National College of Ireland
BSc in Computing
2016/2017

Jefferson Tolentino
X13452702
X13452702@student.ncirl.ie

Adrift

Technical Report

Contents

Executive Summary

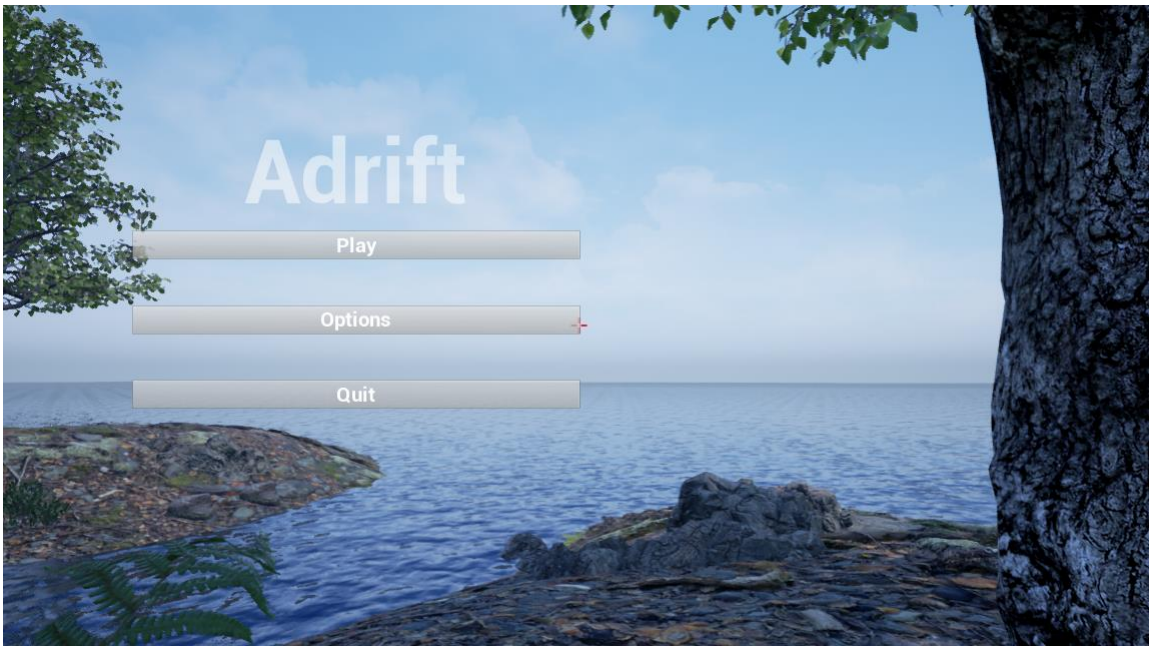
Adrift is an open world adventure game made using epic's unreal engine 4 to deliver the highest possible graphical fidelity.

The player will take control of an unnamed character in a first-person perspective, having a character without a name give the player the chance to fully immerse themselves into the game and project their own image into the game, this place the player itself as the main character. As the player, you wake up inside a castle ruin beside a river and from here you must uncover what has happened to you by exploring the open world landscape presented to you. You will encounter friendly and hostile entities during your adventure so be prepared to survive.

Adrift will have the following key features:

- Explorable maps: The player is not confined in a hallway or a room, the player has the option to explore the map and complete the objective whenever they please.
- Combat system: Ensuring that the player can fight back whenever a confrontation is triggered within the game
- Non-playable characters: The NPC's will be implemented with an advanced A.I which has decision trees, audio, player detection and friendly or non-friendly responses

The game will also incorporate virtual reality as one of its features. Virtual reality has a projected revenue of 30 Billion by the end of this decade, most VR based games confines the player inside a small room. This is a perfect opportunity to create a game where the player can explore a large game world using virtual reality.



1 Introduction

1.1 Game Concept

1.1.1 Initial Concept

Adrift is an immersive casual first person game set in multiple settings to allow the player to explore worlds. The player will take control of an unnamed character in a first-person perspective, having a character without a name give the player the chance to fully immerse themselves into the game and project their own image into the game, this place the player itself as the main character. The player will be able to select maps and unlock mini games within those maps. The game will have a combination of several genres such as first person, adventure, and open world.

The game will also tap in the unexplored are of virtual reality gaming which is an open world or large playable areas.

1.1.2 Updates to the concept

Based on a small survey these game concepts will be added to the game

- The game will now have a narrative instead of just an open world exploration game. Because of this change the maps will now be structured into sequences instead of a hub world selection to convey a sense of progression.

  Having a structure to the maps will also immerse the player in the game because the player will not hop in and out of a map breaking the immersion.

- The player will still be able to play mini game but the mini games are more simplified (example: platforming to advance through the level).
- A simple combat system will be implemented so the players can protect themselves when they are in danger

1.1.3 Final Concept

Adrift is an immersive casual first person game set in multiple settings to allow the player to explore worlds. The player will take control of an unnamed character in a first-person perspective, having a character without a name give the player the chance to fully immerse themselves into the game and project their own image into the game. The game will also give a casual experience giving the player a relaxing experience when they are playing.

The level design for the project has also change from being solely open world to a hybrid map design, the way the user experiences the maps also changed from being to select a world from the menu to a more streamlined per level basis.

The player may also encounter Easter eggs in the game which are secret areas that can be found.

1.2 Background and Research

The initial research for this project started back in early September, where I researched current or immerging technologies from the gaming industry. The second stage of the research was investigating on how, where and what type of games were being released in the recent years based on the technological leaps of the gaming industry and there was also research done on what makes a good video game.

1.2.1 Technology research

In the past, few years there has been a leap in the gaming industry specially with Virtual Reality and Portable Gaming with these new development video games are now becoming more versatile.

