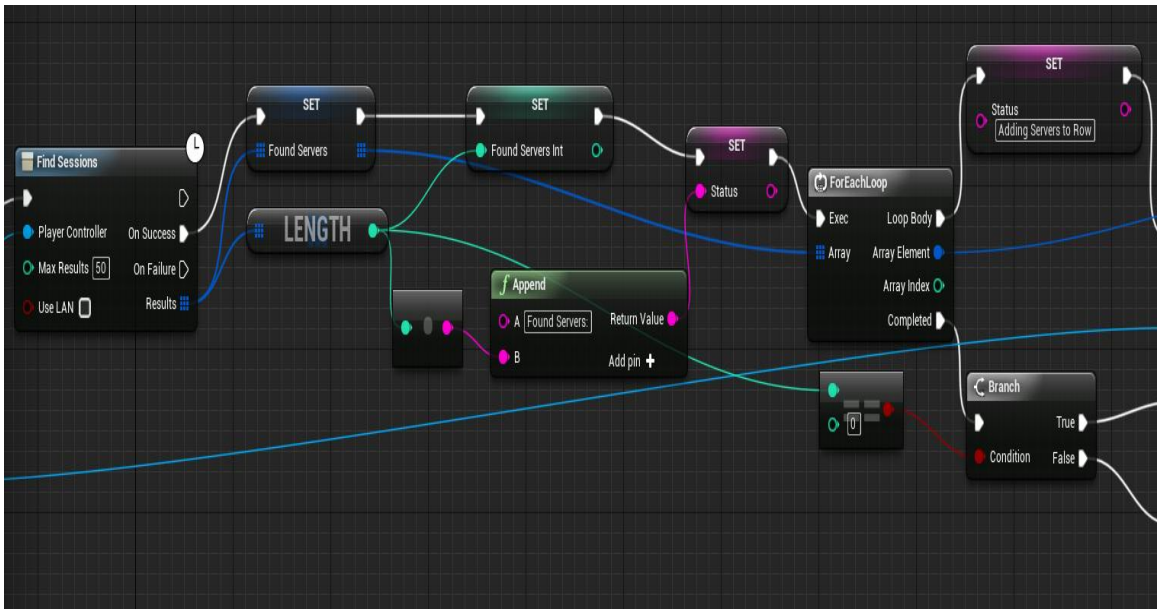
The multiplayer in my game works where the user can set the max amount of players which can enter the server. The user can then create their LAN server or join another server which is already created.

Search for Server:

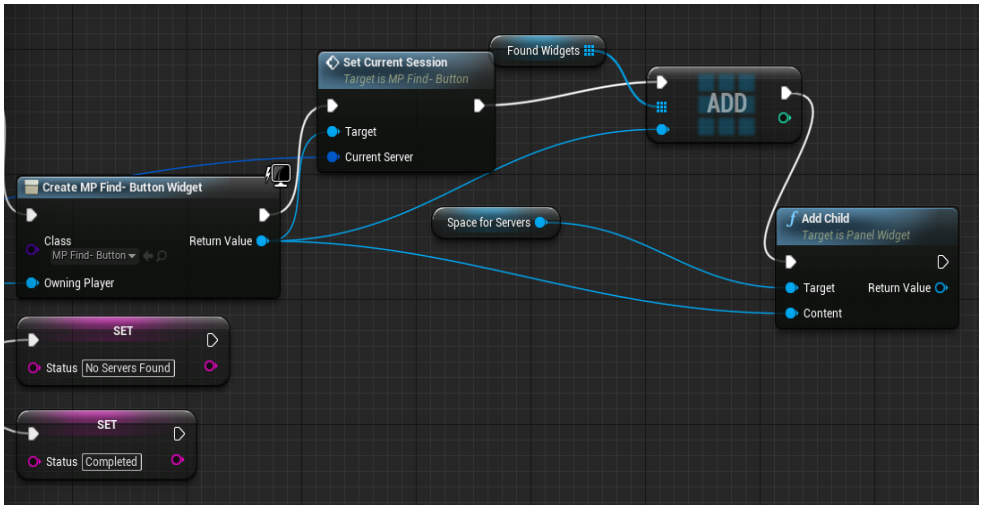
When the user clicks to find a server, it participates in a ForEach loop and removes any servers found before by clearing out the array and then sets the text to "Searching for Games". From searching from games we call Find Sessions, the Player Controller is the player and max results is 50, so the game can hold a maximum of 50 servers.



When the Find Sessions is successful, the onSuccess calls the found servers and sets this to the resulting array, gets the length and passes how many results where found and sets that to FoundServers integer. Then I set the status to the servers that where found and how many where found. Now I create a ForEach loop which comes off the status and takes in the servers which was found from the array, so when this is looping through, it is adding each server into a row and calling the button widget which I created and is used to join the games. On completion of the ForEach loop I created a branch which has a condition that is if there is 0 servers displays “No Server Found” if it is false, otherwise display “Completed Search”.



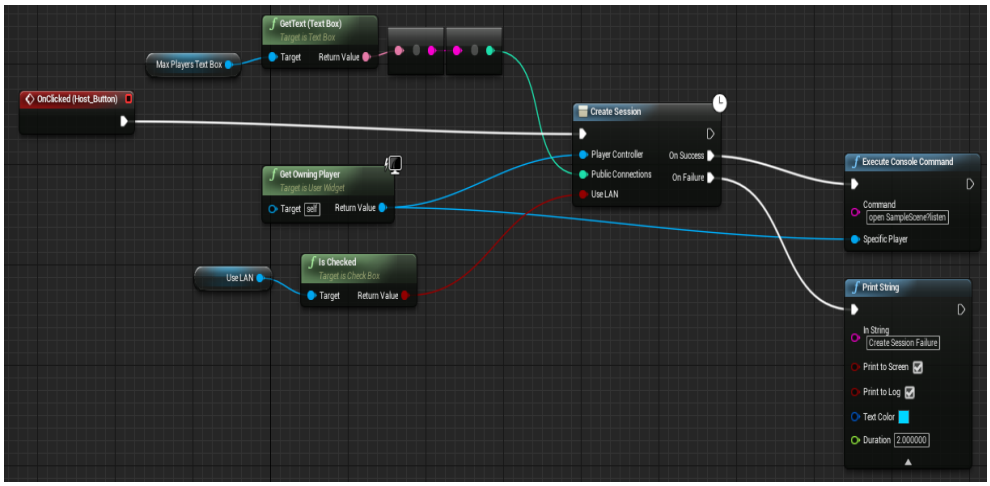
From the widget I call the Set Current Session pass this into the Found Widgets array and then add this to the child.



Host Game and Create Server:

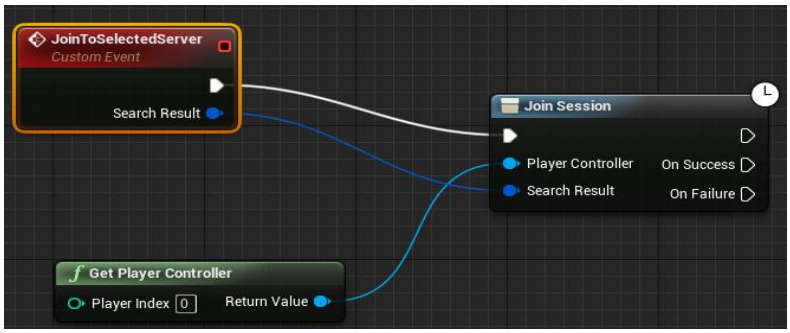
When the user clicks the Host button, I call Create Session. In the Create Session I need to pass in the Public Connections, this is the max number of players which the user set. The Public Controller is the Owning Player which is the user. Then I needed to pass in

whether or not the user wants to use LAN, so I passed in the value of the checkbox from the widget. Now onSuccess I execute the console command “open SampleScene?listen”, this opens the main level of my game, and set the specific player to the owning player, which is the main character. Also onFailure, I alert the user “Create Session Failed”.



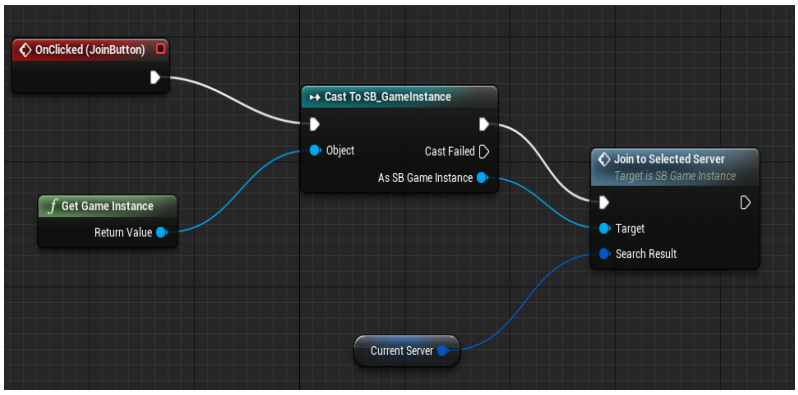
Join Session and Server:

To join a server I call the selected server which has been found by the search, so when the user click on the server I call the Join Session method. The Player Controller is the Player Character and the Search Result is the server which was clicked on by the user.

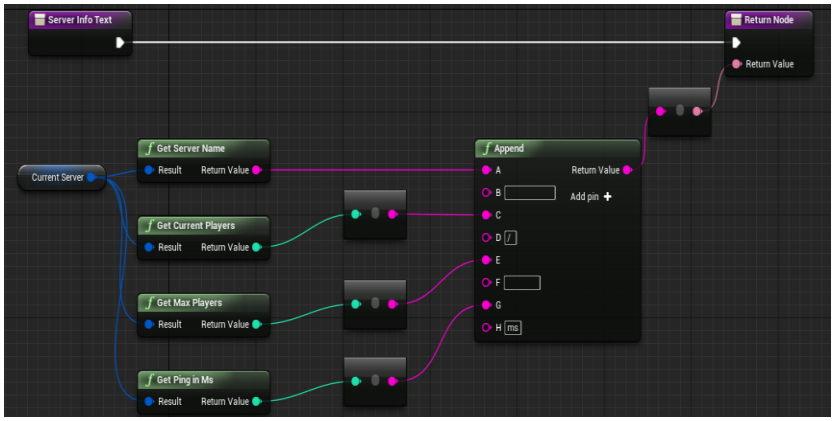


Join Session Button:

I first had to cast this to the session being joined which has an object of Game Instance, then this will go to Join Selected Server, this has a target of the Game Instance and a search result of the Current Server.



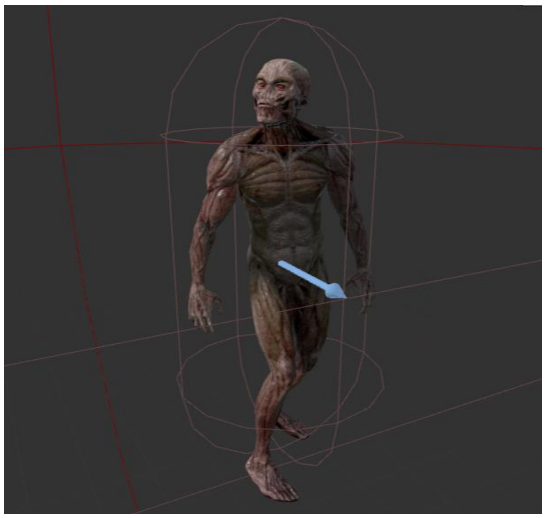
In order to get the text of what is on the button I make new string based on what the results are, these all come from the Current Server variable. So first I call in the server name, Current Players, Max Players and then Ping. Then this gets passed into the return node.



9.1 ZOMBIE AI:

Nav Mesh:

This controls the area that the AI can walk and calculate where to go in.

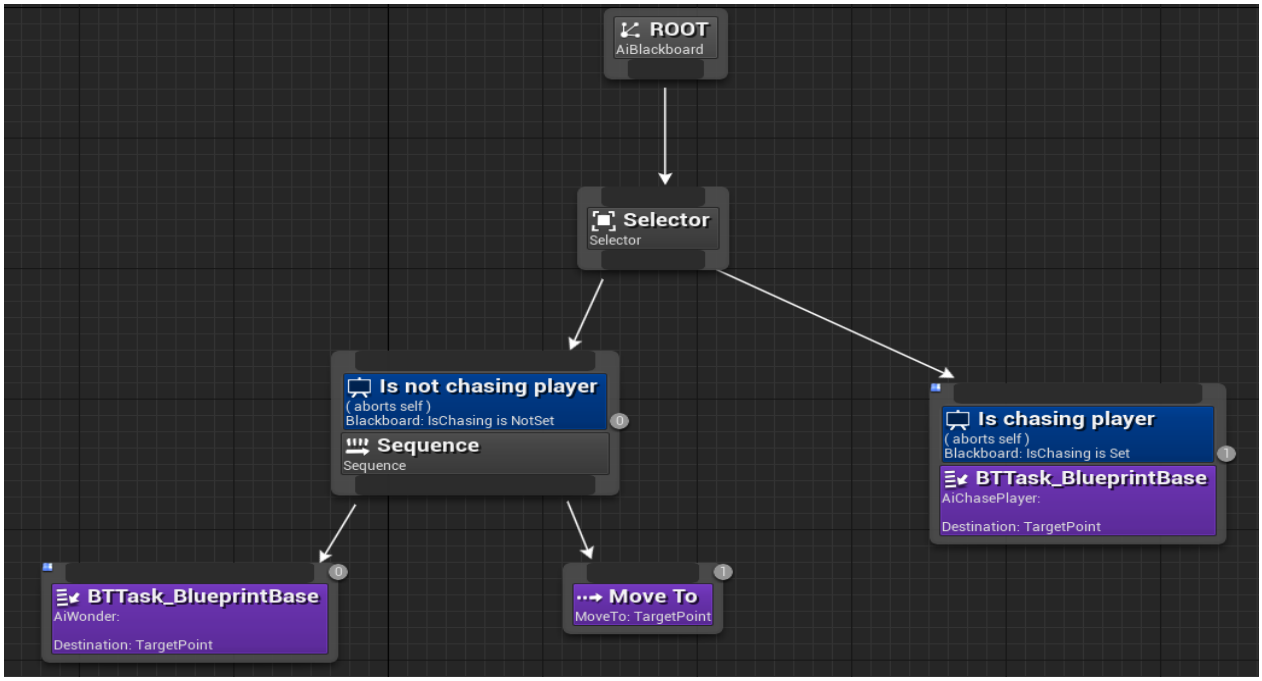


AI Behaviour Tree:

Controls the behaviour of the AI. Event-driven behaviour trees avoid doing lots of work every frame. Instead of constantly checking whether any relevant change has occurred, the behaviour trees just passively listen for "events" which can trigger changes in the tree. Having event driven architecture grants improvements to both performance and debugging. Since the code does not have to iterate through the entire tree for every tick, performance is much better. Conceptually, instead of the AI constantly asking “is the player in my line of sight”, the program can rest until they are prodded and told “there’s the player, now attack!”. When stepping forward and backward through the behaviour tree's execution history to visually debug the behaviour, it is ideal to make the history show relevant changes and not show irrelevant ones. In the event-driven implementation, it is not necessary to filter out irrelevant steps that iterated over the tree and chose the same behaviour as before, because that additional iteration never had to happen in the first place. Instead, only changes to the location of execution in the tree or to blackboard values matter, and it is easy to just show those differences.

In my Behaviour Tree below I start this off with the Root node, I then pass this into a selector and the selector has 2 options to either chase the player or not to chase the player. These are both called from the Blackboard Boolean I have created. So if the player is

getting chased this will call the AIChasePlayer class. If the player is not getting chased it will then call the AIWonder class and begin to move the AI to them spots.



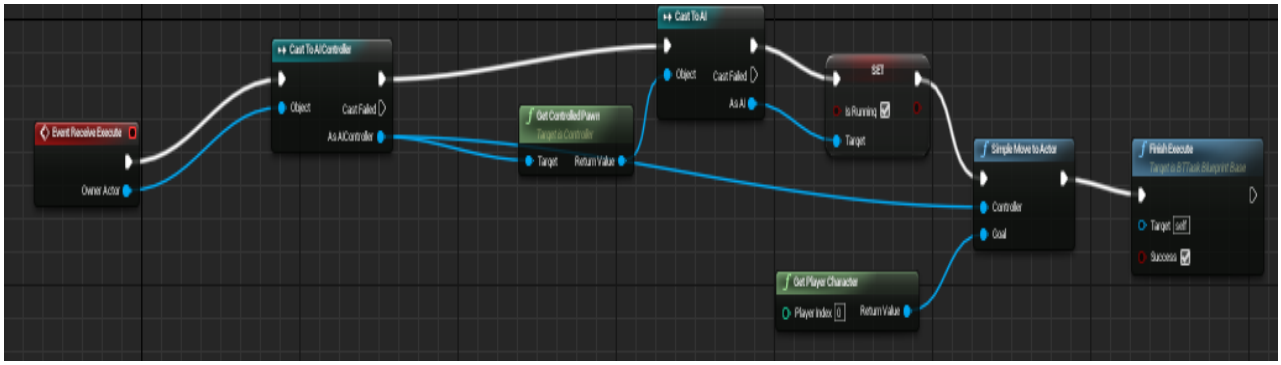
AI Blackboard:

This stores the variables that the AI needs, so the behaviour tree can access them at any time. The photo below shows the variables I needed to create and push out to other classes. The TargetPoint which is the bounds of the points that the AI can wonder to, IsChasing which is a Boolean of whether the player is getting chased or not, then SelfActor which is a reference to the AI Zombie itself.

AI Chase Player Class:

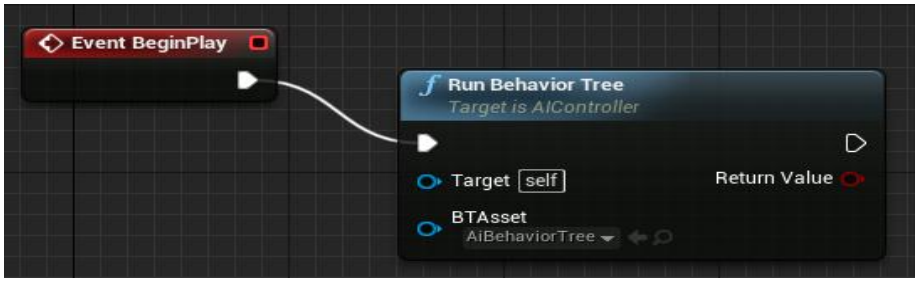
The AIChasePlayer class makes the logic so when the AI Zombie is in sight of the player he will chase after him.

Firstly, I call the AI Controller, this basically controls the whole AI. Then I bring out the pawn and pass that into the AI itself, so this is passing the AI logic into the AI character. Then I set is running to true, so the Zombie will run after the player. Then I can call simple move to actor in which the goal will be the main player. Then I can finish execute.



AI Controller Zombie Class:

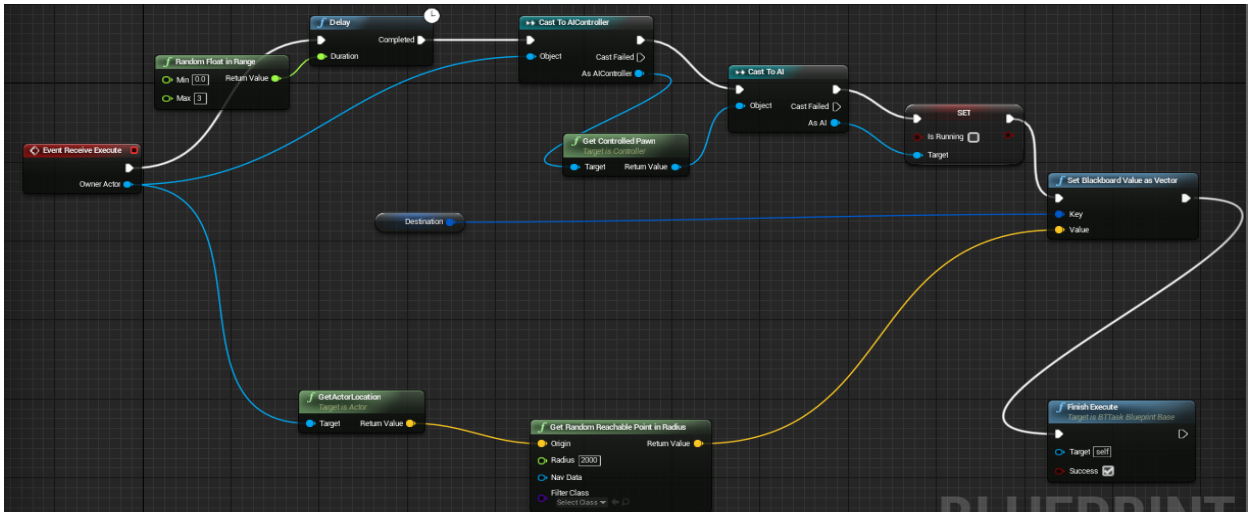
This controls the AI Zombie, which will run the behaviour tree and critically start the process.



AI Wonder Class:

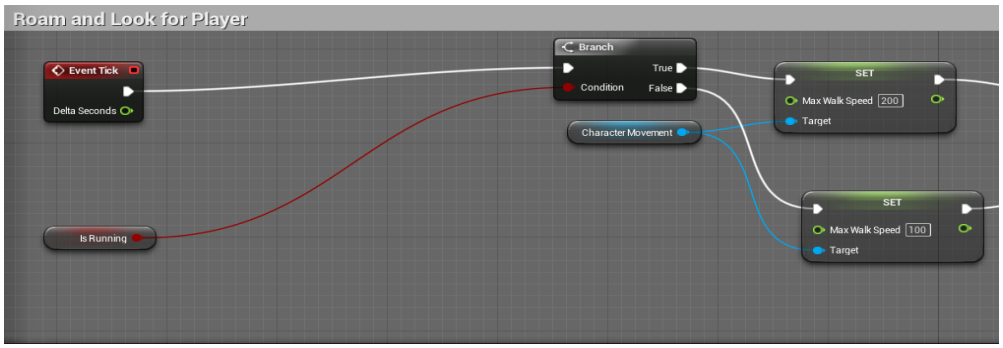
The AIWonder class sets out the random point within a given radius that the AI can go. So the AI Zombie can pick random spots in the map in search of the player.

In the code below I am first setting a delay on the random point in range, this means that when the AI Zombie hits their destination they will stand in that spot in an idle pose for 3 seconds, this gives the zombie a feel of a thinking effect of where they are going to go next. Now I call the AI Controller, this basically controls the whole AI. Then I bring out the pawn and pass that into the AI itself, so this is passing the AI logic into the AI character. Then I can set IsRunning to false so they are walking around to the random spots in search for the user. Then I am getting the random point in a given radius of 2000, and setting that random point to the variable Destination from the Blackboard. Then finish the execution.

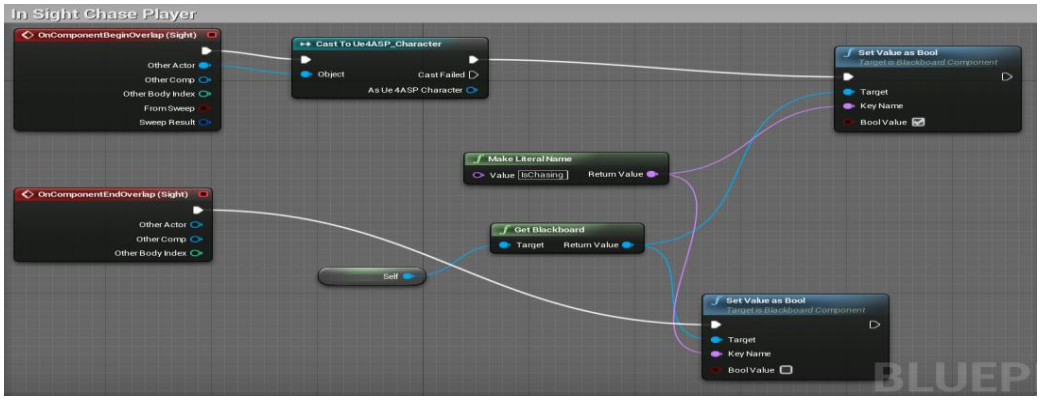


Code on Character Class:

The below code is based on the AI Zombie itself. Firstly I called an Event Tick, so this will call every second. Then I created a Boolean variable called IsRunning, then brought this into a branch. A branch is basically an IF Statement. So if the player is running make the walk speed 200 and if he is not running then make the walk speed 100. This enables to AI to run towards the player when they're in sight of them, and walk when they're roaming the map in search for them.



Also in the AI Zombie character I had to create an algorithm that triggers when the player overlaps their vision it triggers the IsChasing event in the Blackboard which inevitably notifies this in the Behaviour Tree so the Ai Zombie now chases the player. So to do this I first has to create the event overlap for the Zombies sight, then cast this to the main character, this is so that only when the main character is in their sight then they will begin to chase. Then I set two functions called set value to bool, and off this I called the IsChasing event from the Blackboard and set this bool to true, which means they are chasing. With the second set to bool I set IsChasing to false and bring this to end overlap on the AI's sight. So when the player leaves the Zombie AI vision he begins to roam the map in search of the player again.



9.2 BOSS AI:

AI Version 2: Girl Scout Boss

This version of my AI is more complex as the boss will have more available points to roam to and also able to see through walls and pick the quickest available path to get to the player once spotted.

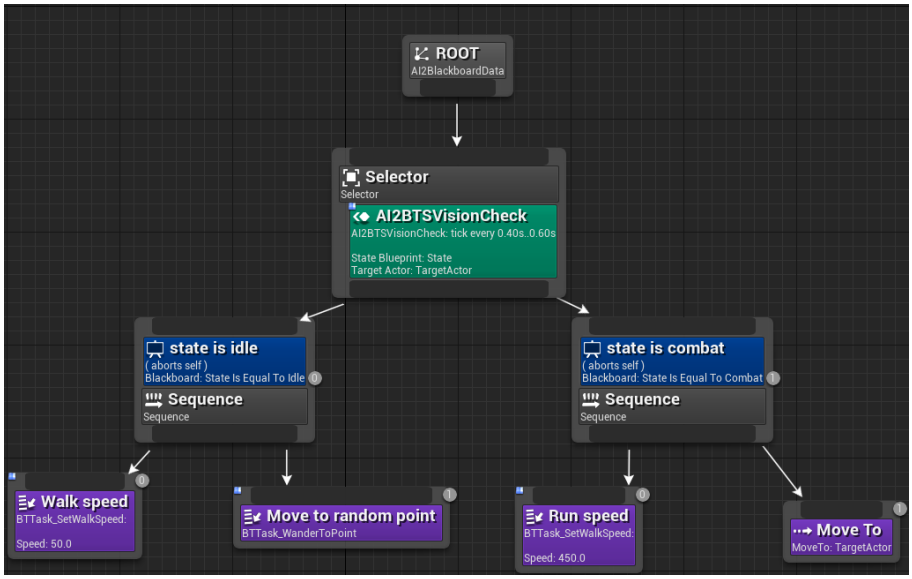
Nav Mesh:

This controls the area that the AI can walk and calculate where to go in.

AI Behaviour Tree:

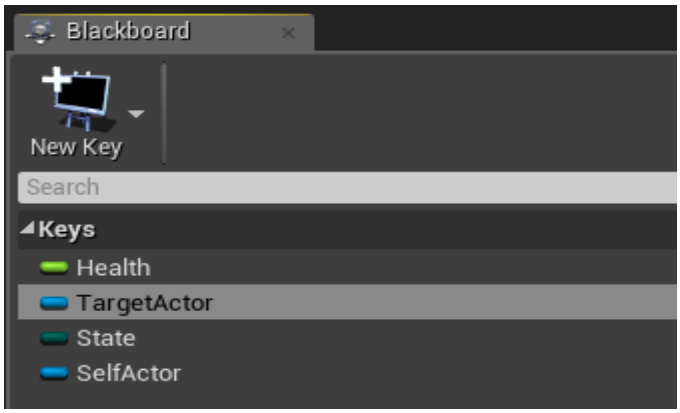
The behaviour tree for my boss AI starts at the root node. From this I bring it into a selector and call the Vision check class to see if the player has been spotted or not. If the player has not been spotted then we got down the left hand side of the tree which calls the idle state, sets their speed to walking pace and sets them off to a random point on the map in search of the user for the AI.

If the vision check class states that they are in sight of the player then we go down the right side of the tree. This states that they are now in combat mode and sets their speed to running pace, and the move to now become the TargetActor which is the player.



Blackboard Data:

This allows the AI bot to store the data that the functions can reference to. In my blackboard I have 4 keys that I need, Health, TargetActor, State and selfActor. The Health would simply be the Health used for the player, the TargetActor would be the next possible sport that the AI can move to, State would be a reference to the enumerator I have created which would toggle between the two possible states that the AI can be in, either Idle or Combat, then finally selfActor is a simple reference to the AI boss itself.

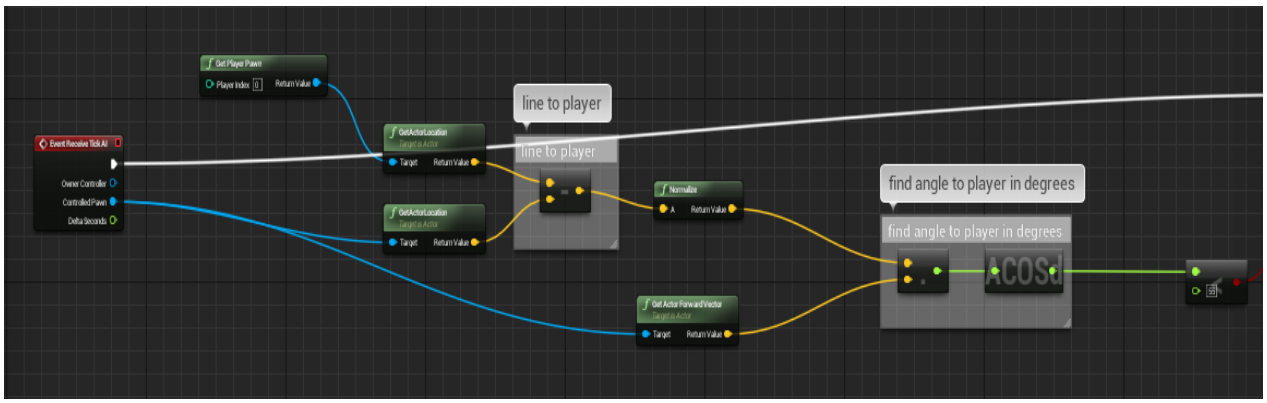


AI Vision Check:

This class is to determine if the player is in the line of sight of the AI boss, and if so then switch between the two enumerators that I created either idle or combat.

To achieve this I first created an event receive tick, so this will tick every second and check this. This will go straight to a branch to say if the player is in the sight of the AI or not. The condition of the branch is the line of sight of the AI, first I get the location of the AI and normalize this to pass into a vector, now I can get the actor forward vector and pass these two into a 3D vector format. I do this to get the location that the AI is in and also the direction that they are moving towards.

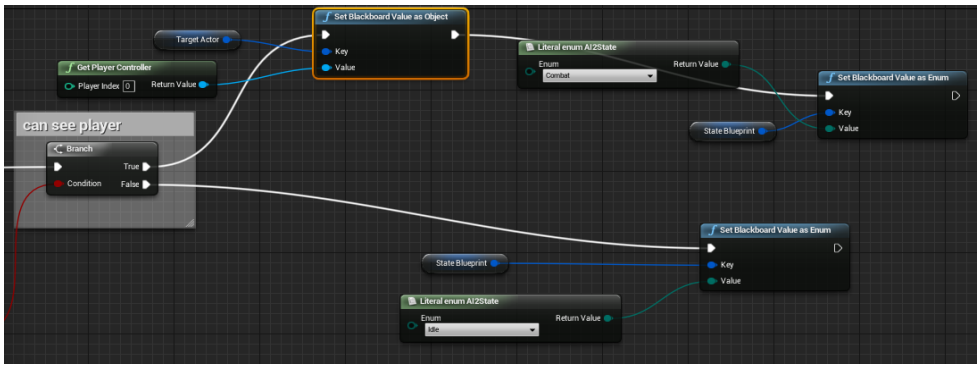
Now I simply transform the returned vectors into degrees and set this to 55 degrees, this allows the AI to see 55 degrees from head height, then pass this as the condition of the branch.



