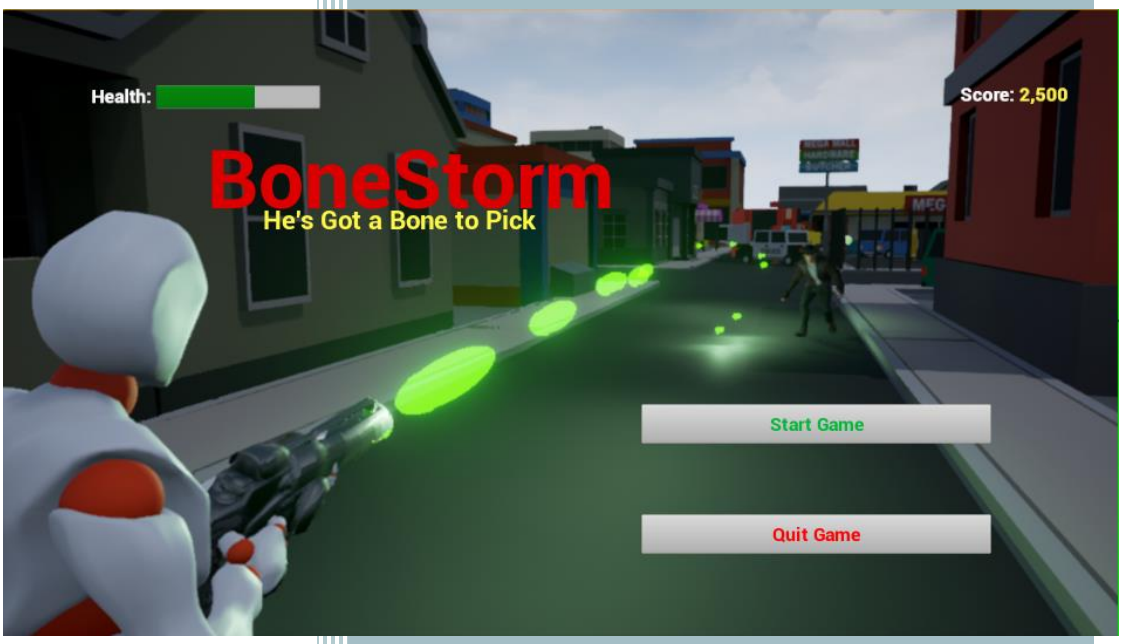
BoneStorm

Technical Report



Student Name: Alan Coleman

Student Number: x12421722

Student E-Mail: x12421722@student.ncirl.ie

Course Name: BSc (Honours) in Computing

Specialisation: Gaming & Multimedia

Project Supervisor: Paul Hayes

Declaration Cover Sheet for Project Submission

SECTION 1 *Student to complete*

Name: **Alan Coleman**
Student ID: **X12421722**
Supervisor: **Paul Hayes**

SECTION 2 Confirmation of Authorship

The acceptance of your work is subject to your signature on the following declaration:

I confirm that I have read the College statement on plagiarism (summarised overleaf and printed in full in the Student Handbook) and that the work I have submitted for assessment is entirely my own work.

Signature: Alan Coleman Date: 07/05/17

NB. If it is suspected that your assignment contains the work of others falsely represented as your own, it will be referred to the College's Disciplinary Committee. Should the Committee be satisfied that plagiarism has occurred this is likely to lead to your failing the module and possibly to your being suspended or expelled from college.

Complete the sections above and attach it to the front of one of the copies of your assignment,

What constitutes plagiarism or cheating?

The following is extracted from the college's formal statement on plagiarism as quoted in the Student Handbooks. References to "assignments" should be taken to include any piece of work submitted for assessment.

Paraphrasing refers to taking the ideas, words or work of another, putting it into your own words and crediting the source. This is acceptable academic practice provided you ensure that credit is given to the author. Plagiarism refers to copying the ideas and work of another and misrepresenting it as your own. This is completely unacceptable and is prohibited in all academic institutions. It is a serious offence and may result in a fail grade and/or disciplinary action. All sources that you use in your writing must be acknowledged and included in the reference or bibliography section. If a particular piece of writing proves difficult to paraphrase, or you want to include it in its original form, it must be enclosed in quotation marks

and credit given to the author.

When referring to the work of another author within the text of your project you must give the author's surname and the date the work was published. Full details for each source must then be given in the bibliography at the end of the project

Penalties for Plagiarism

If it is suspected that your assignment contains the work of others falsely represented as your own, it will be referred to the college's Disciplinary Committee. Where the Disciplinary Committee makes a finding that there has been plagiarism, the Disciplinary Committee may recommend

- that a student's marks shall be reduced
- that the student be deemed not to have passed the assignment
- that other forms of assessment undertaken in that academic year by the same student be declared void
- that other examinations sat by the same student at the same sitting be declared void

Further penalties are also possible including

- suspending a student college for a specified time,
- expelling a student from college,
- prohibiting a student from sitting any examination or assessment.,
- the imposition of a fine and
- the requirement that a student to attend additional or other lectures or courses or undertake additional academic work.

Executive Summary

BoneStorm is a 3rd person survival shooter game created in Unreal Engine 4, which delivers fantastic capabilities in graphics and performance.

Built to be an entertaining engaging game filled with mayhem in a chaotic world, so that the player can really enjoy the chaotic absurdity of the plot and gameplay.

BoneStorm is a game full of features:

- ✓ Artificially Intelligent enemies, these enemies have been given audio and sight detection to make them a realistic threat. They have also been given health.
- ✓ Weapons, using object classes, I have created a gun and projectile that our main character can use.
- ✓ Fully realised 3D world, the perfect arena for this game, large enough to escape enemies, small enough so you won't get lost.
- ✓ Difficulty settings so a player can higher or lower the difficulty settings to match their ability.
- ✓ Realistic characters, using Mixamo, I was able to create character models for our main character and enemies.

The objective of the game, is to survive hordes of enemies that roam the level. The player can go anywhere on the map but should try to avoid becoming close to groups of enemies. The player will have a laser gun, which he can use to harm enemies, but must be careful as he can also be harmed by enemy characters.

The game is built using Unreal Engine blueprints and it is developed primarily for windows PC's. All 3D character models involved were created using Mixamo, some however, may also been edited in Maya to be closer to how I envisioned the characters. Data for the game is stored locally on the user's computer.

Introduction

Initial Game Concept

BoneStorm is an engaging third person shooter set in a chaotic alternate universe. The world is one consisting of Skeleton creatures. When this world is attacked by strange flesh covered creatures (humans!), our hero must take arms in an attempt to save his town from the carnage caused by these monsters, with nothing more than his trusty laser rifle. What follows is an entertaining survival shooter full of danger and mayhem.

Background & Research

The game industry is one of the most rapidly growing industries in recent years and projections predict that this trend will continue. It provides exciting opportunities to developers who wish to begin a career where they can be creative. With the tools that have been made available, such as Unreal Engine 4, Unity or GameMaker, it is now easier than ever to get involved in Game Development from your own PC, in a way that was never before possible.

Unreal Engine 4 is a game engine developed by Epic Games. It has become one of the most popular tools used in game development. Largely as it is free to use as the only costs is a very reasonable and affordable 5%, of any money made through selling product developed in Unreal Engine after the first $3000 per game per calendar quarter.

Survival Shooters are a type of game where the player must defeat large amounts of enemies and complete objectives, so that they can survive a level as long as possible. These games often involve weapons, hordes of enemies and health packs. Generally, the longer a player survives, the better items and resources they will receive. It is also considered more of a success, the longer the player survives.

This type of game is currently one of the most popular and a lot of the enjoyment comes from the simple premise and being able to compare your survival time against friends in a competitive matter. The simplicity of this type of game is why I chose to develop one, as I primarily wanted my game to be enjoyable to play and therefore enjoyable to create while learning a lot of useful skills in game development.

I believed that using the consistently popular “Survival shooter” type game as my initial basis and using a premise as absurd as flipping the human-monster hero-enemy

relationship usually found in these games, that I could make it possible to create a game based in a world full of fun and absurdly chaotic features.

Aims

The aim of this project was to create a fully functional 3D survival game. The game will contain several aspects which feature heavily in most modern games. The game will allow the player to explore and fight enemies. The Game includes a large map that the player can explore.

The aim of the game in terms of style is the following:

- ✓ Cartoonish styled hero character.
- ✓ Cartoonish styled Map and buildings.
- ✓ Realistic styled enemy characters, providing a stark contrast to the main character and world, enforcing the sense that these characters do not belong to this world.
- ✓ The Character and Object models and assets were created or used from Mixamo, Adobe Fuse and the Epic Games marketplace.

The aim of the game in terms of functionality is the following:

- ✓ Menu & Settings screen.
- ✓ A fully fleshed out 3D map.
- ✓ Main character that can run, jump, aim down sights, shoot and take damage.
- ✓ Enemy characters that can detect the main character, run, chase, cause damage and take damage.
- ✓ Character health system, character's will be destroyed when their health becomes 0 or lower.
- ✓ Gun & Projectile functionality, the gun object will be able to spawn the projectile object, which will cause damage when overlapped with enemy characters.

Technologies

Hardware

The player will be able to play the game using either a standard keyboard and mouse or by using a Controller.

Standard Computer Keyboard & Mouse

It will be possible to play this game on a PC through a standard keyboard and mouse. The player can look around with the mouse, use right button to aim down sights and shoot with the left button. They will be able to move using either the W, A, S and D keys or the direction keys. They will also be able to jump using the space bar.

Controller (PlayStation/Xbox)

Alternatively, the player can use a controller instead of a keyboard and mouse if they choose to, this involves setting up a controller to their PC if they do not already have one set up. The player will be able to move using the left analog stick or direction buttons to move, the right analog stick will allow the player to look around, the left trigger will allow the player to look down the sights and the right trigger allows the player to shoot. The bottom button on the right (A on Xbox, X on PlayStation) will be the set jump button on the controller

Software

Software being used to create this project is Unreal Engine 4, the Epic Games Marketplace, Mixamo (& Adobe Fuse) and Maya

Unreal Engine 4

Unreal Engine 4 is the latest version of the Epic Games cross platform game engine. It was primarily used to create first-person games, but has since been updated and been successfully used for developing a variety of types of games.

In March of 2014, Epic Games released Unreal Engine 4, and all its tools, features and complete C++ source code, to developers willing to pay a monthly subscription fee, plus 5% of gross revenue resulting from the sales of anything created using Unreal Engine. In September of 2014, Epic Games released Unreal Engine to schools and universities for free. And as of March 2015, Unreal Engine has been available to everyone for free and will only cost a 5% royalties fee on any successful product developed in Unreal Engine and sold, this also only becomes active after the first $3000 per game per calendar quarter. These have been important moments in the engines history as it has given many developers the opportunity to begin learning how to develop games. It has become a very popular tool used by game developers today.

I will be using it for level design and for setting up the mechanics of the game and characters,

Epic Games Marketplace

The Epic Games market place is an asset store where resources can be bought or sold by developers. These resources can be used in Unreal Engine and can be textures, materials, characters, objects or many other things. This is a very important asset because without this, most objects in a game would have to be created individually and brought into the game. This would be very time consuming, especially given the time frame available for this project. The items available in this market place have been invaluable to the progress of this project.

I will be using this for textures and materials so that I can focus more on the functions of the game.

Mixamo

Mixamo is a free online service using Adobe. It allows characters to be created and rigged in Mixamo Fuse, complete with a default animation blueprint to allow the character to run, jump, aim and perform an array of movements. This is an extremely useful tool as it makes creating the initial setup for a character much simpler and quicker process.

I will be using this, along with the Adobe Fuse features for creating and rigging my characters.

Maya

Maya is a 3D model creation software tool. It can be used to create and edit models which can then be exported and used easily in Unreal Engine. It is also possible to import models from other sources such as Mixamo, allowing for some minor changes to be made to make models look more suitable with one another.

I will be using this for any further editing of models I may wish to do.

System

Requirements

All requirements should be verifiable. For example, experienced controllers shall be able to use all the system functions after a total of two hours training. After this training, the average number of errors made by experienced users shall not exceed two per day.

Functional requirements

Requirement 1 <Play Game>

Description & Priority

The first thing a user will see is the game menu. From here the user will be able to enter the game. This requirement is essential to the system as it is how a user loads the game level and gives control of the main character to the player.

Use Case

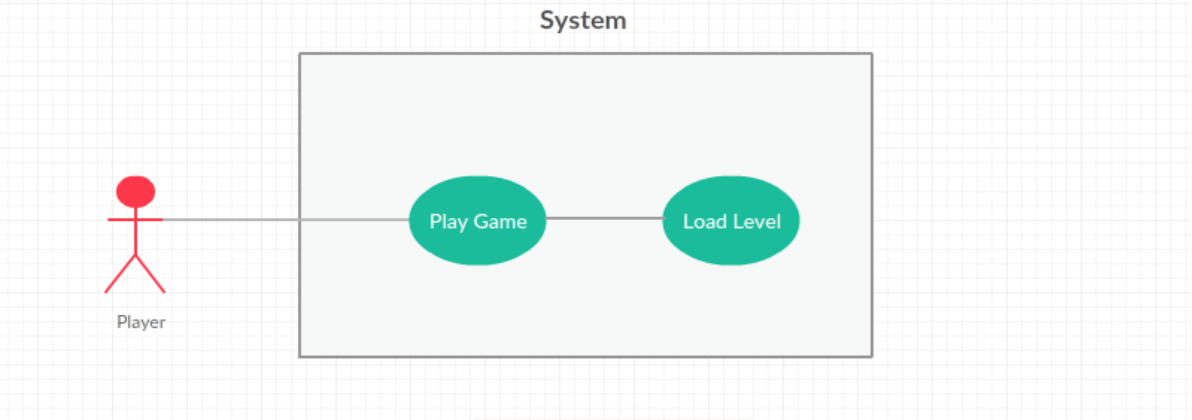
Scope

The scope of this use case is to allow players to start a new game.

Description

This use case describes the process of starting a 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 level.

Alternate flow

A1: Unexpected Error.

Termination

The system presents the first level to the user.

Post condition

The system goes into a wait state.

Requirement 2 <Quit Game>

Description & Priority

Allows a player to quit out of the game. This is essential as it allows the player to end a game session.

Use Case

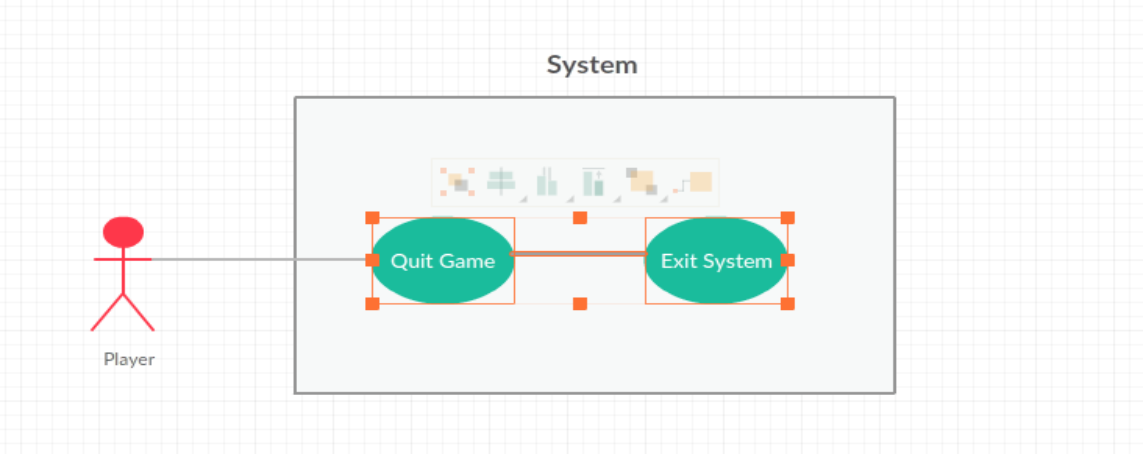
Scope

The scope of this use case is to allow a player to quit and leave the game.

Description

This use case describes the process by which a player can leave the game.

Use Case Diagram



Flow Description

Precondition

The system is in initialisation mode.

Activation

This use case starts when a player wishes to leave the game.

Main flow

1. The player chooses the Quit option.
2. The player quits the game.
3. The system exits the game.

Alternate flow

A1: Unexpected Error

Termination

The system returns the player to the initial game menu screen.

Post condition

The system goes into a wait state.

Non-Functional Requirements

Performance/Response time requirement

The minimum frame rate must be thirty frames per second. The average frame rate must be 60 frames per second on the user's machine. The average response time between click and reaction must be less than 0.5 seconds. The maximum response time between click and reaction must be two seconds. The user can personally modify the graphics settings to ensure the game runs optimally.