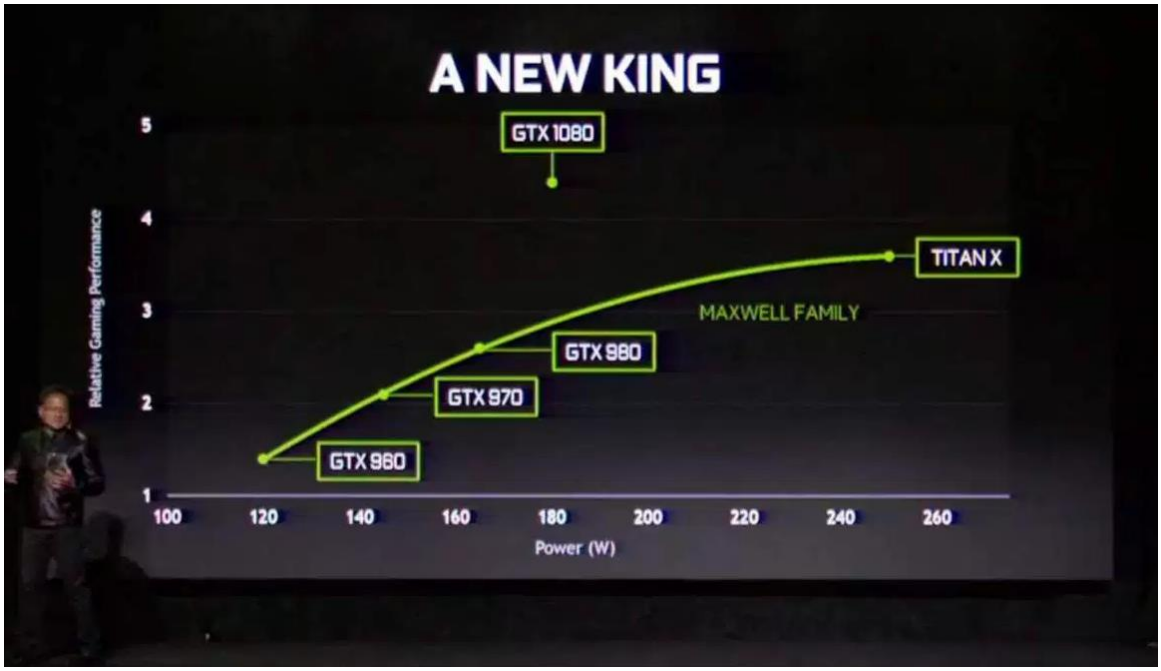
Game Engines – There are about five game engines that are dominating the current gaming market; Crytek's CryEngine, Dice's Frostbite Engine, Steam's Source Engine, Epic Games' Unreal Engine 4 and Unity 5

Engine	Company	Cost	Results
Frostbite	Dice	EA Restricted	Not chosen
Unreal 4	Epic Games	FREE/PAID	Chosen
Source	Valve	FREE	Not chosen
CryEngine 3	Crytek	FREE/PAID	Not chosen
Unity 5	Unity	FREE/PAID	Not chosen

Portable Gaming – In July 2016 NVidia has released its new graphics processing units (GPU), the company has developed a new architecture for its GPU's, the new architecture is code named Pascal. With this new architecture, the new GPU's are more optimized and requires less cooling and power to run meaning that manufacturers can now use full desktop classed graphics cards inside laptops meaning that gaming is now more portable and portable virtual gaming is also a possibility.

Comparison of Pascal and Maxwell Architecture

Screenshot from NVidias Pascal Conference



Virtual Reality – There are three main virtual reality headgear that are currently out in the market; Oculus Rift, HTC Vive and PlayStation VR. With the release of these virtual reality gear gamers will now be able to immerse themselves inside the thousands of virtual gaming worlds. Virtual Reality will have a projected revenue of 30 Billion by the end of the decade.

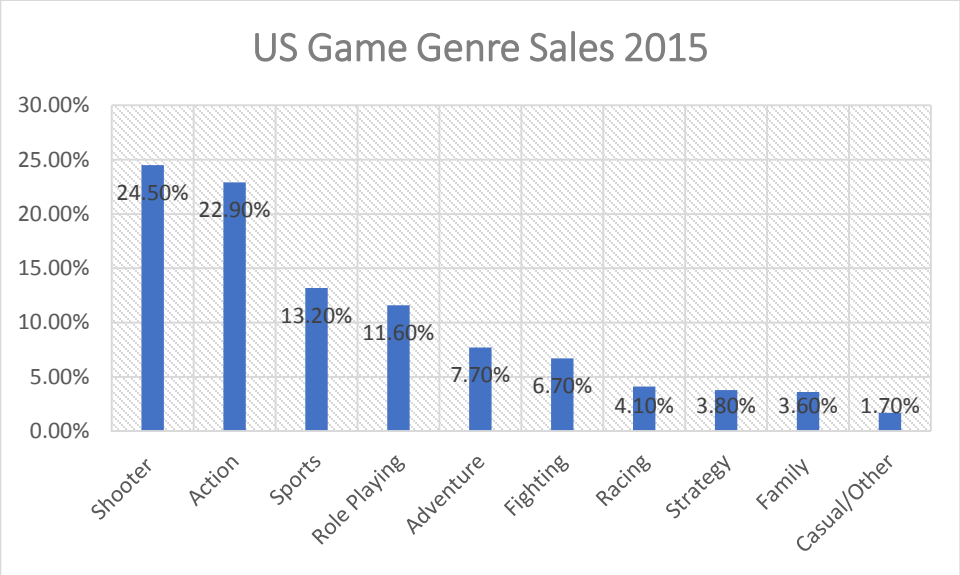
Projected revenue of augmented and virtual reality

Portable Virtual Reality – Companies like Alienware and MSI are developing backpack computers that can be used for virtual reality gaming allowing more freedom for the user.