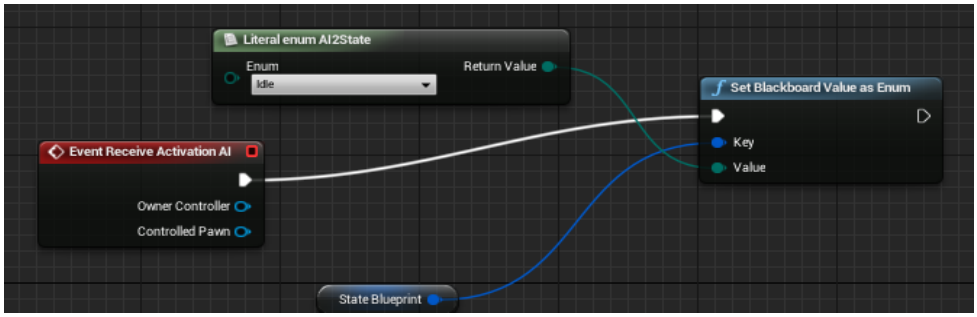
If the branch is true and they can see the player, we set the TargetActor from the blackboard data and set the value to be the player controller which is the main player. Now we have to set the Blackboard Value as Enum and the target would be the state and

the value then is to switch to Combat state so the AI boss now moves more quickly and starts to attack the player.

If the branch is false we Set the Blackboard Value as Enum again the key would be the State blueprint and the value would be Idle state, so the Ai boss moves at walking pace and returns on her route to look for the player.



Here I firstly had to set the Event Receive Activation of the boss AI to the Idle state so the she will roam the map in search of the player as her default move until the behaviour tree tells her otherwise.

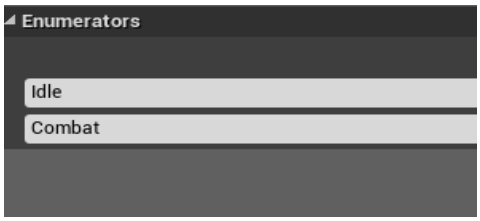


AI Controller:

Similarly to the first version of my Zombie AI, the boss AI needs to run a behaviour tree, this will run through the tree the second the character is spawned in game.

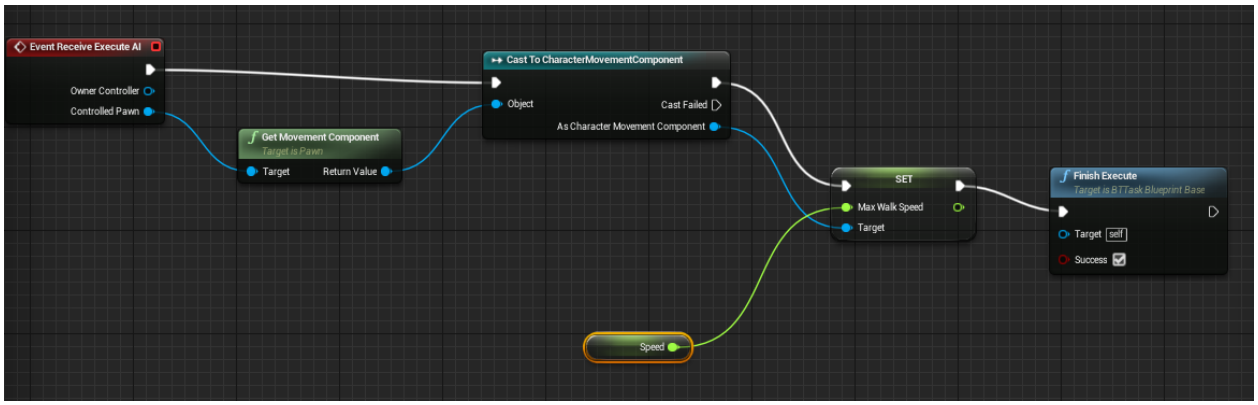
AI State:

Here I had to create an enumerator, these are used for defining options and states in a strut format. In my enumerator I have to states that the boss can be in, either idle or combat. These will called and referenced in the behaviour tree.



Set Walk Speed:

This class simply sets the speed that the AI boss is moving at. For this version I have created one variable called Speed. This variable is a reference from the behaviour tree as this is ultimately where the speed is being changed. So first I call an Event Receive Execute AI, so when the AI executes a move I can get the movement component, then cast this to the character component the set the walk speed to the Speed variable from the behaviour tree, the finish the execution.

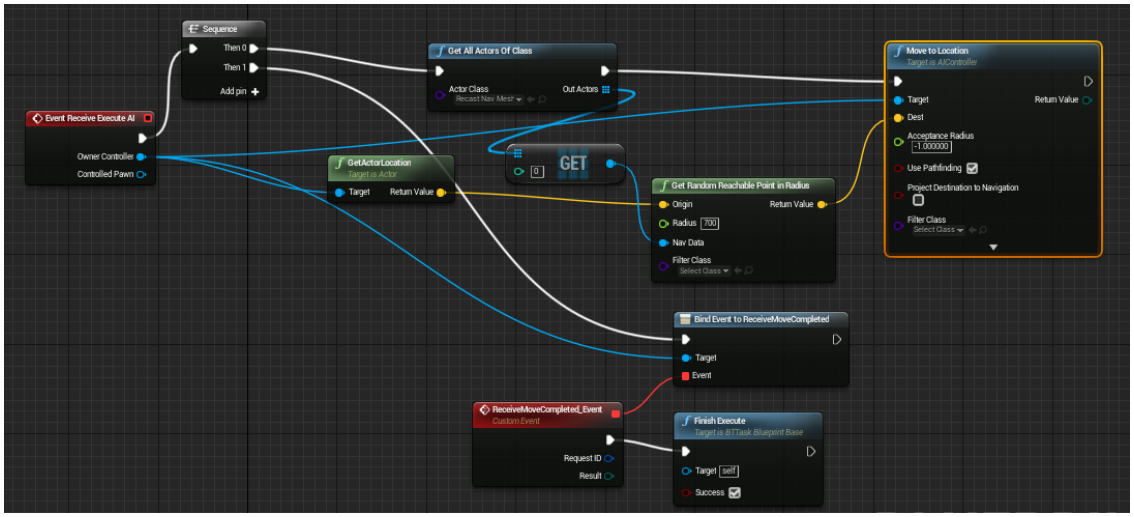


Wander To Point:

This class generates the locations that the Boss AI will move to.

Firstly I had to call Event Receive Execute AI and bring this to a sequence so it can call two functions at the same time. Then we get all the AI actors that are within the nav mesh get them from the array and move them to the calculated location. Off the other sequence we bind this to the move completed and the target would the AI boss, and when the Event is completed we call finish execute until another move needs to be calculated. To calculate the location I get the actor location then set the return value to the Get Random

Point in Radius and this radius is 700 and set this as the Destination in the Move to Location node.

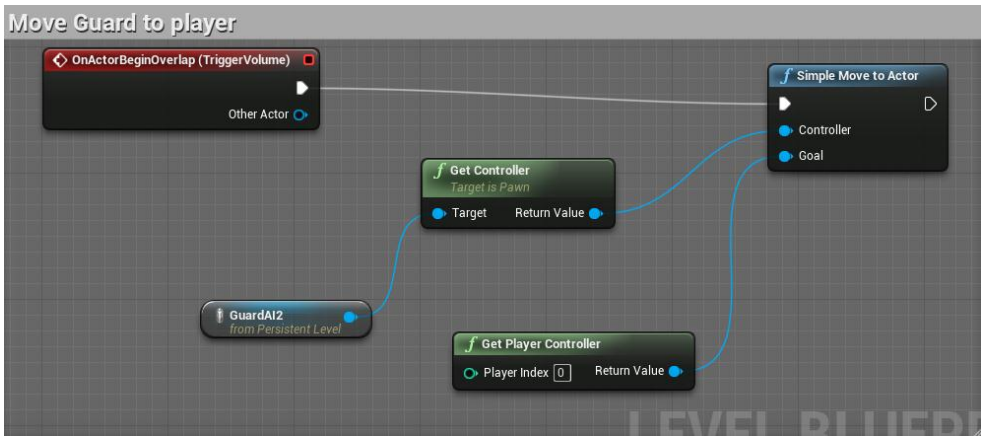


9.2 GUARD AI:

This AI protects a given door on the map which is the door exit the building. When the player comes within a given distance of the door, this will trigger the AI to move to the player and attack, and when the player leaves the location the AI will return to the door and guard it once again.

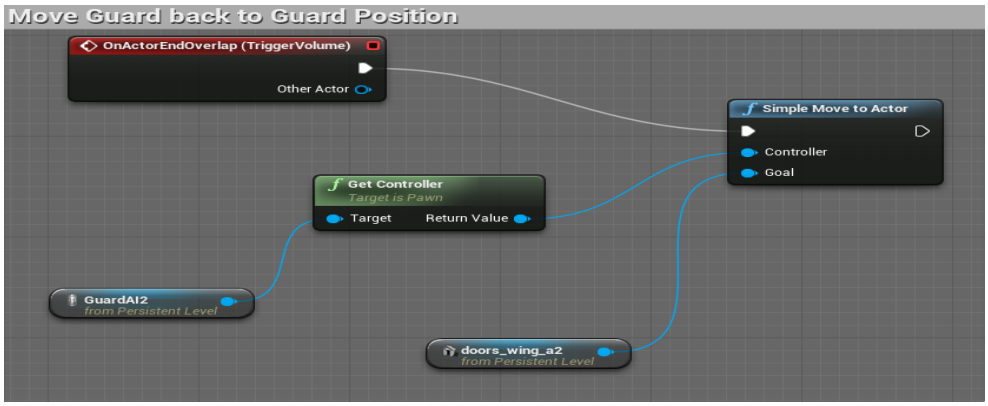
Overlap Trigger Box:

When the player overlaps the trigger box this triggers a move to the actor with the controller being the GuardAI and the goal being the player.



End Overlap Trigger Box

When the player leaves the location we call the move event for the actor, again with the controller being the GuardAI, but the Goal is the Door he is protecting.



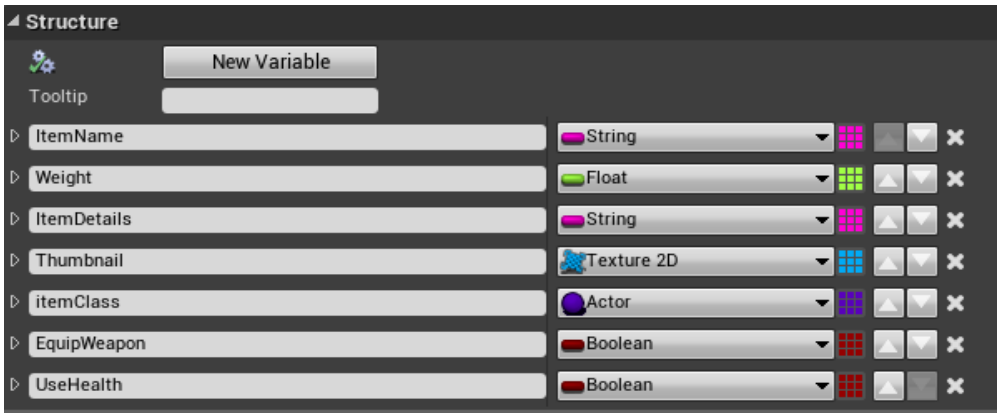
9.3 INVENTORY SYSTEM – STORE/EQUIP/DROP WEAPONS:

The inventory system allows the player to equip all the different items that they find throughout the level. When the user clicks on an item this will bring up information about that specific item, like item name, weight and description. From the inventory the user can equip and drop items. The user can only carry a max weight of 25, so they have to really think strategically which items they need to survive and escape the level.

Firstly I need to create what the widget for what my inventory would look like, and also assign variables that I would pass values into that would change, like name, weight and image. This is what is going to be displayed to the user when they activate the inventory in game. Then this would store the items that I pick up and then I can drop or equip the weapons and items

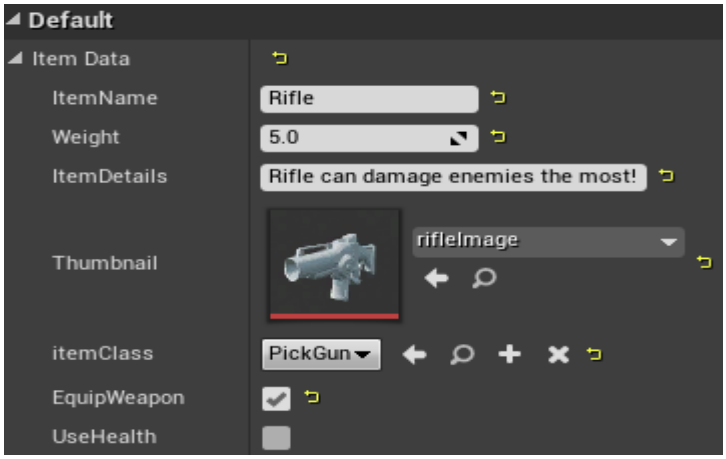
Structure:

To first step in the process is to create a Structure with each of the variables that is needed to pass them into the Array. This would be the Name, Weight, Details, Thumbnail, Class, Equip and Health.



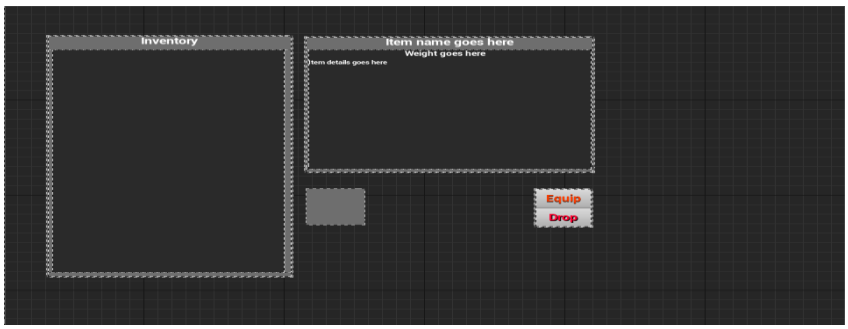
Weapons to Equip:

The user is able to equip numerous weapons throughout the map, like guns, cleavers, etc. In each individual class for the weapons I need to state the information and pass it into the array, so I need to give the name, weight, item details, thumbnail, class and whether the item is equipable or not. This needs to be done for every item that can go into inventory.

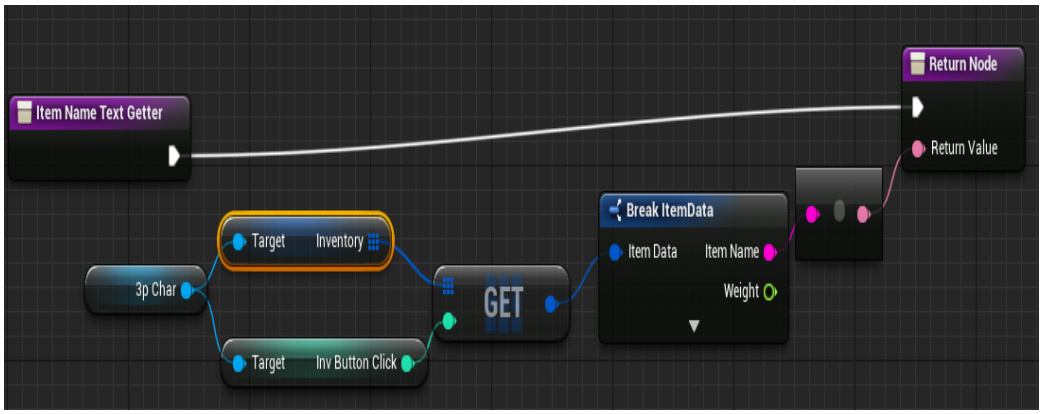


Inventory Widget:

In the widget I designed the look of the inventory which is going to be presented to the user.

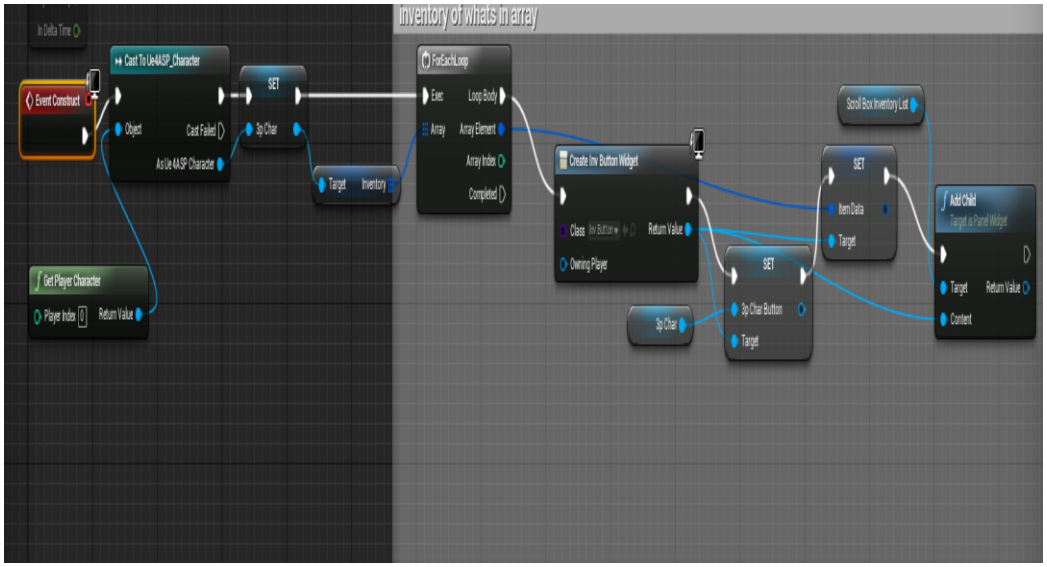


Behind each text box there is code which states which item has been picked and to display information based on that specific item. The below code shows that when the item is clicked by the user this gets the name of the item from the array and passes out the name to the return node. This is the same process for the image, details, weight, and class.



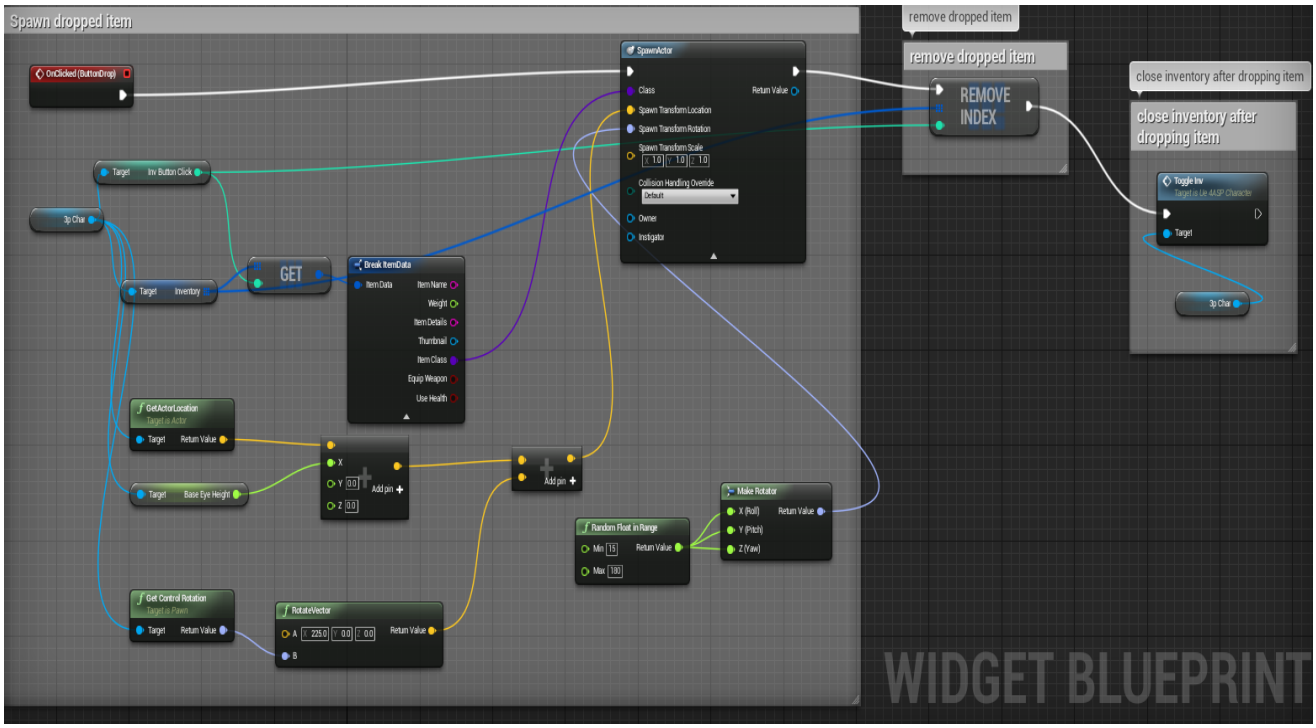
Get what is in the Array:

This starts by calling Event Construct to build the array. Then I need to cast it to the character as this is what is going to be using the inventory. Now I construct a ForEach loop of the inventory array. Now I call the widget of the and pass this on the itemData which holds all the information of each item. Then lastly simply add child, to pass in the items.



Drop Item from Inventory and Spawn it in Game World:

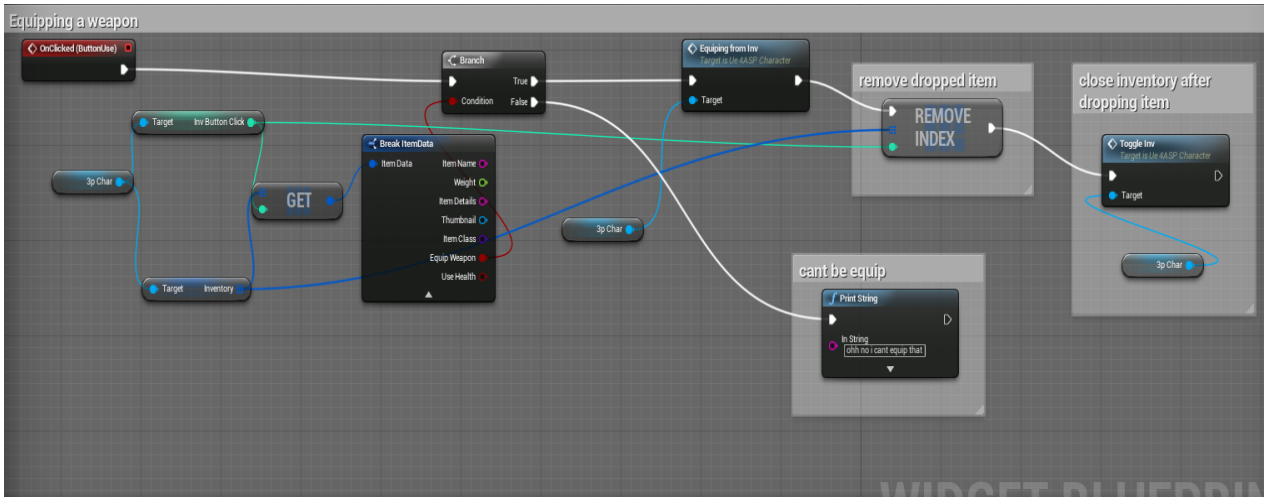
In order to drop the weapon from the inventory to make room for other items, the process is as follows. When the button is clicked this calls the SpawnActor function this takes in the class from the itemData Structure I set up earlier, I call what is in the inventory and get all the data to call the class. For the location of where the item will be spawned back into the game world I first GetActorLocation, so where the actor is in the game world and the EyeHeight, so it spawns out in front of the user, also then for the transformation I add random values in between 15-180 so the items rotates in certain directions to give the dropping effect a better feel. Now lastly once the code runs through the spawn actor I call RemoveIndex, so whatever index that item was in the array it is now gone from the inventory because the player has dropped it and then the inventory closes.



Equip Item from Inventory:

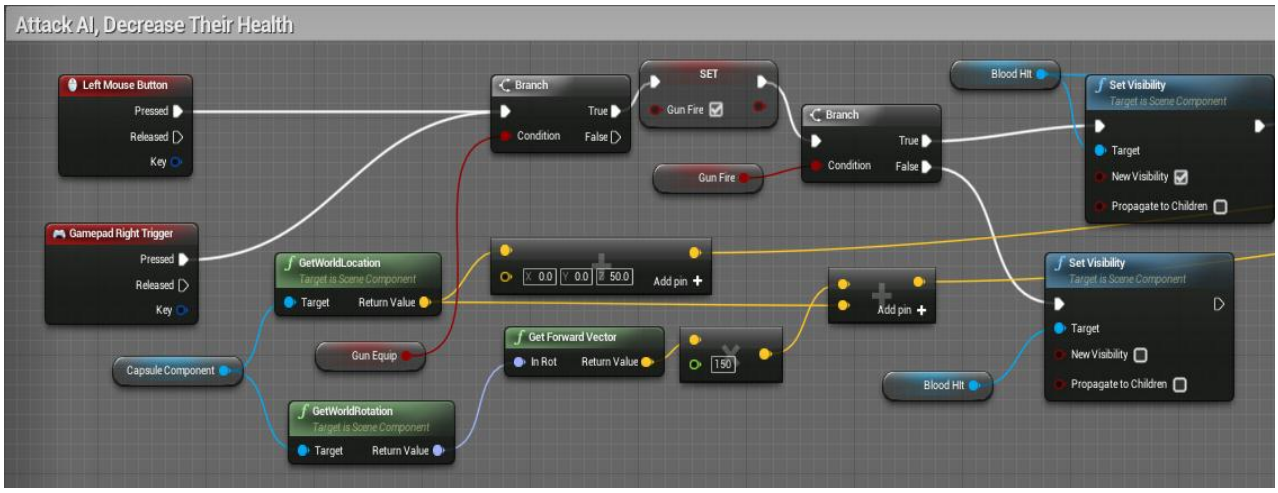
When equipping a weapon from the inventory I first call the button which is being clicked by the user. Now bring this to a branch. I get the ItemData again just like what I did from dropping the item, which retrieves all the information about the items. The condition of the branch is whether or not the EquipWeapon is true or false, if this is false then we pass a notification out to the user to say “Sorry that can’t be equip”. If this

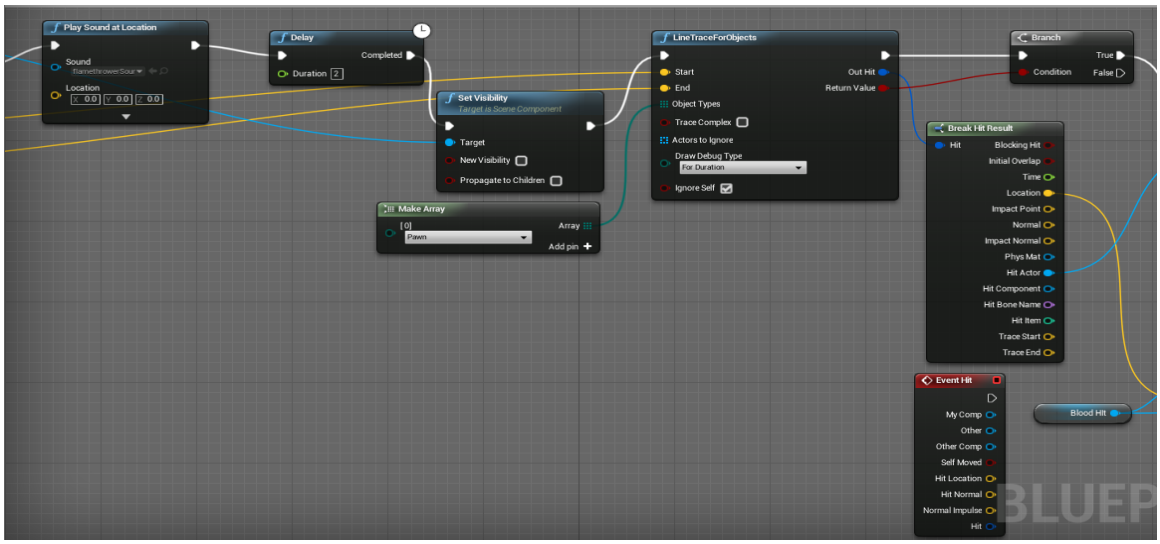
returns true I call the EquipingFromInv function and then remove the item from the index so it is not in the inventory anymore and then close the inventory

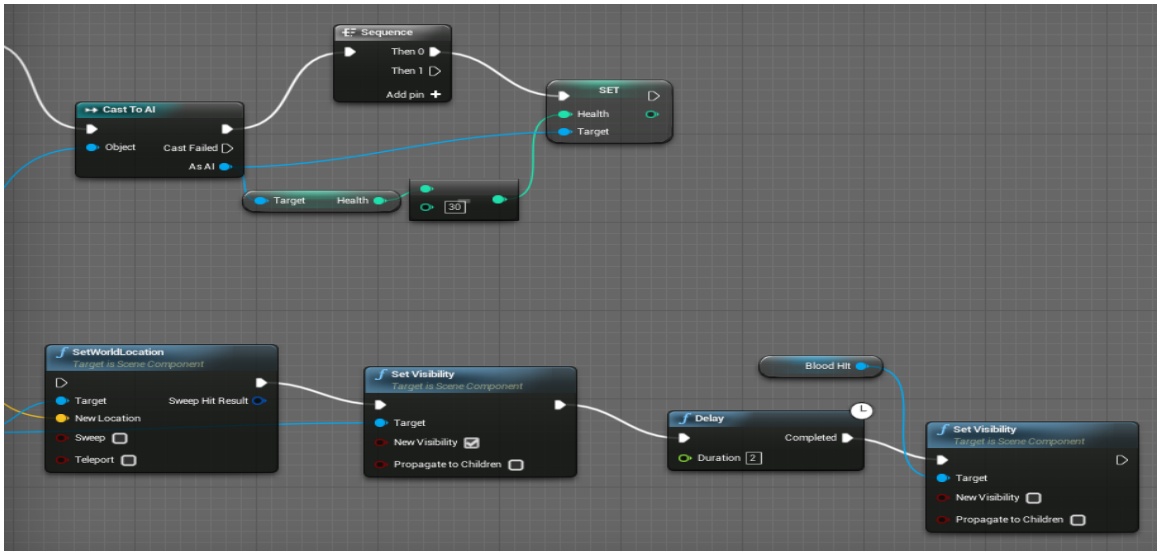


9.4 DAMAGE ENEMIES:

Firstly I get the capsule component of the enemy we want to do damage to, then I get the location of the map. This then gets passed into the line trace of the object which can only be activated by pressing the left mouse button. For the start point of the line trace I calculate an origin 50 on the z access, then this ends by getting the world rotation which gets passed into the forward vector and then multiples the forward vector and the world location. Now, gets put into a branch and casted into the AI as this is what we want to do damage to, set the health as 100, and for every hit we reduce 10 and then display this by printing a string. When they hit 0 this will kill the zombie. Then we just need to display the health widget on the screen.