Availability requirement

The game must be able to run on windows 7 or higher. And should be available to a player whenever they wish.

Recovery requirement

The game will be able to recover all saved game data stored in the event of the system crashing. This will ensure that in the event of a crash, a player will be able to continue their game from a save point rather than being made to start from the very beginning.

Reliability requirement

The game will be available to the user always once the game in downloaded and installed on the user's machine.

Maintainability requirement

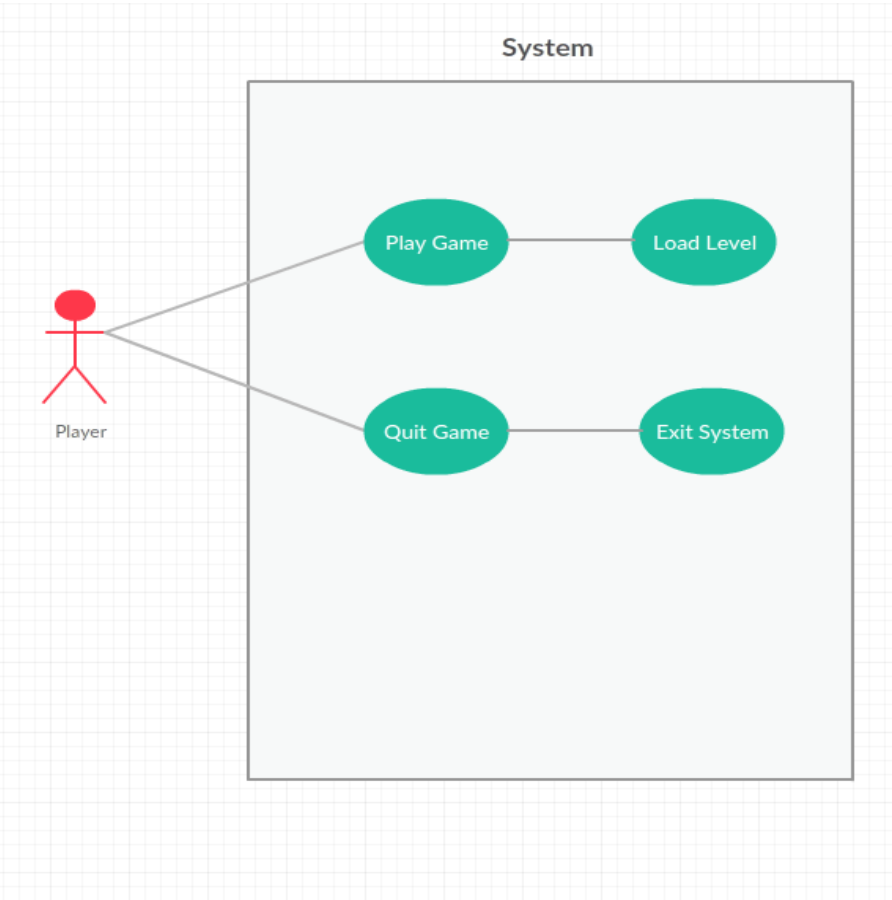
The game will be supported after launch for an indefinite period. Any issues or bugs that arise after the games launch will be worked on and a patch fixing these issues sent to players as soon as possible.

Portability requirement

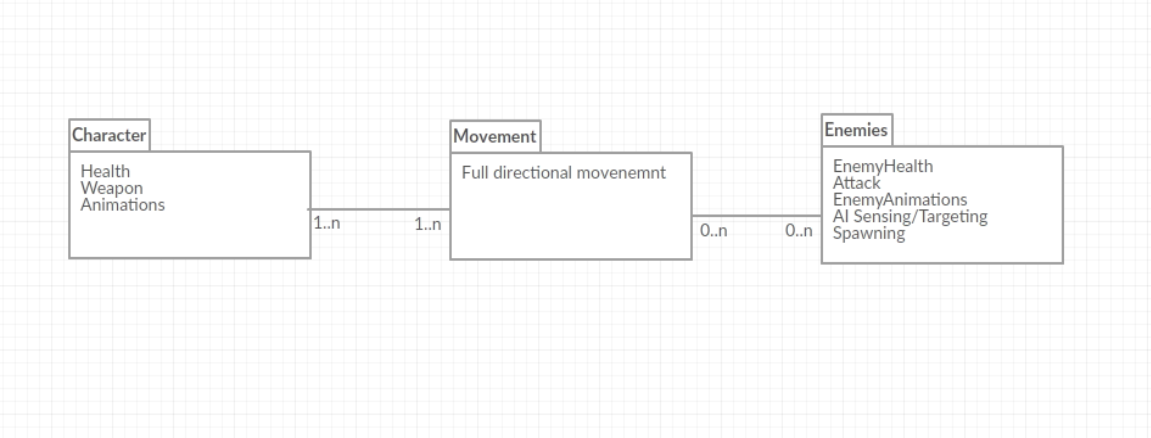
The game will be available to users on any PC or laptop they choose, once the game is stored on it.

Design and Architecture

Use Case Diagram



Class Diagram



I feel like this architecture is well suited to the game I am creating as it covers all of the core components that I want to implement while also keeping it simple and clear. The first class is the 'Character', this contains and updates information about the character, including their current 'Health' and their weapon. We also have the 'Movement' class,

our character and enemies will be able to move in any direction. It will also access animations that will correctly animate the character based on what the player is doing.

The game will have enemies involved, and these characters will use the 'Enemies' and class to determine how and when these enemies can sense and chase/attack the player. It will also determine when the enemy should be harming the player and when to play animations such as the run, attack or death animation.

System Evolution

Over time I wish to add new levels and implement new aspects to the game. I would like to add different weapons capable of different damage amounts.

I would like to create more levels and enemies, each with significant differences to the original level and enemy character, I'd also like to add difficulty options in future to make the game more inclusive and engaging.

I hope to eventually make the game multiplayer in the future, with a Co-Op mode.

I also hope to make it possible for the main character to interact with the environments much more.

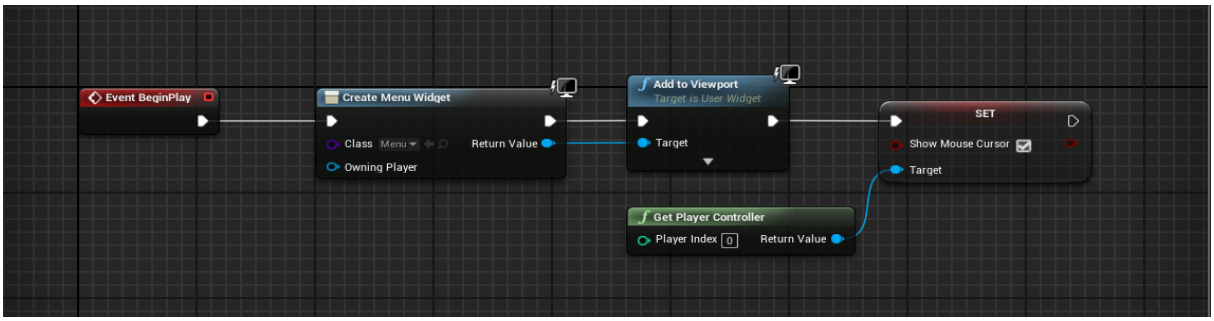
Along with these possible additions, I would like to update and improve on the original project regularly over time.

Implementation

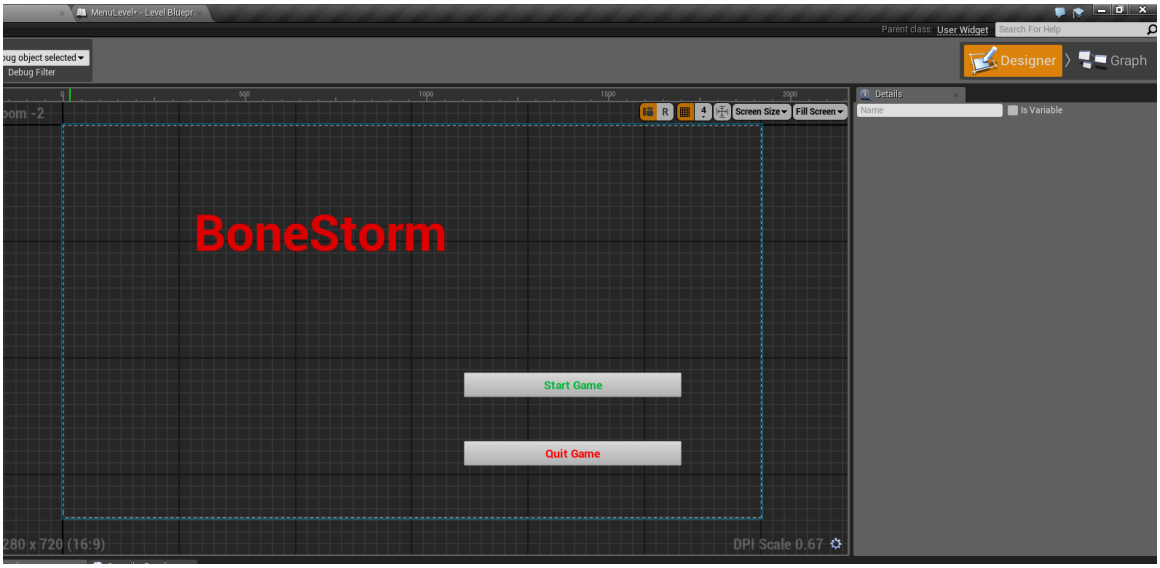
Implementation is where this project became difficult, as a beginner using Unreal Engine 4, the learning curve early on was steep. I found early on that the best way to approach this was to break the game down to each function and focus on completing one at a time.
The following will detail of how each function and feature was made possible using screenshots of the blueprints for reference.

Menu Level Implementation

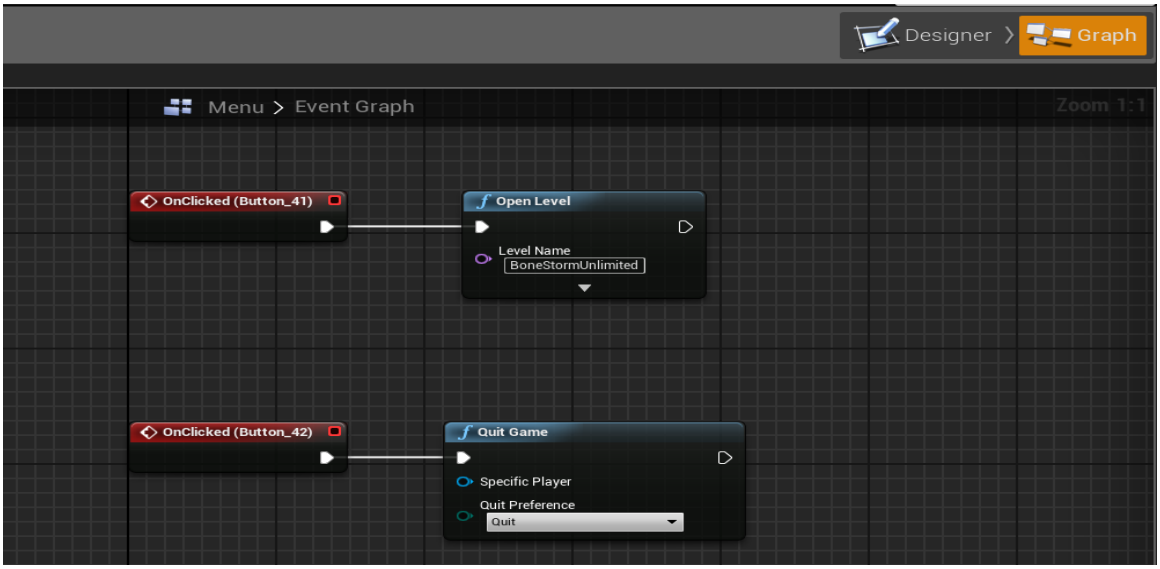
The menu level in this game is our default menu and launch point to the game. I created the menu by first, creating a blank level. Then I created a widget blueprint named Menu. Once that was created, I went to the MenuLevel blueprint and used "Create Menu Widget" to select the menu widget blueprint and add to viewport.



In the menu widget blueprint I was able to drag and drop the layout of what the menu level would then show. Including a title and buttons I could then add functionality to.



I then switched into the "graph" section and added functionality to the "Start Game" and "Quit Game" buttons. I used an Open Level function on an onClick event to open the level titles BoneStormUnlimited, which the name of our main level.

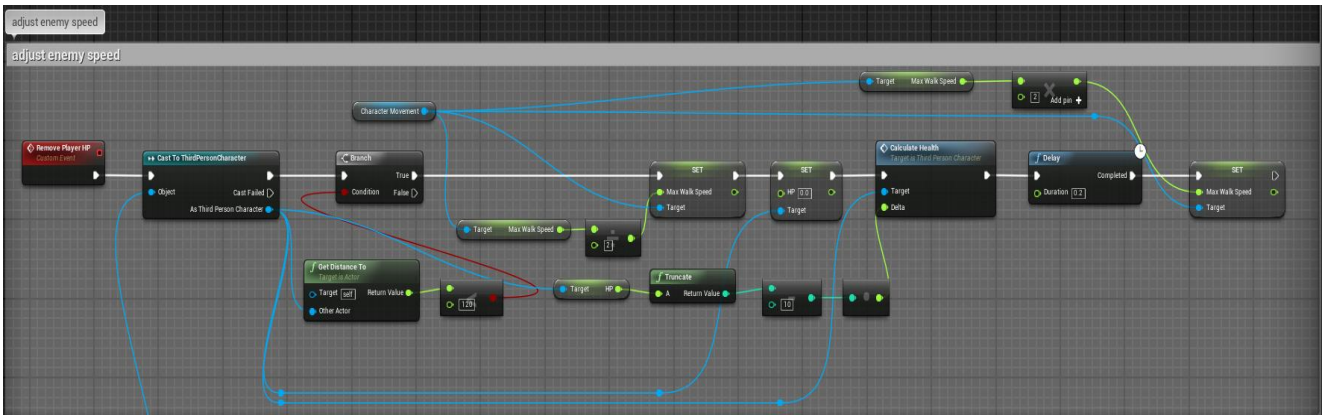


AI Enemies Implementation

Our enemies in this game use Artificial Intelligence to detect, chase and attack our main character, I will discuss their programming, along with the EnemySpawner which determines when to spawn more enemies into the map and the BlendSpace that animates the enemy characters.

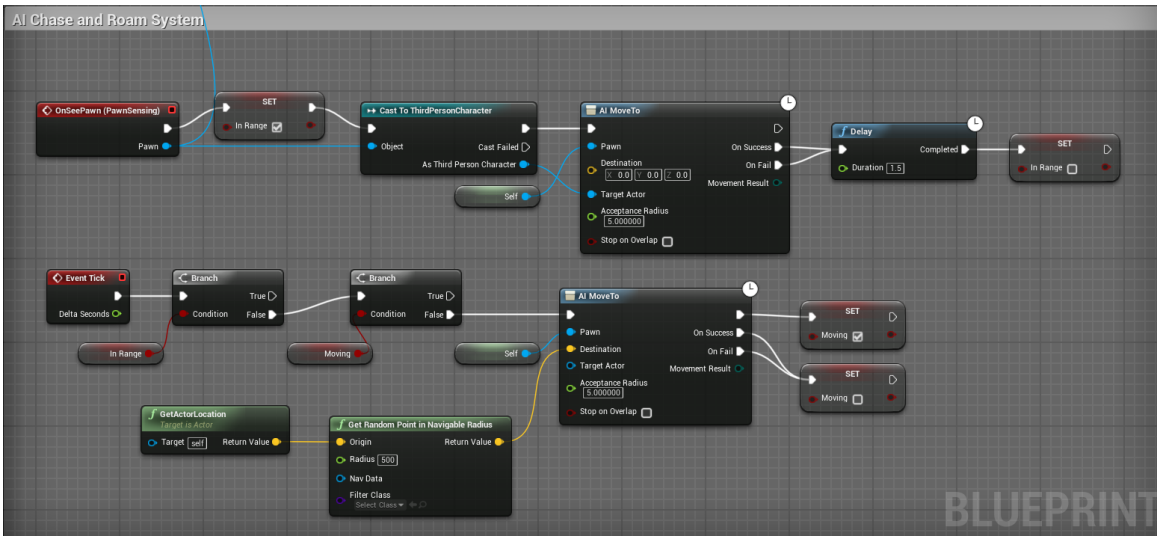
Adjust Enemy Speed

In the AI blueprint, I can control the speed of the enemy characters, while enemies are close enough to the main character to damage the player, the speed of the enemy is halved, this is to prevent the AI from being animated as running on the spot while close to the main character, there is also attack animations we will discuss that makes sure that the AI enemy is always animated as it should be. Adjusting enemy speed is linked heavily to the Chase & Roam functionality that I will discuss next.