1.2.2 Game Release Research

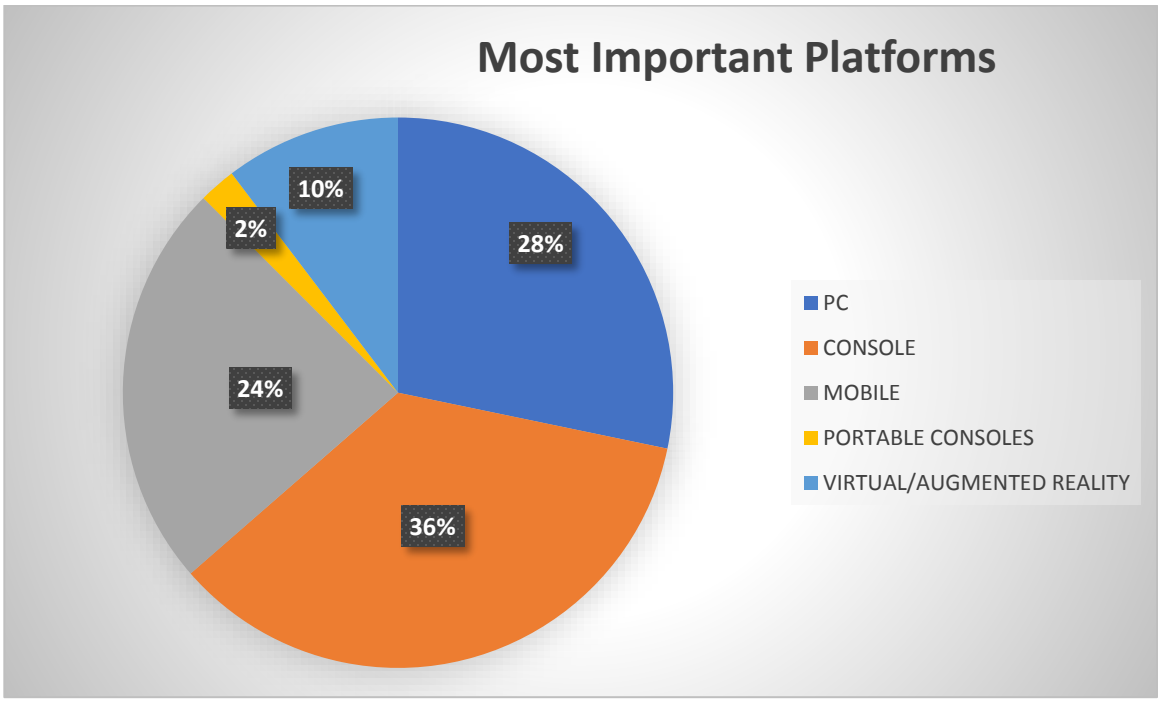
The second part is to research what type of games are being produced, where the games are being released and how are they being released. Below are a few statistics for each fields of research.

Genres:



The genres listed in the above graph is just a generalised list of genres, most modern video games tend to mix and match different types of genres together to create a unique game. There are also sub-genres within each of the genres, the number of them would be too much to list out bout here are some example of sub-genres; First Person Action Adventure, Role Playing First Person Shooter, Etc. Given that most modern games combine genres developers have more freedom on developing what type of game they can create. For this project a combination of these genres are required. Shooter, Action and Others.

Platforms:



From this pie chart, we can conclude that there are three top platforms; PC, Console and Mobile. If we go more in-depth with the information we have here we can narrow the best possible platform to develop for.

Even though the console market has the biggest share, it is divided into at least 5 different markets since there are limited cross-platform capabilities on the console platform. The market is divided between a few devices; Xbox 360, Xbox One, PlayStation 3, PlayStation 4 and WiiU/Wii. Based on this, we rule out console development for now and focus on the next biggest platform which is the PC platform.

Currently the PC platform is the fastest growing in the industry in terms of hardware and graphical fidelity. It also has one of the most open development community, with the release of Unity 5, CryEngine 3 and Unreal Engine 4 more and more users can make and develop pc games, with more people having access to gaming tools there are more resources online that can help with developing a game for pc.

Release Type: There are three main types of releasing video games on the PC platform; Full Game release, Episodic and Early Access.

Full Game release: Full game releases are usually done by triple A developer working for big publishers and the occasional indie developer who has finished the game fully. This type of game release would not be suitable for the project because of the short amount of time available for development.

Early Access: Early access allows the developer to release the game to the public either in a playable state or in a buggy state, most indie developers release games in the early access program so they can have a pool of people to test out the game and give feedback to the developer. This would be suitable for this project since it would get more feedback and users can also suggest improvements and new game mechanics they would like to see in the future iteration of the game.

Episodic: Episodic releases have become popular in recent times, most of the games released in this manner are story driven and the release window for an episode is usually a month a part. A great example games released in this format are games made by TellTail Games, which releases episodic point and click games. This is another suitable release type for this project because it's the best platform to release a story focused/structured game.

1.2.3 What makes a good video game?

Games depends greatly on the person who's playing it, some players might think a game is bad because of the learning curve they must endure before becoming good at the game but regardless of what the person prefers there are some blueprints or criteria that must be included. Below is a list of what elements make a good game but as mentioned previously this list will not apply to a player's personal taste.

Originality – Any game must have at least a hint of originality may it be in concept, gameplay or storyline.

Suspires and rewards – Surprises and rewards in game will keep the players engaged and therefore play and enjoy the game longer without getting bored.

Learning – Giving the player the chance to learn the gameplay mechanics and improve upon as the game progresses.

Challenge – Giving the player challenges while playing the game be it through a mini jumping puzzle or enemies to fight with.

Interesting story line/narrative – Having a narrative to go on will give the game tension and an objective that you must finish.

Level design – A good level design can make or break the game, the level should give a reasonable amount of freedom so that the user can explore the game.

Freedom – The player must feel that the game is not playing itself the player must be able to think that they gave accomplished the things they do in the game and not being hand held and just watching a sequence of events they have no control over.

Winning – The player must have a theoretical possibility of winning the game, each user will play the game differently and they will have different skill levels but the possibility of them beating the game must be present.

1.3 Aims

The overall aim of this project is to create a game where the player can explore a large open world. The game will provide challenges to the player in the form of small mini game inside the game world. It will also provide NPC (non-player controlled characters) that will interact or be hostile towards the player.

The game will allow the player the player to roam around the map as long as they want, unlock hidden mini games and get treasure. The player will also be given the freedom to attack any NPC in the game world.

1.4 Technologies

1.4.1 Hardware

- Laptop/Desktop PC
  - A personal computer will be used to develop and play the game.
- Mouse and Keyboard
  - Mouse and Keyboard will be the primary input of the game. Most players are familiar with the “WASD” Movement but if the player is new a tutorial will be available.
- Controller (Optional)
  - The controller is used to play the game. A controller is needed if the player chooses to use a virtual reality headset like the oculus rift

- Oculus Rift (Optional)
    - The Oculus will be used to test the games potential on immersing the player inside the game world.

1.4.2 Software

- Unreal Engine 4 – Unreal Engine 4 will be used to develop the game
    - Level designing
    - Game Mechanics Development
    - Cut Scenes
    - AI Development
    - Rigging character animations
    - Blueprint development and C++

- Maya Suite and Blender – The software will be used to create assets that are needed in the game.
    - Asset Models
    - Character Models
    - Enemy Models

- Adobe Suite – The software from this suite will be used to create logo, UI elements and voice clips for the game
    - Photoshop (Image Editing Software)
    - Audition (Audio Recorder and Editing software)

2 Definitions, Acronyms, and Abbreviations

GUI – Graphical User Interface: A user interface that shows the players the menu for the game. A GUI can display health, points or even hints inside the game.