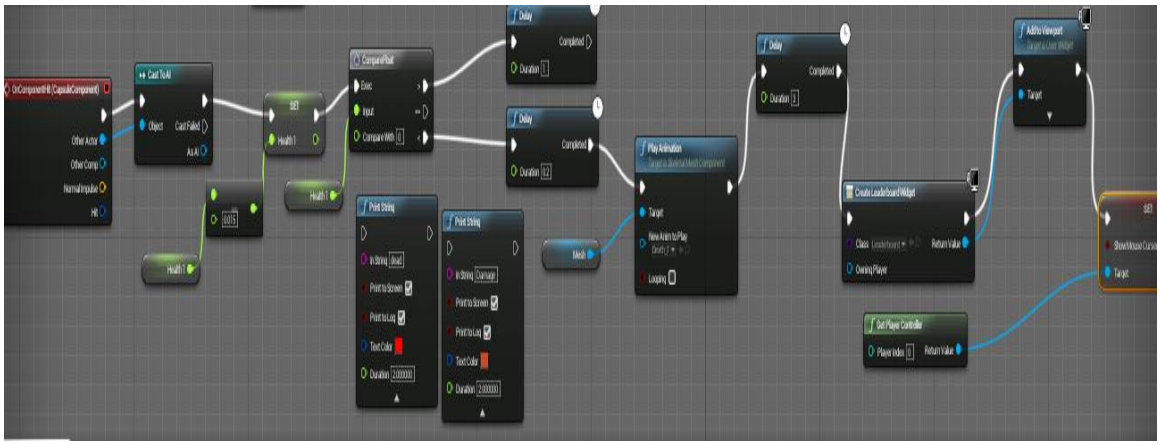




9.4 AI DAMAGE TO PLAYER HEALTH:

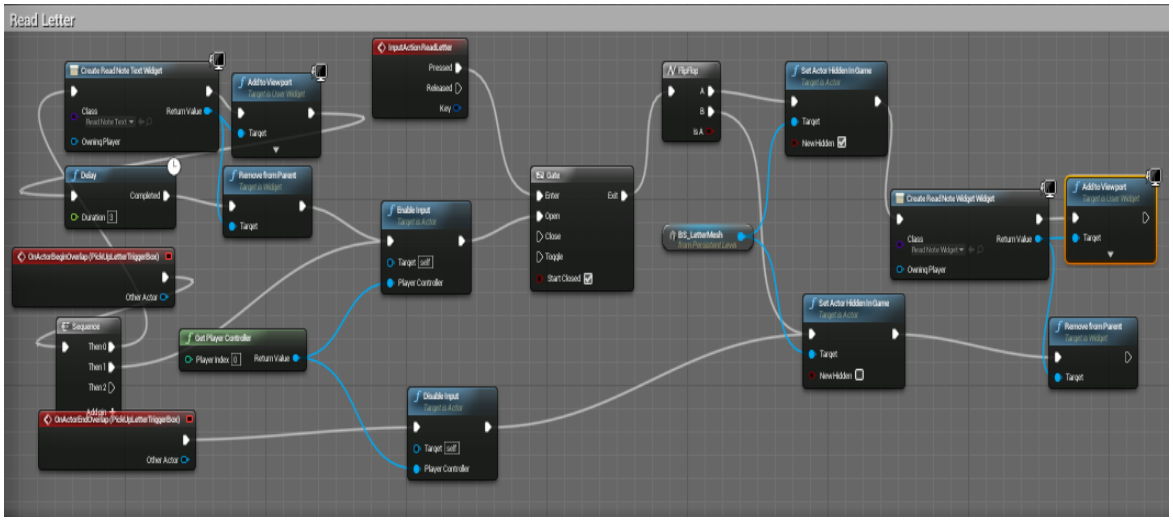
This allows the AI characters to do damage to the player, so for each hit they will do 15 damage to the main player in the game. This will also check if the player still has health after each hit and if the player doesn't have sufficient health after one of the hits this will play the death animation for the player.

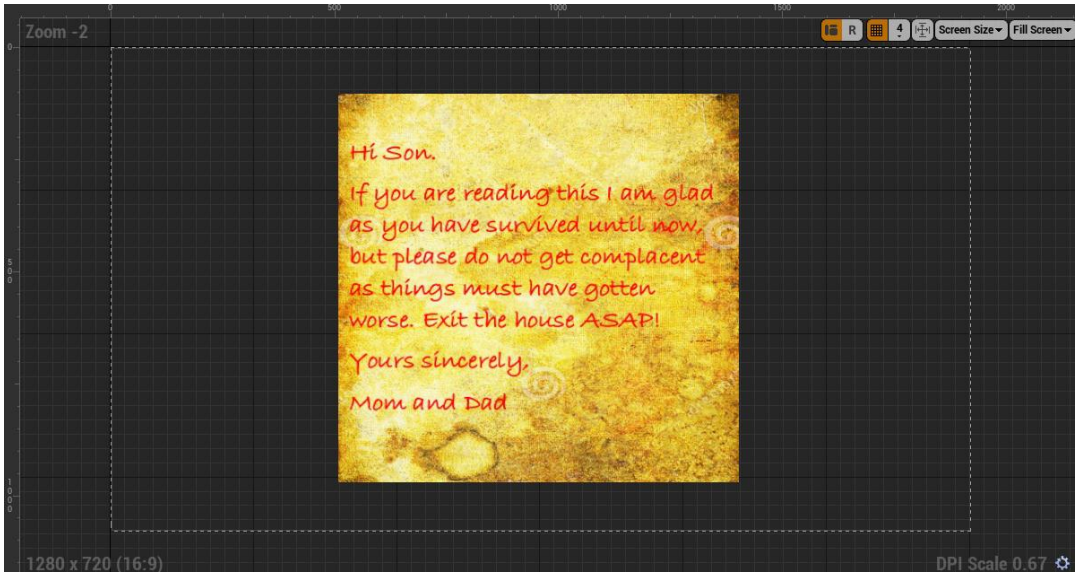
As soon as the AI makes contact with the player capsule I set the player health and subtract 15 damage from this. Then if the health remaining is greater than the damage nothing happens. If the health remaining is less than the damage I play the death animation and prompt the user to enter themselves into the leaderboard as it is game over.



9.6 READ LETTERS:

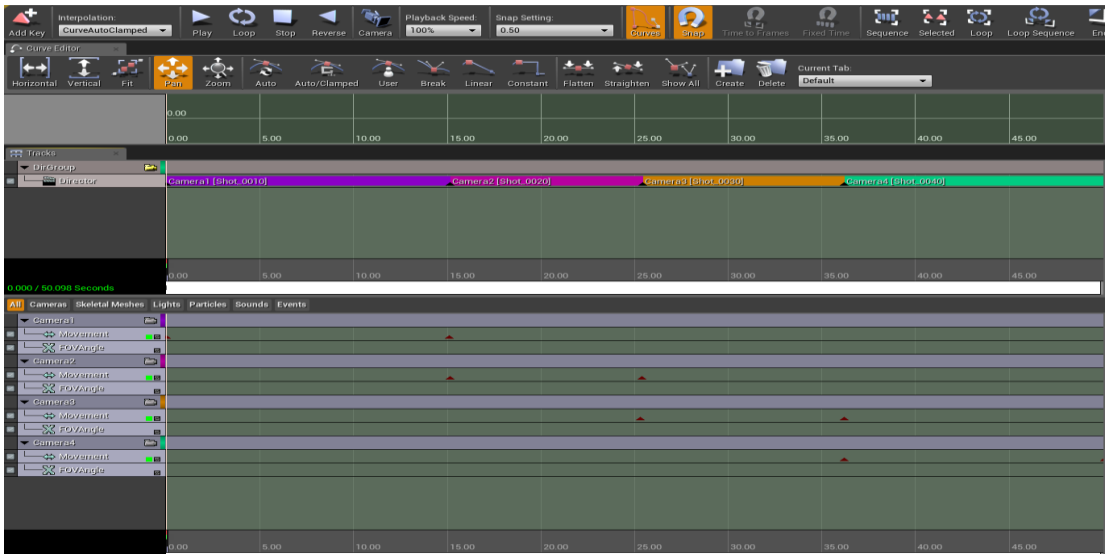
This bit of functionality is so the user can read letters placed around the map so they can get clues to how to escape, get better weapons and increase knowledge of background story. When the user overlaps a triggerbox this will prompt the necessary key the user needs to press to read the letter, which is “R” in my case. Off this then I enable input which means I now enable the key for the player to press and send this to gate, the key “R” goes to enter and enable input goes to open in gate. Now off the exit I need the flipflop function, this means that ‘A’ will always run before ‘B’ in the blueprint. So when the character picks up the letter this will be hid in the game as it creates the illusion that the character is actually reading the letter, now I create the blueprint of the letter I used and add to viewport, which adds the letter to the screen. Now off ‘B’ in the flipflop, I set the actor hidden in game and remove this from viewport when the user leaves the triggerbox, this is to ensure that the player can’t read the letter from halfway across the map for example.

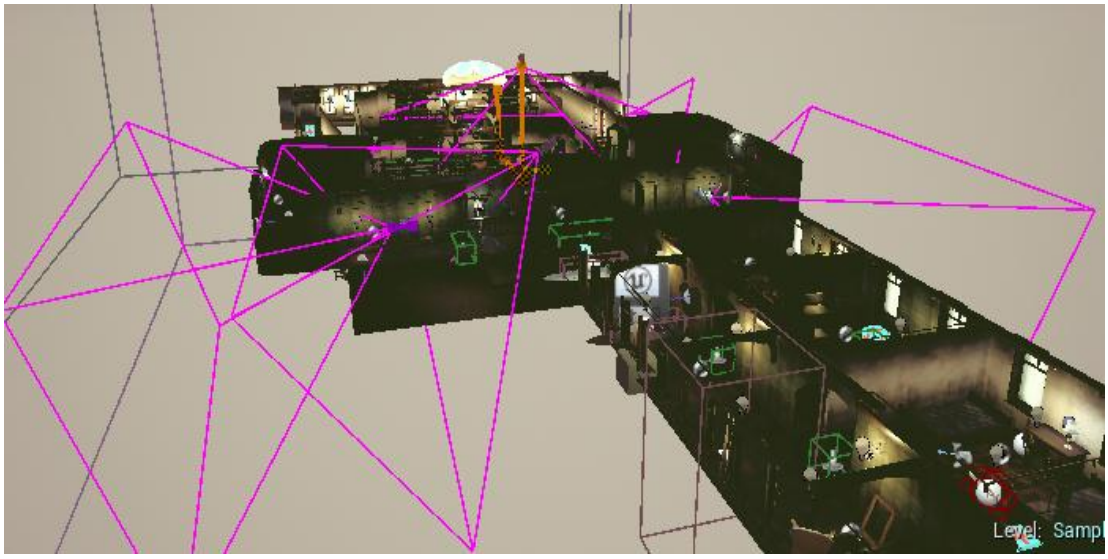




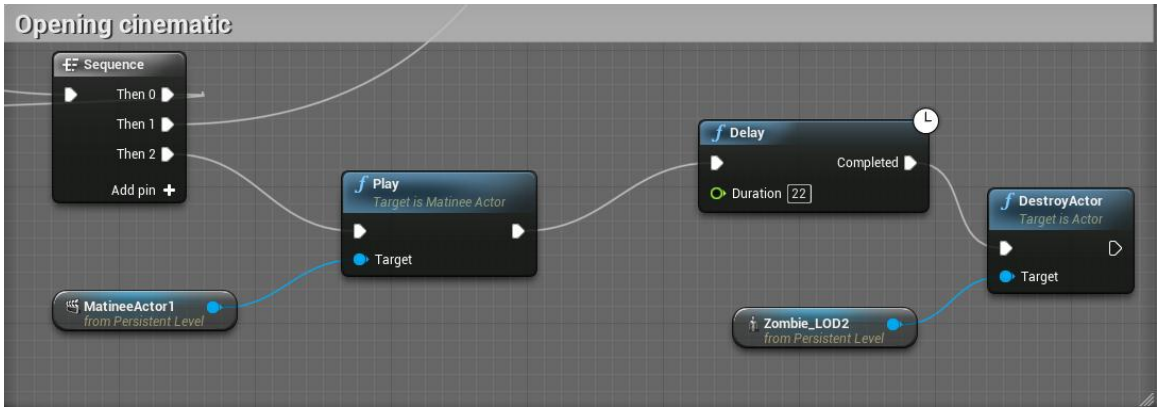
9.7 CINEMATIC/MATINEE:

In order to make the opening cinematic for my game, I first had to create a matinee which you can see below. Firstly I created 3 different cameras and then moved them all around the map in the direction I want my cinematic to look. As you can see the purple lines stand for the direction that the camera is going in. To move the camera all you have to do is create key frames, and from key frame to another key frame you can edit the movement of the camera you wish. Once all the movements was set, I then need to create an actor group, this allowed me to play the cameras in the order I wanted. So in my director group I play the camera in the order 3, 1, 2. To add sound then I added a sound track and chose the music I wish, I made this the same length as the cinematic.





When the level opens I call the play function and target the matinee cinematic I just created.

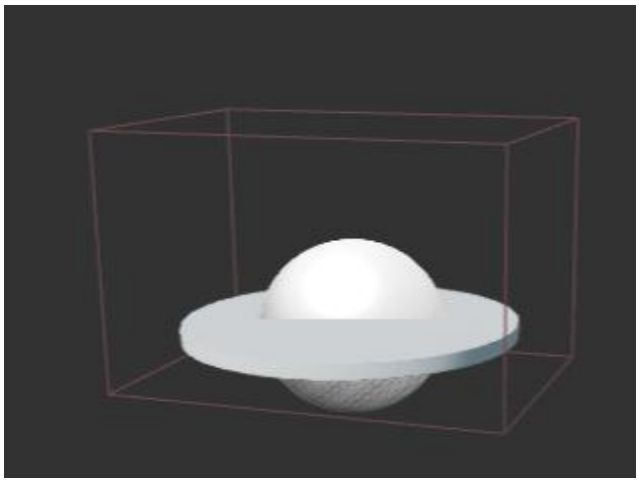


9.8 CHECKPOINT:

The checkpoint feature enables the user to save their progress that they have made if they need to exit the game they can easily load back in straight from where they have left off. They will resume with the same time, weapons, and enemies that they have killed.

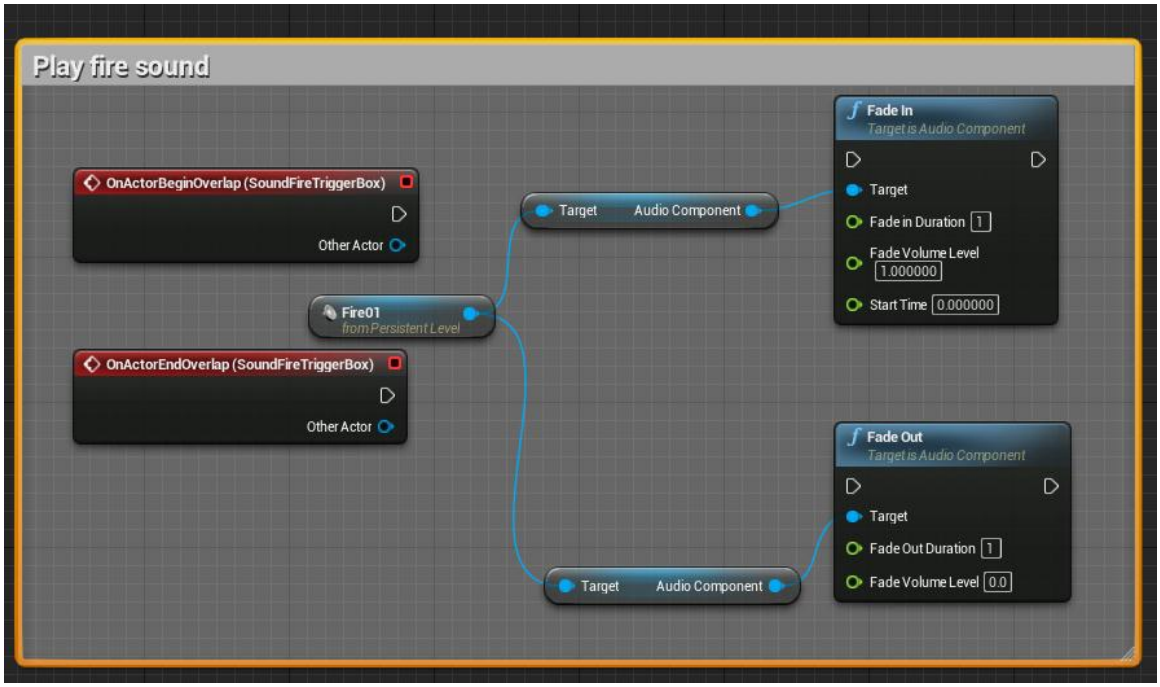
Firstly I need to cast the checkpoint to the main character so the checkpoint only saves when the main player walks over it, then we need to set the transform. The transform is the location of the sport that the player is in, this target of the transform is the main character. Off the transform I need to get a reference to the checkpoint mesh itself, then set the target to the world transform and break this to return at the location “z 128”, if I

didn’t set this and the scale of the make transform to 1 across the x, y and z access the player would disappear off the map when they respawn. Now I can simply send this to my save function and load function also.



9.9 FADE SOUND IN AND OUT WHEN PLAYERS GETS CLOSE:

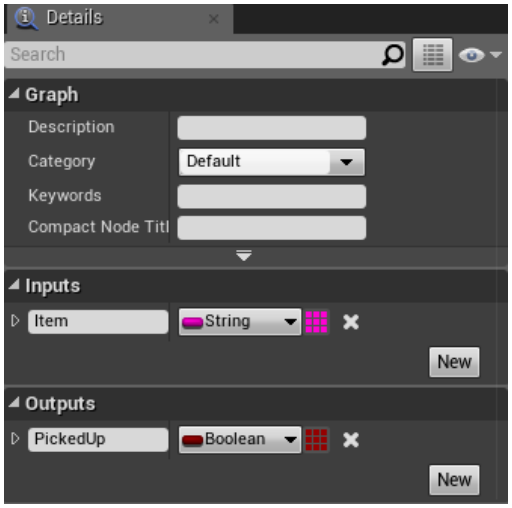
Below I have that when the player walks near an external source such as fire, the closer they are the louder they will hear fire noise, and if the player walks away from it the sound of the fire will fade away. For this I use an ambient sound where and connect the sound to fade in and fade out.

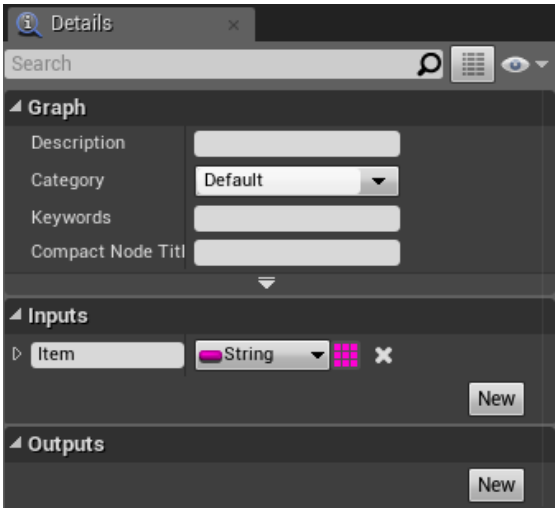


9.10 PICKUP KEY AND UNLOCK DOOR:

For the keys to unlock the doors I first created an Interface which has two functions, Add Inventory, this contains only one input which is a String called Item.

This also has a CheckInventory function, this contains one input string called Item, this also has one output boolean called PickedUp.



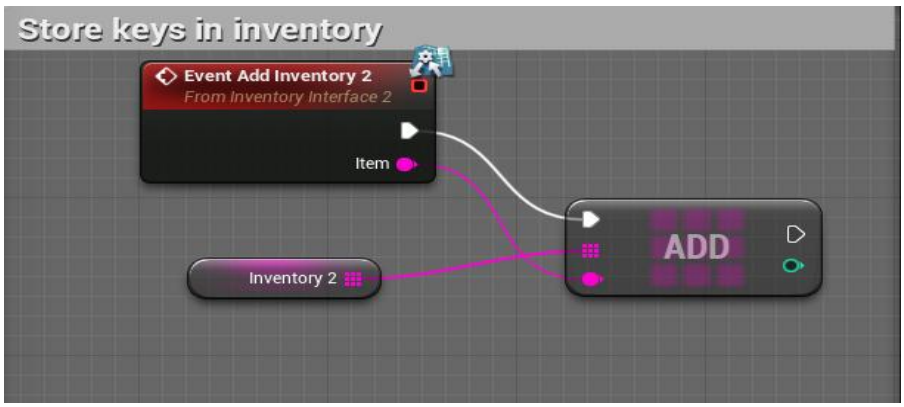


Create Array:

Now, in the character class I created a string array, in this array we inherit the interface which we created in the previous step.

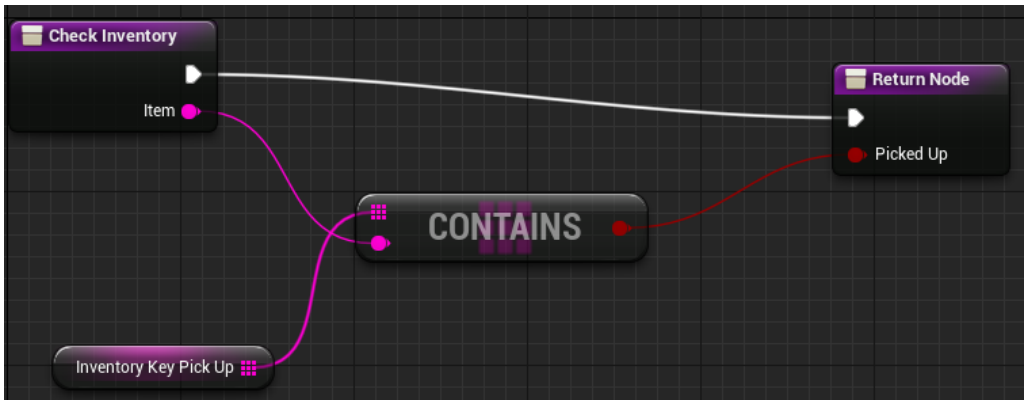
Add to Array:

Now we add the key that was picked up into the array, so call out the variable we created can add it to the function from the interface.



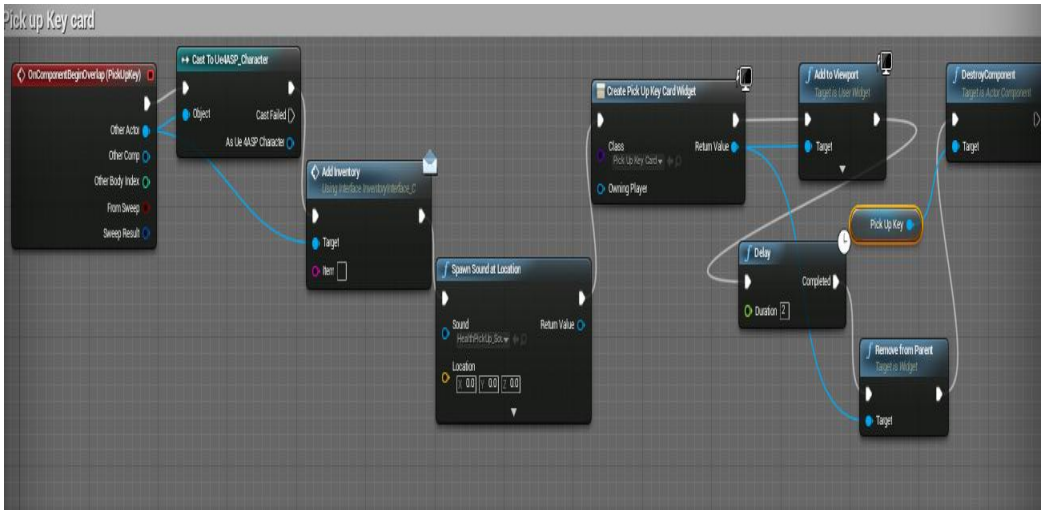
Check Array:

Now in the CheckInventory function we call the variable array we just created and see what it contains and return the result into the return node.



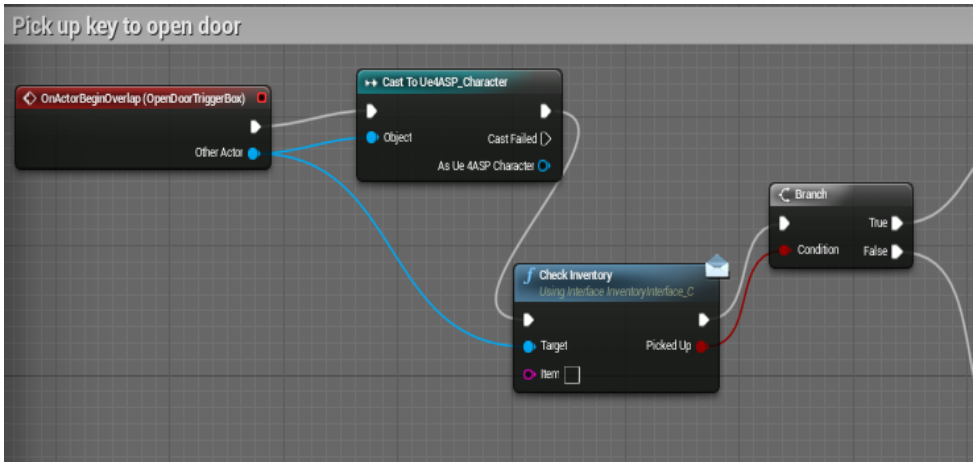
Store Key in Inventory:

When the player makes contact with the key we the call the Add Inventory function the makes a noise to alert the user that they pickup the key, now we display a widget to the user which says “Picked Up Key”, this widget will last two seconds then remove from the screen and then destroy the component.

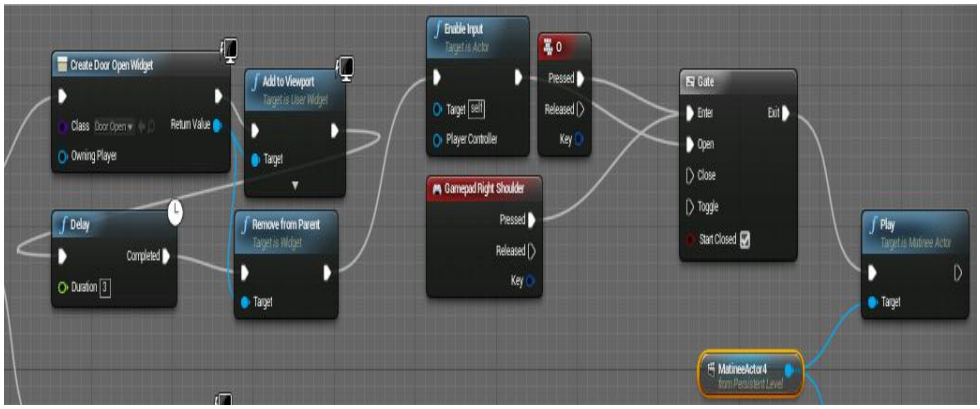


User Unlock Door:

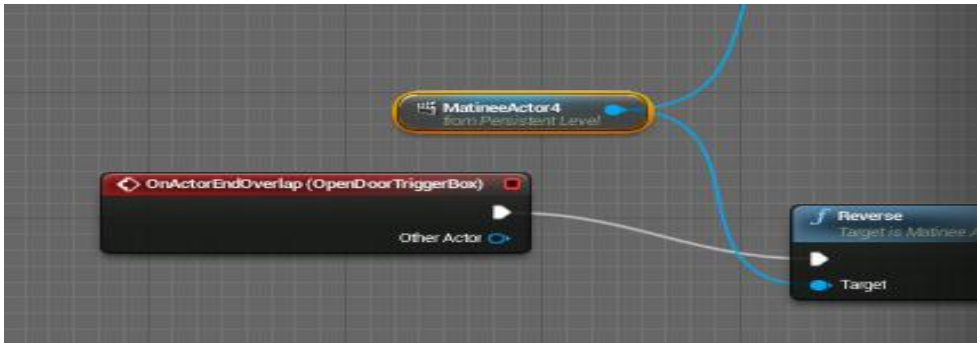
When the user walks up to the door this will check the inventory for the key



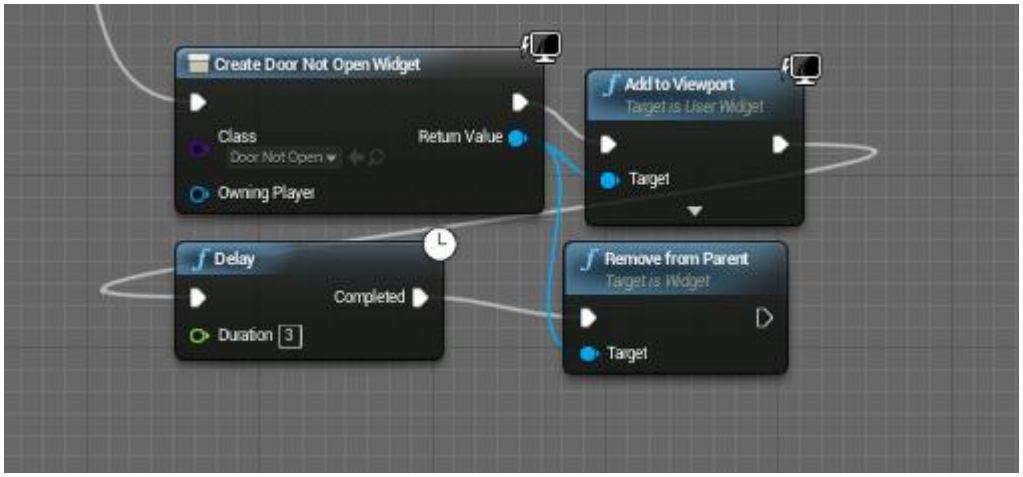
If the branch returns true and we did collect the key then we display a widget to the user which says Access Granted. Now I enable the input, and call the button I wish the user to use to open the door, in my case “O”. Now in the gate function this allows me to have a button open the door so I connect my button to this. Then off the gate, I selected and play the matinee I wish, which is to open the door.



When the user leaves the trigger box, I reverse play the matinee which closes the door when they leave the area.



Now if the branch returned false I display a widget to the user saying, “Please find the key for access”.

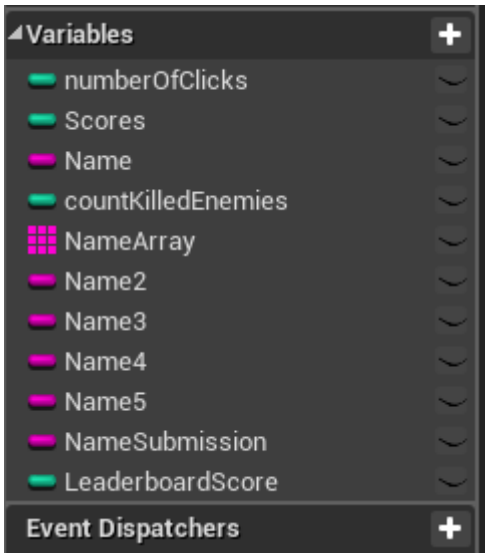


9.11 LEADERBOARD:

After the user completes the game they will be prompted with a widget where they can see their score and enter their name to enter into the leaderboard. This will be saved to a file which will be loaded back in to the game which can be viewed from the main menu. This will display the top 4 results that has been achieved in the game. This will display the players name and score and the rank they are in.

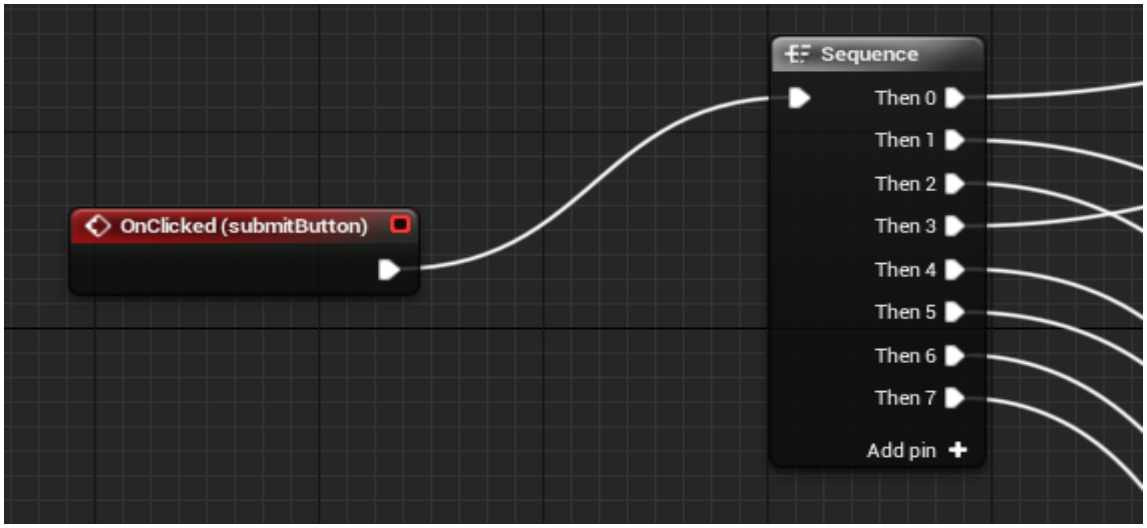
Save Game Class:

This holds the main variables which will be saved to the file which will hold all the scores for the different players throughout the game. These will be saved to the file and loaded back in by the widget.