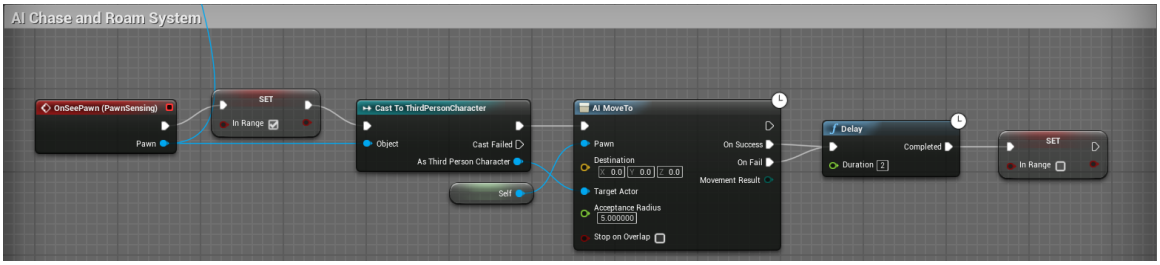


AI Chase and Roam System

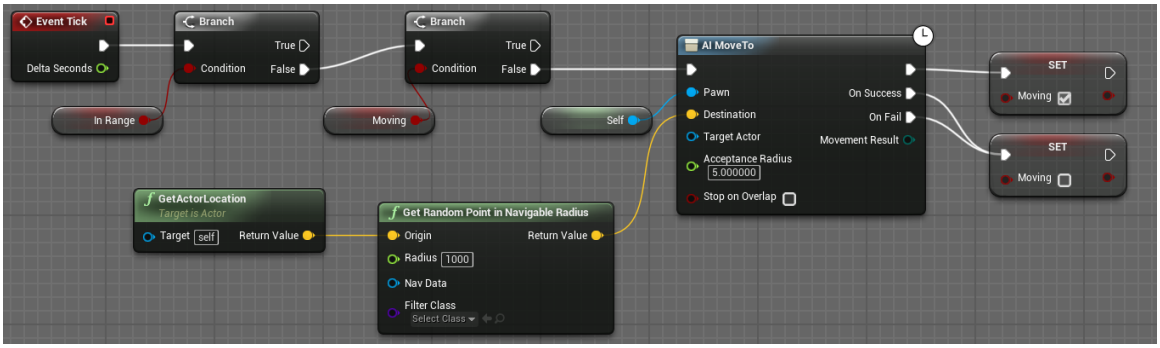
The AI Chase and Roam system I implemented can be broken down into two parts, when an enemy can see the main character and when he cannot. The Chase and Roam system defines how the AI should behave during both events.



When AI enemies can see the player, and are in range, they will move towards the player's location and continue to do so unless the player can get away from the chasing enemies enough to be no longer In Range.

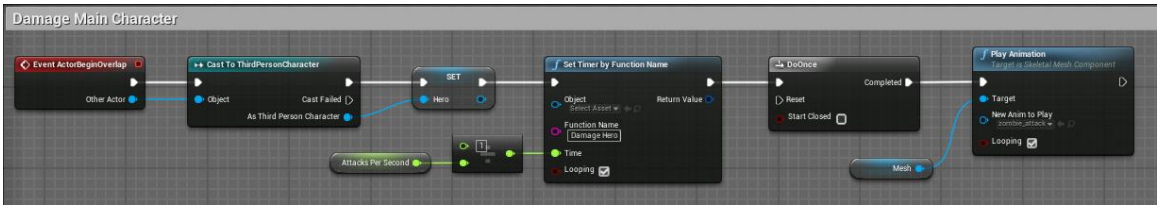


While enemies are not in range or are unable to see the main character, an event tick will regularly ask if the enemy is moving, and if it is not, then it will get the enemies location and a random location on the map that the AI enemy will be sent to, while the Enemy is moving, the event tick will not find another location to send it towards, but when the enemy either fails to reach the location or has made it, the process is repeated. Enemies will be taken out of this process if they detect the player, in which case the enemies will switch into the Chase half of this system.

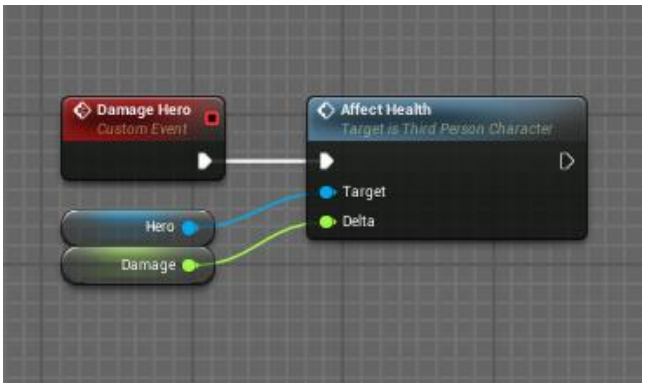


Damage MainCharacter

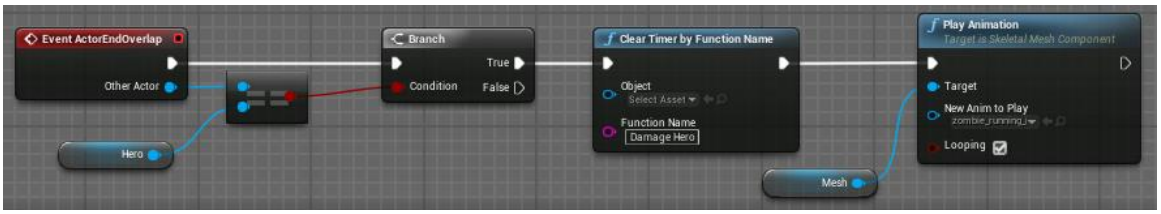
The AI enemies needed to be able to damage the main character, to accomplish this, I gave the main character a health variable, then I gave the AI characters a Collision box, when this collision box overlaps with the main character, it causes the Damage variable value to be taken form the main characters Health value.



While the overlap is taking place, the Damage Hero event takes place and the amount set in the Damage variable is taken from the player's health. Once the overlap is no longer occurring, Damage Hero stops taking place and the deductions are no longer made from the main character's health. How many time damage is deducted is set in the Attacks Per Second variable above.

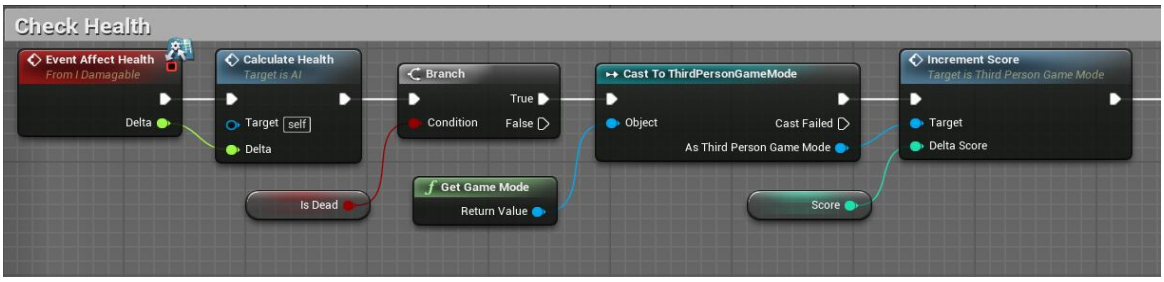


When the overlap stops taking place, the Damage Hero event stops and the player is no longer effected, without this, the player would continue to take damage even if they escape from enemies.

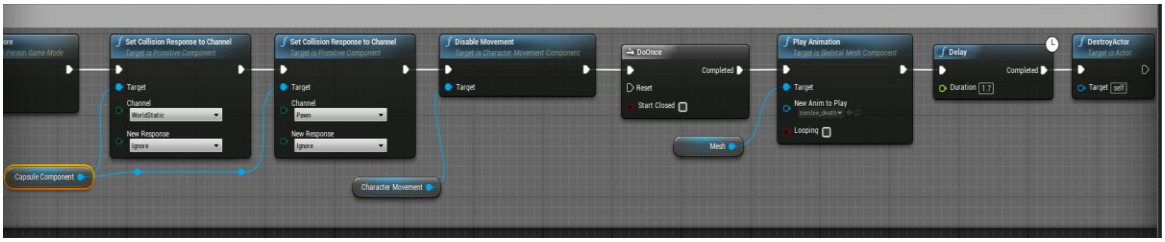


Check Health

The Check Health functionality for the enemies, regularly checks the enemy's health and whether IsDead is true. The enemy health is affected by our projectile class and if IsDead is true, the value set in the Score variable is added to the players score which is set up in our HUD class.



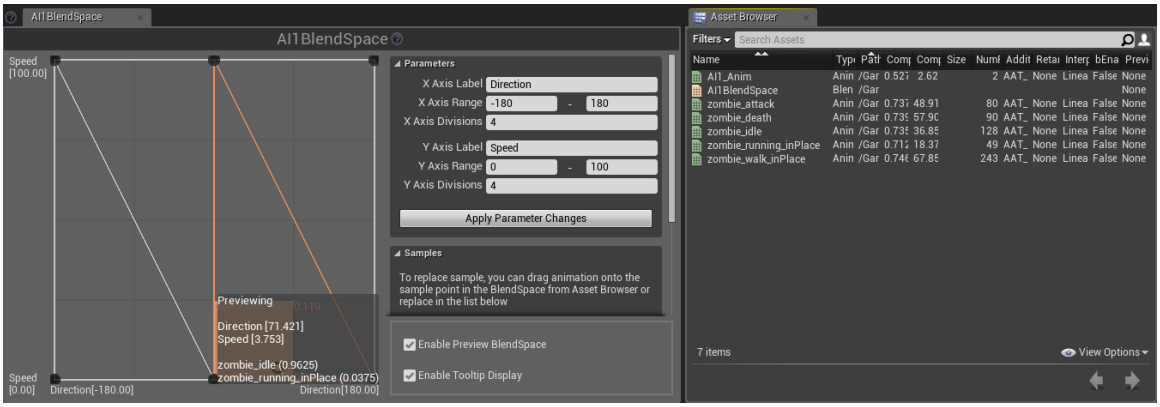
But it then continues, it then disables collision functions and movement so that enemies don't continue to chase or damage the player during the death animation that plays following the enemy's death, the enemy is then destroyed and removed from the game following the delay that allows the death animation to play.



When enemy characters are killed, the EnemySpawner spawn's new enemies using the GameMode. How many enemies can be spawned per second until the map contains the amount set in Max Enemies, is defined in the GameMode.

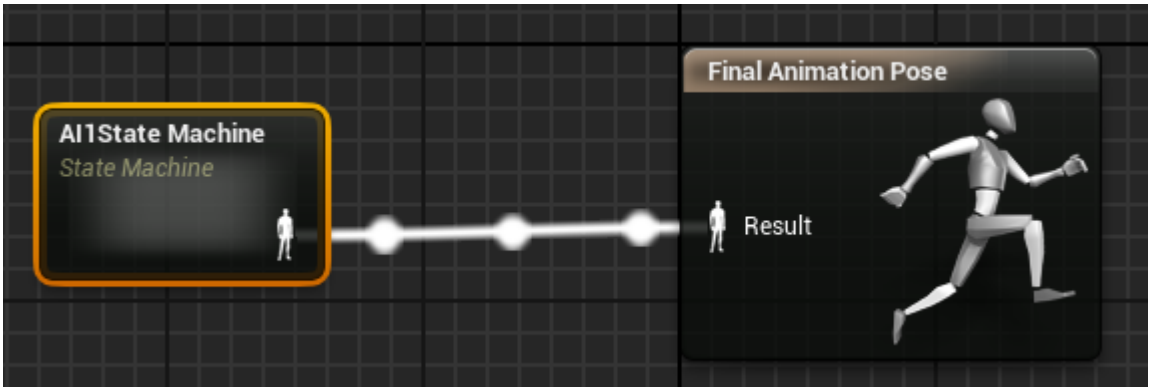
BlendSpace

To create the enemy's animations, I uploaded my enemy character that I had created to Mixamo, there I used Mixamo to add animation assets to my character. When I imported this into Unreal Engine I then created a BlendSpace for the AI character, setting the X axis as Direction giving the character full 360 degree movement, and the Y axis as Speed from 0 to 100, I was able to add the appropriate animations from the asset browser by placing the animations at the correct positions in the BlendSpace, the this then blends the animations when the character is in-between points.

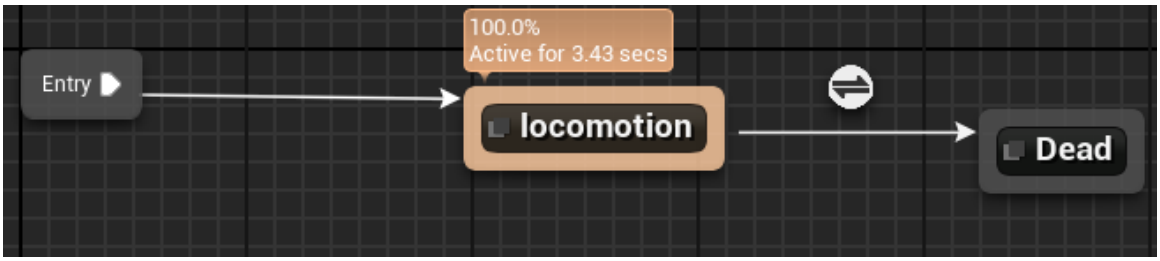


AI AnimBP

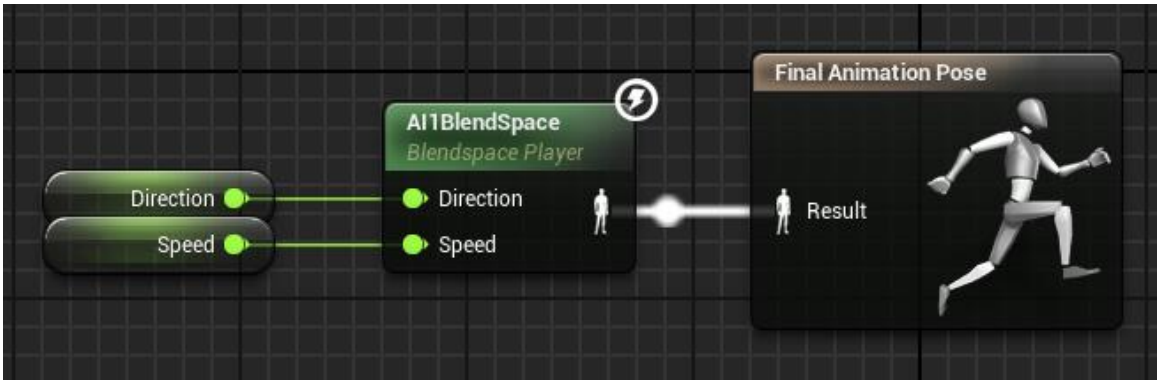
To get this blend space to take effect, I then created an Animation Blueprint that would power my enemy character. In this blueprint, I created a state machine for the enemy.



In the state machine, I created two states, Locomotion and Dead.



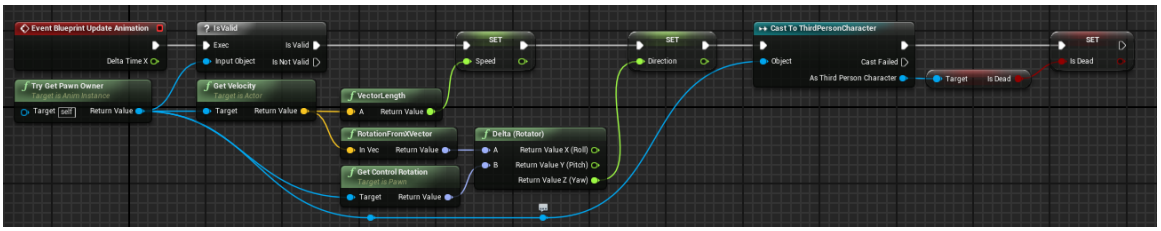
In our main state, Locomotion, we have our BlendSpace takes the enemy characters Direction and Speed and powers the enemy's animations.



In our dead state, the death animation is played.