RIFT – Oculus Rift: This is a virtual reality headgear mainly used in gaming.

AI – Artificial Intelligence:

PS – Photoshop: An image editing software that can be used to make, edit, crop and publish textures to be used in game.

HP – Hit points/Health Points: This term refers to how much health the player has or have in game, HP is usually represented by a bar or by numbers. When the player's health reaches to zero the player dies.

NPC – Non-player character: This term refers to characters within the game that the player does not control, this character may be hostile or friendly to the character, it may or may not interact with the character.

Easter Eggs – Easter eggs in the game are clever references to other games or other popular mediums like movies, songs and books.

CPU – Central Processing Unit: In gaming this calculates physics and other variables.

GPU – Graphical Processing Unit: In gaming this component processes all the graphical elements of the game such as textures, 3D model etc…

RAM – Random Access Memory: Ram is where the machine keeps data and programs while they are processed.

VR – Virtual Reality

HitScan – A type of system where a calculation is performed on a straight-line to find a point where it intersects an object

Teleport – A system that displaces the character from one place to another.

3 System

3.1 Requirements

The game will be designed with modern controls in mind which means that most players will not need a tutorial to be able to play and enjoy the game, for new gamers the game will offer a tutorial that will teach them how to move, interact and teach them basic combat. The basic movement of a modern first person shooter are bound to the keys “W, A, S, D,” for movement “Spacebar” to jump “E” to interact with objects. Additional controls will be shown on screen when needed.

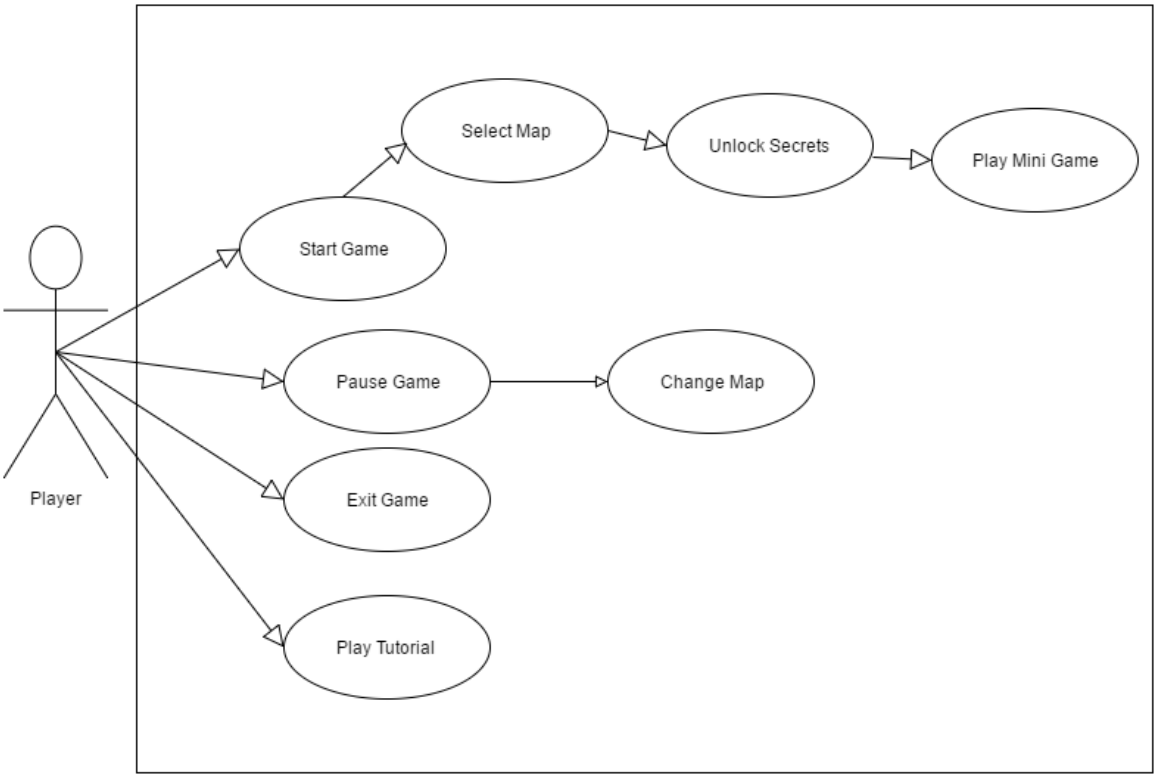
- The user should be able to load the game within a minute of the application being launched
- The users game should be saved after the option tutorial
- The user should be able to skip the tutorial at any time
- The user should be able to revisit the tutorial at any time.
- The user should be able to load a map with no problem and interact with the world. The user should be able to exit the game at any point the wish.
- The user will be promoted if they are a beginner or experienced game when they first start up the game. Picking an option will determine whether the player needs to be loaded in a tutorial map.
- The user should be able to play and unlock mini games and unlock hidden areas on the map when the players discovers them.

3.1.1 Functional requirements

The functional requirements define the function of our system. A function being a set of inputs and outputs of a system. Below is a list of the functional requirements of the game system which is ranked in order.

1. Start game
2. Select Map
3. Unlock Secrets
4. Play mini game
5. Pause Game
6. Change Map
7. Exit Game
8. Play tutorial

3.1.2 Use Case Diagram



3.1.3 Requirement 1 <Start Game>

3.1.3.1 *Description & Priority*

The user must be able to launch and start the game. From here the player can explore hub are of the game where the player can select the map they want to play on.

3.1.3.2 *Use Case*

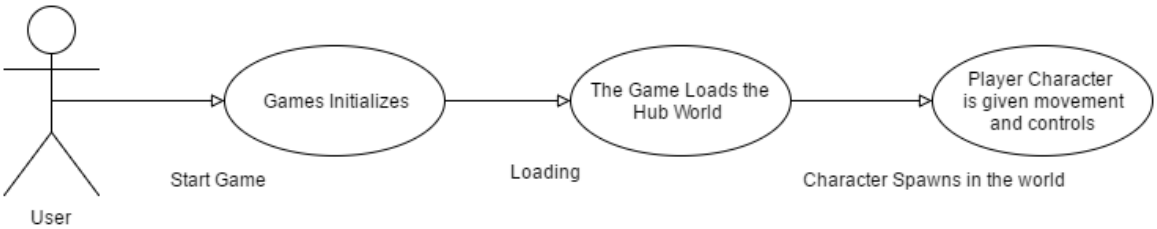
Scope

The scope of this use case is to allow the user to start the game and let the player move around the hub area.