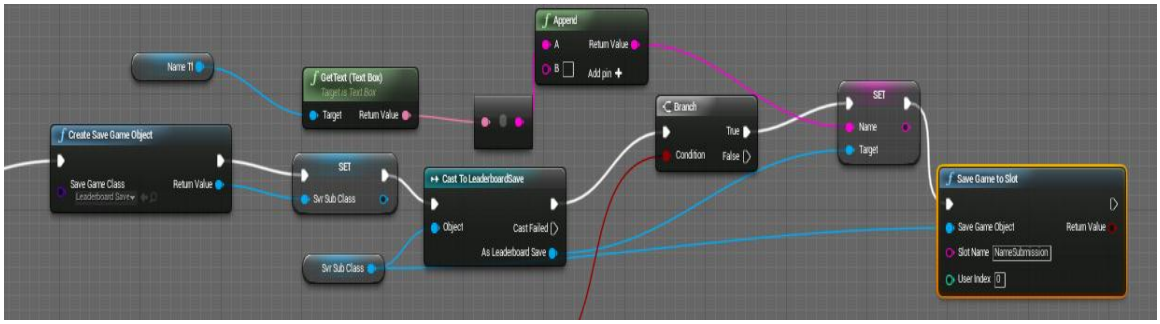
Determine Users Place in the Leaberboard:

When the user clicks submit this enters into a sequence node to check each place and name while working its way down the leaderboard, so first it will check the players name and score of 1st place, then determine whether the players name and score should go in there, then move onto 2nd place and if not and move onto 3rd and so on.



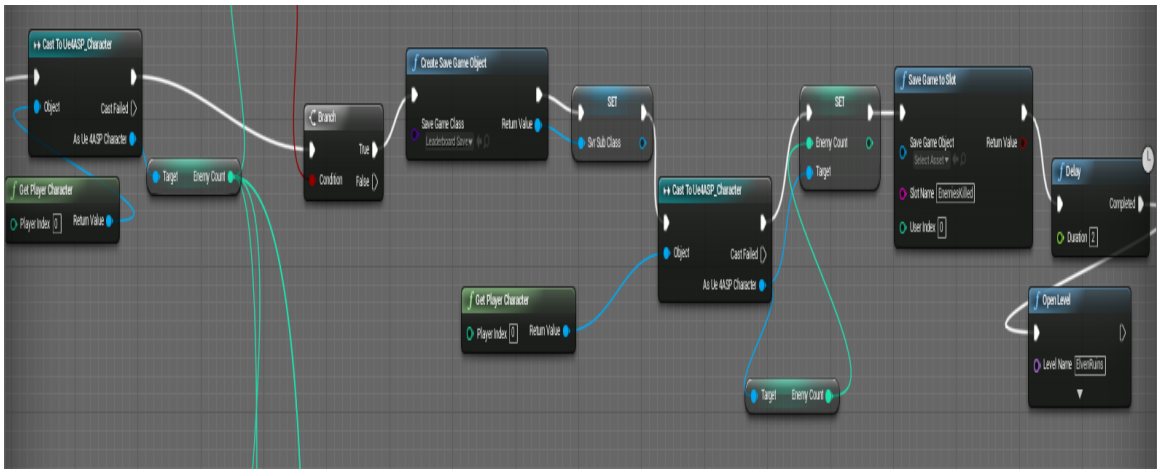
Enter Players Name and saving them to a file:

Firstly I create a save game slot which stores the users name and then I need to set the save game class which holds the variables which are being saved. Then I cast this to the LeaderboardSave, the object of this is the save game class. Now I bring this to a branch which has a condition of the players score, so if this score is greater than the score in first, then store the users name in first place, if not than move onto the next step in the sequence control node. So if it is I need to store the users name, so I set the name which is being taking in by the text box in the widget which the user types in. then finally I save this to a game slot called NameSubmission. This is done 4 times to check for each of the save game slots in the leaderboard.



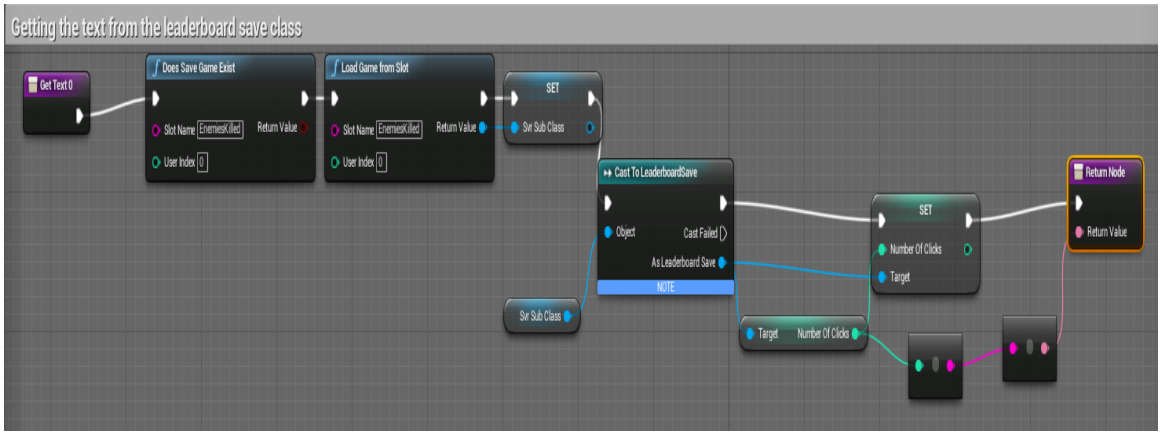
Enter Players Score and saving them to a file:

First I cast this to the Player class to retrieve how many enemies they have killed, the object of the player is the player character. Then off the player I create a branch which has a condition and if the players score is greater than the score already in that slot then this score now enters into that slot along with the player's name which I discussed in the previous step. So when this is true I create a new save game object which has all the variables for saving the players name and score. The set this class and cast it to the Player Character to get how many enemies have been killed and set this and save it to a new save game slot called EnemiesKilled and then open a new level to ask the player if they want to play again. This is done 4 times to check for each of the slots available in the leaderboard

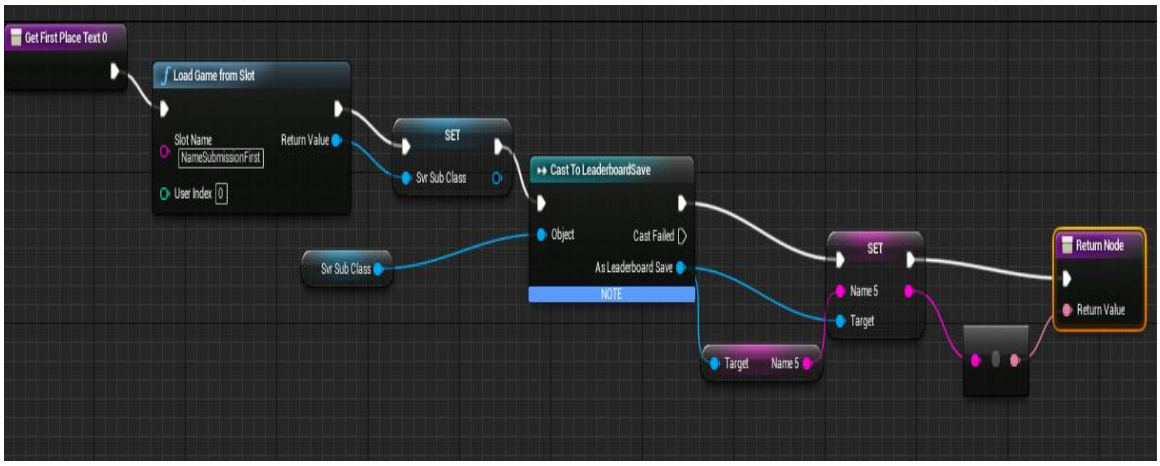


Display Leaderboard and loading scores back into Leaderboard:

In order to load the players score back in from the file, I first check if the save game file exist by calling the function and entering the exact name of the file into the save slot, so then if it does I call Load Game from Slot and enter the file name again, now I have to set the save game class and cast this to the LeaderboardSave which holds the variables and get the EnemiesKilled, set this to the new variable and enter it into the return node.



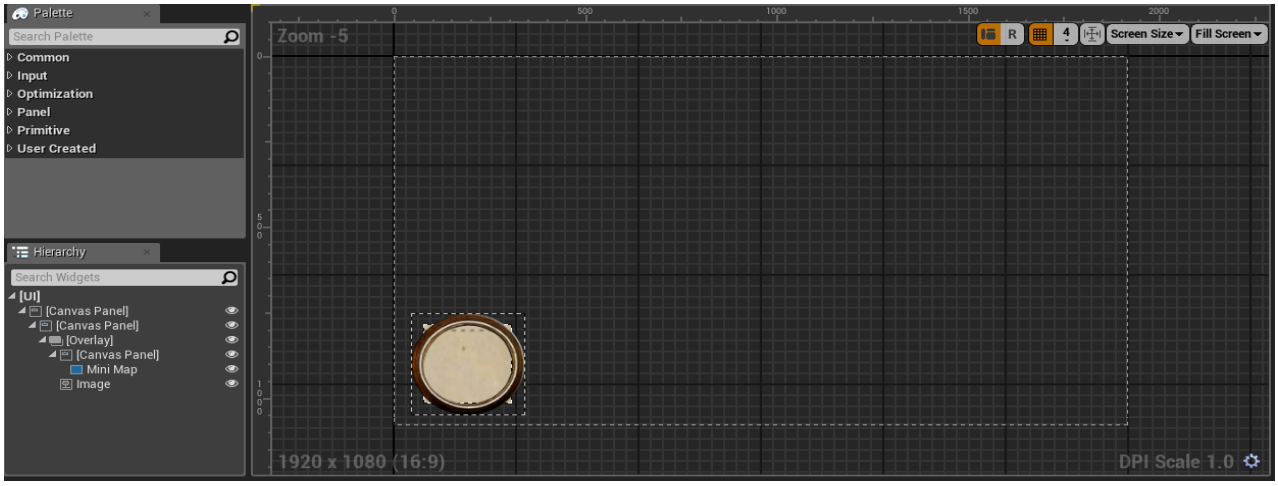
Now in order to get the player name to display alongside the score, I first check to see if the Load Game file exists and set the Save Game file, then cast this to the LeaderboardSave game class and call the Name variable which has been saved with the players name and then enter this into the return node.



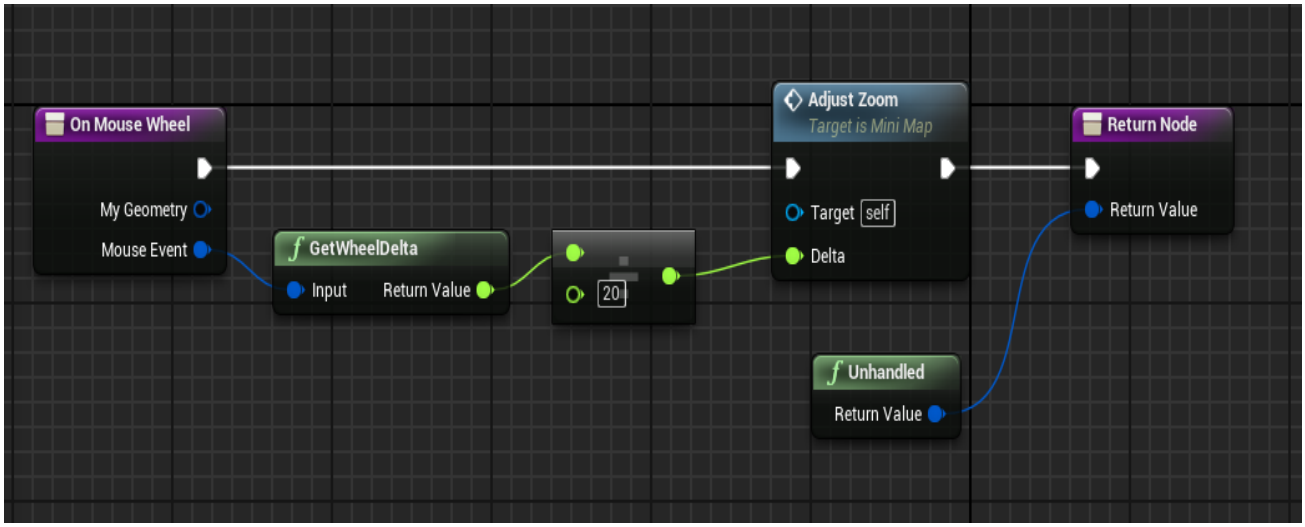
9.12 TRACKER MAP:

The purpose of the tracker map is to track the different enemies and their locations while the user is playing the game. This will get their location in terms of their coordinates and print them visually in the bottom left hand corner of the screen. If they are out of site of the map they will still be displayed but instead of them being a red dot this will transform into a red arrow.

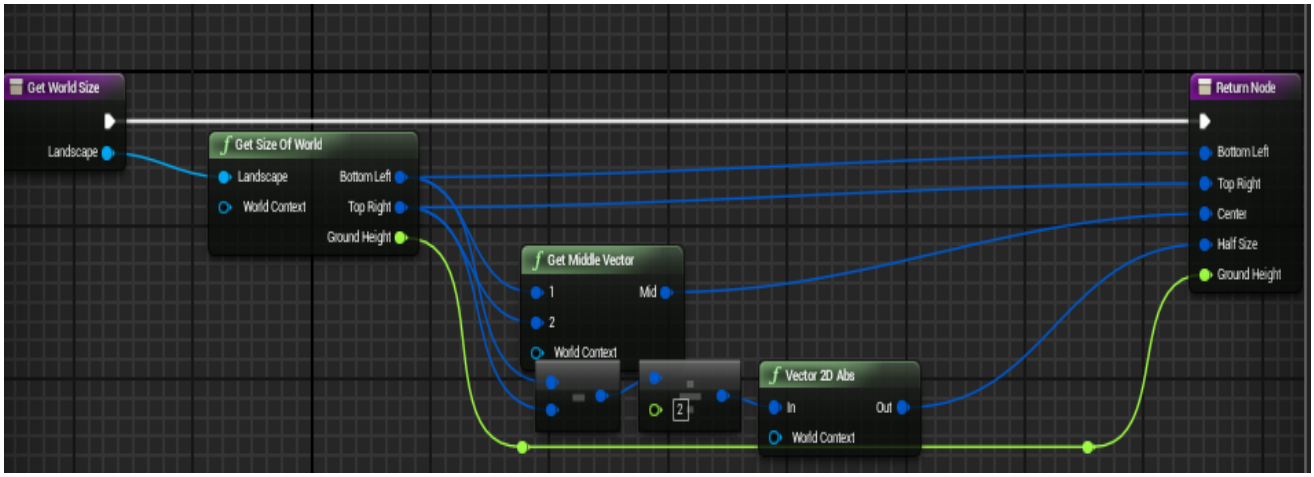
The first step of this is to create the widget which will visually display the enemies out to the user. So for this I added a border which went around the map and then a background which the enemies will display on.



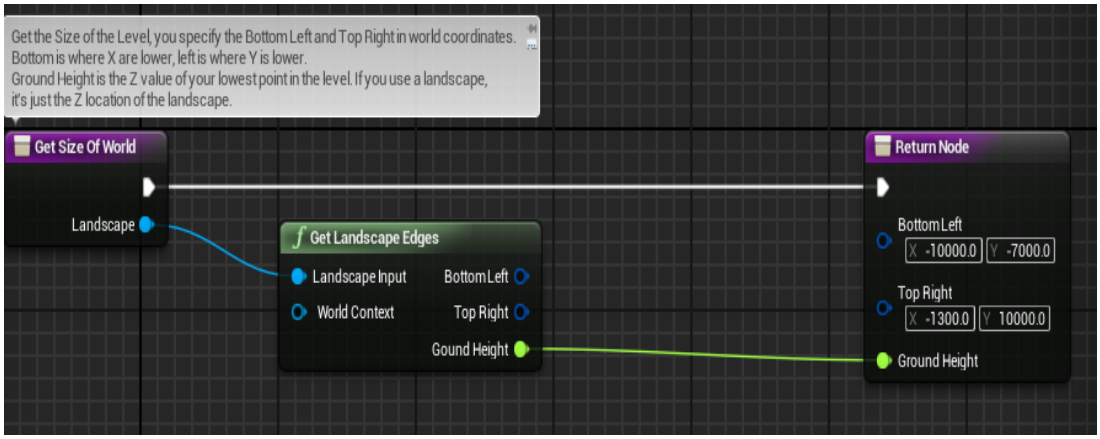
In order to allow the user to zoom in and out of the map to get more of an overview of the area and enemies which are out of sight I first call On Mouse Wheel event and send this to Adjust Zoom function. Off this I get the delta from the GetWheelDelta and the return value is then divided by 20 and then this is the Delta for the Zoom. Then the Return Node is the Unhandled event.



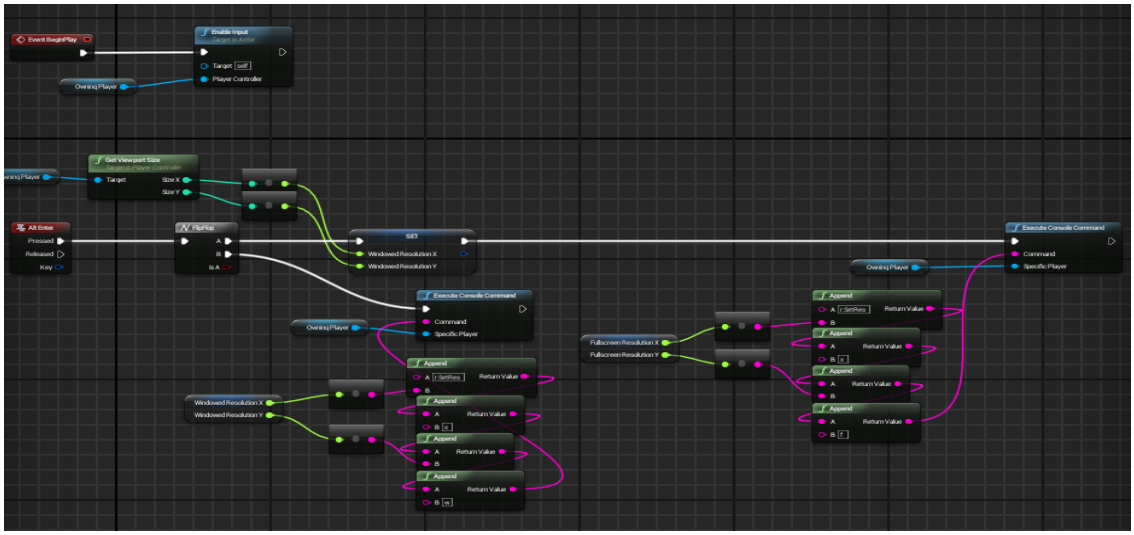
In order to get the location of the Enemies I first need to retrieve the size of the game world. For the landscape I call the Get Size of World Function and the Bottom Left runs to the Bottom Left of the return node, and the same procedure takes place for the Top Right. The Bottom Left and Top Right also needs to get the Middle Vector and then this takes the Middle from this and runs to the centre of the Return node. Now for the height this is a Float and runs to the Ground Height of the Return Node.



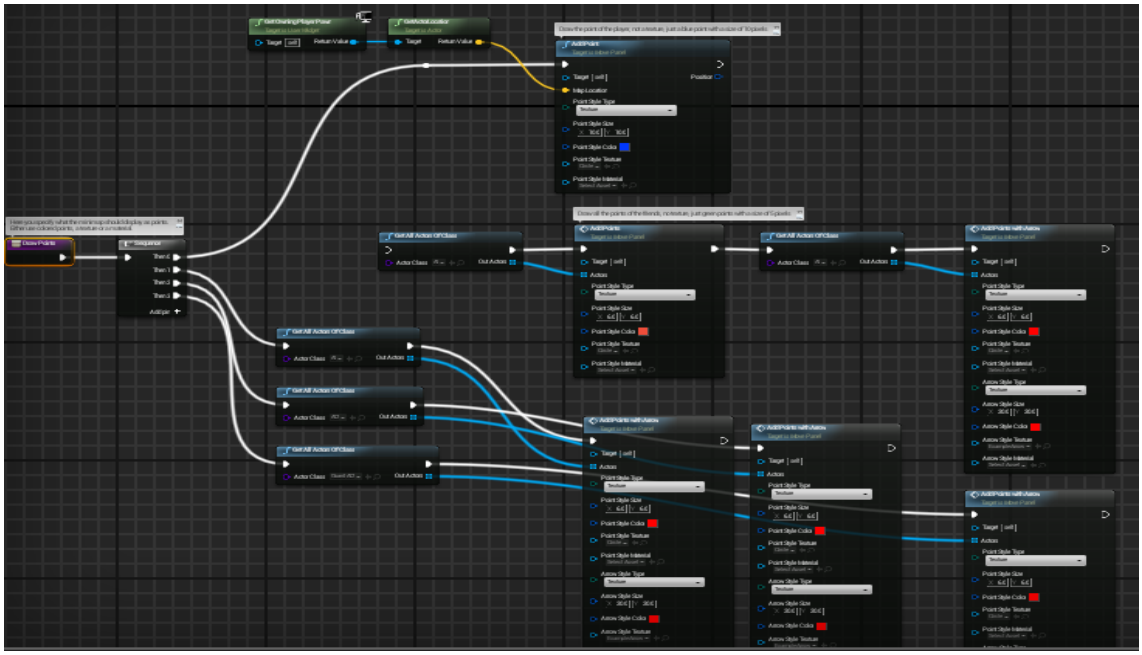
Get the Size of the Level, you specify the Bottom Left and Top Right in world coordinates. Bottom is where X are lower, left is where Y is lower. Ground Height is the Z value of your lowest point in the level. If you use a landscape, it's just the Z location of the landscape.



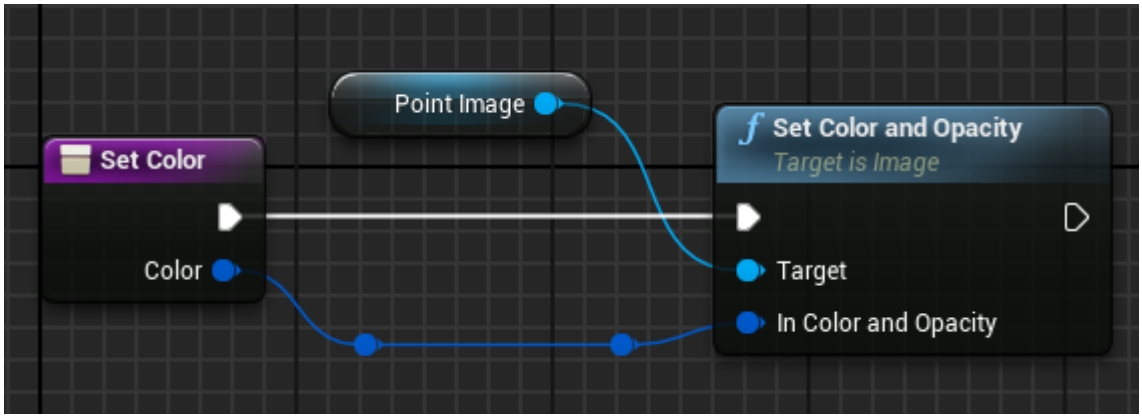
Now when the user clicks Alt+Enter this will return the map to the centre of the users location, in order to add this feature I first enable input for the player when the game first begins, this is achieved by calling Event Begin Play and bringing this to Enable Input then the Player Controller is Owning Player which is the player. Now when the player clicks Alt+Enter I bring this to a Flip Flop. Off "A" I change the Windowed Resolution X from the viewport and covert this to a float, I repeat this process for Windowed Resolution Y. Now off "B", I Execute Console Command which is the return value from Windowed Resolution X and Windowed Resolution Y. Also the Specific Player is the Owning Player which is the player itself. Now the result from the Windowed Resolution X and Windowed Resolution Y and this goes to the Execute Command, and the command off this is the Fullscreen Resolution X and Fullscreen Resolution X.



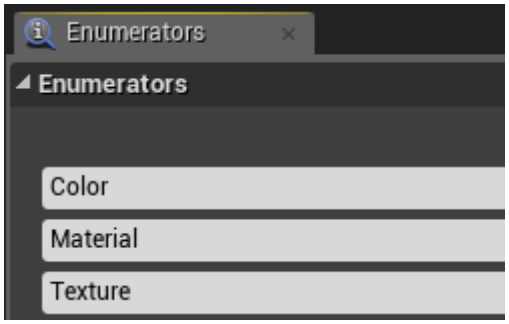
Now to add the enemy pointers to show on the map I first create a Draw Points function. Now on the first sequence I add the point to the map which is the player, for the location I retrieve the users location and then the owning players pawn which is the player. Now for sequence 2, 3, and 4 I the retrieve the class which represents each of the enemies. Now each of these goes to add points with arrow function, so then each enemy which is on the map will become a red dot when close and transform to a red arrow when out of sight. This needs the array of actors from each class, and then the type of arrow I want them represented by.



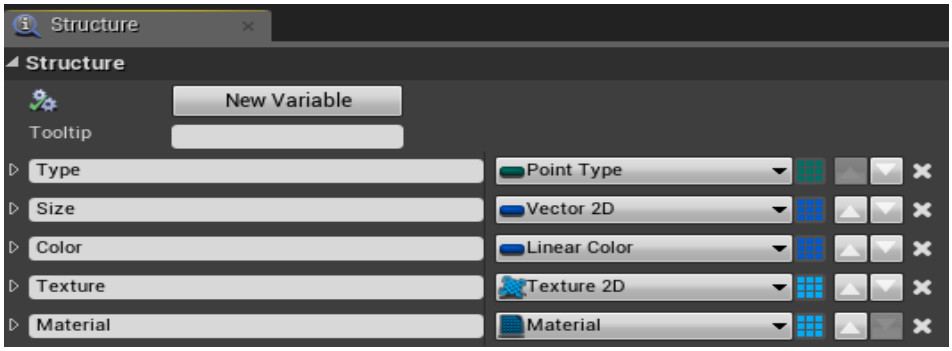
To set the arrow colour I call the Point Image in the arrow widget class and the call the function Set Colour and Opacity, then the target I want to set the image as is the Point Image and the Colour then changes to the In Colour and Opacity from the function.



The Enumerators needed in this whole process is for the colour of the arrows, materials and texture. You can see these being used throughout the process in the previous steps.



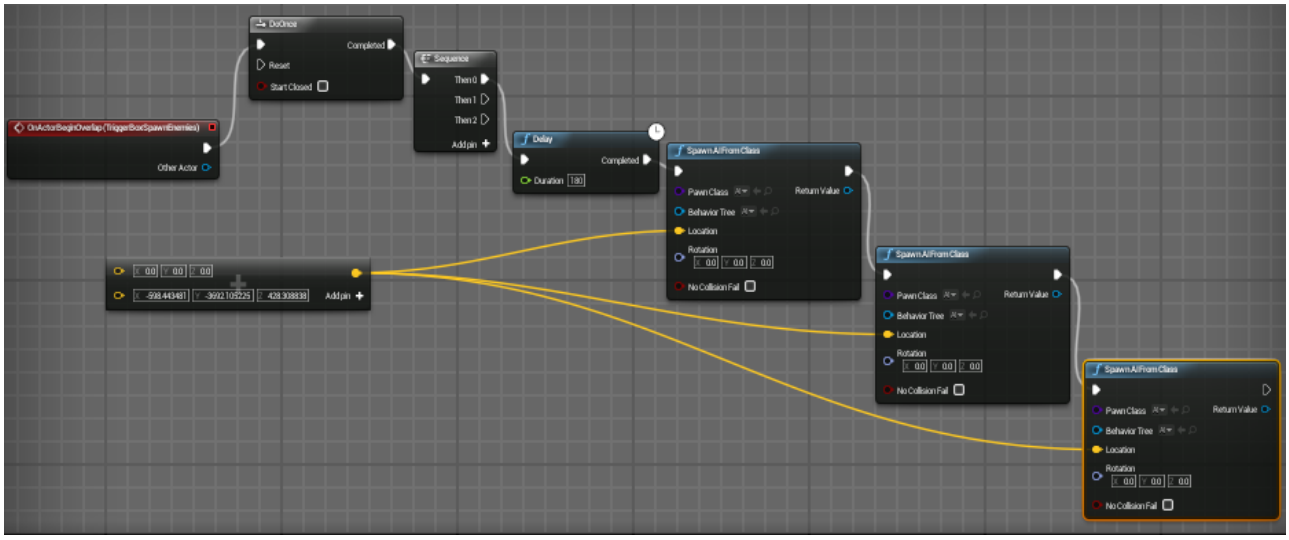
The Structure is used to create the different variables and object I needed to make this functionality work perfectly. So for type I needed to create a Point Type, Size is Vector 2D, Colour is Linear Colour, Texture is Texture 2D and Material is simply a Material type.



9.13 SUMMONE ENIMIES EVERY 3 MINUTES:

This adds the functionality spawning 3 zombies into the environment every 3 minutes to add difficulty to the game so the player finds it harder to escape the longer they take.

This is achieved when the actor leaves the first room they will walk through a trigger box which triggers the 3 minutes countdown every time. So this has a delay of 180 seconds which is 3 minutes. The when the time reaches 3 minutes. I spawn 3 actors from the class and that class is the AI enemy, and this has the behaviour tree of the AI as well. Then the location is the X, Y, X coordinates of different locations throughout the map.