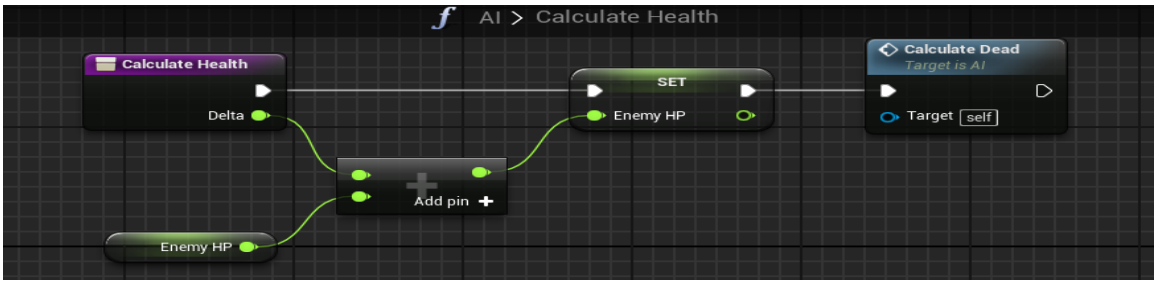
The transition between Locomotion and Dead decides when the state should change from the Locomotion state to Dead state. I set it so that when IsDead is true, the transition takes place.

All of these values for speed and direction are received and set using the following in the event graph.



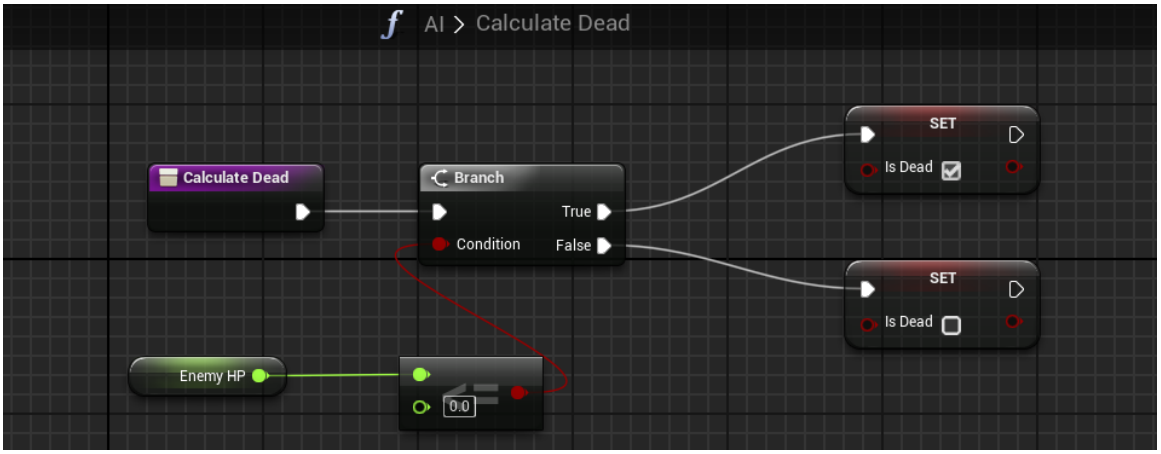
Calculate Health

The enemy character calculates and sets the health in the CalculateHealth script. This is connected to the CalculateDead script.

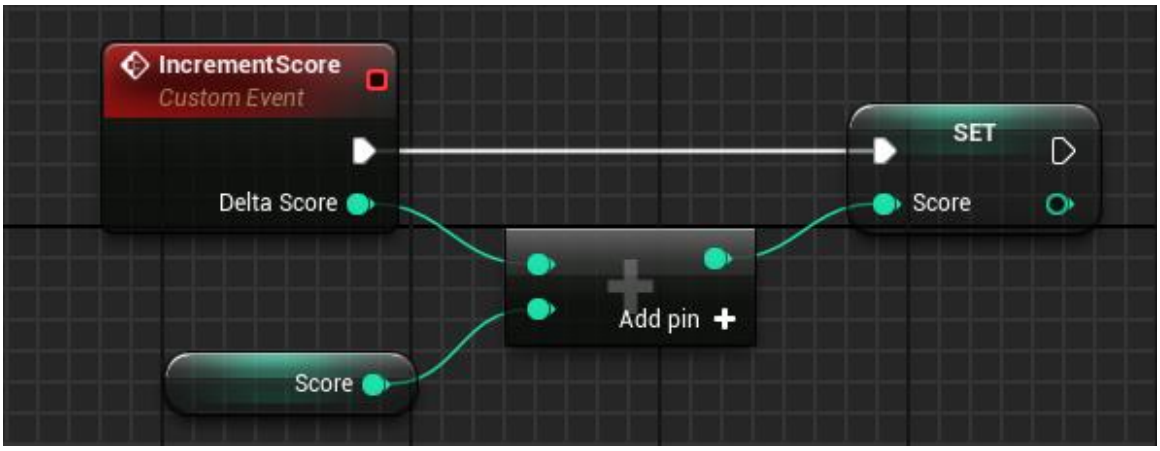


Calculate Dead

The CalculateDead script is what sets IsDead to true, if the EnemyHP is equal to or less than 0, otherwise IsDead is set to not true.



When IsDead is true, the GameMode has the following, where a value for score is added to the score the player sees as part of the HUD.

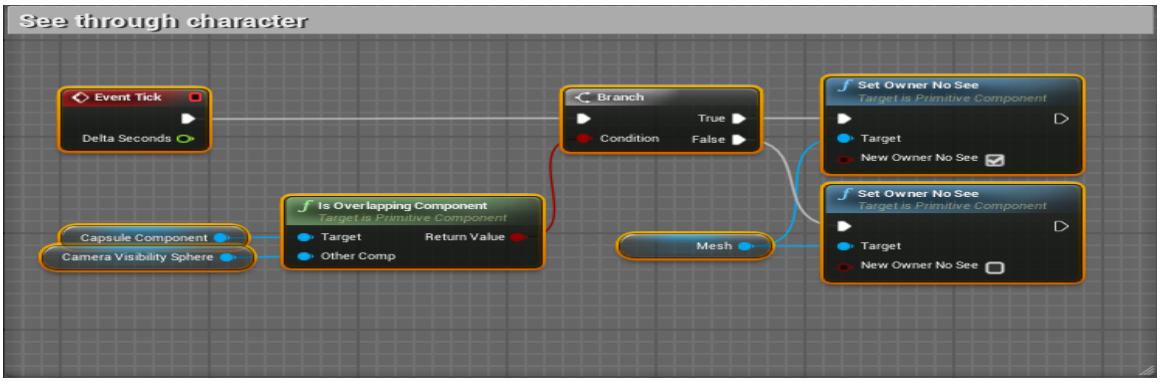


ThirdPersonCharacter (Main Character)

Our main character has the most functionality. I will be discussing the several aspects I have programmed in the ThirdPersonCharacter, including Health, Shooting, Respawning, Aiming Down Sights and Orienting Character.

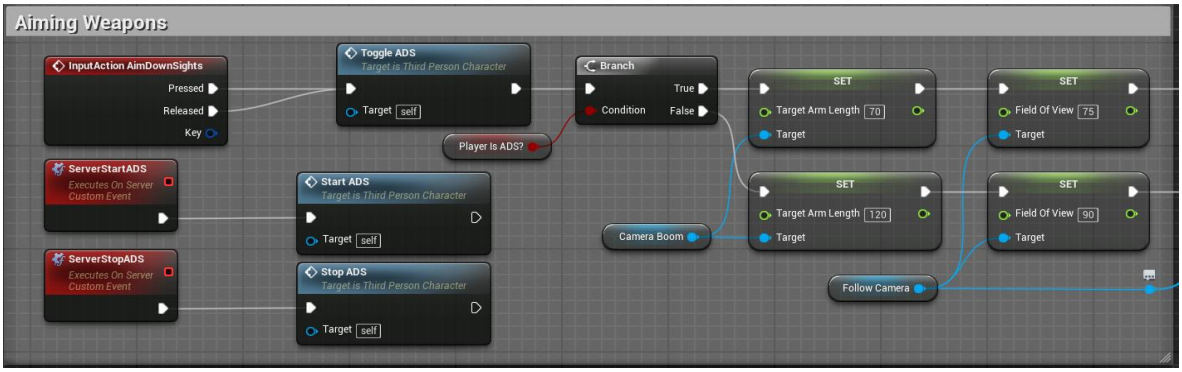
See Through Character

The first thing I changed about the default ThirdPersonCharacter, was to add a Camera Visibility Sphere to the follow camera and added an overlapping component so that the camera will not overlap with walls or with the character when the camera gets too close, which would look very glitchy.

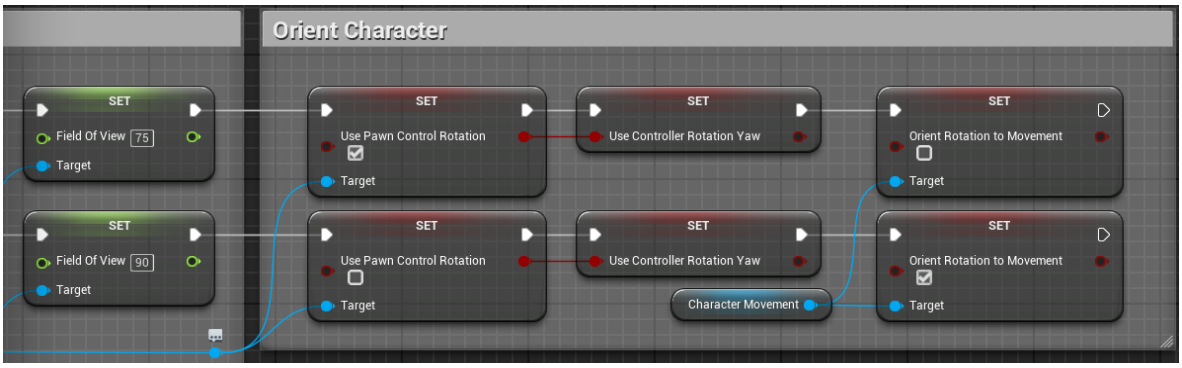


Aiming Weapons & Orienting Character

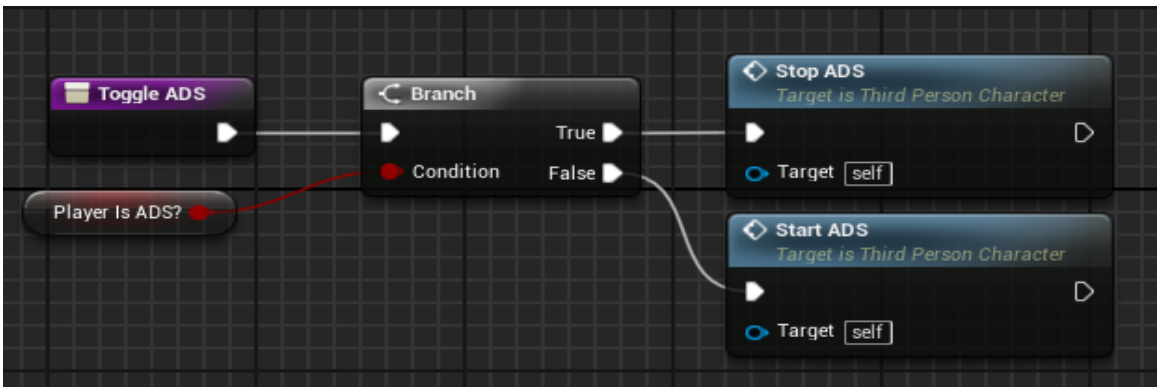
For Aiming down sights, I created scripts for Toggle ADS (aiming down sights), Start ADs and Stop ADS, which I will discuss shortly, Player Is ADS? Is a Boolean that is set to true while the appropriate button is pressed (right mouse button / left trigger). The Camera Boom is set to shorten the Arm Length and lessen the Field of View, when Player Is ADS in not true the Arm Length and Field of View is set to the standard values.



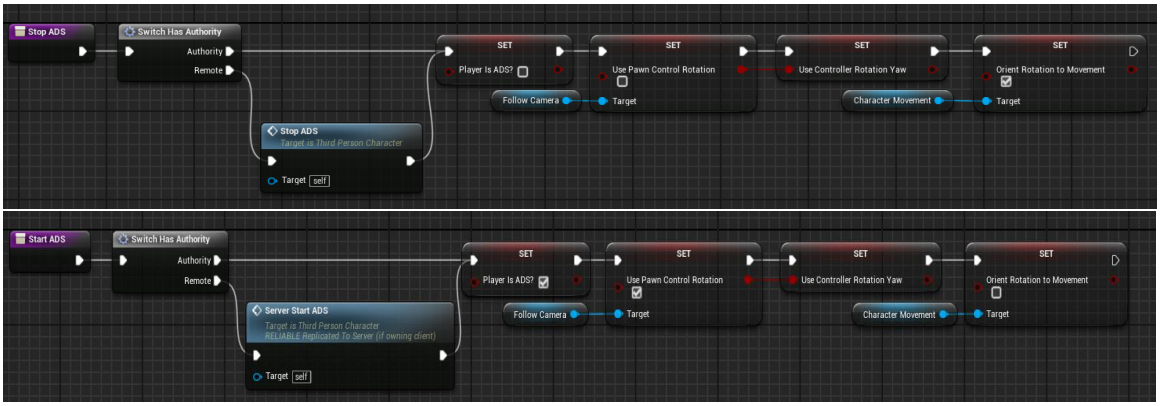
This then continues into the orient character nodes, when the character is aiming down sights, the movement is set to use Control Rotation, this allows us to move one way while aiming another so that we don't have to be still or running towards enemies to aim at them.



The Scripts for ToggleADS, StartADS and StopADS are what sets the Player is ADS Boolean to either true or false. Every time the button for aiming down sights is either pressed or released, ToggleADS finds whether Player is ADS is true or false and changes it.

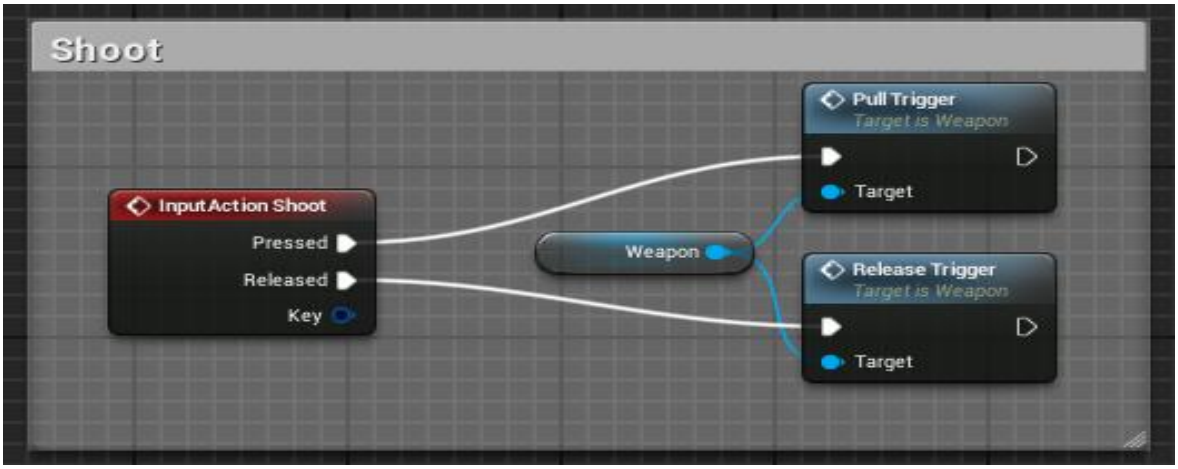


In StartADS and StopADS we can see how the characters movement and orientation is set to be different during each being set as true.



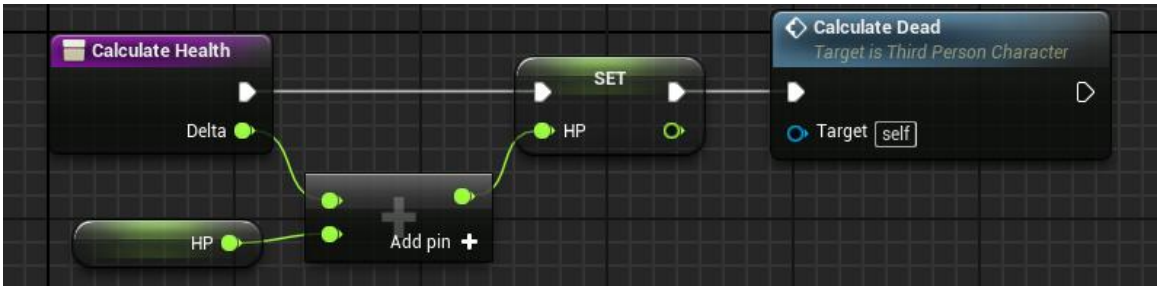
Shoot

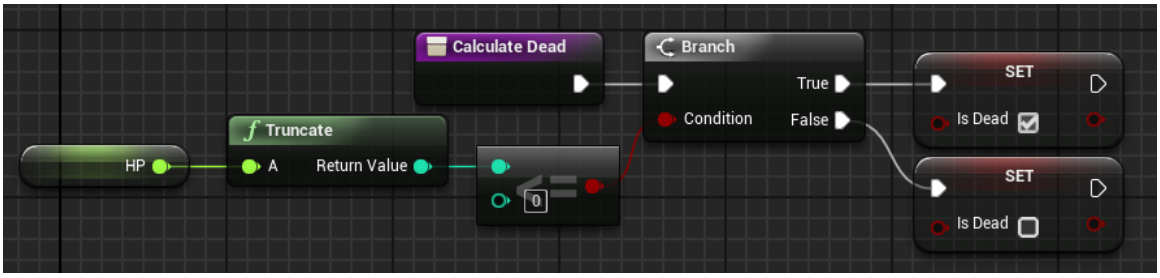
The shoot functionality is set in this script, the input action is set as the left mouse button/right trigger. The action of this is set up in the Weapon script.