Description

This use case describes the function that allows the player to start the game and explore the hub area.

Use Case Diagram



Flow Description

Precondition

The game needs to be installed on a desktop or laptop machine. The machine also must meet the minimum requirements.

Activation

This use case starts when the user launches the application and starts the game.

Main flow

1. The game launches
2. The User starts a new game (see A1)
3. The Game loads the map “Hub” (See E1)
4. Character is given controls

Alternate flow

A1: <Game Prompts for tutorial>

1. The system displays a prompt for training mode
2. The User selects an option
3. Use case continues at position 3 of the main flow

Exceptional flow

E1: <Corrupted Save>

4. The system will not load the save file
5. The system will make a new save file
6. The use case continues at position 3 of the main flow

Termination

The system gives the player controlled character movement controls

Post condition

The system goes into a wait and idle state

3.1.4 Requirement 2 <Unlock Secrets>

3.1.4.1 *Description & Priority*

3.1.4.2 *Use Case*

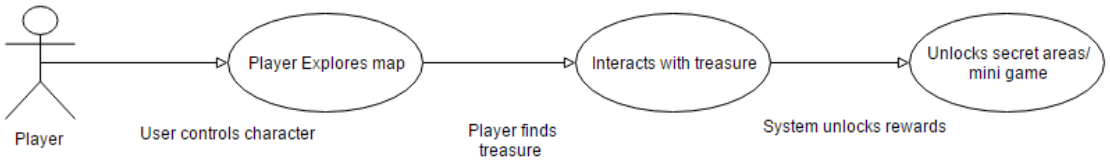
Scope

The scope of this use case is to allow the user to unlock hidden mini games and Easter eggs on the game world

Description

This use case describes the functions that allow the user to unlock hidden secrets

Use Case Diagram



Flow Description

Precondition

The system must have loaded a map/level and have given the user control over the character

Activation

This use case starts when the player explores the map and interacts with a specific object on the map.

Main flow

5. The system gives control to the character
6. The player roams the world and interacts with items/objects
7. The system unlocks new elements (See A1 and A2)
8. The player is given control by the system

Alternate flow

A1: <Unlocks secret area>

7. The system unlocks a new area of the map
8. The system plays a pre-rendered cut scene of the area
9. The use case continues at position 12 of the main flow

A2: <Unlocks mini game>

10. The system unlocks a new minigame
11. The system loads a pre-rendered cut scene
12. The use case continues at position 12 of the main flow

Termination

The new area or mini game is unlocked and the cut scene is played

Post condition

The system goes into a wait state

3.1.5 Requirement 3 <Play Minigame>

3.1.5.1 *Description & Priority*

3.1.5.2 *Use Case*

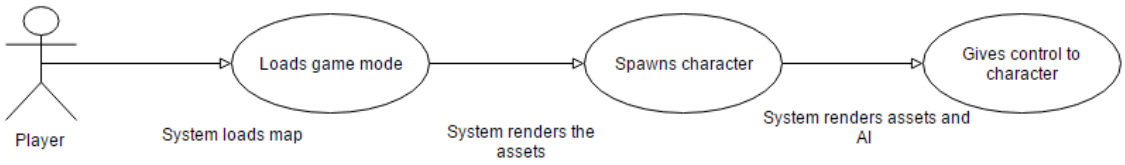
Scope

The scope of this use case is to allow the user to launch and play a secret mini game

Description

This use case describes the function that allows the user/player to play a mini game

Use Case Diagram



Flow Description

Precondition

The player must have unlocked the mini game by finding a hidden section

Activation

This use case starts when the player touches or interacts with the hidden treasure

Main flow

9. The user interacts with the secret object
10. The system loads up the game mode
11. The system spawns the character in a small map
12. The system gives the players control
13. The player plays the mini game

Termination

The playable character is spawned and the mini game is activated

Post condition

The playable character is spawned and system waits for inputs from the player

3.1.6 Requirement 4 <Pause Game>

3.1.6.1 *Description & Priority*

3.1.6.2 *Use Case*

Each requirement should be uniquely identified with a sequence number or a meaningful tag of some kind.

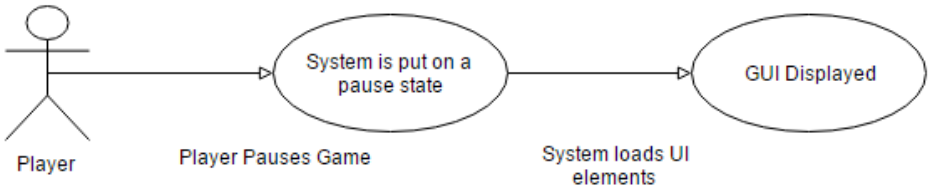
Scope

The scope of this use case is to allow the user to pause the game at any time

Description

This use case describes the function that allows the player to pause the game

Use Case Diagram



Flow Description

Precondition

The player must have spawned in the hub world or in any other playable map/level

Activation

This use case starts when the player wants to take a break and pause the game

Main flow

14. The player hits the “esc” key to pause the game
15. The system loads the GUI elements
16. The system displays buttons [See A1 and A2 for alternative flow]
17. The system is on idle mode

Alternate flow

A1: <Pause Menu for Hub World>

13. The system displays only three buttons (Resume game, play tutorial and exit game)
14. The use case continues at position 21 of the main flow

A2: <Pause menu for other levels>

15. The system displays four buttons (Resume game, choose a different level, play tutorial and exit game)
16. The use case continues at position 21 of the main flow

Termination

The system presents the pause screen GUI

Post condition

The system goes into a wait state

3.1.7 Requirement 5<Change Map>

3.1.7.1 *Description & Priority*

3.1.7.2 *Use Case*

Each requirement should be uniquely identified with a sequence number or a meaningful tag of some kind.