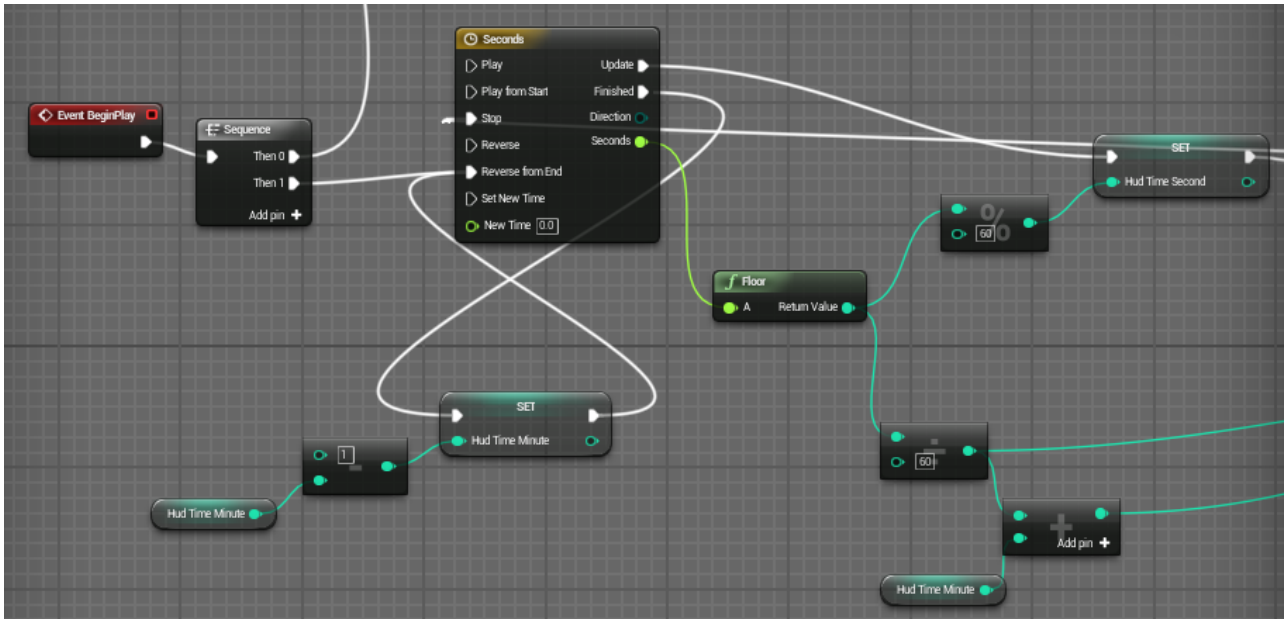


9.14 TIMER:

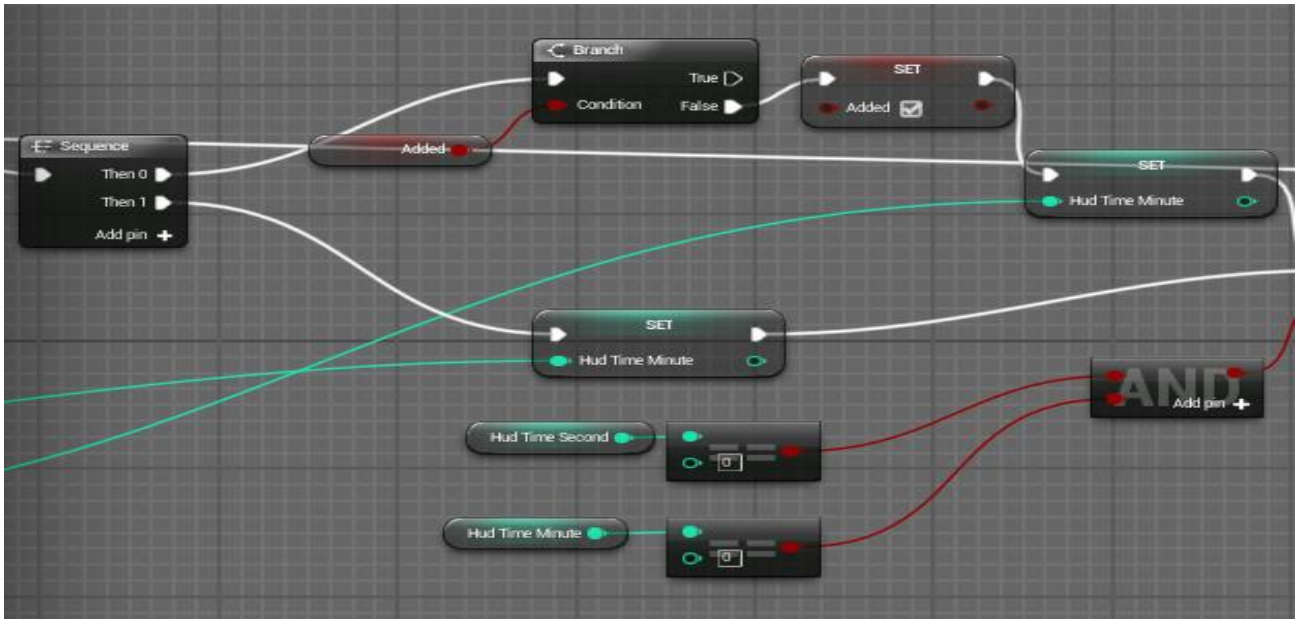
The timer in my game is used to trigger events as the game goes on. So after 3,6,9 minutes this will summon enemies, and when the timer reaches 0 the zombies will over run the game and the player will die. If the player completes the level this will send the time to a variable to calculate to calculate the players score based on the time left and how many enemies they have killed.

The code for the timer gets placed in the HUD for the game as it uses a timer on every level. The first thing I have to do is call Event Begin Play, so this triggers the timer the second the game begins. This now runs off a seconds as this calls the Health Widget too. In the HUD I had to create numerous Integer variables to calculate the time, so these are for Minute, seconds and then Added, which is a Boolean. This now first gets called to a timeline of seconds, as this is going to be counted down from 10 minutes, I used Reverse from End for this. Now off this I set the minute in which the game is at while playing and reduce the minute by 1, when that minute is up and then send this result into Reverse from End after the minute has been calculated.

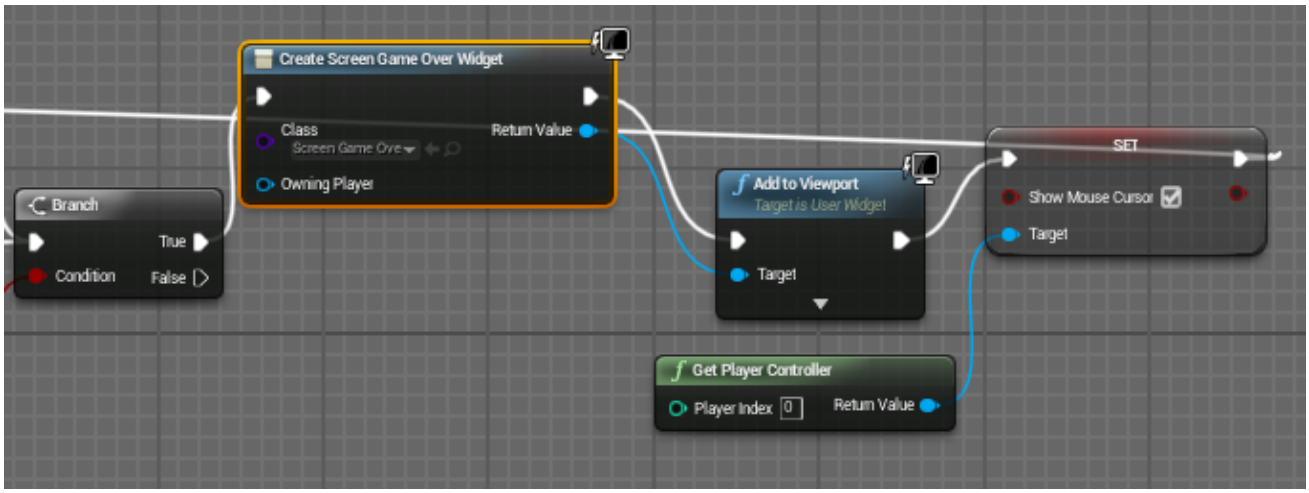
I calculate how many seconds is in a minute by calling the Floor function and return the value by dividing the returned value by 60 and sending and setting the result by as the second. Also off the Floor function the returned value gets divided by 60 and that result gets added to the minute which is now the result gets set as the new minute.



Now I bring this to a sequence to participate in two algorithms at once. Off 0 I bring this to a branch with a condition of whether this has been added to the timeline and if not I set the added Boolean as true and then set the new HUD time minute. Off 1 from the timeline I set the HUD time minute so when the HUD time second is zero and the HUD time minute is zero the player dies.



To cancel the game, I create a branch which has a condition of whether the whole time is zero. So then if this is true I bring up the death widget to let the player know that they have run out of time, by adding this to the viewport and set the mouse cursor for the player to decide whether they want to play again or not.



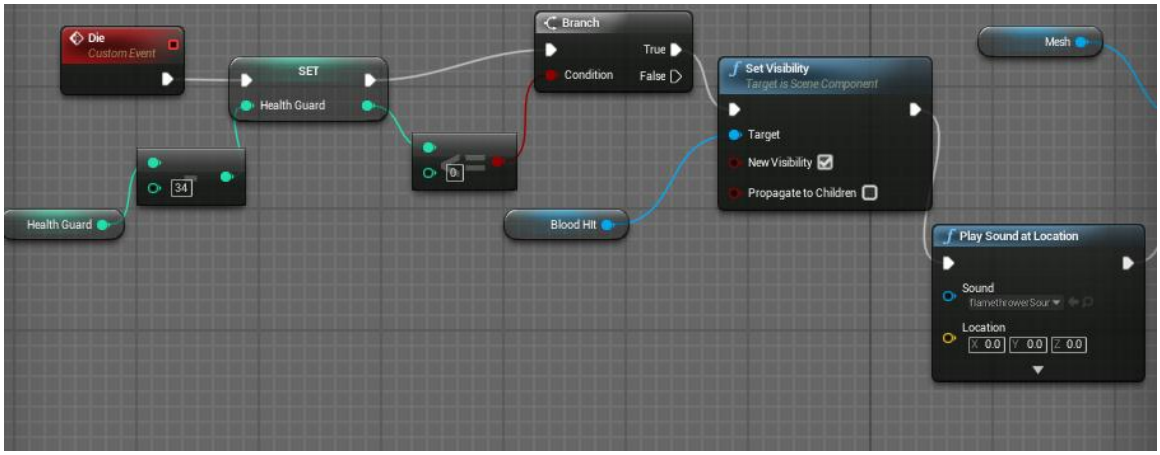
From the set mouse cursor this come back to the timeline to onStop, to stop the timeline altogether.

9.15 ENEMY KILLED COUNTER:

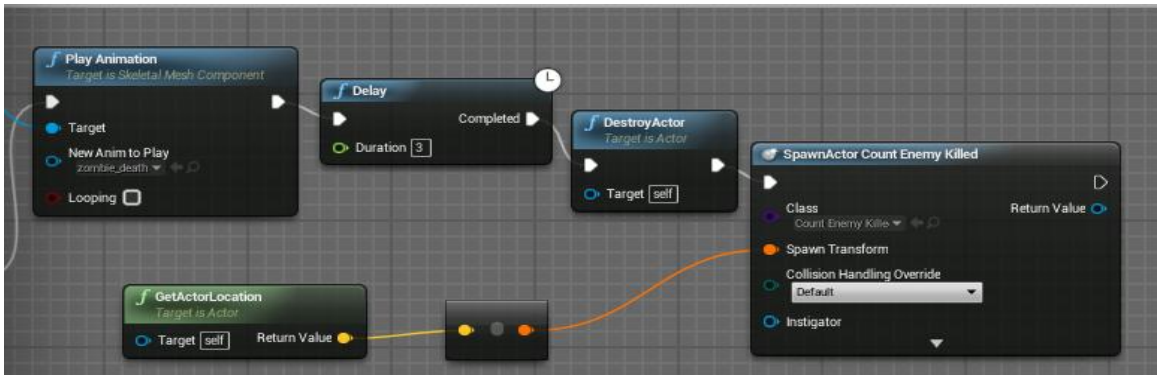
This creates an object which spawns when the enemy is killed and when the object is collected it adds one to the total score for the player. This will be displayed to the player on the top of the screen.

Enemy Killed Spawn Counter Object:

When the enemy dies this will spawn the object which the user can collect to add to their total score. First I created a custom event, I set the health of the enemy and pass in their current health at that moment in the game. Then I check if the health is less than or equal to zero, and if this is true I set the visibility of them becoming on fire and play the flamethrower sound as the player has successfully killed the enemy.

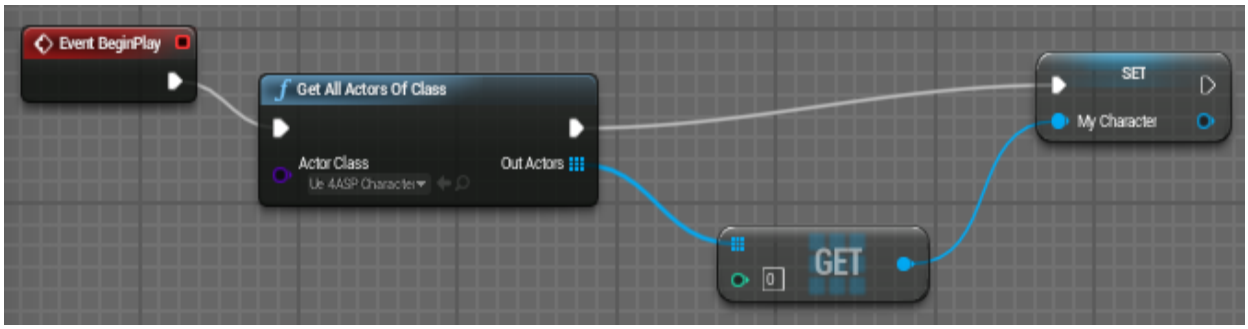


Now I play the death animation for the enemy as they are dead, and then after three seconds destroy the enemy and spawn the counter next to where the players location is on the map.

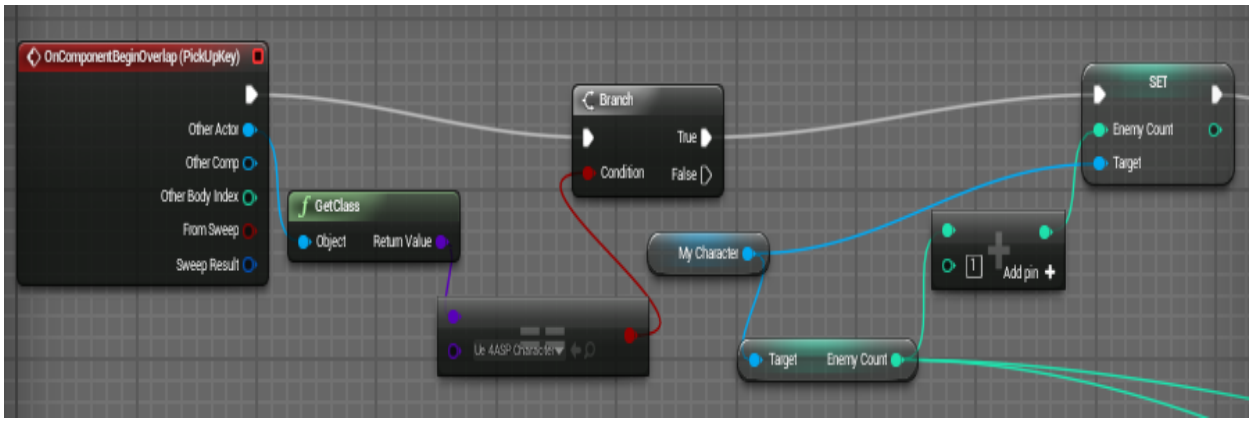


Counter:

When the level starts I get the players class and pass this into an array and the get the result and then set the result of the array and pass this into the character class

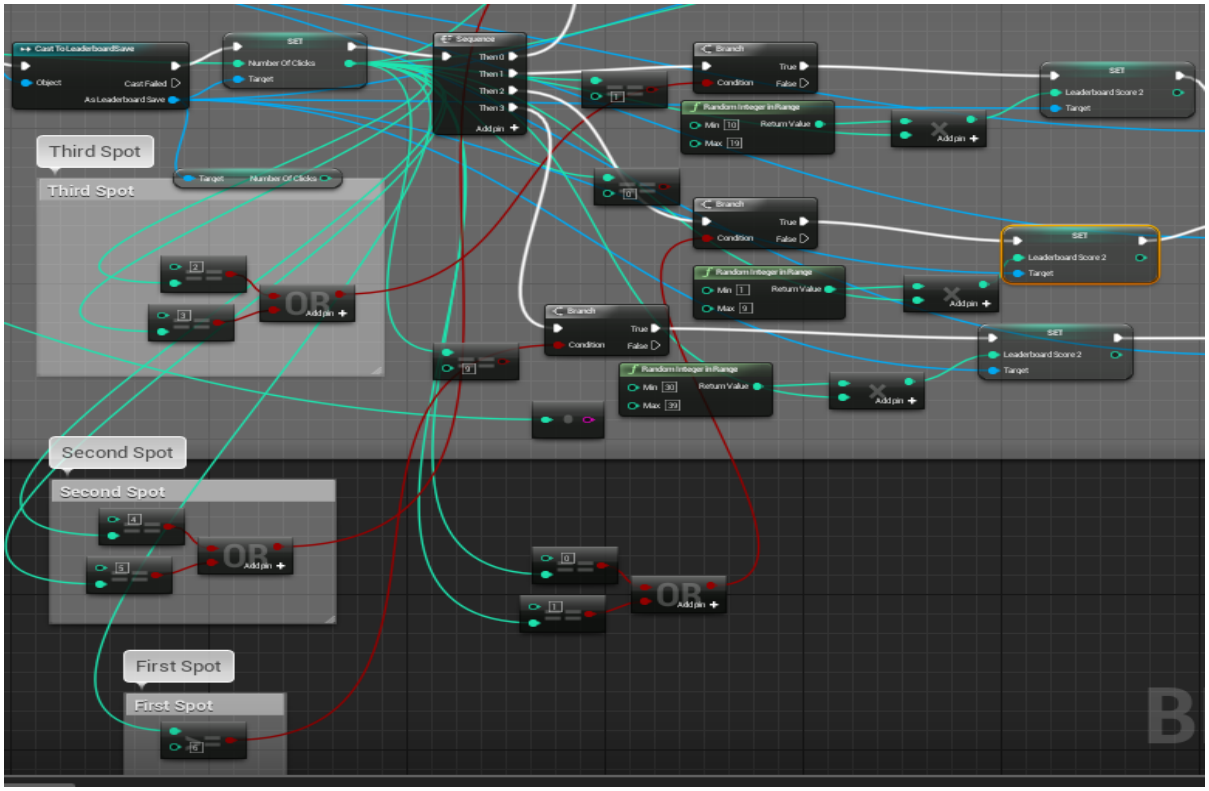


When the character walks into the object we want the object to disappear and add one to the enemy count, so for this I first have to call OnComponentBeginOverlap and reference it to the PickUpKey. The actor for this I need to get the class of the character and add that as a condition to check if the object is being collected by the player. If this is true then I set the enemy score and add one to the total.



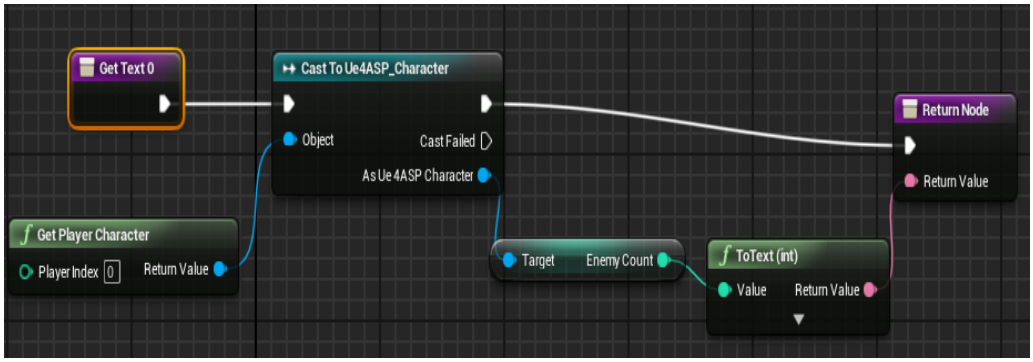
So after this adds one to the total, this will destroy the object so then I call the save game class and set the enemy count from the saved file to the same amount of the total enemies being killed. Off the enemy's variable which is being saved in a file I need to run sequence to check how many enemies have been killed, so if 0 or 1 enemies have been killed I update the saved file for this and set the new value and name. The off sequence 1, I check if 2 or 3 enemies has been killed and if this is true then we update the save game

for this, and so on. However, if 6 or more enemies has been killed this will slot into first place if the time is good and the score is better than the previous opponents.



Display Enemy Count:

In order to display the enemy count on the screen, I get the text which has been set in the widget and cast this to the player which holds the count. The object of this is the player character. Off the character I call the EnemyCount and pass this as the return node.

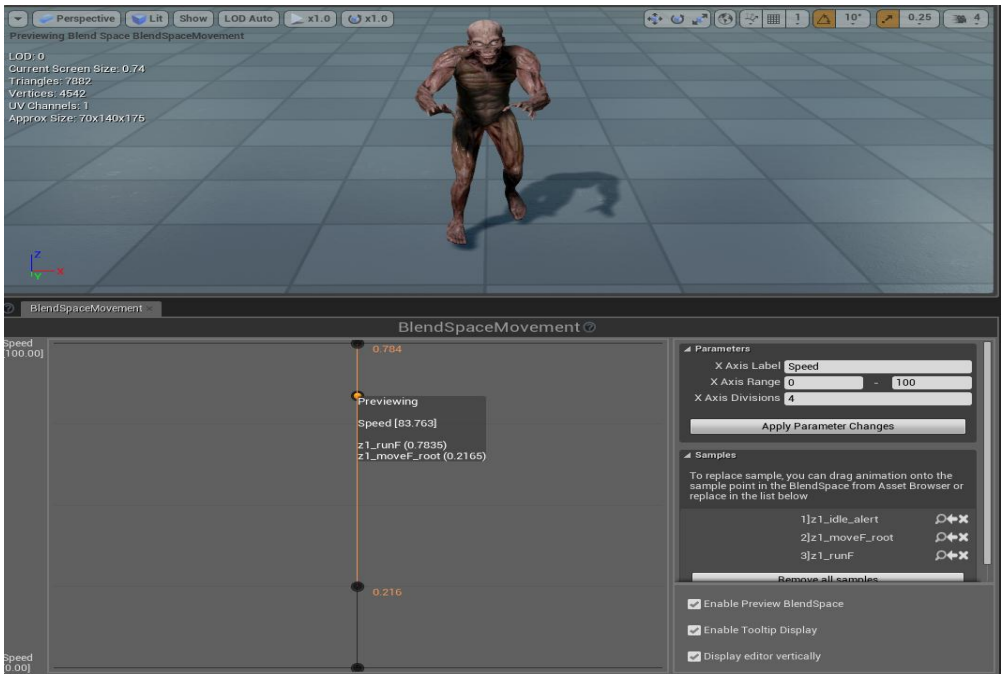


9.16 ANIMATIONS

For my game I had to incorporate 5 different sets of sets of animations, one for each character. These determine how each character moves, attacks and wonders around the map. Each set of animations involves the same process but use different animations to make each character unique.

Blendspace:

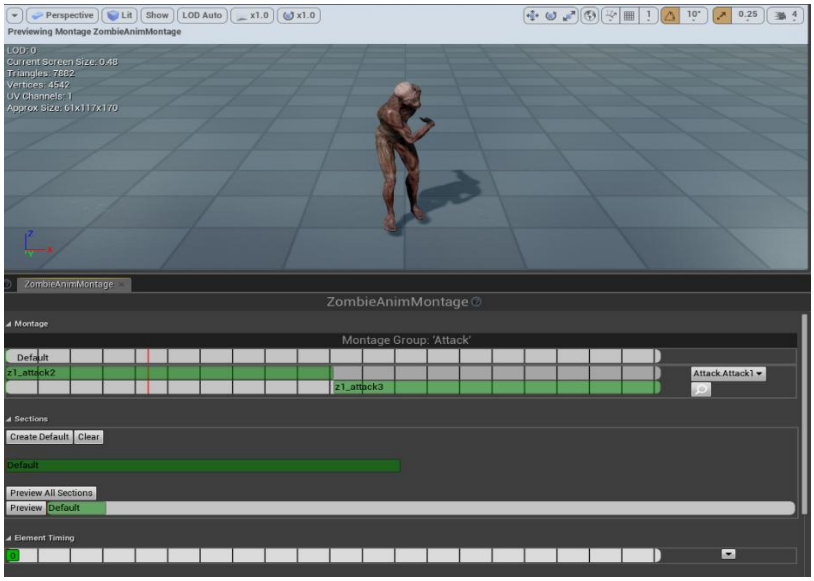
First I created a blendspace, this is used to blend different animations together. So when the character is idle this will trigger the idle animation, then when they are walking this will trigger the walking animation and finally when they are running this will trigger the running animation. As you see in the image below the time triggers different animations based on the speed the character is moving at, if they are static this means idle animation, if they are barely moving this will trigger the walking animation and then when they are at full speed this means the running animation.



Animation Montage:

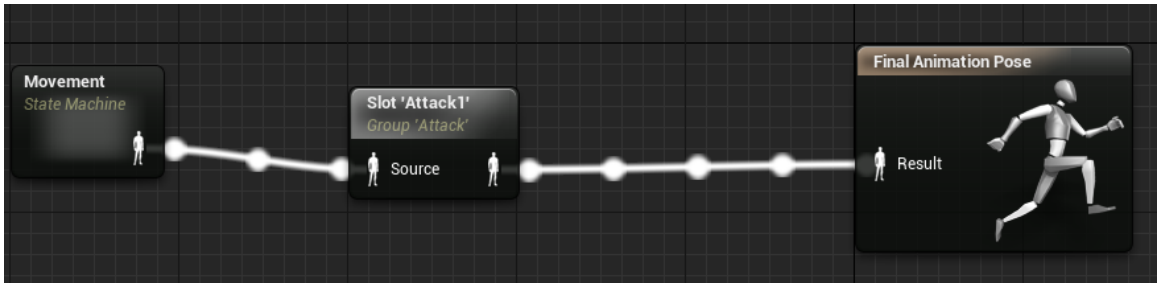
This creates the attack animations for each character. This can contain as many attack animations as the developer would like. As you can see on the montage timeline I have used 2 different attack animations. When you have selected which animations you would

like to use and happy that they are running smoothly you now need to give the Montage a group name and slot name, this is crucial for calling the specific montage in the Animation Blueprint.

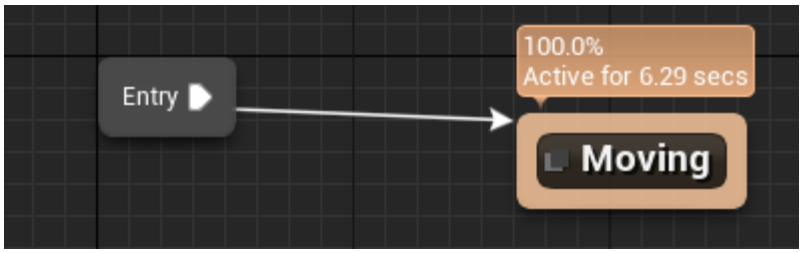


Animation Blueprint:

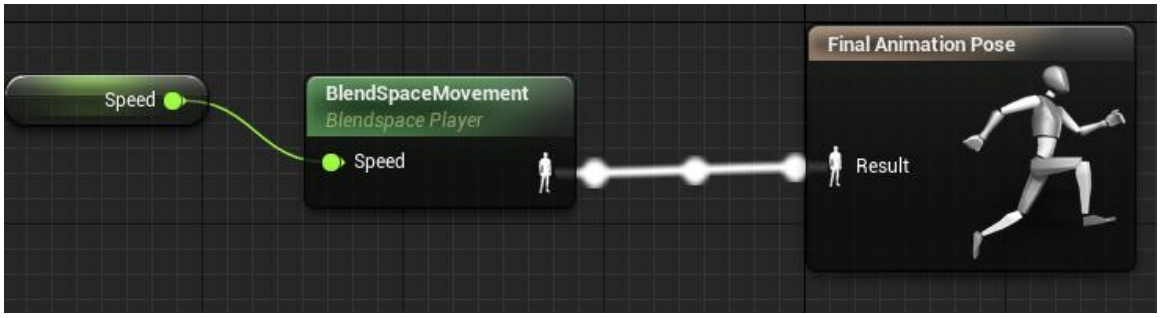
Now this process contains how the animations will run. The first step involved is creating a finite state machine to determine which state the animation is in. This runs to the slot that we have created in the Animation Montage to trigger this event, then I finish the animation by running it to the final animation pose.



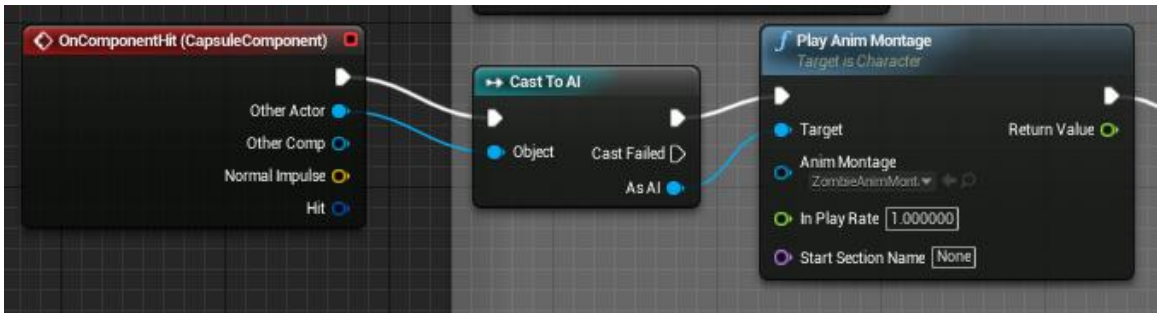
It is always crucial to determine the speed that your character is moving at, so for this I go inside the state machine that I have called Moving and set the entry to the speed that the character is moving at.



I create a variable called speed which is referenced to the state machine to get the speed then set this to the blendspace and send it to the final animation pose to finish the process in the Animation Blueprint. Then this will move the character at walking pace until they see the main character then chase you and run after the main player until you are out of sight.

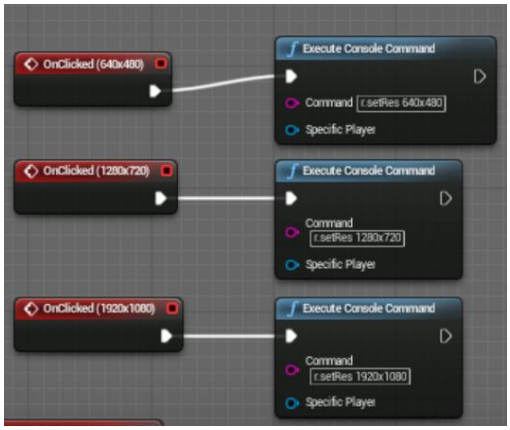


To trigger the Animation Montage when the zombie is attacking the player I first go into the code where I have set the Enemy AI attacking the main player of the game and say when they are decreasing my health I cast to the AI and play the Animation Montage I have created earlier in the steps.



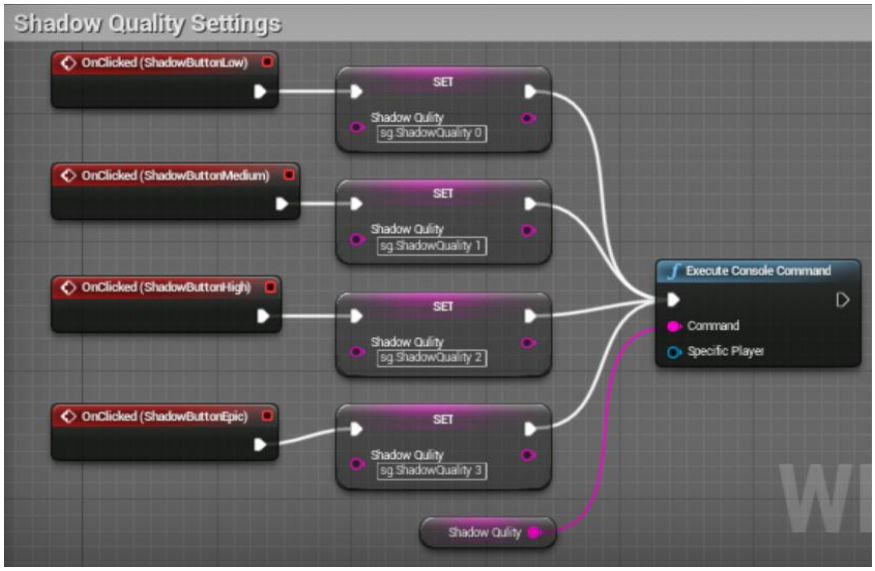
9.17 RESOLUTION

The below piece of code shows the functionality that executes the resolution change from 640x480, 1280x720 and 1920x1080. For this I need to run the blueprint Execute Console Command and run the command r.setRes (type in the resolution you want to change to e.g(640x480)).



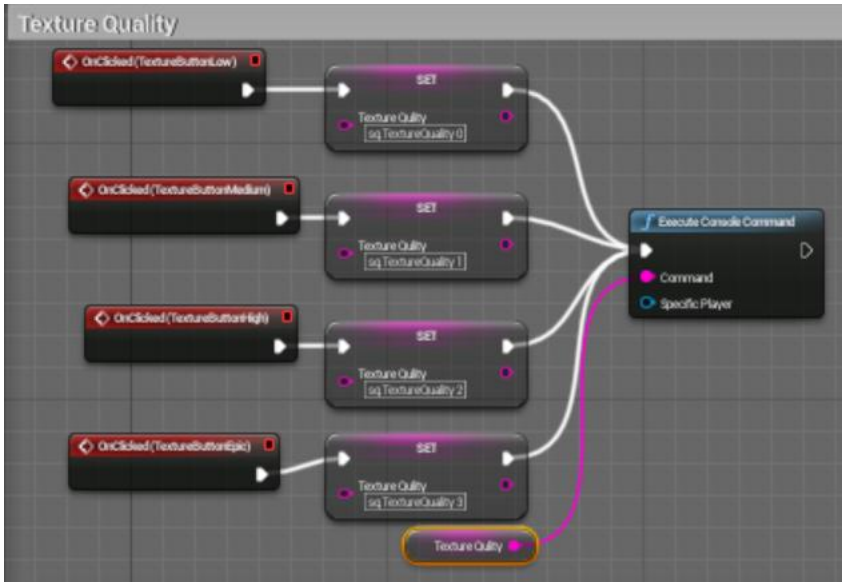
9.18 SHADOW

In order to change the shadow quality I first had to create a string and set the shadow quality of this based on the user input. The command for setting this at epic settings is "s.g.ShadowQuality 3". After this the result of the user input gets passed into the Execute Console Command.