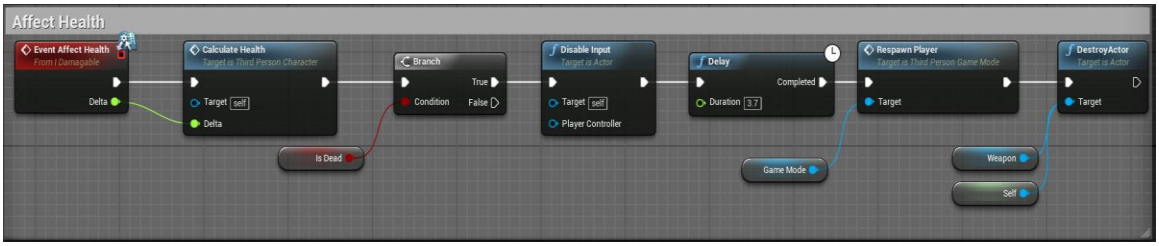
Affect Health

Just like I had for the enemy, I have also created both a calculate health and calculate dead script for our main character. These two are connected and when the players HP is equal to or below 0 IsDead is set to true.



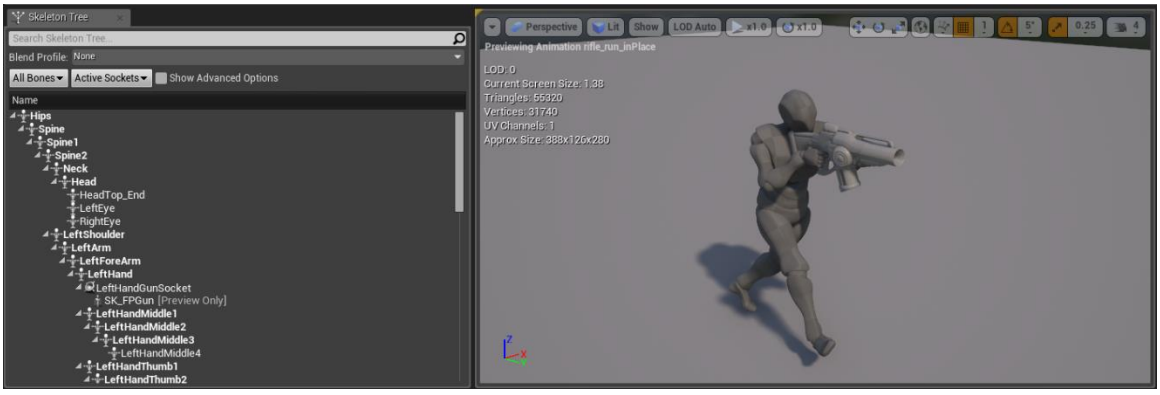


After calculating the player's health and whether or not isDead is true, I created the Affect health functionality in the main event graph for the main character. When IsDead is true, the players input is disabled, preventing the character from continuing to move, there is then a delay long enough to allow the character death animation to play before the character is respawned and the old version destroyed.

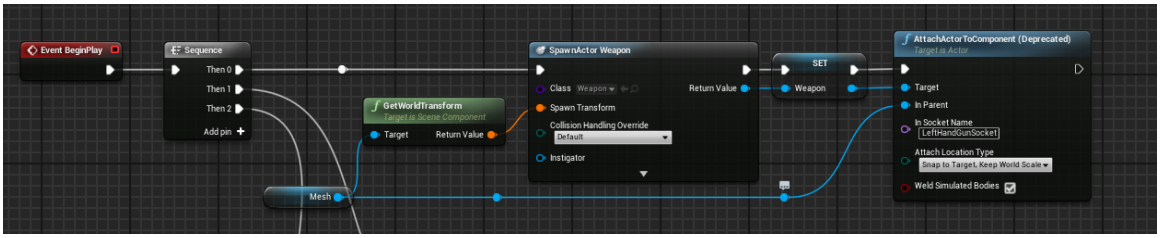


Attaching Gun

Attaching the weapon turned out to be simple enough. After creating and setting up the Weapon and Projectile objects, which will be discussed shortly, I was able to go to the character skeleton, add a socket named LeftHandGunSocket to the skeletons left hand as the right changed too much between animations making it difficult for the player to aim consistently. I was then able to rotate and position the gun to how I wanted it. This of course is just the preview and set up and would not change our character until we added the gun to the socket in our script.

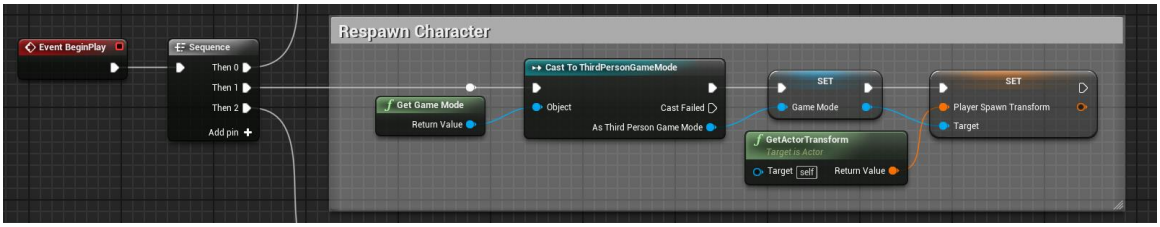


In the ThirdPersonCharacter script I then set up spawning of the weapon and attached it to the socket using an AttachActorToComponent node, where I used the LeftHandGunSocket I had created in the preview.

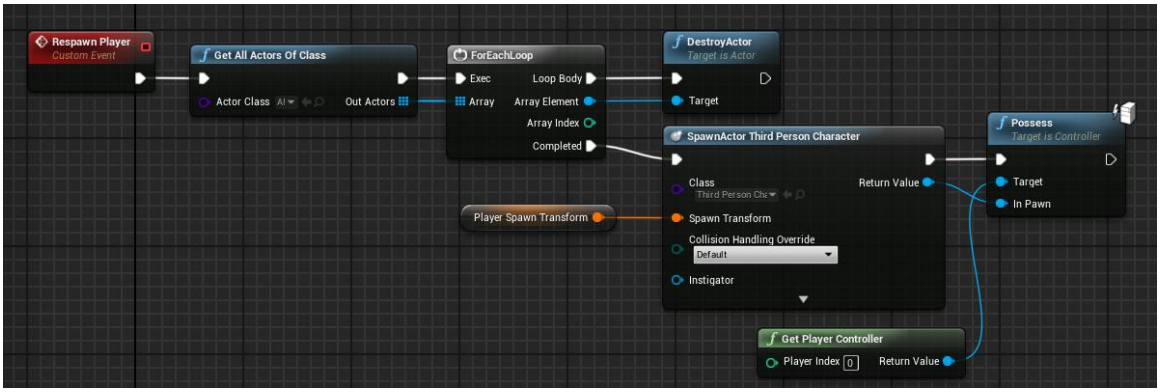


Respawn Character

In the section about Affect Health, I have shown when the player is respawned, but this is how the player is respawned. This includes the following in the ThirdPersonCharacter script, which calls from the GameMode.

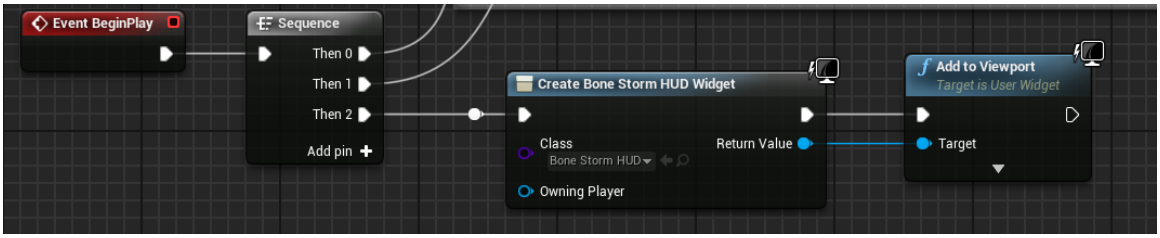


In the GameMode I have set that when the character is destroyed, a SpawnActor Third Person Character node is used to respawn the player and give the player controller to the newly spawned character.



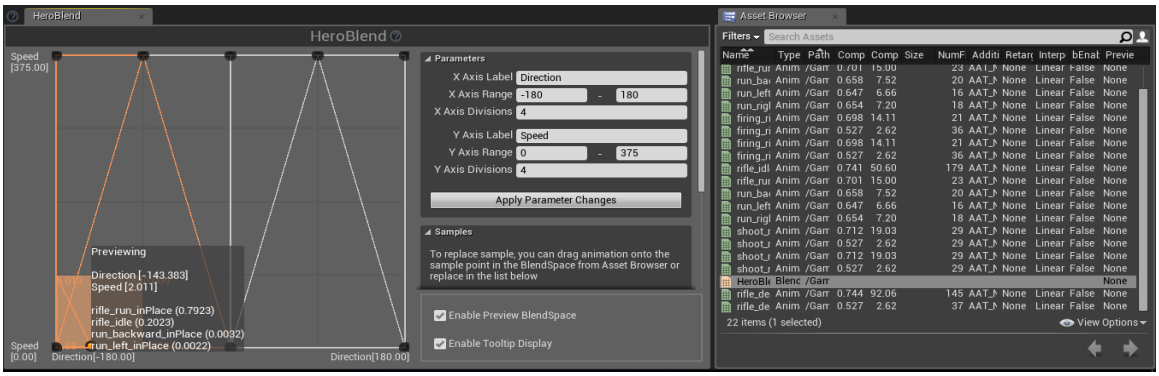
Display HUD

I created a HUD which will show the player their health and score during the game. This HUD is then added to viewport



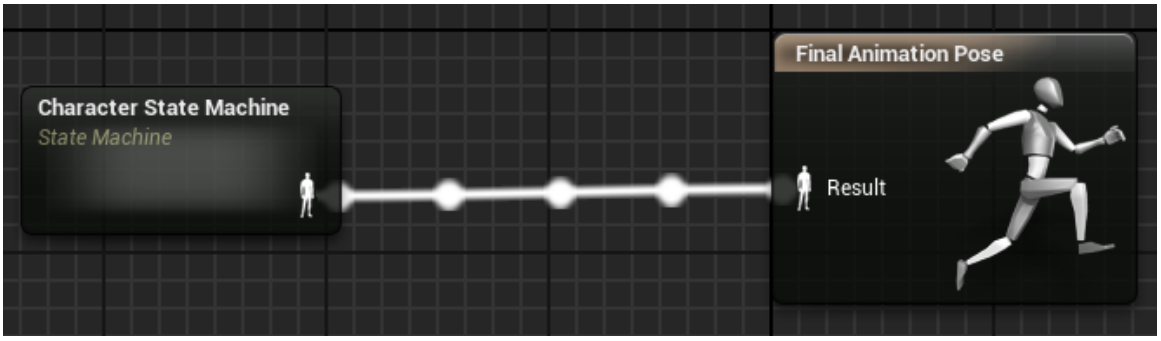
HeroBlendSpace

Much like the enemy character, I used Mixamo animations to create the character BlendSpace. Unlike the enemy character however, our hero character uses a much more varied set of movements. The Hero character needed to be able to run backwards, left, right, forward and any way in-between those movements. Largely because our main character can aim one direction while moving in another direction. The axis in this graph are Direction and Speed, the character is idle when Speed is 0. And the character is animated as running forward, left, right, backward or a mix of these depending on the direction axis.

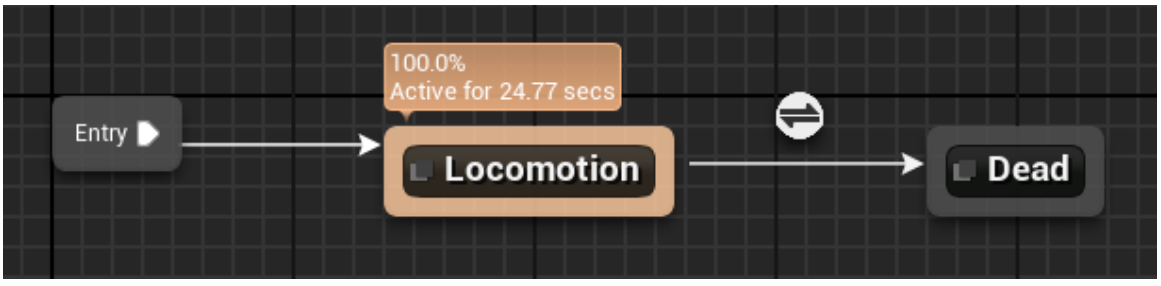


CharacterAnimBP

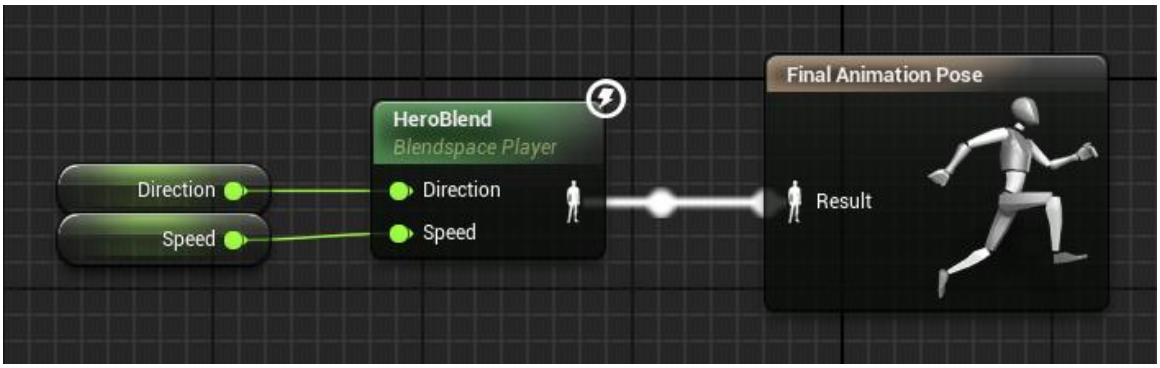
The HeroBlendSpace is then used to power the character through the CharacterAnimBP. In the CharacterAnimBP I created a Character State Machine.



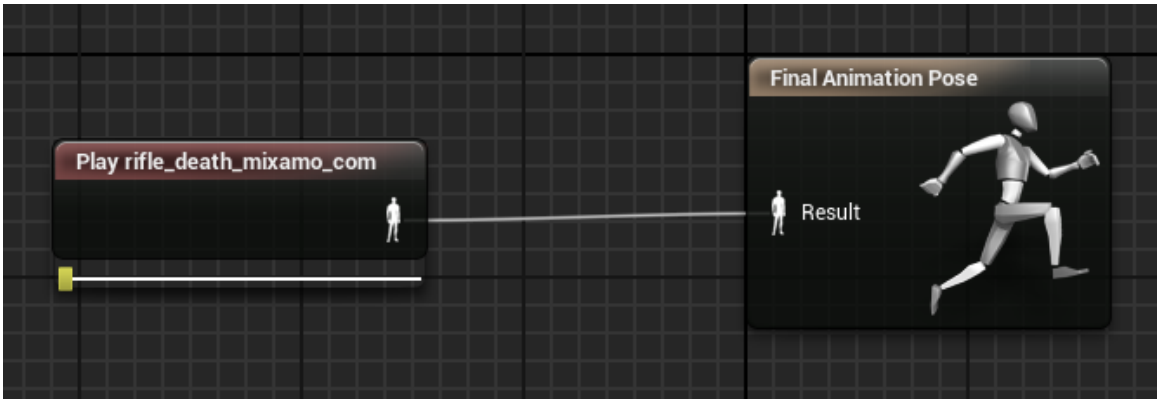
In this state machine, we have two states, Locomotion and Dead.