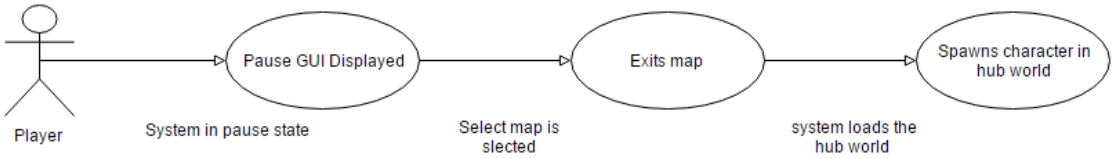
Scope

The scope of this use case is to allow the user to change maps when the players character is out of the hub world

Description

This use case describes the function that lets the user change maps

Use Case Diagram



Flow Description

Precondition

The playable character must be out of the hub world

Activation

This use case starts when the player is on the pause menu and selects change maps

Main flow

18. The System is sitting idle on the pause menu
19. The player selects to change the map
20. The system displays a prompt asking the player if they want to return to the hub world
21. The player selects an option (See A1 for alternate flow)
22. The system loads the hub world

Alternate flow

A1: <selecting no>

17. The system does not load the hub world
18. The use case continues at position 22 of the main flow

Termination

The system loads the hub world or the system sits in the pause menu waiting for an input

Post condition

The system goes into a wait state

3.1.8 Requirement 6 <Exit Game>

3.1.8.1 *Description & Priority*

3.1.8.2 *Use Case*

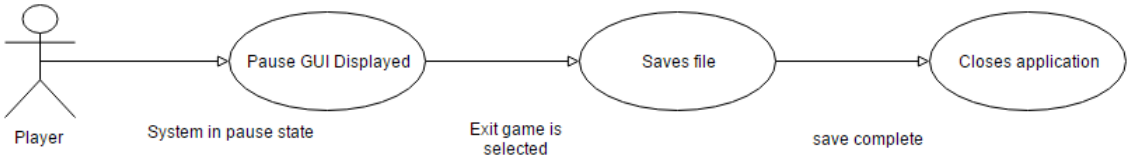
Scope

The scope of this use case is to let the player exit the game any time

Description

This use case describes the function that lets the player exit the game any time

Use Case Diagram



Flow Description

Precondition

The system must be in the pause menu screen

Activation

This use case starts when the player wants to close the application

Main flow

23. The system displays the pause menu screen
24. The player chooses the close game button
25. The system displays a prompt confirming if they want to exit the application
26. The player selects an option (See A1 for alternate flow)
27. The system closes and returns to desktop

Alternate flow

A1: <Selecting no>

19. The system does not exit the game
20. The use case continues at position 27 of the main flow

Termination

The system exits or the system waits in the pause menu screen

Post condition

The system goes into a wait state or closes itself

3.1.9 Non-Functional Requirements

Performance/Response time requirement

The performance requirement will be decided by the user's machine, if the users machine does not meet the minimum requirements the use will have trouble running the application.

Minimum Specification:

- Operating System: Windows 7 or Above
- Memory/Ram: 4GB Minimum, 8GB Recommended
- CPU: Dual Core Intel or AMD Minimum, Recommended Quad Core Intel, or AMD CPU
- GPU: NVidia GeForce GTX 470 or AMD HD 6870 or higher

Recommended Specifications:

- Desktop or modern gaming laptop
- **Operating System:** Windows 7 or above
- **CPU:** Quad-Core AMD or Intel i5/i7
- **Memory/Ram:** 4GB Minimum, 8GB Recommended
- **GPU:** NVidia 760 or Above for desktop and NVidia 970m for laptop

The users machine will determine the response time in game, when the users machine does not meet the minimum requirements the application will respond slower and vice versa.

Availability requirement

The application must be able and ready to launch at any time once it installed in the user's system.

Recover requirement

The application should be able to load a previous working save file in an event that the application crashes.

3.1.10 Security requirement

The security requirement for this game will be very minimal since the user does not have to have an internet connection to play, the user just needs to install the application locally once. Once installed the user can launch the application anytime without needing an internet connection.

Reliability requirement

The game must always be able to launch from the exe file. Once the game is launched the engine will take over.

Maintainability requirement

Patches will support the game, patching any bugs found in development or bugs discovered by the users/players. The users can email feedback or post in forums.

Extendibility requirement

The game after release, additional content could be developed for the game, adding more maps, items, and gameplay mechanics.

Reusability requirement

The assets that will be used for this game will be reusable custom made or downloaded.

3.1.11 User requirements

- The User machine must meet the minimal requirements of unreal engine 4, these requirements are as follows:
    - Operating System: Windows 7 or Above
    - Memory/Ram: 4GB Minimum, 8GB Recommended
    - CPU: Dual Core Intel or AMD Minimum, Recommended Quad Core Intel, or AMD CPU
    - GPU: NVidia GeForce GTX 470 or AMD HD 6870 or higher

- For the best experience the user's machine should meet the recommend specifications, these requirements are as follows;

    - Desktop or modern gaming laptop
    - **Operating System:** Windows 7 or above
    - **CPU:** Quad-Core AMD or Intel i5/i7
    - **Memory/Ram:** 4GB Minimum, 8GB Recommended
    - **GPU:** NVidia 760 or Above for desktop and NVidia 970m for laptop machines
- The user must have a controller or keyboard and mouse, these will be used to control the in-game character
- The user must have working copy of the application
- The user may use a VR headset like the oculus to play the game.

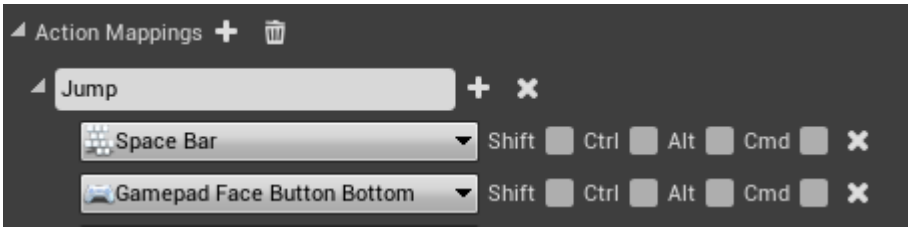
3.2 Implementation

3.2.1 Control Implementation

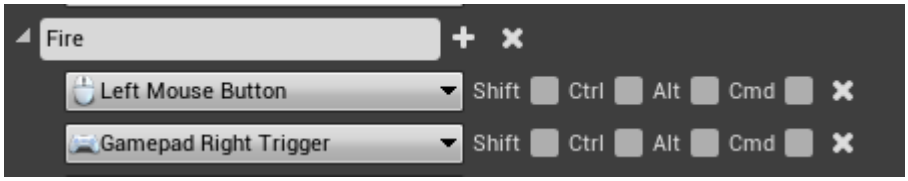
Key binding

Every project in unreal has a default input section, the project implements the usual control scheme for first person shooters. For example "W,A,S,D" keys are used for movements and Spacebar is used for jump.

Jumping - Jumping is bound to the Spacebar and Bottom face button on the controller.