9.19 TEXTURE

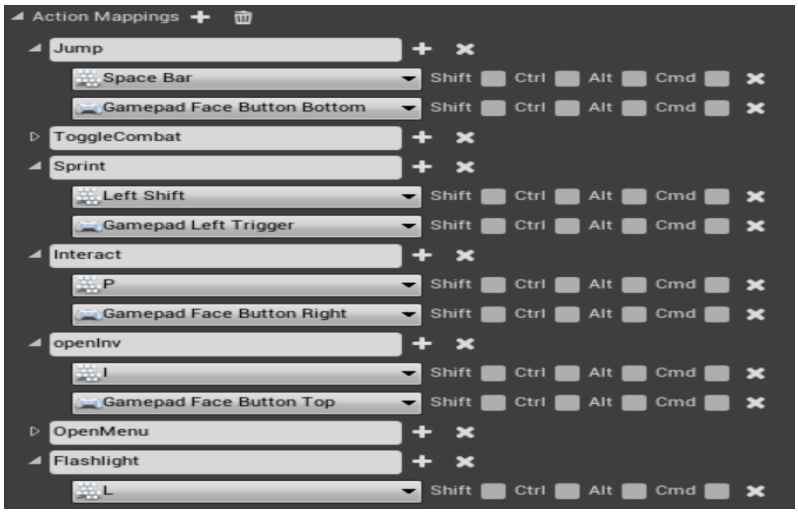
The texture is done in a similar manner as shadow quality. I first had to create a string and set the texture quality of this based on the user input. The command for setting this at epic settings is “s.g.TextureQuality 3”. After this the result of the user input gets passed into the Execute Console Command.



9.20 CONTROL INPUT

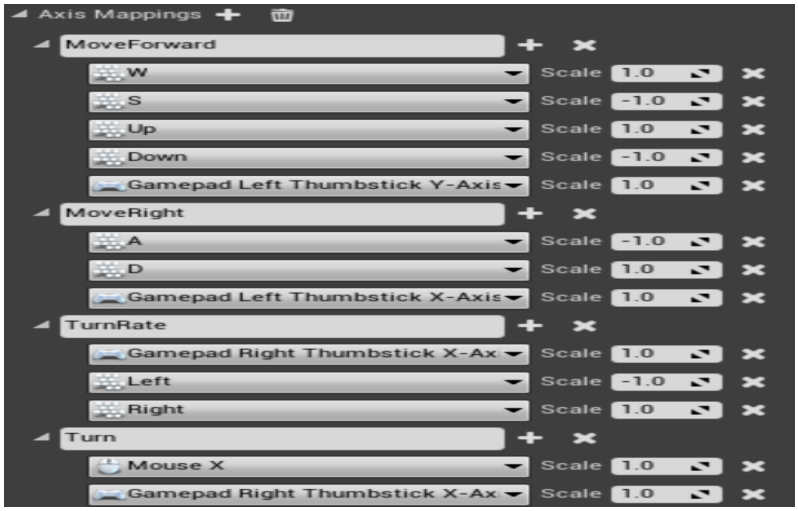
Action Mappings:

The action mappings provide the main input for the controls of the game, this can include Jump, Combat Mode, Interact with objects, Open Inventory, Flashlight, etc. To incorporate these in the game you first have to set the controls in the project settings by creating a new action mapping and then under this set the control you want to use for example keyboard and mouse or gamepad controller. These can get called through your character blueprint of where you want this to be used.



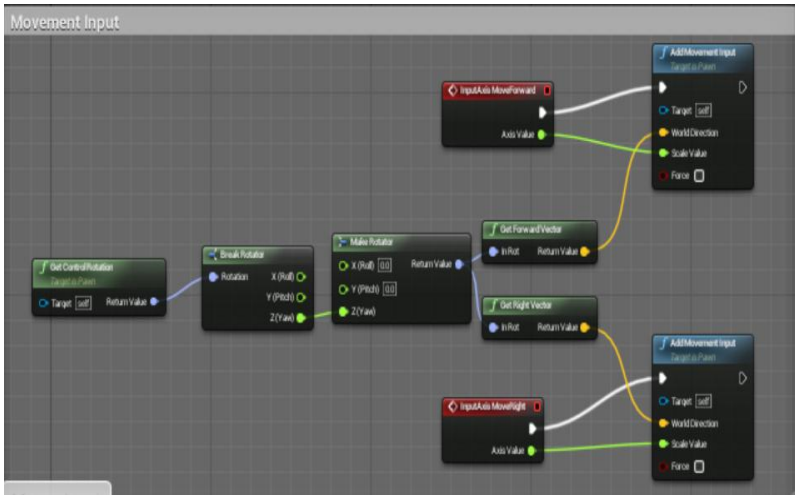
Axis Mapping:

The Axis Mappings are the controls on the character itself, this would be for moving the character forward, right, left and backwards. This would also include, the turn rate, looking around, etc. They are added in the same way as the Action Mappings.



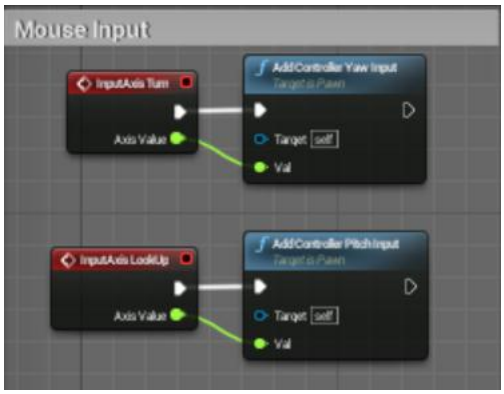
9.21 MOVEMENT INPUT

The movement input provides the functionality for the user to move the character by using either keyboard and mouse or gamepad controller. In the code below I called the InputAxis MoveForward as this is the controls I want to use for moving the character around. This connects to AddMovement Input blueprint as this provides the functionality for this, the world scale would be the axis value that is user is doing while playing the game.



9.22 MOUSE INPUT

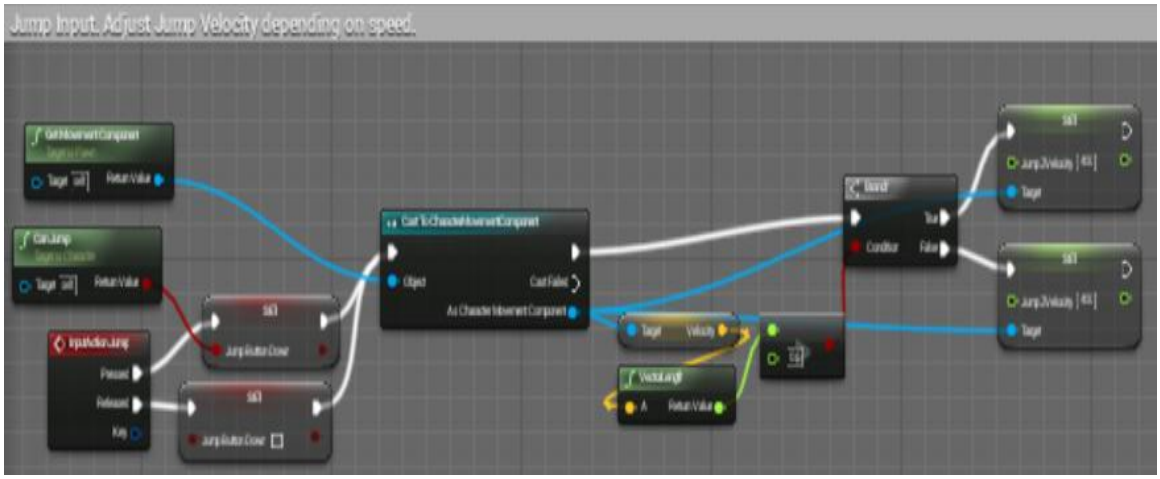
The mouse input allows the user to look up, down and all around the environment. I needed to call the InputAxis Turn for turning the player around. This went to AddController Yaw Input and the value from this is goes to the axis value from InputAxis Turn.



9.23 JUMP INPUT – DEPENDS ON VELOCITY

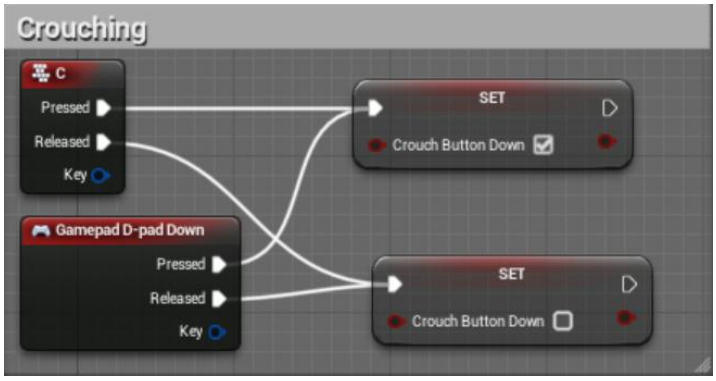
To enable the character to jump I had to find and get the velocity of the environment, the speed the character is moving and also the timing of the jump. Firstly, I call InputAction Jump from the controls I set earlier, now when this is pressed I set jump button down and the condition is if they can jump then allow the character to jump. When this is released set this to false and make the character come down from the jump. Off the jump button down variable I cast this to the main character with the object being the get movement component. The blueprint component from the player gets the velocity in a vector length

and if this return value is greater than 0 this will return true and the jump velocity will be set to 400, if this is false the jump velocity will be 400.



9.24 CROUCHING

In order to allow the player to crouch and sneak past enemies when you press ‘C’ I set crouch to true as the player would be crouched, and upon release set crouch to false and they will stand back up.



9.25 TUTORIAL LEVEL

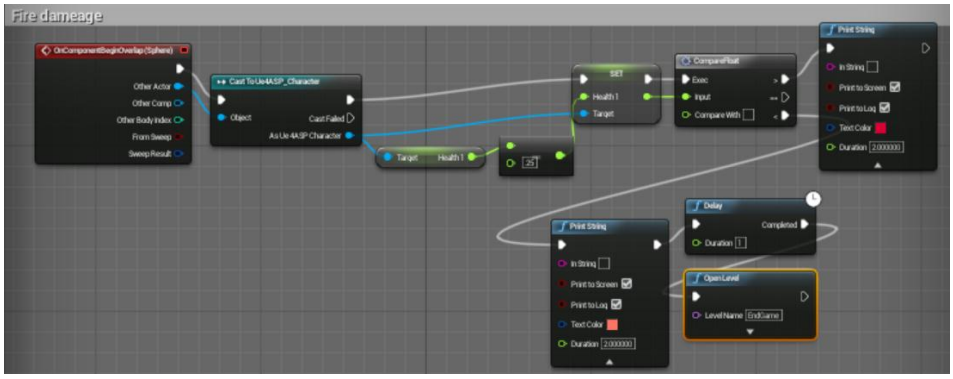
The tutorial level provides the player with the bases to play the game. This will give them the chance to get to know the controls and also give them knowledge of the background story of what is happening and why. The level will be in day time as it is an introduction to the environment and what the story is about. The goal of the level will be made clear at the start so the user knows what to do in order to progress.

This level inherits some of the functionality which is in the main level. This was all explained in the previous headings. It uses:

- Key pick up to open locked doors
- Gun fire
- Inventory system
- Timer
- Health for player
- Damage to enemies
- Enemy kill counter
- Read Letter

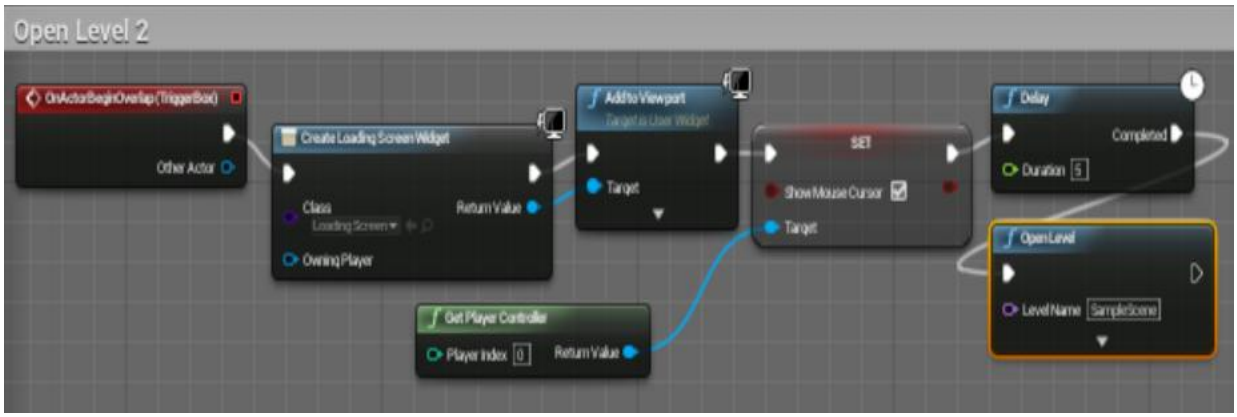
For this I made a new cinematic which is an introduction to the map. I found this gives the user a better overall experience.

When the user finds the key and opens the correct door, this will be full of fire as the world is only beginning to turn. To enable the fire to do damage to the player I had to create an OnActorBeginOverlap to the sphere and cast this to the main character of the game, so this is triggered only when the player walks over it. From the character I call the Health variable and decrease 25% of their health and then set it back to the Health so this is decreased on the widget which gets displayed back out to the user. Now I run a CompareFloat to check if the players health is less than or greater than zero. If this is greater than zero nothing happens, whereas if this is less than zero the player is dead and the try again screen is displayed.



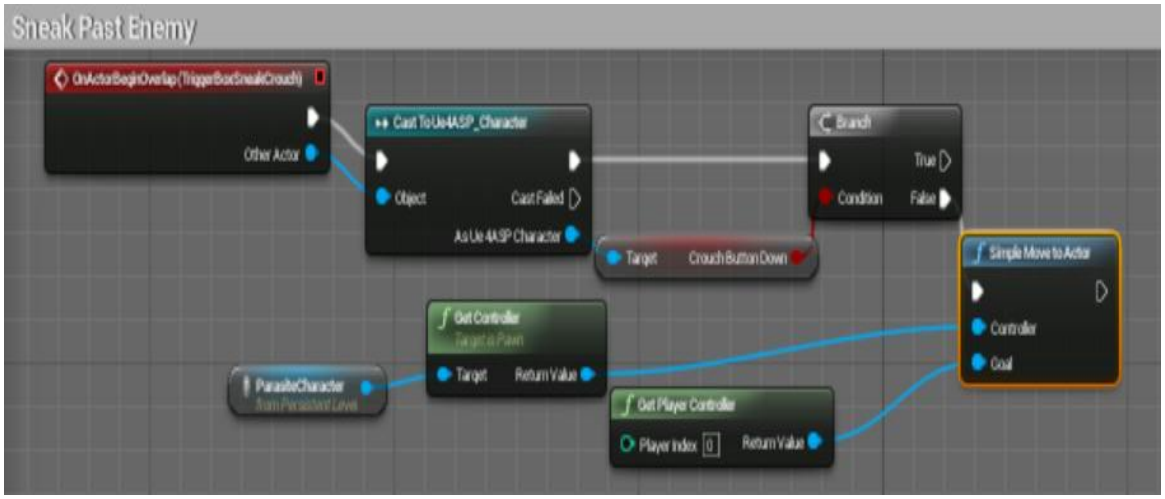
In order to show the text gives the user hints, controls, goals and features that pops up throughout the level the level I first had to create a trigger box which triggers them in different spots throughout the level. This is casted to the main player so they only show when the main player makes contact. I then call the DoOnce function so this only happens the first time they make contact, this is to prevent the display showing every time they overlap the trigger box. Now I begin to show the widget, and add the selected widget to the viewport, I also show the mouse cursor so the user can exit from this view when they are finished.

Once the user completes the goal I now need to trigger the main level of my game. This is done through the functionality of the trigger box. Once this is overlapped by the main character this will begin to load the main level. It will create the loading screen widget, add this to the viewport, then begin to open the main level.

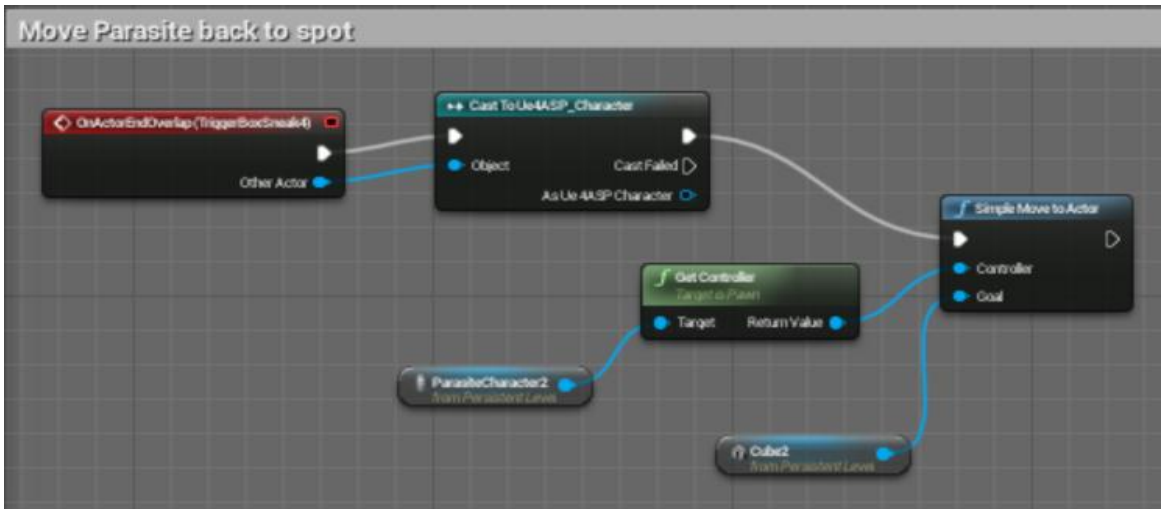


9.26 PARASITE AI

The purpose of the Parasite enemy is that he is blind, and only attacks when he hears your footsteps. This allows the user to sneak by him when you are crouched. If you walk into the trigger box and you are not crouched then the enemy will come to you and attack. This is because the enemy can hear your footsteps.

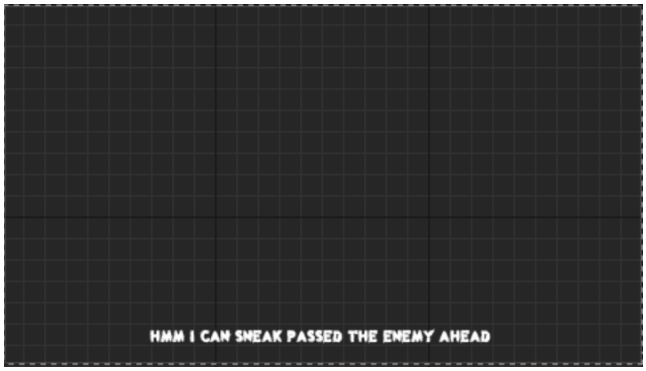


When the user leaves the hallway this will end the triggerbox zone which will tell the parasite enemy to return to their position as you are not a threat anymore. This is because the footsteps from the player is at a distance so the enemy will return to their guarding position.

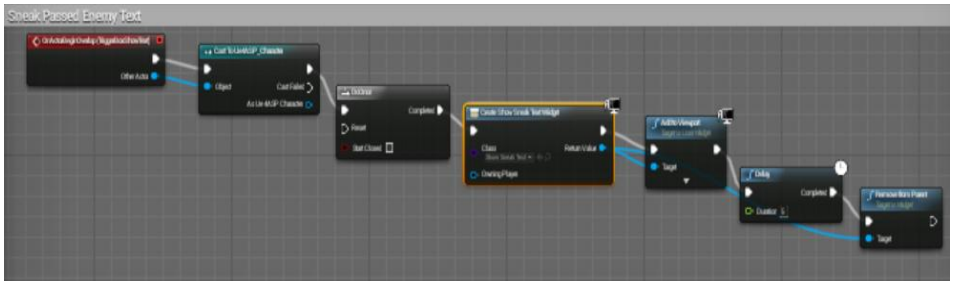


9.27 SHOW HINTS THROUGHOUT THE LEVEL

I created numerous widgets to enhance the users knowledge of what their goal is and what type of enemies are ahead. One of the widgets is shown in the image below.



The code below is triggered when the user enters into a specific zone of the map. When the user enters this area it will display the specific widget, add this to the viewport for 5 seconds and then remove it.

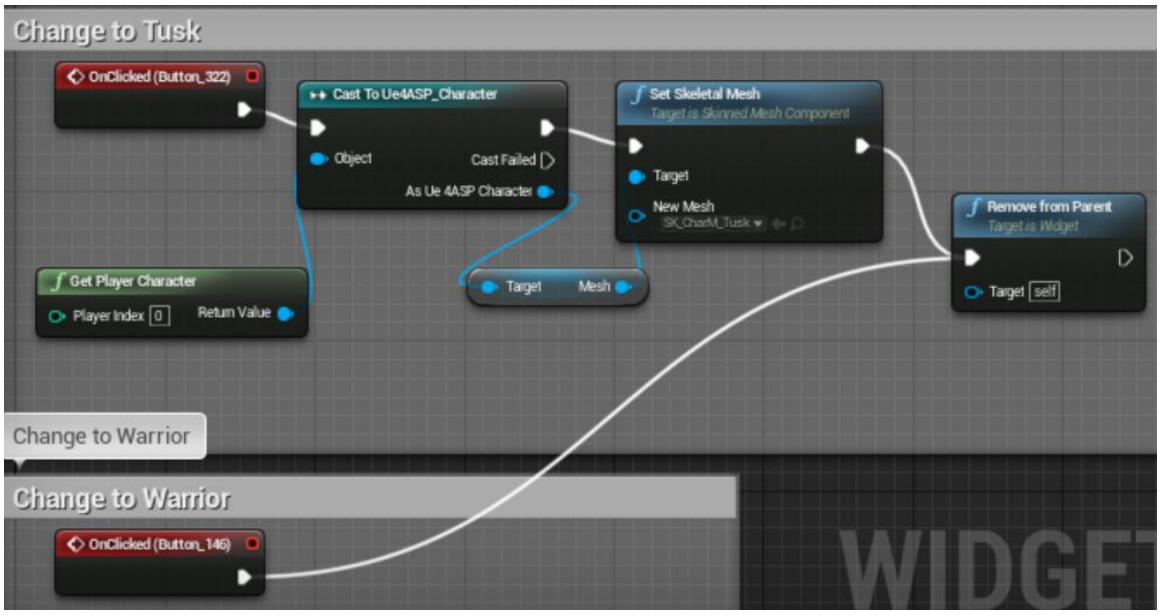


9.27 CHARACTER SELECT

This allows the user to select from different characters when they begin the main game mode. The first step was to create a widget which gets displayed when the main level begins.

When the user selects to play as the Tusk it called the OnClick Event for the button. This casts this to the main character, then sets the skeletal mesh, the target would be the mesh that is set on the character by default and the new mesh would be the mesh for the Tusk.

Now if the user select to play as the warrior this will simply remove the widget from the viewport and play the game as this is the default mesh on the character already.



10 TESTING

Testing in many different forms has been carried out in order to ensure that the software acts as expected in all situations as well as ensuring that the requirements have been successfully met.

For testing I gathered colleagues from around the college to test out the beta version of my game. For this I gathered 11 colleagues to gain a better overall perspective of the game. This will give me the opportunity to find out if there are any hidden bugs that I haven't come across, or even errors. I will run usability testing to determine how users interact with the UI of the game, along with unit testing for testing specific parts of the code in isolation with test code that was written specifically for this purpose, and then I finally ran assertion testing this was used to check the Boolean conditions of certain variables as they should all return true once the test is carried out.

I feel these tests would be great for the development of the game and bring it to the next level for commercialization.

USABILITY TESTING

The purpose of usability testing is to find out how different users interact with the UI of the game. The usability test was carried out with the aid of 11 participants. Usability testing is able to help any developer improve the GUI by providing eye maps which showed where the user was looking when the game encountered a static background such as a menu or jump scare, during battle etc.

UNIT TESTING

Unit testing was used in this project in order to test parts of the code in isolation with test code that was written for this sole purpose. Unit testing wasn't written for every aspect of the project. Some test driven development was employed for the most important functionality of the project, this ensured that these aspects of the code worked as intended.

ASSERTION TESTING

Assertion testing was carried out for the most important functionality in the testing process. It was primarily used to check Boolean conditions of certain variables. These should return true once the test is carried out. If they return false then this makes it clear that there is a bug with the program. Information provided by this type of testing is not user friendly, it is therefore important to make that this is primarily used to notify the developer that there is an error in a specific condition. This type of testing has many benefits and can speed up the implementation of many features during development. It has helped identify very subtle errors that may have otherwise gone unnoticed and solved them in a timely fashion.

11 GRAPHICAL USER INTERFACE (GUI) LAYOUT

11.1 MAIN MENU:

The screenshot below shows the main menu requirement where the user can start a new game, multiplayer, view the leaderboard, select options or exit the game.

The screen below shows the options menu where the user can change the resolution or change the shadow and texture quality of the game all to suit the graphics card of their machine

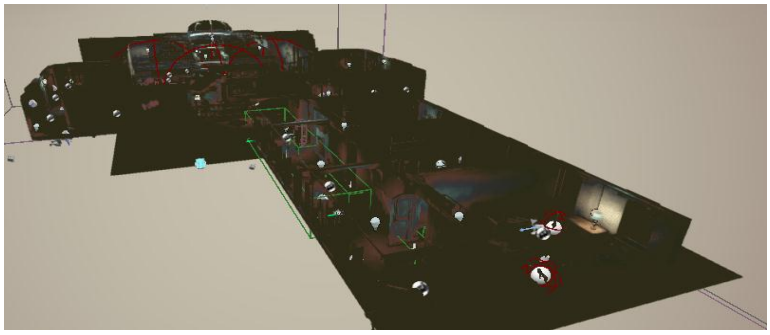
The resolution that they can change from is 640x480, 1280x720 and 1920x1080. The resolution that you will be able to run the game on depends on the graphics card in the machine and the monitor you are using.

Below you will see the shadow and texture quality settings available, the user can chose from multiple options ranging from, low – epic.

11.2ABANDONED MANOR:

This level will be the main level to my game, the aim for the user here is to kill the zombies/boss's and escape from the environment.

This level will be the main environment of the game for both single and multiplayer. The player will be first presented with a cinematic of the situation they find themselves in and then spawned in a scary immersive environment where they will have to beat the clock to escape while staying alive and killing as many zombies as possible to register a high score. They will be provided with a lot of features like flamethrower, crowbar, hatchet, inventory system, health, mini map, clues etc. which will all aid the player in their escape.



11.3MULTIPLAYER:

The multiplayer allows the user to create servers to play their friends from two separate locations, and their friend can then join their friend's server to play together. This will be a race to the finish as there is only one of everything so if you don't find the correct weapons in time you are defenceless to the zombies attacking you, but mainly there is only one key to the exit door!

Below you will see where the user can create their own server. The options available to them is whether they want this to be on LAN

Below you can see the game searching for available servers:

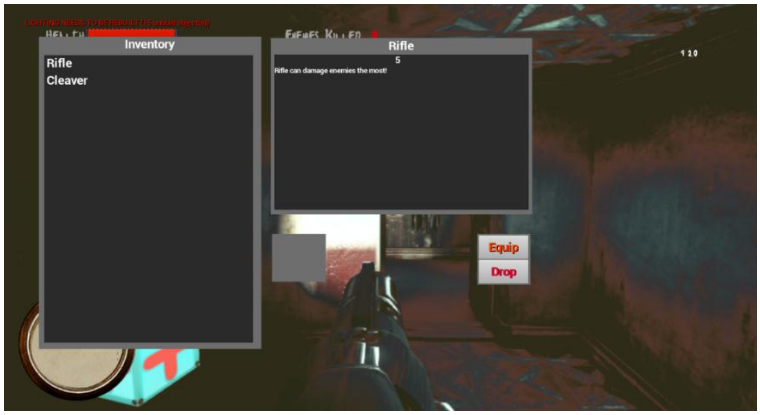
Now you can see the servers available to the player, once clicked this joins that specific game mode and begins the level

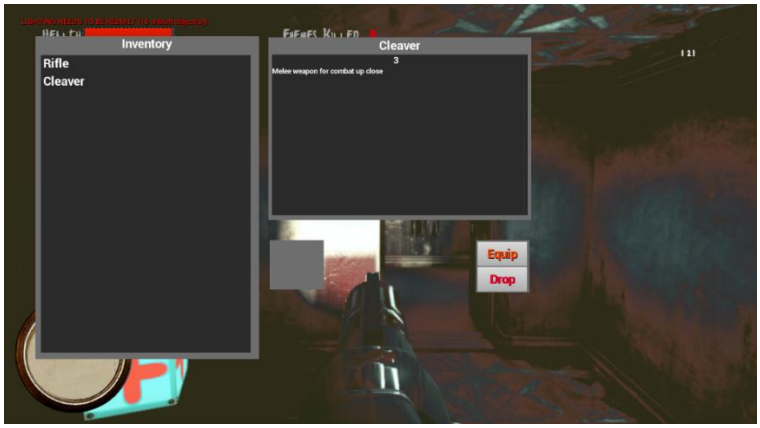
When no servers are found the result is as follows:

Now you can see the result of multiplayer with 2 main characters playing the game from different locations:

11.4INVENTORY SYSTEM:

This shows the inventory system which I incorporated, when the player picks up an item they find this will be placed in the following inventory, when they click on an item they will be shown all the details got to do with that item like weight, name, description etc. they can also equip or drop the item if they wish. The player can only hold a total weight of 25, so this means they have to be careful of choosing which items to pick up and equip in order to have the best chance of completing the game.



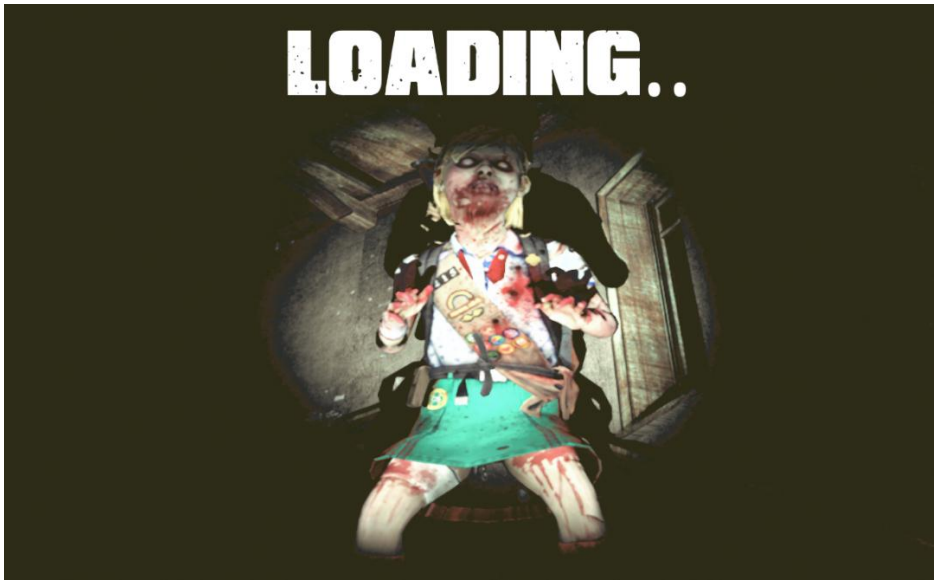


11.5DEATH SCREEN:

When the player dies they will be presented with the following screen where they can load there last save which would be the checkpoint which they walked through or they can quit the game. If they chose to load the last checkpoint they will reopen with the same amount of bullets, health and weapons that they had at that time. As they are all getting saved in the save file and getting loaded back in.