In the Locomotion state, Direction and Speed are taken by the HeroBlendSpace and this animates the main character while in this state.

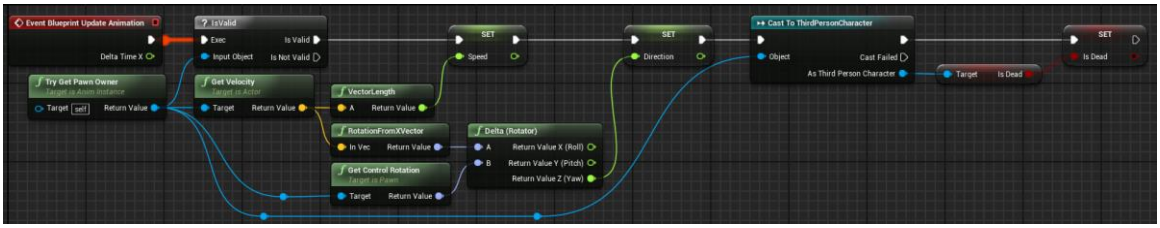


In the Dead state, the hero death animation is simply animating the main character.



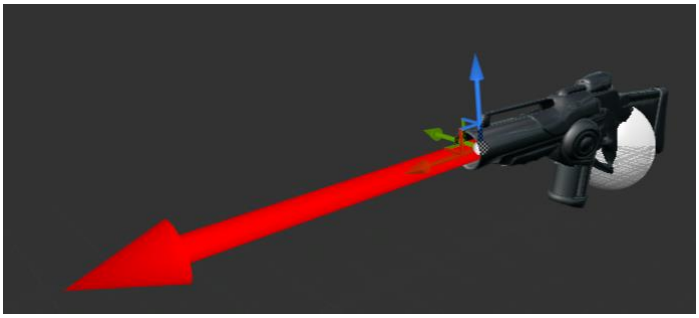
The transition between the Locomotion and Dead states is what defines when the character goes from Locomotion to Dead. The transition occurs when IsDead is true.

The values for Speed and Direction are received and set in the AnimBP's event graph.

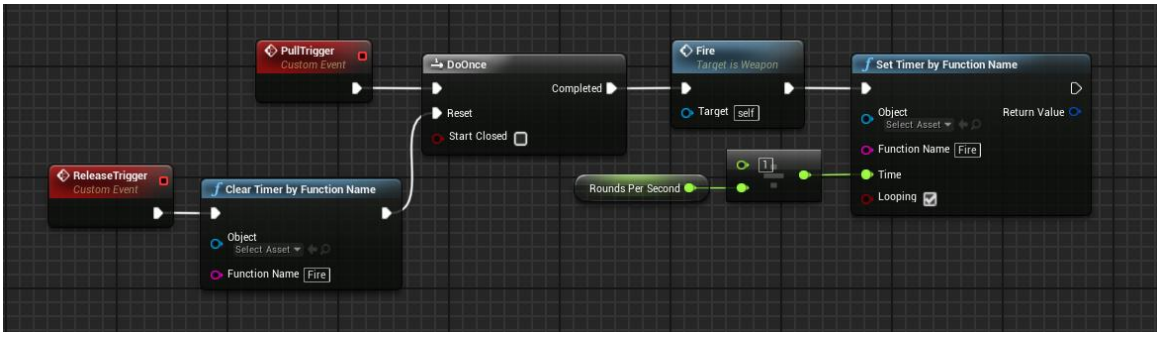


Weapon

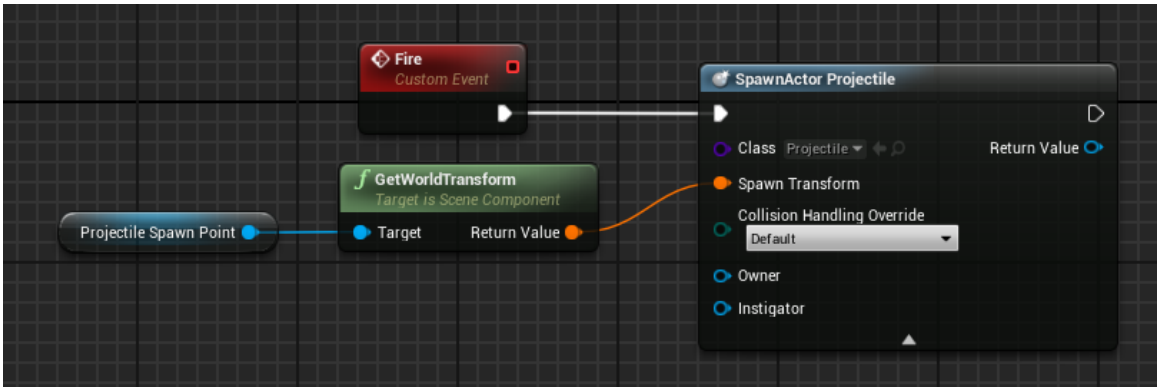
For the gun the player uses, I created an object named Weapon. I then used a gun mesh from the sample assets and changed its material to a chrome asset to give it a futuristic look that would suit the projectile it would be spawning. I also added an arrow which is the ProjectileSpawnPoint for the weapon.



In the event graph for Weapon, I created custom events for Pull and Release trigger, these events call upon the Fire function I also created and uses a timer to set how many projectiles can be spawned per second.



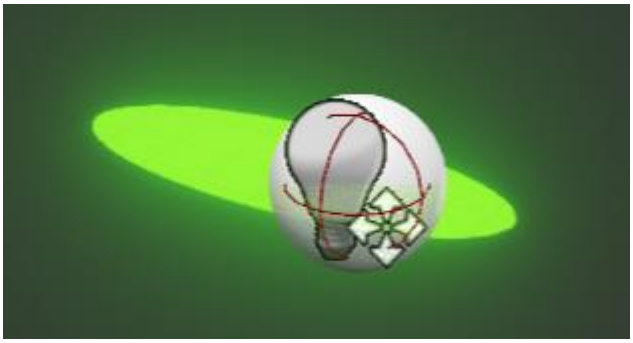
The Fire custom event I created is what spawns the Projectile object from the Projectile Spawn Point arrow that I added to the gun mesh.



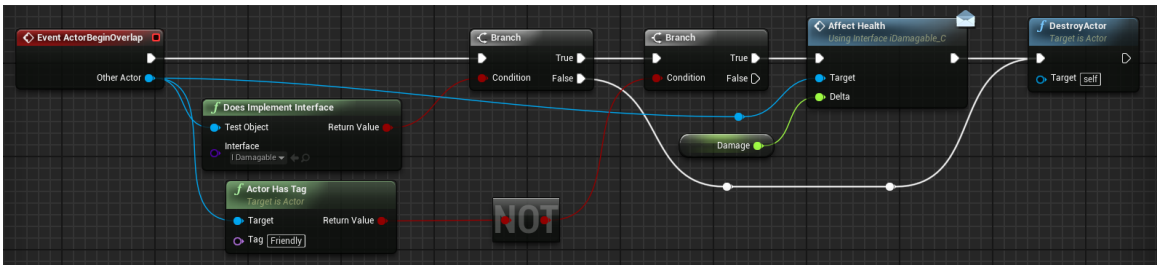
Projectile

For the projectile, I created an object named Projectile. I made the projectile long and thin and gave it a bright green colour so that it would look like a laser blast, I then added a light component to give off a pale green colour so that the projectiles that are

spawned in the game would give off just a little light to give it more of a laser effect. This effect works particularly well in the shadowed areas on the map and even causes enemies to cast shadows from this light source.

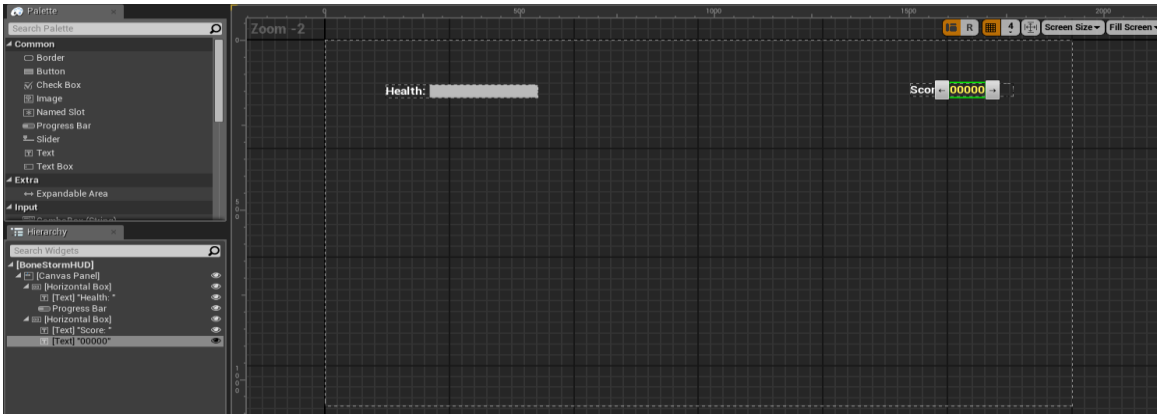


In the event graph, I set the projectile to damage characters that it overlaps with, unless the character has the tag Friendly, which I gave to the main character to prevent any self-inflicting damage. This is where the value for the damage it can do to enemy characters is set. I also created an interface called iDamagable, which contains the function Affect Health, and is what allows the projectile and to communicate to the enemy character and deduct the Damage value from then enemy's health value.

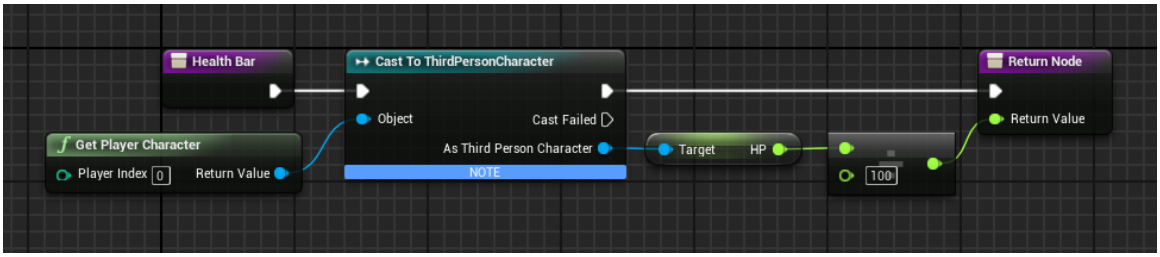


BoneStormHUD

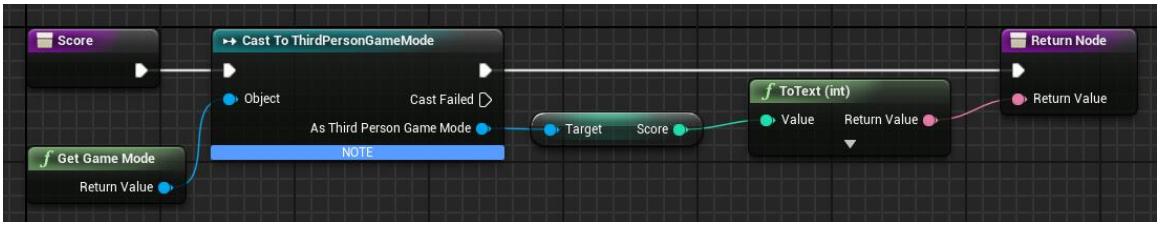
I created a Head-Up Display for this game that would show the player their Health and Score as they play the game. To do this, I created a Widget Blueprint that I named BoneStormHUD. In the designer part of the widget blueprint I added a progress bar for health and a text box for Score. I also set the progress bar to have a green colour and the Score text to have a yellow colour.



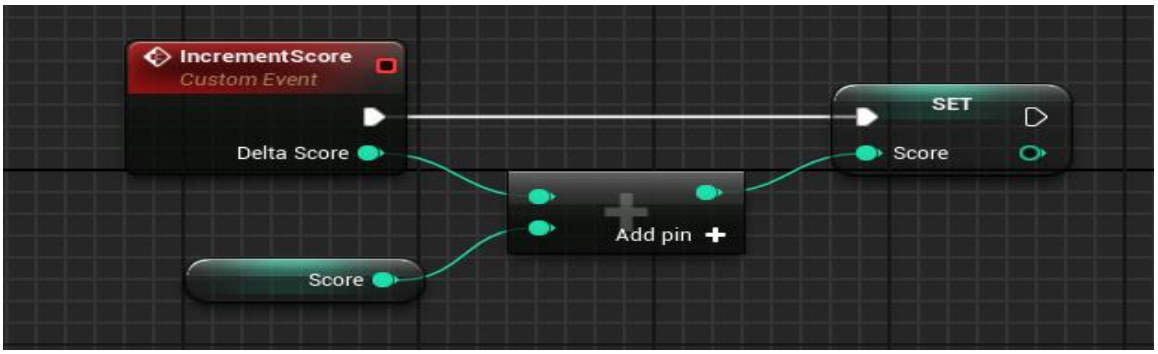
Then, in the graph section I gave the progress bar functionality by taking the character's health (HP) and dividing it by 100 as the progress bar goes from 0 to 1.



Then I added functionality to the Score text box, I set the text box to display the players score, which is taken from the GameMode.



In the GameMode, the Score that appears on the HUD is calculated by adding the Score value to the current score each time an enemy is killed.



Testing

Testing projects is essential to any good project, throughout the course of this project I regularly tested features and components to ensure all features were functional and non-disruptive to the project while implementing. Testing was carried out using a few different methods.

Unreal Engine 4 Simulation Mode

Unreal Engine 4 offers the ability to play a project in simulation mode, this is incredibly useful as after each change or addition made, it was possible to use this to see if the desired effect had been made, it also allowed me to compare different values and variations on what I was adding as I could quickly test the game with different values and variables to choose which was most fitting with the project.

Unreal Engine 4 Test Levels/Projects

While learning how to implement different features, most were originally developed and tested in alternate or duplicate levels or projects in Unreal Engine 4. This allowed me to experiment with creating functions and components in a risk-free environment. When I managed to implement a function as I wanted it to be, I was then able to add it to the BoneStorm project.

BoneStorm Beta

I regularly packaged my project in development mode, allowing me to play the latest version of my project in its current form. This allowed me to witness some minor differences that can be seen when comparing the simulation mode with the packaged version.

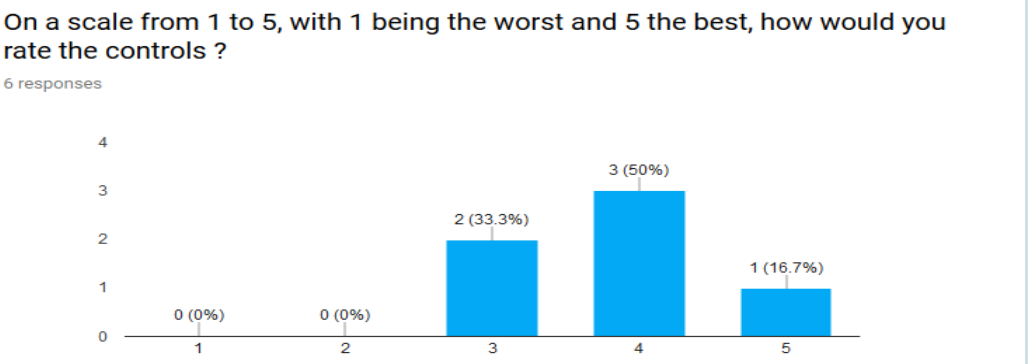
Reviews

I regularly sent the beta versions out to people, to gain their insight, thoughts and suggestions on the project. I tried to make this group of testers as varied as possible, sending the project to both fellow students/ graduates, and people who would not have a large amount of experience or knowledge of Gaming or Computing.