Firing Gun – Firing weapon is bound to the left mouse click and right trigger on the controller

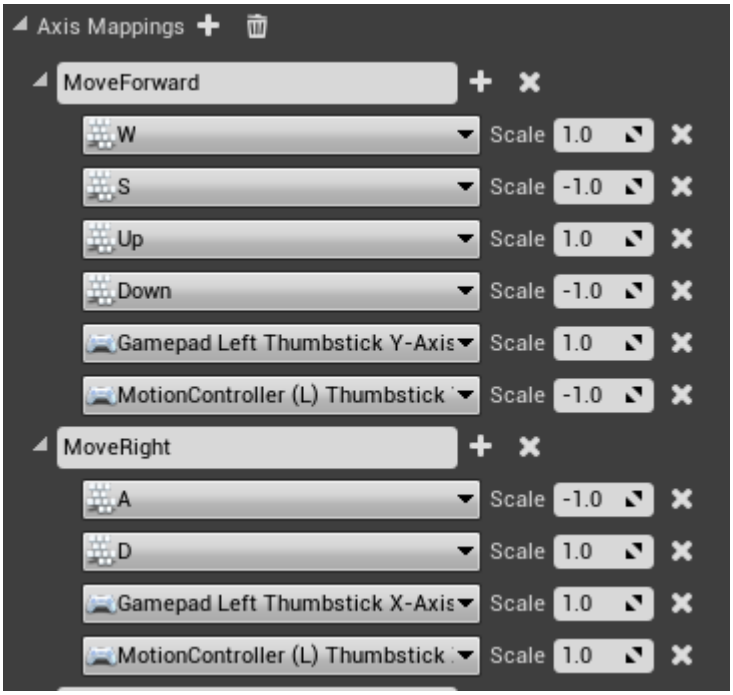


Moving – Movement in games are split into three different Axis; X, Y and Z. The project uses Y axis to move forward and X axis to move side to side.

For the sake of keeping the project clean only two main mapping were used “move forward” and “move right” and each mapping has two opposite keys with positive and negative values.

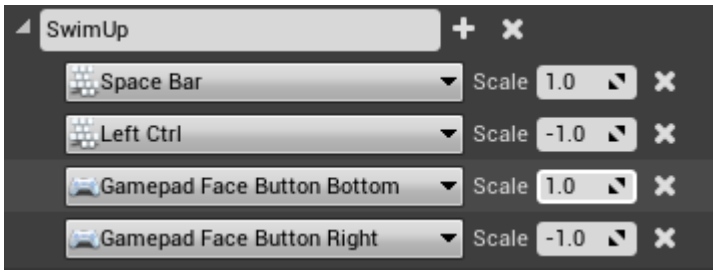
Example: on the move forward mapping the key bindings are “W” and “S” keys. The “W” key has a positive value so the character moves forward and the “S” key has a negative value which is the opposite of the “W” which means that the character will move backwards.

The typical movement keys for games are “W, A, S, D”. Controller support is also supported as seen on the diagram below.

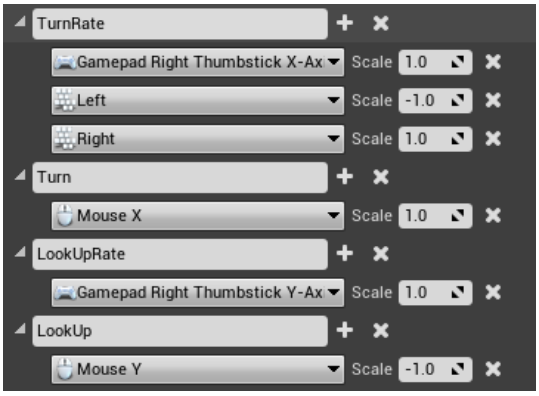


Swimming- The same can be said with the swimming control mapping, it works by using the Z Axis to move up or down. As seen above it uses the same positive and negative values to move up or down the axis depending on what keys are pressed.

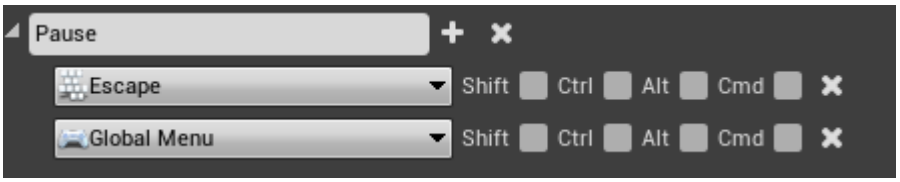
The Typical movement options for swimming are “Space bar” to swim up and “Left control” swim down. Controller support is also supported as seen on the diagram below.



Looking- For looking and aiming around we use the right thumb stick on the controller or using the mouse on the computer.