11.6LOADING SCREEN:

Each time the game loads the user will be presented with the following screen which was made using a widget.

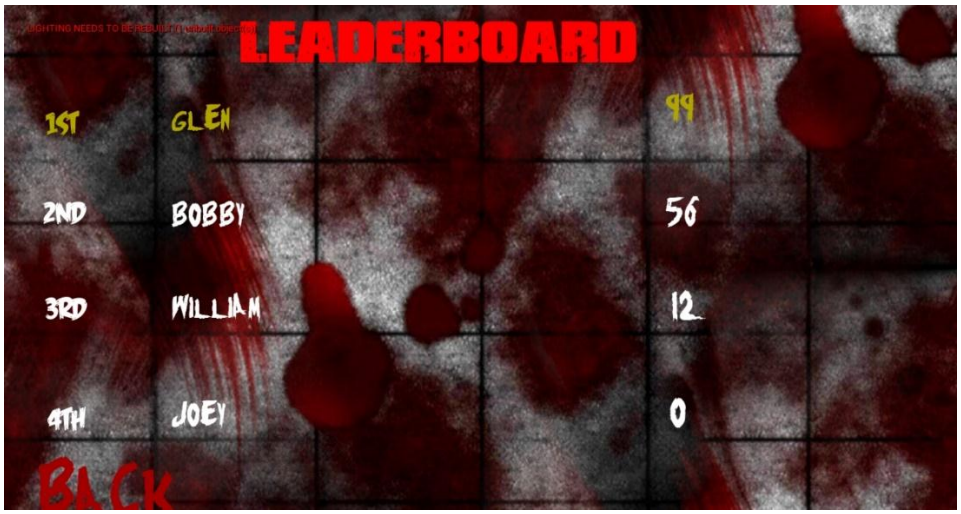


11.7SUBMIT YOUR SCORE:

Here the user can submit their score based on the enemies killed and how long they spent on the level.

11.8VIEW LEADERBOARD:

Here you can view the local Leaderboard and see everyone's performance who has played the game, the image below shows the top 4 results in the leaderboard.



11.9OPENING CINEMATIC:

Here you can see the opening cinematic for the game, this gives the user an overview of the environment which they find themselves in. They will be able to see the zombies spawning from the ground, eating and searching for you.

11.10 CLOSING CINEMATIC:

Here you see the closing cinematic for my game, as when the user finds the escape from the Abandoned Manor they will then enter the real world where the zombies are taking over.

11.11 TUTORIAL LEVEL:

The tutorial level provides the player with the bases to play the game. This will give them the chance to get used to the controls and also give them knowledge of the background story of what is happening and why. The level will be in day time as it is an introduction to the environment and what the story is about.

Throughout the level the user will trigger pop ups that will show them how to how to play the game, the image below shows the controls that the user will need to play the game.

During this level the user will be presented with a widget of what the goal is in the level. The goal is to find the key to the locked door.

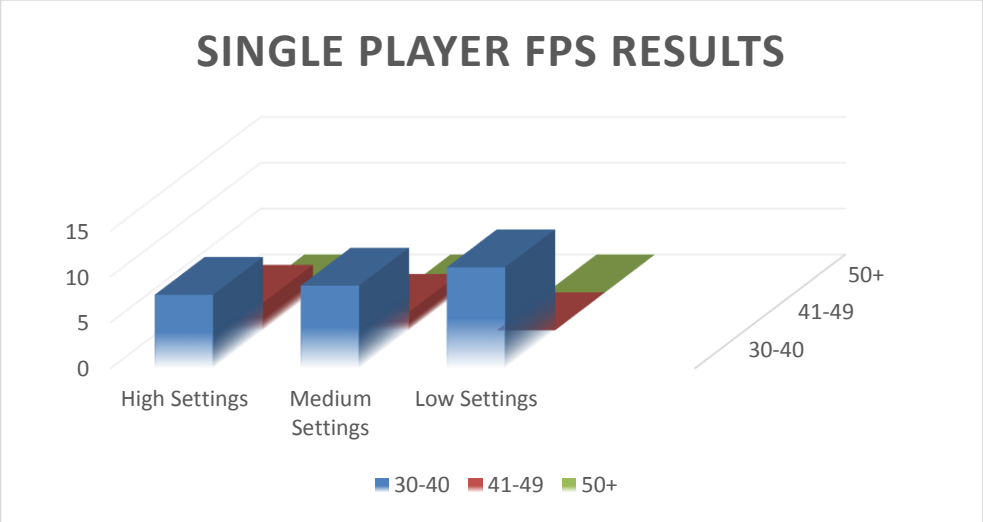
However during their experience of this level they will experience the use of the main weapons and also most common type of zombies, this will be just a taster of what is to come for the main level.

When moving throughout the level the user would encounter be pop ups that enhances their knowledge of the background story so they can engage in the story better.

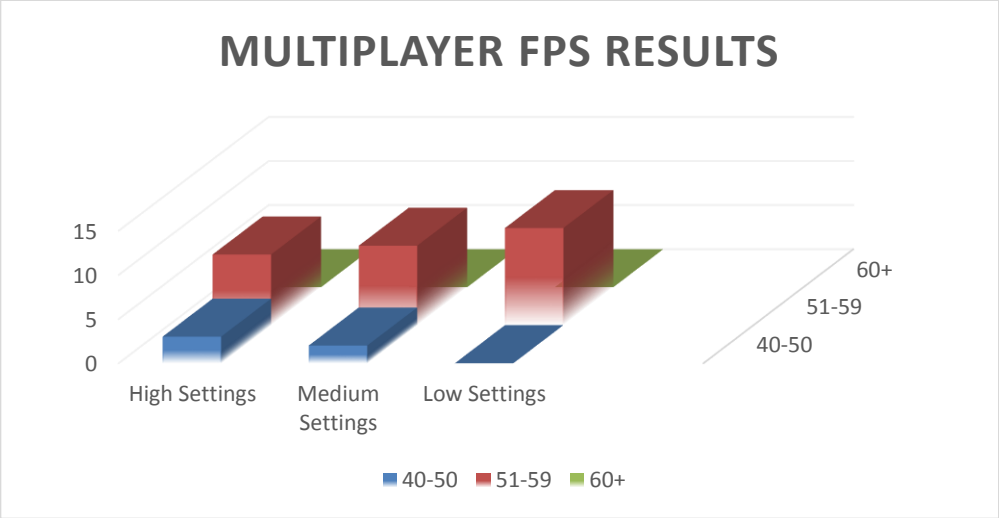
12 CUSTOMER TESTING

Extensive user testing took place. The idea of this was to evaluate the game performed when an average user tried to interact with the software, this was done during single and multiplayer game modes. The reason for this was benchmarks are significantly different for single and multiplayer. Frame rate tests the overall performance of the game, the optimal performance for a game running offline in single player is 30FPS and this transforms to 50FPS for an online multiplayer game. To run these tests I gathered 11 colleagues and took the results throughout the process of them playing the game. I ran the benchmark tests at high, medium and low settings to gather an overall view of the performance on a standard spec pc.

The results of single player FPS where very satisfying. 7 out of the 11 participants were getting optimal performance at high settings with 4 getting great performance, 8 where getting optimal performance at medium settings with 3 getting great performance and everybody was getting optimal performance at low settings. I was very happy with these test results as it showed that my game runs without any issue on a standard spec computer.



Secondly I ran benchmark tests for each user joining the same server and playing in the one game mode together. I ran these tests for both the hosts and clients. The results couldn't have been much better as the performance was right where it should be. On high settings 7 users got optimal performance, with 4 getting great performance, on low settings 8 got optimal performance with 3 getting great performance and everybody got optimal performance on low settings. Again these test throw up get results on standard spec pcs.



13 EVALUATION

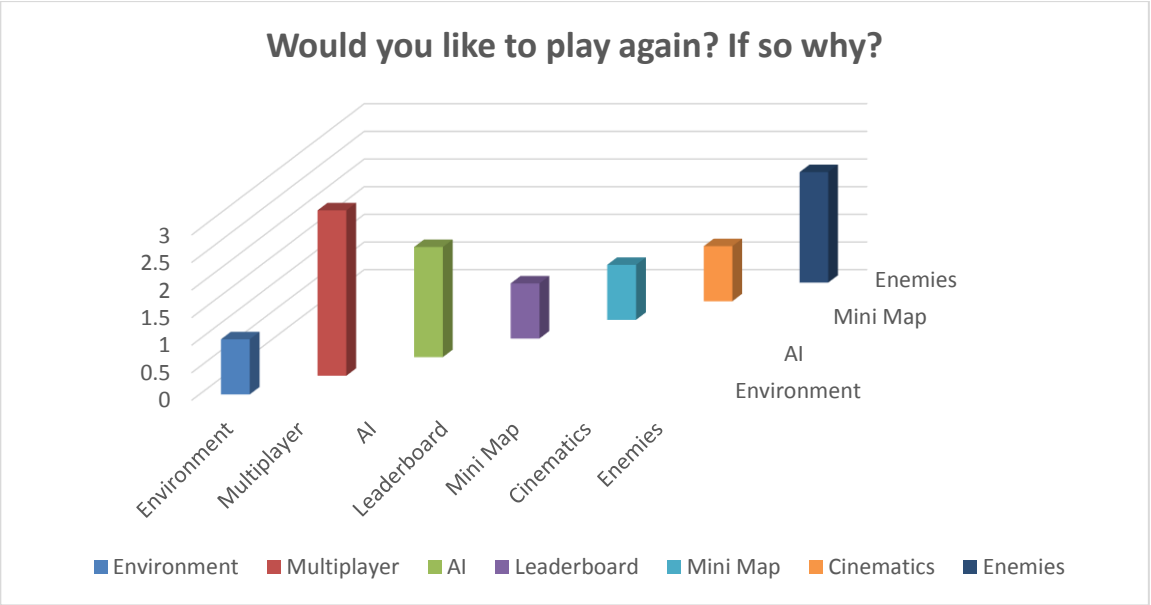
The 11 testers I gathered were provide with a survey to fill in that gave general feedback about the game in different areas. The graph below shows the questions and results each participant was asked. I gathered 7 males and 4 females which where a variety of different ages. There was 5 in the age range of 16-20, 4 between 21-29, 1 between 30-40 and 1 at 40+. These where all people who felt they could give good judgement of my project as they are game related. 5 participants said they play games occasionally and 6 said they play games every day. So overall I feel I am getting a good opinion from all aspects. Below are the questions and the results from the post-game survey.

Question 1) Do you feel the game fulfilled your expectation of a horror game?

The results of this question show that 6 people strongly agree and 5 people agree, I found these results really positive as it showed I addressed the features needed to create a zombie horror game.

Question 2) Would you play this game in your free time, if so why?

The results of this question where very positive as everybody said they would like to play again, and for a variety of different reasons. This showed that my game has a variety of functionality that appeals to many different audiences.

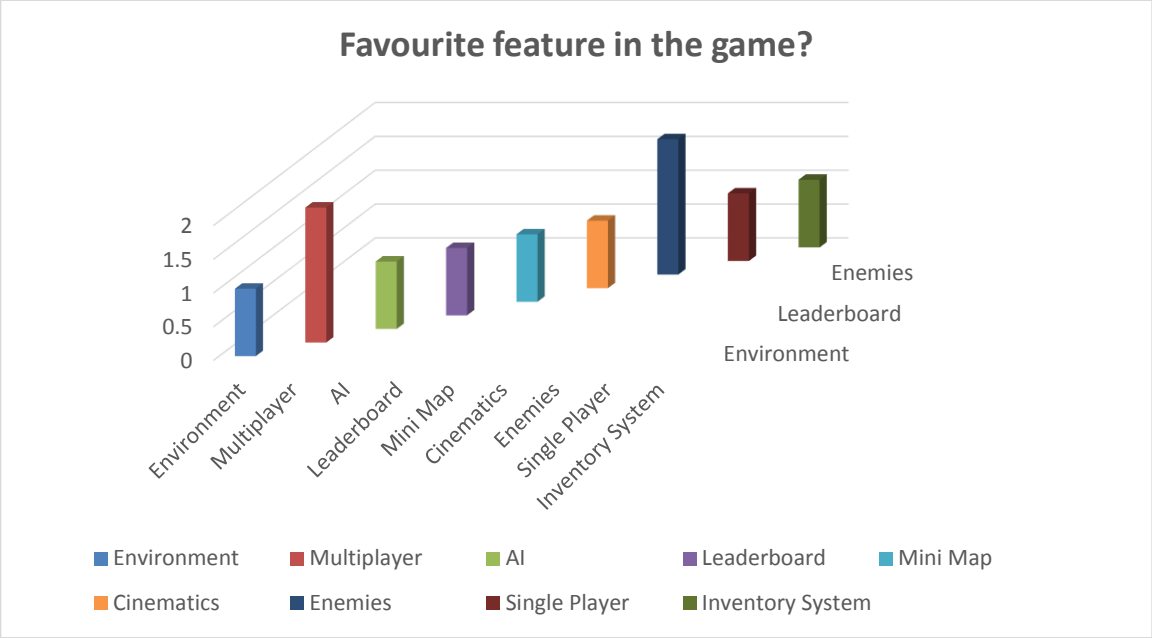


Question 3) After you complete the game would you play it again to beat your or your friends high score?

The result of these questions showed that the leaderboard plays a vital role in the users playing the game again. This can either be to beat their own high score or their friend's high score.

Question 4) What was your favourite feature of the game?

The result of this question showed that the users prefer different features of the game, this shows that the game appeals to a variety of audiences which is what I set out to do at the start.



14 CONCLUSIONS

From this experience I have learned a lot of new skills as I have never developed a game before so this was a completely new experience. The main reason for selecting to create a game for my final year project was that there was an interest in the area gaming from myself of how games were created, and also I had a chance to create a game in a style which was interesting. Having a genuine interest in the project area made working on it much more enjoyable, trying out different methods of implementation and game functions. As I would like to go on to have a career in game developed this type of project gives me the experience and skills needed to succeed in this field. This game takes an established genre and attempts to innovate on it in several different ways. This also has a lot of complex features incorporated to help give the game a unique feel. NCI gives us an introduction to Unreal Engine in the Graphics module where we were tasked with creating different sorts of functionality. This functionality wasn't enough to go on and develop a fully functional game, more research was required but it did act as a stepping stone to get a good start. With more research I was able to use Unreal Engine 4 potential in beautiful visuals, architectural visualisations and simulations to create a terrifying environment for the player. Incorporated for single and multiplayer modes using LAN so players can join other servers to play together.

The game has a variety of other functionality which includes:

- Single Player – Main game mode

- Multiplayer – Creates the servers through LAN
- Zombie AI –Artificial intelligence uses behaviour trees which help determine the best path to find the player.
- Cinematics – Gives more effect of the environment and situation to the player
- Leaderboard – Users can store their score and compare against other competitors

Gameplay features:

- Timer – The set amount of time the player has till the zombies overrun the level
- Inventory system – For the weapons scattered throughout the game
- Health pickups – If the player is attacked they can regain their health
- Pickup keys – To advance through the game and progress through the story
- Mini Map – Players can view where enemies is on the map so they can avoid and judge where to go

However no project goes without its disadvantages associated with the development stage, it was evident during development of nearly every game that they will be extremely large and complex to develop thus requiring a long time to even get it to a playable state. The project was faced with some limitations namely the small development team which had one developer with 9 months of time to develop this project. With a larger team and more time I feel this could go a lot further and gain high attention from customers. Although I am extremely pleased with the outcome of what I was able to accomplish and I feel this has commercialisation potential as it is unique and could do well on Steam Greenlight with some advertising as zombie horror games are very popular in the market at the moment.

15 FURTHER DEVELOPMENT OR RESEARCH

In future development with this game I would love to incorporate the Oculus Rift virtual reality headset with Motion Controller. This is still without a release date but early versions have already blown us away with the devices ability to immerse wearers in virtual worlds. At the beginning of development you could play games while wearing the Oculus headset while seated in front of your keyboard and mouse, or while holding a controller. However, developers found when your display gives you the ability to look

up, down and all around you with life like head tracking people will want more immersive, realistic type of input as well.

Teddy Lipowitz, a developer from Australia, put together a demo of a first person shooter horror game that uses Motion Controllers. The simplistic demo pits a single player with a gun against Alien soldiers. Physically ducking your body lets you hide behind crates to dodge their blasts, and this setup allows for precise control of the gun. The Motion Controllers aren't just for gun control, they also track the movement of the player's torso as the Oculus Rift tracks your head not your body.

I feel this would be a great addition to the future development of New Age of the Dead as this is completely new technology on the market and would be really well perceived. As Oculus are bringing out new features these could be added to the game as they get released for future upgrades.

I feel my project has efficient potential to be commercialized, as it is unique, incorporates the best features from different games in the same genre and puts these into one package. One way of commercializing my project is through Steam Greenlight. Steam is the largest market for PC gaming amassing 7.5 million concurrent users, providing the game with a large market straight away. This is a system that enlists the community's help in picking some of the new games released on Steam. Developers post information, screenshots, and video footage for their game and seek a critical mass of community support in order to get selected for distribution. Steam Greenlight also helps developers get feedback from potential customers and start creating an active community around their game during the development process. This new process of picking which game should get chosen gives smaller game developers more of an opportunity to get their game on Steam. The old way of deciding was not the most efficient, not only was the decision made by just a handful of reviewers, they may have also missed opportunities by declining games that would have been interesting to a larger audience.

This could all be incorporated with a variety of different methods in order to get your project known. This could include:

- Development of a website, whether this acts as a home base for all of your games, or just the one you're currently working on, it needs to be updated frequently and departmentalized. The home page should feature an extended overview, captivating screenshots and a media page with images and videos.
- Social media is also a very powerful tool to get your game known, this could include a Facebook and Twitter profile, in theory you could subscribe to dozens of social media outlets.

- A development blog could be used also, while these are less essential than social media and a website, most gamers and developers love reading about personal struggles and celebrations during the development process.
- Trailers could be made when the game development is nearly finished. This is one of the most important things to do in order to get people excited about playing your game. Just a simple trailer which includes gameplay or cinematics can get the hype of a game through the roof and can contribute to making a successful game.

This is a great opportunity for young developers like myself to get noticed and have early success with a self-made project online for customers to buy or download.

16 REFERENCES

Bibliography:2016 (2004) *Game engine technology by unreal*. Available at: https://www.unrealengine.com/blog (Accessed: 8 May 2016).**In-line Citation:**(2016, 2004)

Bibliography:VR, O. (2016) *Natural gestures and movement*. Available at: https://www.oculus.com/en-us/touch/ (Accessed: 8 May 2016).**In-line Citation:**(VR, 2016)

17 APPENDIX

17.1 PROJECT PROPOSAL

New Age of the Dead

Student Name: Glen Ward

Number: x12436692

Email address: x12436692@student.ncirl.ie

Degree Program Name: BSc (Hons) in Computing

Specialisation: Gaming and Multimedia

Date : 1/10/15

Objectives

The main objectives of my project would be to create a fully functional survivor horror game on Unreal Game Engine. This will be aimed at the Desktop PC market. The main game is being made on unreal engine as I find this is much stronger software then unity engine. I plan on incorporating a lot of functionality throughout my project, this would include Health for player and AI, health regen packs for the player which they can restore a percentage of their health, unlockable doors using specific keys, cinematics using matiness throughout the game at key points, inventory where the player can pick up items they find and use these to equip/drop or even craft new items by mashing other items together and much more!

This game will be a 3rd person shooter survivor horror game where the user will have battle for survival and pass through the different levels in order to save the world from a zombie apocalypse. The 3rd person game is a genre of 3D action games which the player character is a visible on-screen and the gameplay consists of shooting. It combines the shooting elements of the first-person shooter with the jumping and climbing puzzles of a 3D platformer and a simple melee fighting system from a brawler. Third-person shooter games almost always incorporate an aim-assist feature, since aiming from a third-person camera is difficult. Most also have a first-person view, which allows precise shooting and looking around at environment features that are otherwise hidden from the default camera.

Background

The area of gaming is one that has seen tremendous growth over the last number of years. Gaming is creating jobs in various sectors.

The reason I chose this project was due to my interest in gaming. I use my free time to play games and relax, like FIFA, league of legends, GTA etc. and I would like to make something that I use as a hobby and incorporate this as my final project. The main reason I chose BSHC Computing Science was to be able to learn how to make games and how I could specialise in this area near the end of my degree. During this summer I took numerous online courses to prepare myself on how to create games so I would have an idea of how games are made and how I could go about creating a game. The online courses consisted of how you could use C# in Unity Engine and incorporate this with modelling to make simple games. Also how you go about using Unity from scratch as a beginner.

The reason I chose 3rd person shooter was because I have a keen interest in that area of gaming as GTA is one of my favourite games, this area of gaming has a huge following of players. The best well know games is this area is perhaps:

- GTA:

Grand Theft Auto V is an open-world, action-adventure video game developed by Rockstar North and published by Rockstar Games. It was released on 17 September 2013 for the PlayStation 3 and Xbox 360, on 18 November 2014 for the PlayStation 4 and Xbox One, and on 14 April 2015 for Microsoft Windows. The game is the first main entry in the Grand Theft Auto series since 2008's Grand Theft Auto IV. Set within the fictional state of San Andreas (based on Southern California), the single-player story follows three criminals and their efforts to commit heists while under pressure from a government agency. The open world design lets players freely roam San Andreas, which includes open countryside and the fictional city of Los Santos (based on Los Angeles).

In order to play this game the player must think logically of what to spend their money on and how to navigate through the level. All of this is a highly enjoyable style of gameplay and one which I feel I could make an adequate contribution to in terms of my project.

I thought about using Unity the main platform for my project. The reason I chose Unity Engine was because I feel it offers a great deal of features that could go a long way to simplify features which otherwise would take a great deal of time on other engines. Unity makes the addition of physics, terrain and models as simple as drag and drop. However Unity does not remove the majority of the complexity that the project will turn out to be, so the coding that will be involved will not be decreased in any way. Unity works really well with blender which will be my preferred choice of program to create 3D models. When you create your model in blender you simply save this file and drag and drop this into Unity, whereas with other Engines you have to save the file in blender to a specific format and then try and import this into the library which may cause some errors.

But I decided to go with Unreal Engine 4, Unreal Engine 4 is a complete suite of game development tools made by game developers, for game developers. From 2D mobile games to console blockbusters, Unreal Engine 4 gives you everything you need to start, ship, grow and stand out from the crowd. Revolutionary new workflow features and a deep toolset empower developers to quickly iterate on ideas and see immediate results, while complete C++ source code access brings the experience to a whole new level. Unreal Engine technology powers hundreds of games as well as real-time 3D films, training simulations, visualizations and more. Over the past 15 years, thousands of individuals and teams and have built careers and companies around skills developed using the engine. I feel this would benefit my project a lot more.

Technical Approach

Research:

I carried out a great deal of research before starting my project. I had to think about numerous factors before coming up with my project idea as I did not want to start a project and make it half way through and not be able to finish it. I discovered the game engine Unity would best fit for making arcade so this wouldn't be the best fit for making my zombie game.

I also did some research into the style of game I am going to be creating. Zombie first person shooter game is very much in the market at the moment where Techland and Warner Brothers created the game "Dying Light" which was nominated for over 50 awards. Dying Light is a 3rd person shooter and this game shows that there is a market for this at the moment. I aim to use this type of game as template to such and use this for ideas. But before I start I plan on laying out all the potential features that I fell would fit into my project and spend time researching them and seeing what would take the longest amount of time so I can adjust my time accordingly. I also plan on reading books and looking at YouTube videos in order to gain the knowledge for this time limit to incorporate these different features too.

Interface Requirements:

- User interface
- Weapons
- User customized
- Hardware Interface
- Software Interface
- Communication Interface

Other functionalities I plan on incorporating into my project are

- Survival tactics
- Dialogue
- Mapping
- Levelling system
- Instances
- Implementation

For my main game which will be built on Unreal Engine this will involve code and blueprints. Blueprints can bring different functionality to games made on Unreal Engine.

Special resources required

There is no special resources needed to undertake this project. Since I am making a game the hardware I need to create this will meet the minimum requirements of having a set of components. The main component we need that will be applicable to the GPU, if this unit is not powerful enough then I would need to procure different hardware on which to create and the game might not even run. Therefore I can't see any issues with regard hardware resources to me or both at the college or at home.

Overall the resources I need are:

- Mouse
- Keyboard
- Screen/monitor
- Power supply

Technical Details

For my main 3rd person zombie shooter game I plan on incorporating the functionality through blueprints and code in a language of my choosing. The **blueprints visual scripting** system in Unreal Engine is a complete gameplay scripting system based on the concept of using a node-based interface to create gameplay elements from within Unreal Editor. This system is extremely flexible and powerful as it provides the ability for designers to use virtually the full range of concepts and tools generally only available to programmers. Through this we can design, implement, modify the games, players, cameras, input, items and environments etc. blueprints will be my main source of coding and will provide a base for everything I do throughout my whole project. Every functionality for every game that is developed on Unreal Engine from mass production products to start up products. For the character development I plan on using Mixamo which is a highly popular piece of software which possess a highly powerful editing tool that allows the user to modify every bone, skin and hair on the character. In relation to animations I decided to adopt Autodesk Maya, which is one of the quickest tools for making different animations, as I have a lot of animations to make which would be for the boss AI, Zombie AI and player character itself. Regarding the cinematics I wish to use matiness to create these, this is a simple feature in Unreal Engine which allows you to create multiple sorts of cinematics which could include animations, sound, fighting scenes or simply an overview of the area.

Evaluation

I plan of evaluating my games through the use of testing that will be carried out by my fellow classmates and other students of NCI as I will host different prototypes at certain stages of my game and get the users to test this at these particular stages.

I also plan on using white-box testing which will test the internal structure of my game as opposed to the functionality that will be tested by my fellow classmates. In general, white-box testing an internal perspective of the system, as well as programming skills that are used to design test cases. So I will ask the testers to give their input so I can exercise different paths through the code and determine the possible different routes and choose the best possible route.

Glen Ward

_____

Signature of student and date

17.2 PROJECT PLAN

See Attached document

17.3 MONTHLY JOURNALS

17.3.1 SEPTEMBER

Reflective Journal

Student name: Glen Ward

Student Number: x12436692

Programme BSc in Computing

Month: September

My Achievements

This month, I was able to come up with my project idea and this is something I feel will be definitely achievable over 4th year in NCI. I feel with the skills I will learn throughout this year and also the skills I have learnt over my stay in NCI and also with my work placement in SAP, this project will no doubt be challenging but will be rewarding upon completion.

The aim this month was to come up with a project idea and make a good project proposal. In this project proposal I have outlined many tasks like, the objectives, background, technical approach, Gantt chart etc. In the Gantt chart I have made numerous timelines for my game, I feel these will help keep me on schedule throughout 4th year and ensure I will not get side tracked. I aim to be completed my step of my game which is outlined on my Gantt chart by the end of October, and so on. If all this goes well I will well forward on my main zombie shooter game at Christmas and fully concentrate on this until the end submission in May.

My Reflection

I feel this month has been very successful and this is shown in the great start I have made, and I was even able to get more work done then I planned at the start of the month. This will be fantastic moving on to next month.

Intended Changes

Next month, I will be able to get started on designing each of my characters and hopefully if this goes smoothly then I can maybe start on animations too.

17.3.2 OCTOBER

Reflective Journal

Student name: Glen Ward

Student Number: x12436692

Programme: BSc in Computing

Month: 5/11/2015

My Achievements

This month I was able to achieve a main menu, along with AI, main character, made a building for level 1, gun with lasers, game over and death for character.

For the main menu I was able to achieve this through interfaces, blueprints, images etc. In order to make the gun, I created a sphere collision and material in the shape of lasers which are going to come out of the gun and kill the enemy. The AI was done in a similar manner with blueprints which act as the enemy and attract the enemy to the player, I found this the most complicated to put in my project so far. At the moment when the enemy hits the player he dies automatically and it shows game over. The enemy has 30 health and the bullets count as 10 health so when each bullet hits the user this shows the remaining health for the AI.

My Reflection

I feel I have achieved a lot this month and managed to achieve more then I set out to do. As we had a lot of CA's this month I balanced my work and project well this month and I aim to do the same next month.

Intended Changes

Next month, I will try to make the next levels for my project, make my project unique by adding stealth into my game. Also I would like to try and get the opening cinematic completed. I feel this would be a lot to get completed in one month regarding other projects and submissions due the same month.

Supervisor Meetings

Date of Meeting: 20/10/2015

Items discussed: We discussed how to make my project unique and add different difficulties into my project after I get the first 2 levels made.

Action Items: I have found out how to make my project unique by adding stealth into this, so when the user runs out of ammo they can stealthy sneak by them by searching the bodies and the enemy will think you are one of the zombies too.

17.3.3 NOVEMBER

Reflective Journal

Student name: Glen Ward

Student Number: x12436692

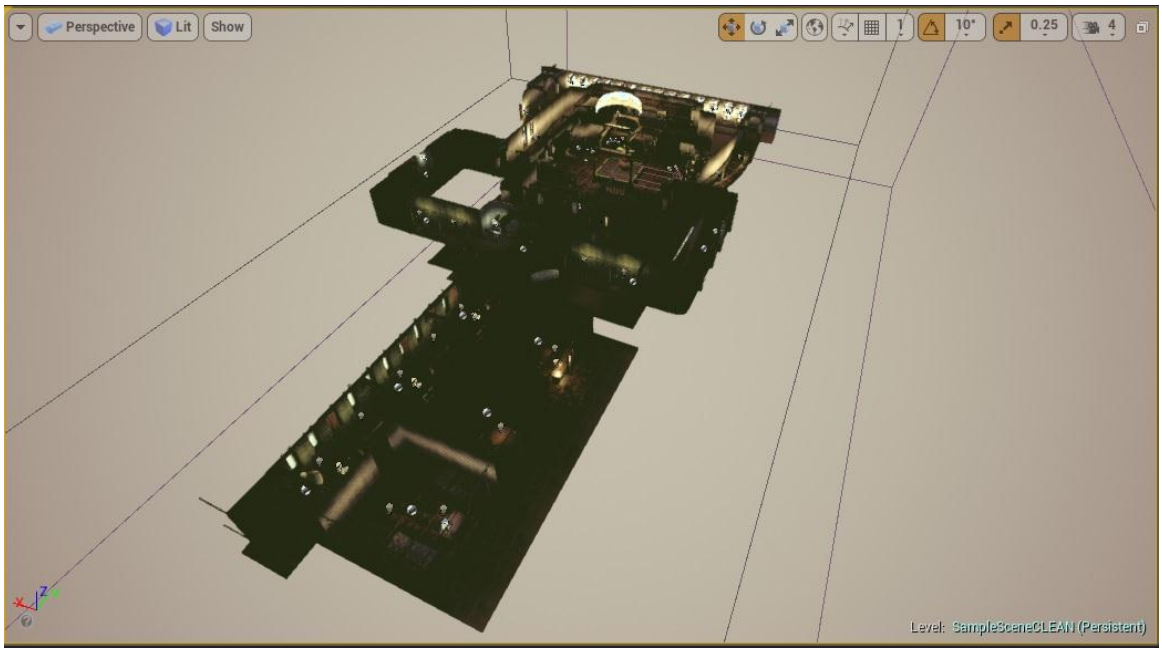
Programme: BSc in Computing

Month: 1/12/2015

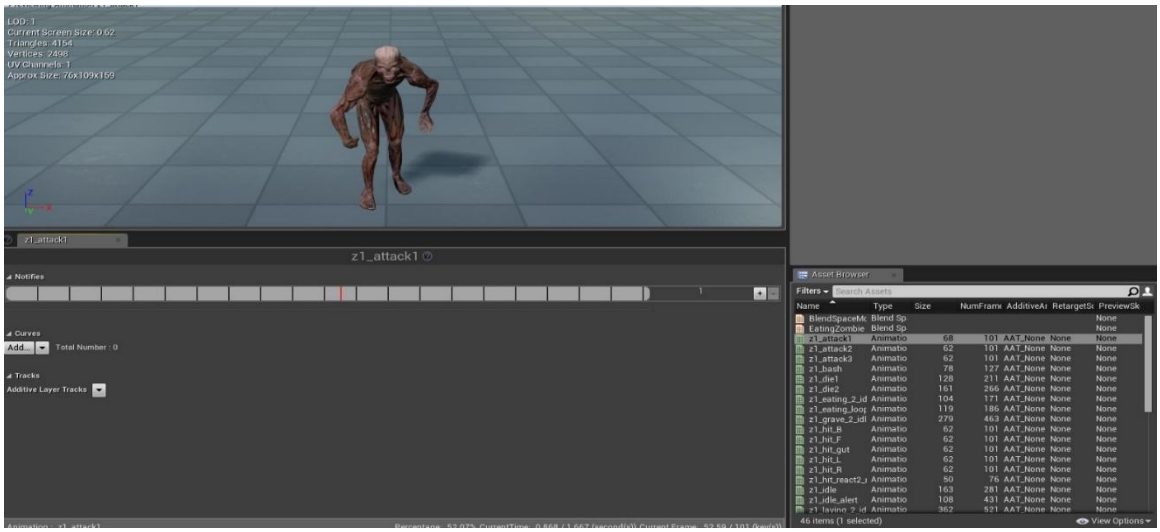
My Achievements

This month so far I was able to achieve fully designing the main level and also completing the design of the first level to my game. I was also able to complete the design of the enemies of the zombies with all the animations.