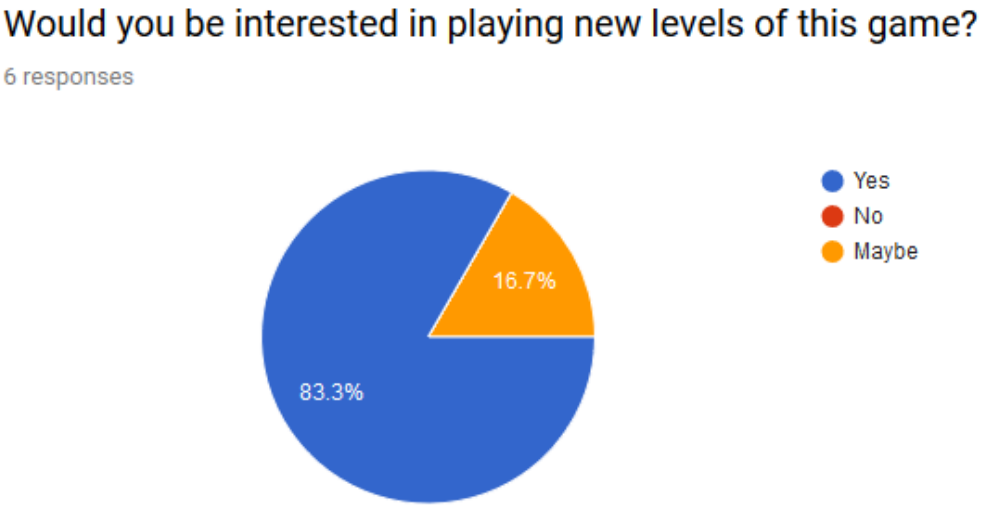
Surveys

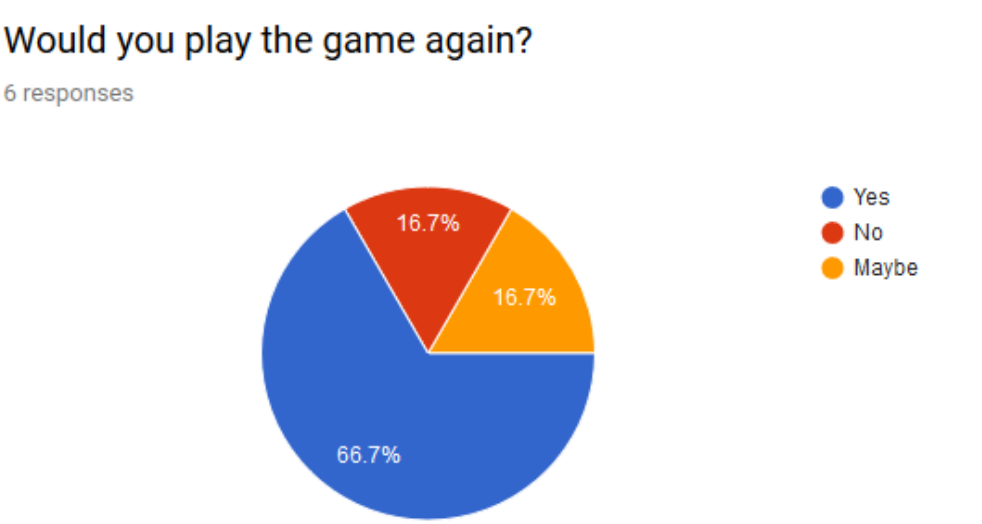
I created a survey on Google forms, along with the beta, I gave many users the survey to complete. This was a great way for me to gauge people's opinion on the project and receive honest feedback. Some interesting results from the survey for the most recent version included:

- ✓ When asked to rate the controls, between 1 and 5, over 80% of testers gave it a 3 or 4, in the optional suggestion field, many testers claimed that aiming accurately was sometimes difficult. Because of this I have narrowed the aiming function through Field of View and the Camera Arm Reach, this has made hitting targets easier already and I will continue to look at ways to improve aiming accuracy further.



- ✓ When asked if they would be interested in playing new levels of this game, over 80% said yes, whereas only 66% said they would play this game (as is) again. From this, it is clear that new levels would make a large difference to the popularity of this project.





Conclusions

Going into this project I set myself goals I wished to achieve, teaching myself to competently develop in Unreal Engine was the goal I wanted to accomplish most of all, but I also set goals for my project. I wanted my project to involve as many of the following aspects as I could make possible:

- ✓ Design a large map for characters to roam.
- ✓ Implement at least one AI agent that could sense the player character.

- ✓ Implement a fully functional weapon and projectile.
- ✓ Implement health and damage systems for all characters.
- ✓ Animate all characters. (attack/death/run/idle)
- ✓ Add audio to the game. (background music/sound effects)
- ✓ Spawn enemies and respawn hero.
- ✓ Set up a simple Score system for the player.

I believed that if I could accomplish these goals, that I would be happy with the end result of my project.

The problems encountered trying to reach these goals included:

- ✓ Getting to grips with Unreal Engine 4, I started this project as a beginner with Unreal Engine and had to learn as I went during this project. Blueprints and the User Interface were difficult to get used to.
- ✓ FBX files, starting out, importing meshes and animating characters were completely new to me and took time to understand.
- ✓ Tutorials available were generally very specific to certain types of games and were rarely for third person shooters, as first person shooters and third person RPG's are far more popular thanks to Unreal templates. It took time to learn how to alter the material covered in tutorials to better fit my own game.

Although I would have liked to add even more to my project, I consider it to be a success and managed to meet my main objectives. The feedback has been generally and clearly positive. I can say that I am very happy and proud of my final year project and the skills that I have achieved over the course of its development. I hope to continue developing and improving this project in future.

Further development or research

I received helpful feedback from my testers that I consider to be very helpful at identifying the next step. I would like to use this feedback going forward. I know where I need to improve the current project and also what other aspects or functions could become a part of this game in order to improve it in the future.

Many testers noted that aiming could have been more accurate, this is where I will try to improve next.

While I achieved all my main objectives for this project, there are a few things I had hoped to achieve if I had the time to do so. I had hoped to add items such as Health or Ammo. I had also hoped to make some extra levels. I hope to achieve these aspects in the future as I continue to develop and improve upon this project.

References

During this project, I used many different YouTube tutorial series and websites to learn and develop and my project. The following is the resources that I used.

YouTube. (2017). *Blueprint Twin Stick Shooter | v4.8 | Unreal Engine - YouTube*. [online] Available at: https://www.youtube.com/playlist?list=PLZlv_N0_O1gb5sdygbSiEU7hb0eomNLdq

YouTube. (2017). *Creating Games For Beginners Using UE4 - Unreal Engine 4 Course - YouTube*. [online] Available at: https://www.youtube.com/playlist?list=PLL0cLF8gjBpqDdMoeid6Vl5roMl6xJQGC

Unrealengine.com. (2017). *Marketplace - UE4 Marketplace*. [online] Available at: https://www.unrealengine.com/marketplace

YouTube. (2017). *Massive UE4 Tutorial Playlist - YouTube*. [online] Available at: https://www.youtube.com/playlist?list=PLZlv_N0_O1gaCL2XjKIuO7N2Pmmw9pvhE

Forums.unrealengine.com. (2017). *Unreal Engine Forums*. [online] Available at: https://forums.unrealengine.com/ [Accessed 5 May 2017]

Docs.unrealengine.com. (2017). *Unreal Engine 4 Documentation*. [online] Available at: https://docs.unrealengine.com/latest/INT/

YouTube. (2017). *Tutorials - YouTube*. [online] Available at: https://www.youtube.com/playlist?list=PLwqnA65RFUf-8xIFQYV8Zjziv4gUqjLZ_

Appendix

Project Proposal

Objectives

Personal Objectives:

My personal objectives for this project is to push myself to build on my limited experience using Unreal Engine and to learn it to a much greater extent, as game development is an area I have a huge interest in becoming involved in after completing my computing degree.

I will also gain valuable experience while learning how to use and become proficient in 3D modelling software, such as 3ds Max and other game development related software.

Development Objectives:

My objectives for the development of this project are to create a game which allows you to have full control over the movement of the main character.

The character will be modelled to be a skeleton.

You will be able to run, aim, look down the sights of a gun, control the following camera and shoot as you move around the level map.

Enemies will spawn on the map and if nearby will become aware of your character and attack.

Enemies will be modelled to look like humans.

Weapons can be used to defeat enemies in order to survive.

Background

I have been studying computing in National College of Ireland for 4 years. I've been interested in the Gaming and Multimedia stream since first looking into the possible 4th year specifications, sometime during the third year of my computing course. After reading up on what is involved in each of the specifications and speaking to friends who were in the year ahead of me, it became the clear option for me as I find it interesting and exciting. Game Development is the area I want to become involved with professionally on completion of my course. Knowing I was going to pick the Gaming and Multimedia stream, I spent a part of my summer before 4th year getting to grips with game development by using Unity. I followed some YouTube tutorials on creating some basic 2D platformer style games to get a basic understanding of the tools and set up. I then tried out both GameMaker and Unreal Engine as final year approached. I quickly decided that Unreal Engine would be my preferred option for developing my final year project. I preferred Unreal Engine to Unity and GameMaker mostly due to its capabilities and ease of use. I then started following some YouTube tutorials on Unreal Engine to gain a basic understanding and feel for the controls. While this means that my knowledge and experience is limited as of starting this project, I have a great deal of confidence and enthusiasm for the year ahead and look forward to building on my current knowledge and experience of Unreal Engine.

Technical Approach

Approaching my project, I need to continually improve on my currently limited abilities and knowledge of Unreal Engine. I will need to continue to follow the tutorials which are available from Epic Games. And, continue to follow YouTube tutorials which teach the many different aspects and functions of Unreal Engine.

As there are many different features involved in this project, I will approach the project by breaking it down into core features and focusing my learning and development of the game by focusing on one feature at a time (how do you eat an elephant? One bite at a time). For example, as I start, I will be focusing on learning and developing the required camera and character controls, before I move on to focusing on customising the environment of a level.

Much of this approach will involve creating small scale projects as a learning practice and to gain the experience required to develop an understanding and improve my abilities. With this newly gained knowledge and understanding I can begin developing the newly learned feature and implement it as an integral part of my final year project.

Special resources required

The Examples and Tutorials available on both Unreal Engine and YouTube for developers to follow and learn are a highly-valued resource in my learning of Game Development.

While it was not an essential because of the availability of the college computers at NCI, I decided to buy the parts to and put together a new PC for myself at home. This was because I quickly became aware of how much my humble laptop would struggle to run and build in Unreal Engine.

Technical Details

The project will be created mostly using Unreal Engine software and in parts C++ programming language.

Evaluation

This project will be evaluated by several factors, such as how well it runs upon completion, how many specified goals of the project have been reached to satisfaction. I also hope to demo this project to a test group and receive feedback and reviews on their experiences.

Monthly Journals

September:

With the college year approaching, my attempts to learn the basics of unity, to get a head start on the year, increase. I've created a few basic 2d platforming games and have the beginnings of some basic 3d games started. I'm looking forward to the college year ahead, knowing that I can focus my final year project on something I thoroughly enjoy doing, but I'm also aware that I will need to focus on the college work and need to be more disciplined in my independent learning than I have been in previous years. I'm aware that I will be learning Unreal Engine during the year in Computer Graphics Design and Animations, but unity was recommended to me and has many a lot of similar functions to it.

I start my final year on the 19th and decide to use Unreal engine during the first week of college to see what it is like and compare it to Unity. I also give game maker a go after receiving it and some features that would usually be very expensive for only €15 as part of a "humble bundle".

After a few days, I choose to do my game in Unreal Engine as it seems to have the most capabilities and provides a great jumping off point for game development beginners. I spent my free time during the rest of the week watching tutorials on Unreal Engine and experimenting with some templates to get a feel for the tools involved.

I have an independent study day on Mondays and decided to spend the three-day weekend getting college work out of the way from the start and to keep practising Unreal engine. By Monday afternoon I feel faded and when I see my counsellor, she tells me I need to take breaks to get some exercise and see people. This seems obvious, but having someone point it out makes me realise I can't just burn through the project immediately and that I need to pace myself and unwind when I can.

During the second week in college I try to build a map I created in unreal, it's nothing too big, just a basic building on a street. But when my laptop tries to build the lighting, it becomes apparent that my laptop simply isn't capable of it with its basic graphics card which unfortunately is integrated. I was expecting this to happen at some point, but I had hoped it would happen much later in the year when building a more complex map. At that point, I would be confident of finishing what was left on college computers. But as I can't do nearly as much as I hoped on my laptop, I decide to buy computer parts and build my own PC for working at home.

I spend the next few days picking the right parts for my PC and looking around several different sites that sell parts before settling on parts from overclockers.co.uk. The PC is pricy but it's a worthwhile investment and something I thought about getting before. I order the parts and know they'll arrive after the weekend.

October:

2nd Oct:

After feeling stressed the previous day from the amount of upcoming college work due, I manage to work through an assignment I happen to find tedious. Now feeling motivated I manage to get a few other things finished, which makes me feel much less stressed and more relaxed. And I feel much more sure of myself again. I also make my Project Proposal PowerPoint and think of a name (at least for now) for my Game I'm developing. BoneStorm!

3rd Oct:

I arrived home from college and find my PC parts have arrived. I spend the evening putting together my new PC. But once I have it finished and the PC is running, I'm receiving a "no vga signal" message on my screen. I try to figure this problem out for

some time but by then I'm tired and only stressing myself out. Also, I had to be up early for college the next morning and decide to figure it out tomorrow evening.

4th Oct:

I have been thinking about how my hour-long bus trip every morning and how it might be a big factor in why I seem to lack motivation once I'm in college, yet do much more work on days I stay home. I had realised that on Mondays, when I had new episodes of the Weekly Planet Podcast to listen to on the bus, rather than just music I was more focussed once I got to college. I put this down to finding the podcast interesting and getting my brain engaged on my commute rather than drifting asleep. So, I begin to try out different podcasts on the bus to engage myself. After trying a few, I find a podcast called WTF with Mark Maron. A podcast where notable celebrities come in and talk about the path their careers have taken. An episode with comedian Billy Crystal caught my eye and I find the loose interview and storytelling incredibly interesting. When I get to college I feel energised and focused and the podcasts become a regular thing. The problems with my PC continue to bother me while I'm at college though, but fortunately some friends who graduated the previous year, invite me to lunch and this takes my mind off it until I get home. Once I get home I identify the problem with my PC as one of the 2 sticks of RAM (8GB each) is faulty. Once this stick of RAM isn't connected, everything works perfectly. And I get to use the new PC to do some work on the Intro to AI project before the end of the day.

5th Oct:

Nervous about the project proposal in front of 3 lecturers. This is where the project would either get the green light or be turned down. Luckily the proposal goes great, the lecturers like to enjoy the idea and have some interesting ideas and comments which I keep in mind and take on board as they contained some interesting suggestions.

7th - 8th Oct:

Spent a large part of the day working with Unreal Engine and make some great progress. Spent some time working on my Intro to AI project and on the following day I make huge progress on it, which creates some more time for me to work on my Software Project.

20th – 22nd Oct:

After spending the previous week working on upcoming assignments and CA's I work through the Project Proposal document including the project plan.

23rd Oct:

After a restless night worrying about all the CA's due in the next week, I talk with my friend who graduated last year. She calms me down and tells me she had a meeting in place for discussing dropping out this time last year and to just not put too much pressure on myself to do well, which makes me feel like I'm not struggling as much as I thought and takes a lot of the stress off. And made plans with other friends to see Hunt

for the Wilderpeople in the Lighthouse Cinema the following day to take time off and relax a little.

November:

1st & 2nd Nov:

Spent a few hours learning how to import character models into unreal engine 4 projects and attempting to set up a model as my main character. This was interrupted by a dinner I had been invited to, to celebrate a friend's graduation. I was happy for the event to break up the college work.

13th Nov:

Spent the last week or two focussing on deadlines and upcoming CA's for Artificial Intelligence and Web Services & API Development, and started on my Requirements Specification. Finally got a chance to pick up where I left off on the 2nd. Looked into how to import and set up Mixamo characters in Unreal Engine and tried doing a few different things with that to get to grips with how to implement that.

14th - 18th Nov:

I spent a lot of time during these few days working to get the initial version of my Requirements Specification for my Software Project completed in time for the upload deadline. Managed to upload my first version early, in time to take a well needed break, to go see a few friends for celebration after their graduation ceremony.

26th & 27th Nov:

Settled on a Mixamo model which was altered slightly in Maya and given different textures to make it look more like a cartoony styled skeleton like creature. It still at this point used a Mixamo Blueprint which I had yet to alter. I also looked on the Epic Games market place for possible textures and materials to use for modelling my level(s) and have a good idea of how my main level should look.

28th - 30th Nov:

Using the Mixamo Blueprint for my character meant that I needed to set up some controls and camera options that I had previously done while using the standard Unreal Engine 4 mannequin. I created a Collision Sphere which I placed in the follow camera, I then put in the event graph that when this Sphere and the mesh surrounding my character got too close, the character would become invisible, rather than causing the player to see through the character, in what would look like a glitch. In a multiplayer scenario, other players would still be able to see the character. I also set the camera to be above the character's right shoulder and closer to the character, which is more suitable for a third person shooter. With the changed camera position, I had to make changes to the camera in order to make sure that the camera would still respect collisions and not go through walls.

I also added an 'AimingDownSights' function to the new character, which causes the camera to zoom in over the character's shoulder while the right button is being pressed. When the weapons system in implemented, this will be used for accurately aiming weapons. I made it so that the character will rotate with the camera when in this state.

December

5th & 6th Dec:

Worked on my Tech Report for my software project and on an Intro to AI CA, trying to get them completed before the upcoming deadlines.