For the main level I used different materials and incorporated them together in order to create a horror environment like an abandoned manor.



The design of my zombies are all designed to be as scary as possible and the animation I used was to attack, get hit, go forwards, backwards, death animation and raise from idle to moving etc.



I was also able to achieve an aim offset for the main player, this means the user will be in combat mode at all times, which I would like. When the player moves around the centre of the screen will be where the user is aiming at all times.

My Reflection

Overall I feel I making very good progress with my project, I feel I am nearly completed achieving the basics for my game so I can kick on and get working on rest of my game and giving it all a very good structure so the audience will be enthralled overall.

Intended Changes

Next month, I will try and get the opening cinematic done for my game. This will be taking from the house in my first level and give the audience an idea of the background story so far. Also I would like to achieve getting the weapons working on the main player so a user can shoot the enemy and each bullet will stand for a certain health of the enemy.

Supervisor Meetings

Date of Meeting: 24/11/2015

Items discussed: We discussed the progress I have made so far and went through how I am progressing and that both parties are happy with how I am progressing. We have scheduled a new meeting which will take place in the 8th December which will be a day for demoing what we have achieved so far and demoing the project to her, we feel this is the best way to approach the meeting from her forward.

17.3.4 DECEMBER

Reflective Journal

Student name: Glen Ward

Student Number: x12436692

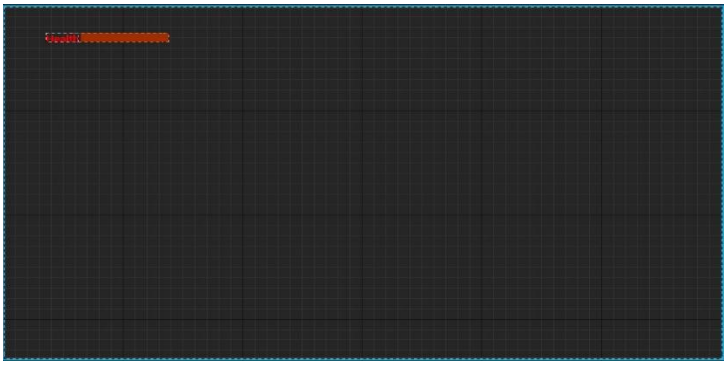
Programme: BSc in Computing

Month: 18/12/2015

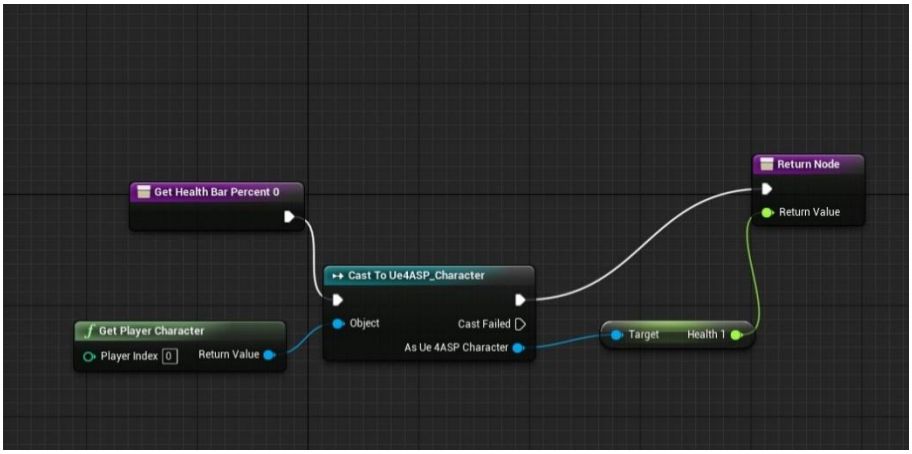
My Achievements

This month so far I was able to achieve fully making the opening cinematic for my game. This included making numerous matinees. A matinee is a feature for unreal engine which allows the user add numerous cameras into the scene and record key frame to key frame of what you would wish. Mine included zombies summoning from the ground, roaming and an overview of the environment.

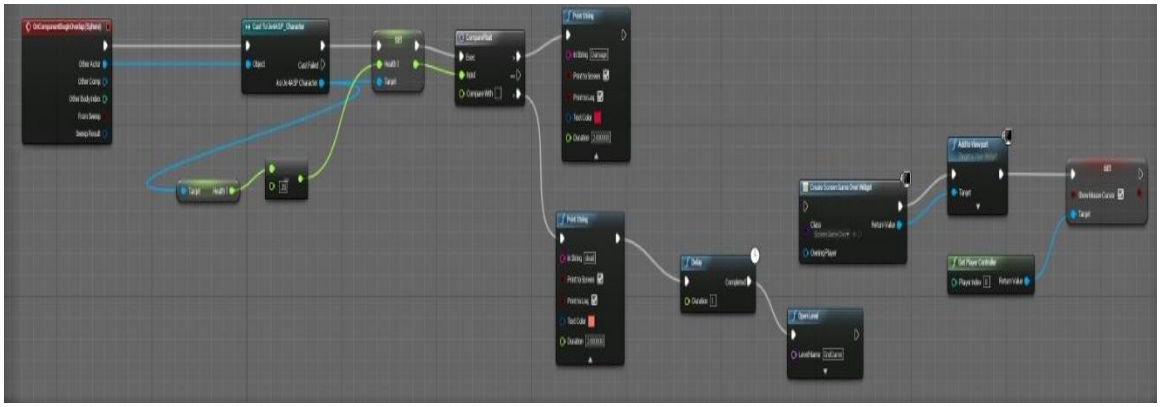
Also I was able to incorporate a health bar for the main character of my game, this includes giving the player 100 health, and for each hit the enemy has on the player it removes 10 health and external damage such as fire removes 4. I was able to achieve this by making a widget to design the health bar, and also the functionality was achieved through blueprints. Creating the widget I was able to show the health to the user as red first and as the player takes damage they see black appearing where they took damage.



On the health bar I was able to create a function which states that the lowest percent which is 0 and cast this to the main character, the return node will take in the health itself and the character.



Now on the character we need to state how the damage will take place, so here I state that when there is an overlap which means when they collide, we set the health, off this we set how much damage this will be, now we can compare the float so if this is > than we show dame and if not then the character is dead and play the death animation and send this to the game over screen.



My Reflection

Overall I feel I'm making very good progress with my project, and should be able to complete my aims I set out in my original project proposal.

Intended Changes

Next month, I will try and create an inventory for my game, where the player can equip weapon they find throughout the level and use these to fight enemies. In this menu the user will be able to equip and drop weapons. Also I will add that the player can equip weapons without using the inventory. If I have time I will hopefully be able to incorporate health pickups, this means that if the player has taken a significant amount of damage then they can restore 25% of their health. I would like to also add jump scares, throughout the level jump scares will appear with an image of what happened and will add to the background story, pickup notes which would be stories from the characters family which will provide clues of how you can escape and stay alive, find keys to open a key amount of doors.

17.3.5 JANURARY

Student name: Glen Ward

Student Number: x12436692

Programme: BSc in Computing

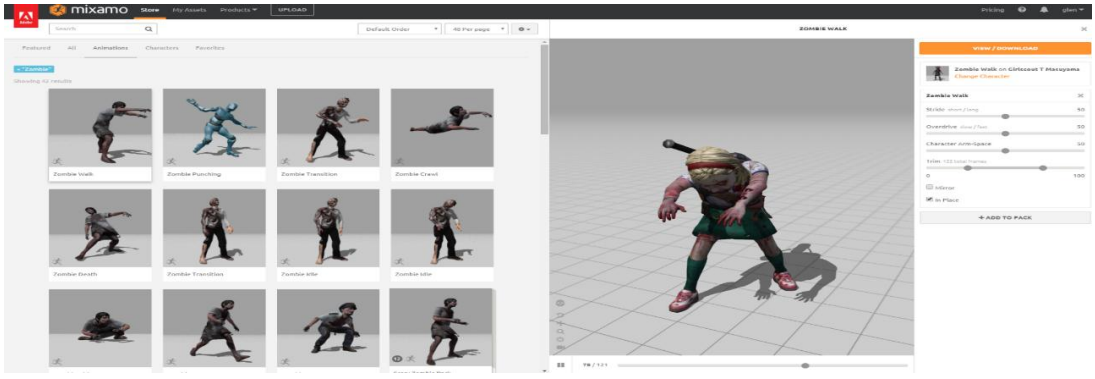
Month: 03/02/16

Achievements

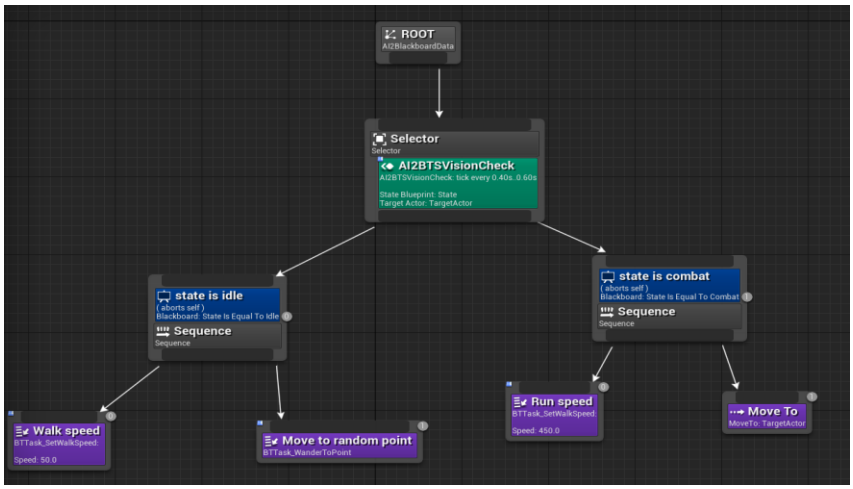
This month I was able to get a lot done, I achieved with making the boss AI for my game and also the character to go with it, made health regen packs so when the player walks through them they gain health, also a fully working inventory system and checkpoints throughout my game so if the player dies they can respawn at a certain point in the level that they last left off at.

Design character and Boss AI:

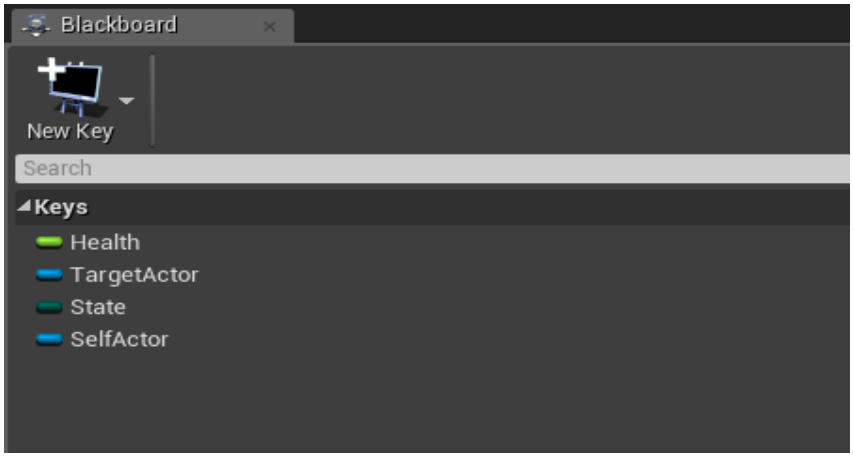
Made character on Mixamo fuse and selected/made the animation I wanted to use



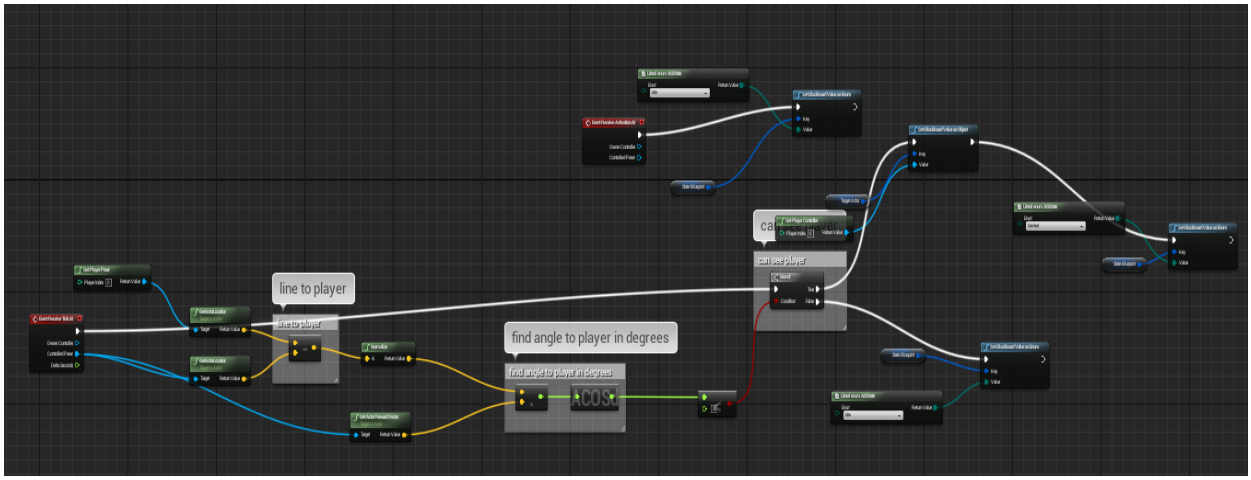
Created the necessary behaviour tree so the AI knows when to roam, be idle and run and attack the player



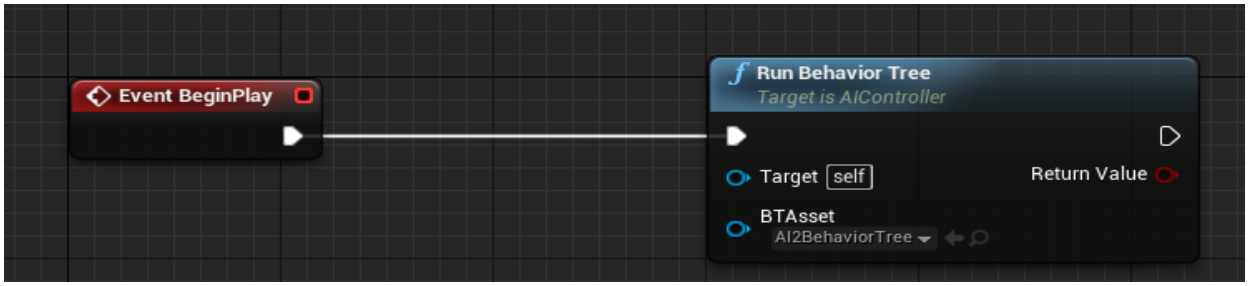
The blackboard is used to create the states that the AI will go through in the game



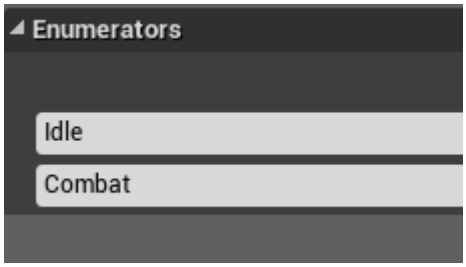
This provides the vision check from the AI looking for the user, this gets called in the behaviour tree when this happens



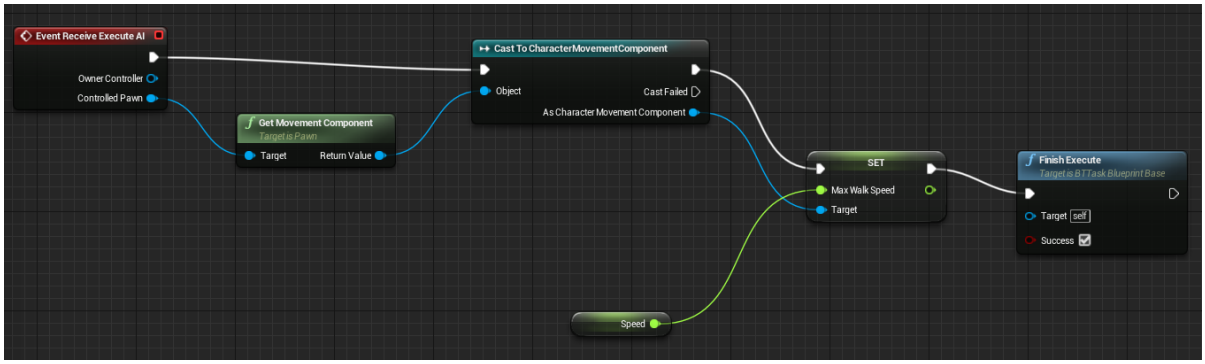
This is the AI controller, this calls the whole process of the behaviour tree when it's set up



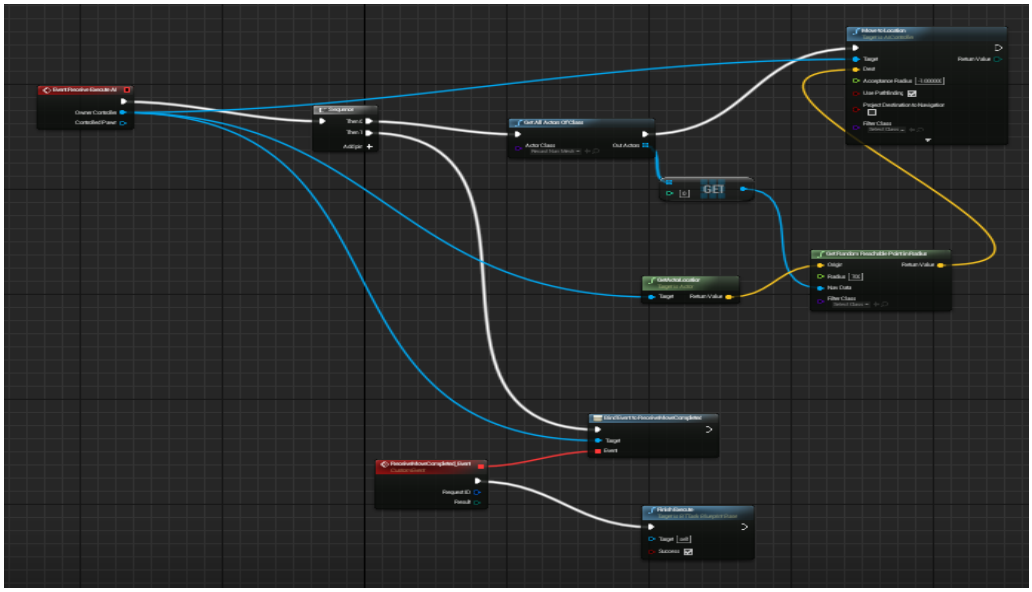
These are the 2 states the AI can go into when walking these have to be enumerators for the AI to be able to access them



This sets the walk max walk speed for the AI



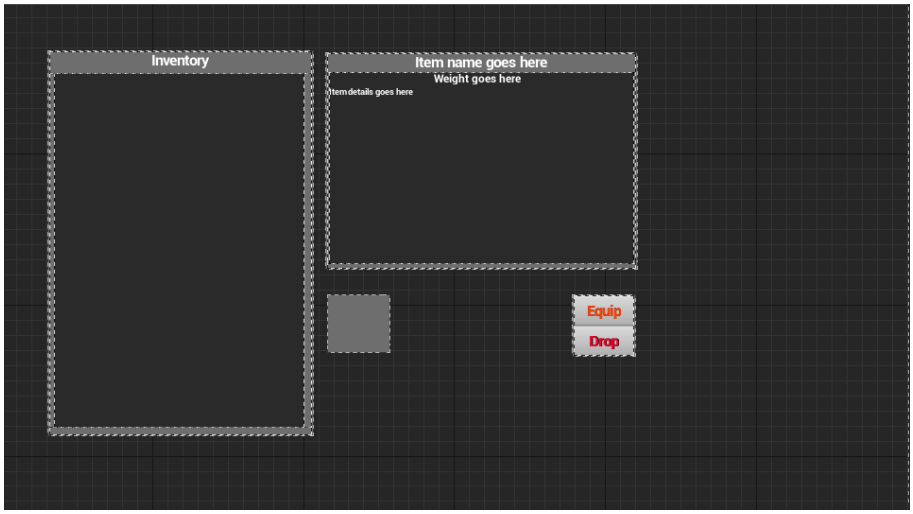
Below sets the AI to wander around the level and pick random points to search, the AI can see in the distance unlike the zombie AI as this boss should be harder to kill and more intelligent than the zombie AI



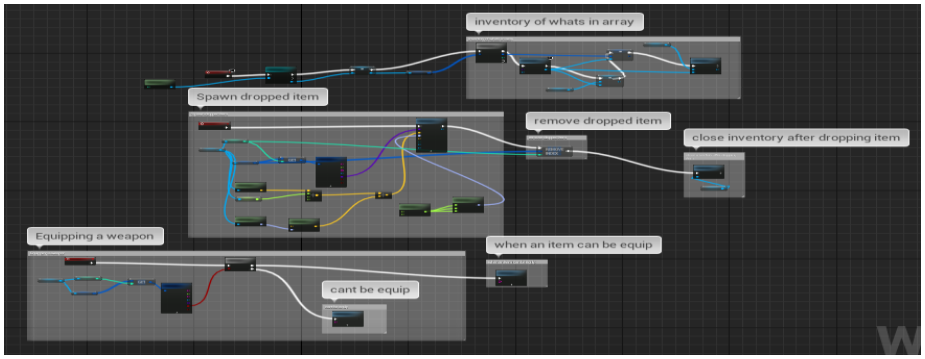
Inventory System:

ItemData provides the Array which everyone will be placed in.

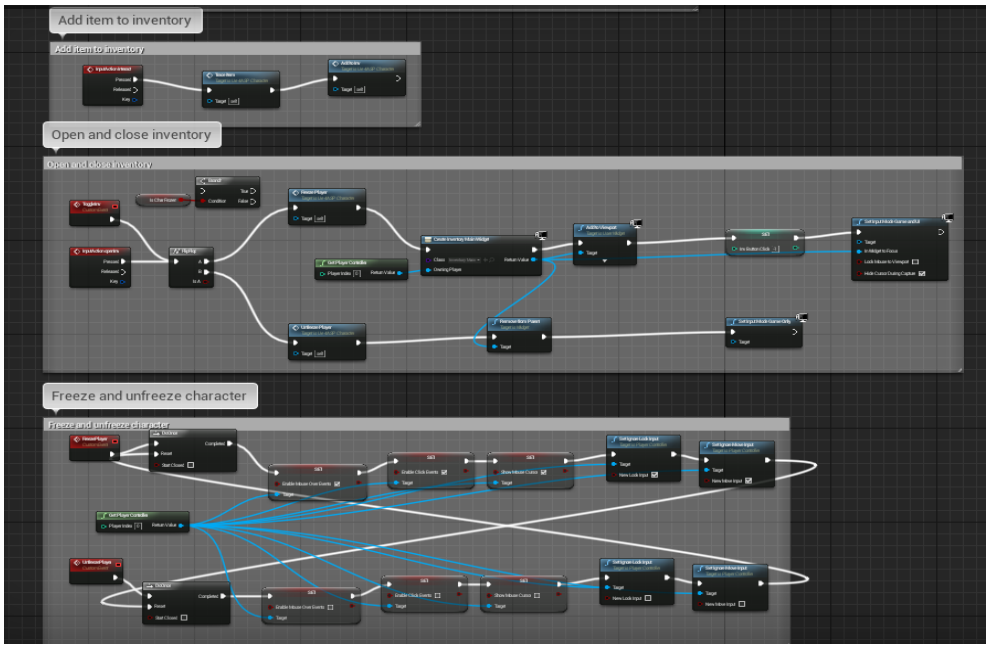
Below is the layout of the inventory widget and what this will look like to the user when they open this in game



The functionality inside the widget calculates whats in the array, how to spawn a dropped item, remove a dropped item from the array and how to equip a weapon from the inventory and what can be equipped

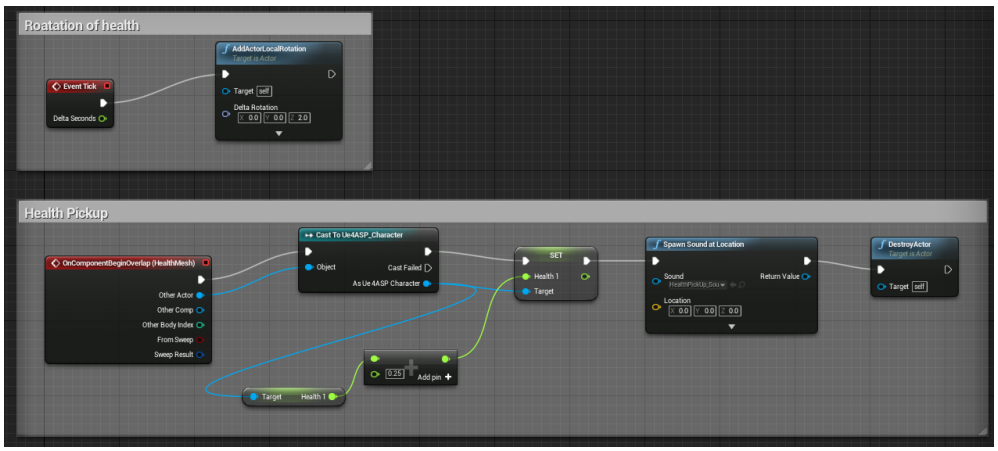


Inside the player blueprint is all the functionality for the player, regarding the inventory the player needs to know what's be added to the inventory, how to open and close the inventory and freeze and unfreeze the player while the inventory is open, this basically means pausing the game while the inventory is open.



Health regen packs:

Inside the health pickup, we need to have the functionality of how the health is being added to the health bar of the player, this will rotate throughout the game to add neat effect for the player



My Reflection:

Overall this month I feel I got a lot done, I feel this is because the stress of exams where gone and I'm starting to get to grip with this project as I know what direction I am going in

Intended Changes:

Next month I would like to add a multiplayer feature to my game where one player could be the main character and the other could player as the zombie, I feel this would be very cool and unique to add as not many, if any, other horror game possess this feature. I am unsure how hard this would be to incorporate but with rigid research and a good understanding of what is required I feel this would be possible.

Supervisor Meeting:

Meeting with Christina is due to take place tomorrow

17.3.6 FEBRUARY

Student name: Glen Ward

Student Number: x12436692

Programme: BSc in Computing

Month: 03/03/2016

My Achievements

This month I was able to achieve getting multiplayer working where the servers are created on Steam so this game can be played from two separate locations. This was very difficult to implement as it is a very complex feature. Also I added a mini map feature where the map shows where he player is and also where all the enemies is. This allows the user to see where they have been and where to be careful if the enemy is nearby. Also I finalised my save and load system where the user can pause the game and save it at any location, they can load the game from the main menu and this will resume the game from the last save. Finally, this month I redone my all my menus with different font, layout and background pictures which better suit my game.

My Reflection

Overall this month I feel I got a lot done, I feel this is because the stress of exams where gone and im starting to get to grip with this project as I know what direction I am going in

Intended Changes

Next month I would like to a login and registration system so each user can have a specific account to my game, this will benefit the game overall as then each player will have a specific game save and load, campaign mode. Also I would like to add a time feature which gets pushed into a leaderboard which the user can see globally and then just their times in general.

Supervisor Meeting

Meeting with Christina took place on the 1st March, we discussed where my game was, and as she felt the main features of my game has already been completed she asked me to implement a login and registration system to add more functionality to the game.

17.3.7 MARCH

Student name: Glen Ward

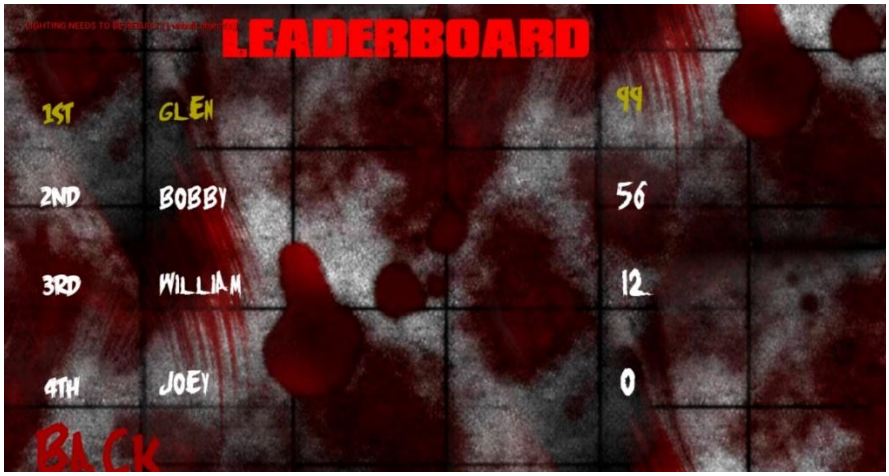
Student Number: x12436692

Programme: BSc in Computing

Month: 04/04/2016

My Achievements

This month I was able to achieve getting a full leaderboard implemented. After the user completes the game they will be prompted with a widget where they can see their score and enter their name to enter into the leaderboard. This will be saved to a file which will be loaded back in to the game which can be viewed from the main menu. This will display the top 4 results that has been achieved in the game. This will display the players name and score and the rank they are in.



My Reflection

Overall this month I feel I got a lot done, I feel this is because the stress of exams where gone and I know where I want my project to be at the end

Intended Changes

Next month I intend to start patching everything together and finishing my project off. When that is complete I intend of adding all my new functionality that I have achieved in my project and start documenting this in the final report.

Supervisor Meeting

Meeting with Christina will took place on the 22nd March. We discussed the progress I have made on my project and she told me she was very pleased with my progress and to finish off getting my leaderboard working and then start working on my documentation to document the progress.

OTHER MATERIAL USED

Any other reference material used in the project for example evaluation surveys etc.

CD containing code should be glued to the technical report.

(**Only applicable to the Final Report in May 2016**)

18 USER MANUAL

Control Scheme – Keyboard and Mouse:	**Control Scheme – Gamepad**
W – Move Forward	Left Analog – Move Forward
A – Turn Left	Left Analog – Turn Left
S – Move Backwards	Left Analog – Move Backwards
D - Turn Right	Left Analog – Turn Right
I – Inventory	Y - Inventory
Left Click – Shoot	Right Trigger - Shoot
E – Equip Weapon	X – Equip Weapon
O – Open Doors	Right Bumper – Open Door
Z – Drop Weapon	Left Bumper – Drop weapon
Spacebar – Jump	A - Jump
C – Crouch	Down - Crouch
F – Toggle Flashlight	Left – Toggle Flashlight
Left Shift – Sprint	Left Trigger - Sprint
P – Store Item in Inventory	B – Store Item in Inventory
Tab - Toggle 1st and 3rd	Right – Toggle 1st and 3rd Person