10th & 11th Dec:

I uploaded my AI CA which has freed up time to spend finishing my Tech Report for the software project.

13th Dec:

Created a PowerPoint for my upcoming Mid-Point presentation, spent the day going over what I have accomplished so far in my project to put into my presentation and so to better explain what I have done.

14th Dec:

Had my Mid-Point Presentation today. Thought it went well. I was able to explain what I had created so far. I also received a lot of advice on how I should continue with my project. Mainly that I should focus on elements in the game such as enemies and functions, and not focus too much on modelling and appearance of the map, as examiners will be more interested in how the parts of the game that interact with the character work, than they will be with how nice or detailed a building front looks.

15th – 23rd Dec:

Projects in both Multimedia & Mobile App Development, and Web Services & API Development due before Christmas. Most of my time has been taken up with these so I haven't had as much time to work on my software project as I would have hoped.

24th – 26st Dec:

Had a nice Christmas and a few days to relax before studying for exams. Younger cousins were around for Christmas and are very interested in the game I'm making. They're both in secondary school and wouldn't have known game development was something that could be done at home on a pc. This means they're generally positive about the idea of creating my own game. I've found they're a good way of gauging how good an idea for the game is, by how "cool" they think certain parts are. After all, they could certainly be considered as a target crowd for a video game.

I'll be spending the rest of December studying for January exams, but feedback from both my Mid-Point presentation and from my younger cousins, is feedback I will be

thinking about, in order to formulate a plan on where to go from here, once I finish my exams.

January

6th & 7th Jan:

Watching a lot of tutorials on creating simple AI enemies that attack the main character, and also how to use Mixamo character models for these enemies.

14th - 16th Jan:

Created some simple enemies in an Unreal Project template to work out how to create them and how I can implement them into what I have created for my project so far.

20th – 23rd Jan:

Created these enemies in my unreal project, have been trying out a few different settings with them, customising them to behave the way I would like them too.

27th & 28th Jan:

Looking at how to create a character health bar and that will decrease from enemy attacks. Been looking at more tutorials for this and would like to implement it soon.

February

5th & 6th Feb:

I set up a health variable for the main character, he starts on 100 and if an enemy is close enough to him, his health declines until he manages to get away from enemy AI characters, which chase. If the player's health hits 0, the character is destroyed.

11th & 12th Feb:

Been using Adobe Fuse to design my own characters, so that I can have enemy characters which won't look all the same.

18th & 19th Feb:

Worked on designing a projectile which will be shot from the character's weapon, then on creating the object that will be the gun.

25th & 26th Feb:

Worked on attaching the gun object to the main character and on making it so that it will fire the projectile object created.

March

6th-10th Mar:
Completed work on the gun and projectile actors, so that when the player clicks, the gun will fire the projectile..

12th & 13th Mar:
Added models to the main level, the map now looks like a cartoon styled town and gives plenty of roaming space for the player to explore during levels.

15th – 17th Mar:
Worked on making enemy characters harmed by the projectile objects. Gave enemies a health variable (100), then made an interface and set up the projectile so that each time an enemy is hit, they will take damage (20, may be changed) and will be destroyed if their health hits 0.

23rd Mar:
Had a meeting with my supervisor after college. Talked about what had been completed and what few pieces were left to accomplish. Helpful, as an outside opinion is helpful to know which aspects should be focussed on.

26th Mar:
Had to change my Hero characters' health system so that it would mirror the enemy health system, spent the day redesigning it and making sure that the hero takes damage when enemies get too close and will be destroyed if health hits 0.

Requirements Specification
Requirements Specification (RS)

Document Control

Revision History

Date	Version	Scope of Activity	Prepared	Reviewed	Approved
14/11/2016	1	Created Document	Alan		
08/12/2016	2	Updated Document	Alan		

Related Documents

Title	Comments
Title of Use Case Model	
Title of Use Case Description	

Introduction

Purpose

The purpose of this document is to set out the requirements for the development of BoneStorm, a game developed in Unreal Engine 4 for my final year project.

The game will be a third person survival shooter in a 3D world. The story of this game is that it will be set in a world of peaceful skeleton type characters. An unfortunate epidemic hits this world, infecting skeletons and turning them into crazed flesh monsters (which look just like humans). As the only skeleton left, you will need to run and shoot yourself to safety while collecting resources such as health and ammunition to escape these creatures. This story is to serve as a deviation on what might be considered an outplayed scenario of a single human surviving monsters and it's my hope that the ridiculousness of it, will inject a lot of fun into the game.

The intended customers are people who enjoy video games, and are looking for a game that is a fun survival shooter and doesn't take itself too serious in tone and feel.

Project Scope

The scope of the project is to develop a game in Unreal Engine 4 made with C#. Which also will use models created, rigged and imported using Autodesk Maya and Mixamo.

Alan Coleman was involved in research and discussions on the internet and with various members of staff at National College of Ireland for the following requirements.

REQ1. Start Game.

REQ2. Quit Game.

Schedule: The game will be created over the space of 9 months and broken down into multiple phases of planning and development.

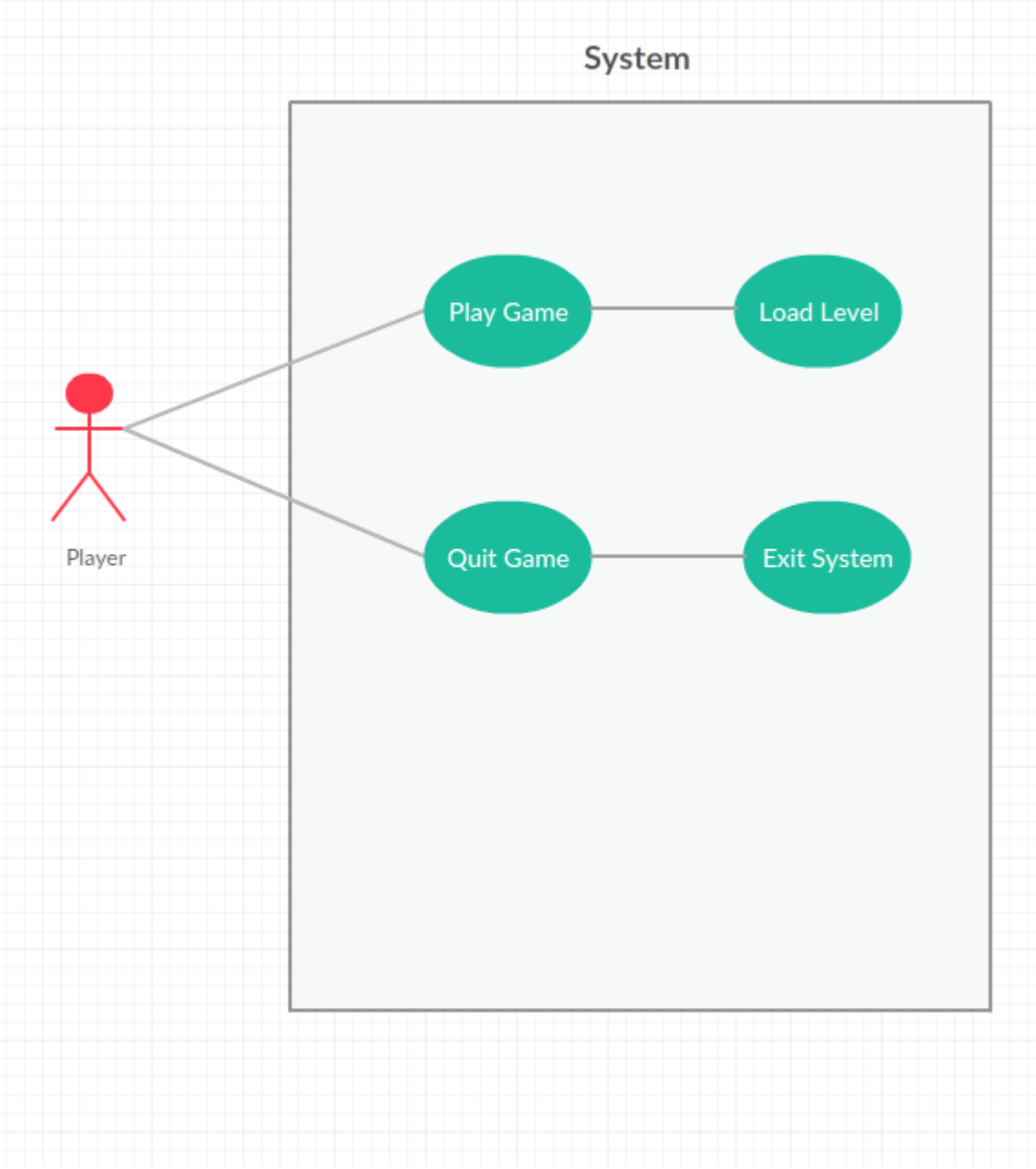
Cost: The game will be mostly free to make with a few exceptions being assets bought on the Epic Games online store.

User Requirements Definition

It is a must that the game will be enjoyable. The game will be stable, coherent and provide a challenging experience. The game will not glitch and will run smoothly.

Functional requirements

Use Case Diagram



Requirement 1 <Start Game>

Description & Priority

The first thing a user will see is the game menu. From here the user will be able to create a new game. This requirement is essential to the system as it is how a user begins a new game and gives control to the player.

Use Case

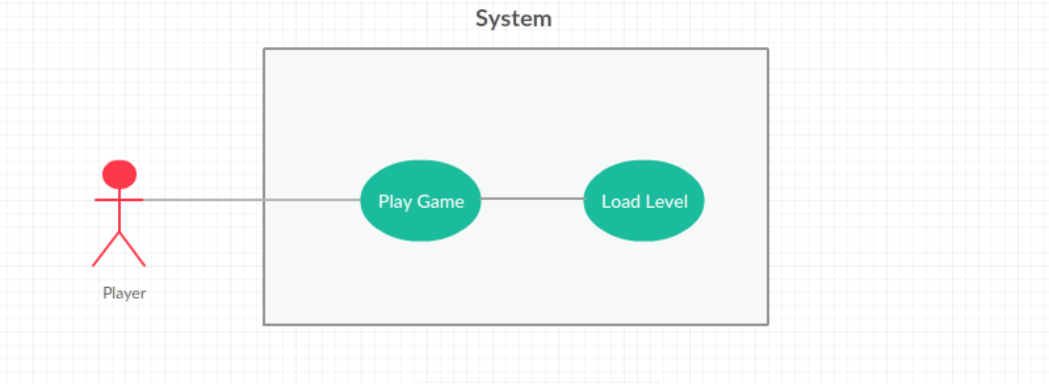
Scope

The scope of this use case is to allow players to start a new game.

Description

This use case describes the process of creating a game.

Use Case Diagram



Flow Description

Precondition

The system is in initialisation mode.

Activation

This use case starts when a player creates a new game.

Main flow

1. The system identifies the player.
2. The player creates a new game.
3. The system creates a new save file.
4. The system loads up the first level.

Alternate flow

A1: Unexpected Error.

Termination

The system presents the first level to the user.

Post condition

The system goes into a wait state.

Requirement 2 <Quit Game>

Description & Priority

Allows a player to quit the game and is warned any unsaved game data will be lost before confirming they wish to quit game. This is essential as it allows the player to end a game session.

Use Case

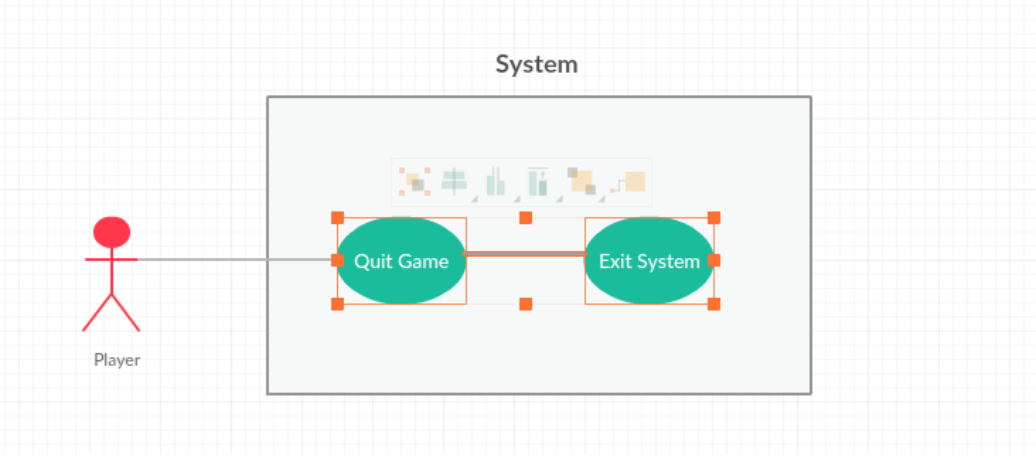
Scope

The scope of this use case is to allow a player to quit the game.

Description

This use case describes the process by which a player can leave the game.

Use Case Diagram



Flow Description

Precondition

The system is in initialisation mode.

Activation

This use case starts when a player wishes to leave the game.

Main flow

4. The player pauses the Game.
5. The player quits the game.
6. The system exits the game.

Alternate flow

A1: Unexpected Error

Termination

The system returns the player to the initial game options screen.

Post condition

The system goes into a wait state.

Non-Functional Requirements

Performance/Response time requirement

The minimum frame rate must be thirty frames per second. The average frame rate must be 60 frames per second on the user's machine. The average response time between click and reaction must be less than 0.5 seconds. The maximum response time between click and reaction must be two seconds. The user can personally modify the graphics settings in order to ensure the game runs optimally.

Availability requirement

The game must be able to run on a windows 7 or higher. And should be available to a player whenever they wish.

Recovery requirement

The game will be able to recover all saved game data stored in the event of the system crashing. This will ensure that in the event of a crash, a player will be able to continue their game from a save point rather than being made to start from the very beginning.

Reliability requirement

The game will be available to the user at all times once the game in downloaded and installed on the user's machine.

Maintainability requirement

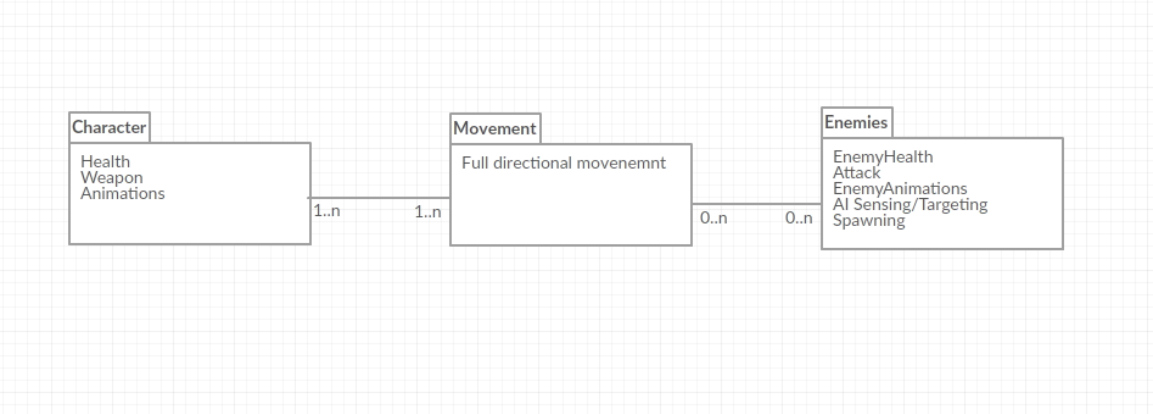
The game will be supported after launch for an indefinite period of time. Any issues or bugs that arise after the games launch will be worked on and a patch fixing these issues sent to players as soon as possible.

Portability requirement

The game will be available to users on any PC or laptop they choose, once the game is stored on it.

Design and Architecture

Class Diagram



I feel like this architecture is well suited to the game I am trying to complete as it covers all the core components that I want to implement while remaining simple to understand. The first class is the 'Character', this contains and updates information about the character, including their current 'Health'. We also have the 'Movement' class, our character can move in any direction.

The game will have enemies involved, and these characters will use the 'Enemies' and 'Target' classes to determine how and when one of these characters attacks the player.

In the game, it will be possible to pick up more than one gun, when a player picks up a new gun, they can then switch between the ones they have collected and use whichever they prefer, this uses the 'Weapons' and 'Guns' classes to do so.

System Evolution

Over time I wish to add extra levels and implement new aspects to the game. I would like to add a more diverse range of weapons, levels and enemies, each with significant differences to the original ones I hope to create.

I also hope to add some vehicle functionality to the game in future, and to make it possible for the main character to interact with the environments much more.

Along with these possible additions, I would like to update and improve on the original project regularly over time.