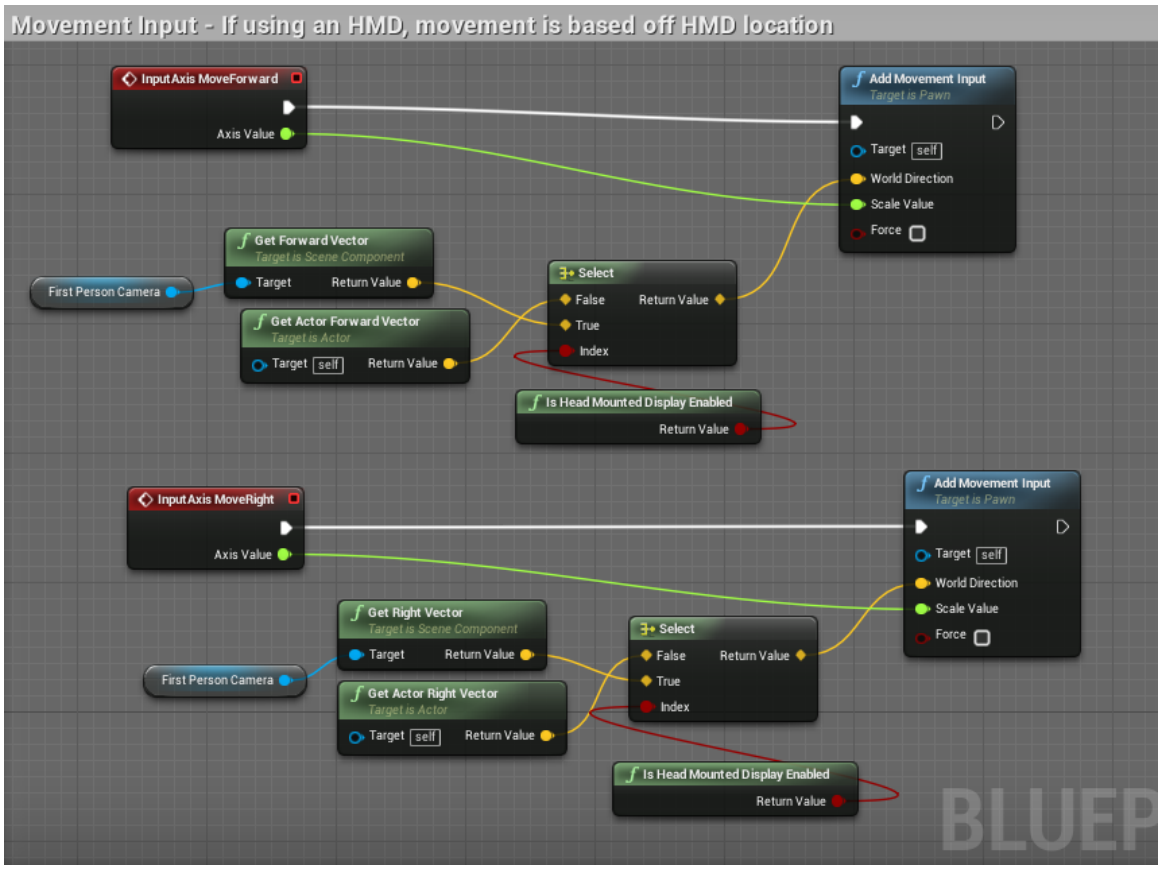
Pause - Every game has a pause menu for when the player wants to take a break. The default for most games are the escape key.



Control Blueprint

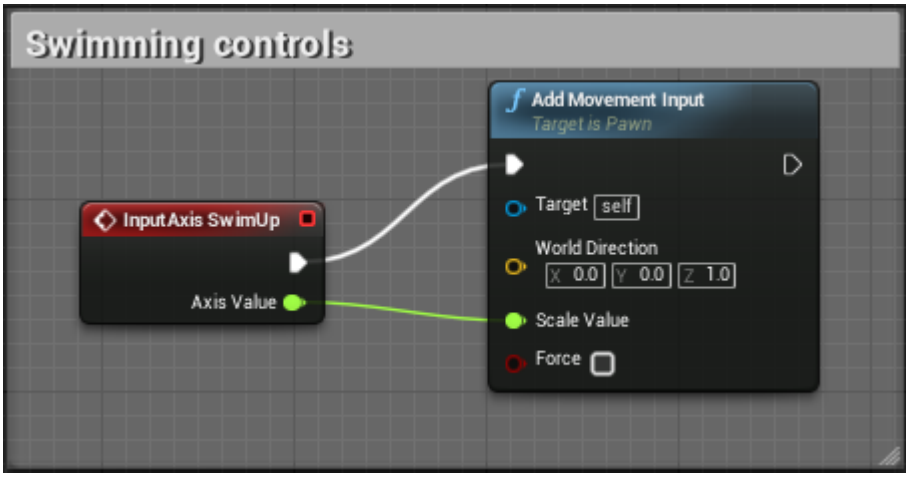
Movement-

For Movement, the “MovementForward” Input changes the Y axis of the player move forwards or backwards and the values of these are set in the key bindings. The same can be said for the MoveRight input, this allows the player to move left or right because of the positive and negative values we set in the key bindings.



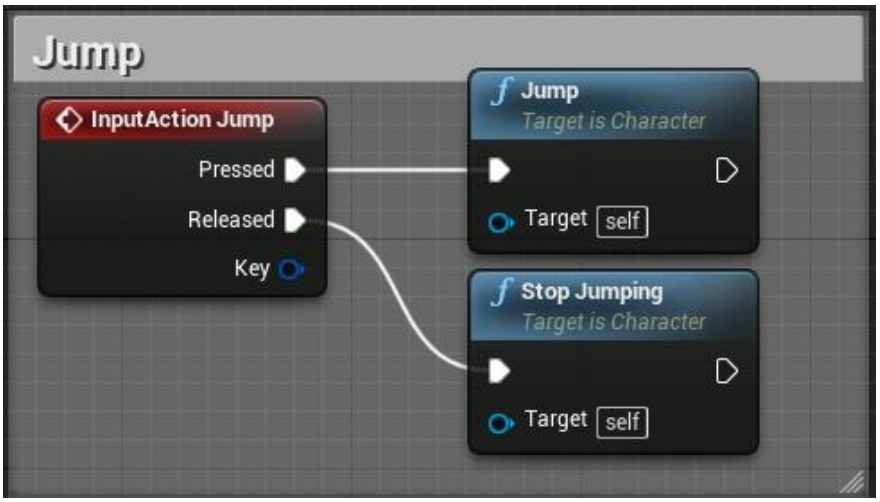
Swimming-

For swimming the "SwimUp" input changes the Z axis depending on keys pressed. On the key binding, we have set a negative value to go down and a positive value to go up when we are swimming.

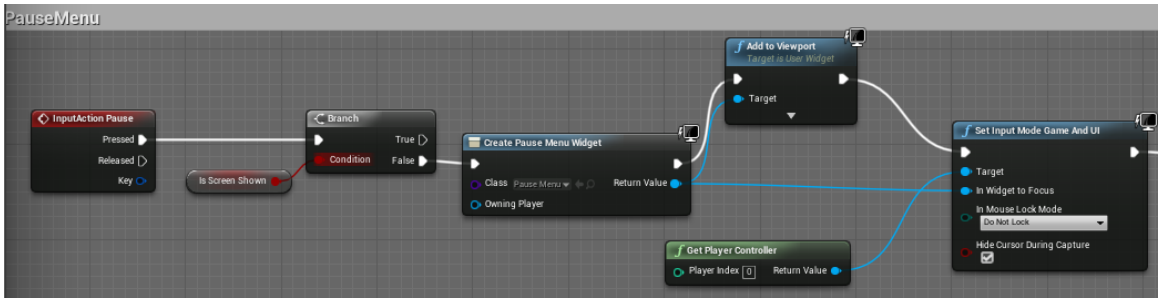


Jumping -

The jump input is attached to two nodes, Jump node and stop jumping node these nodes are built within unreal engine so we just need to map them correctly.



Pause-



3.2.2 Combat System Implementation

Player Combat

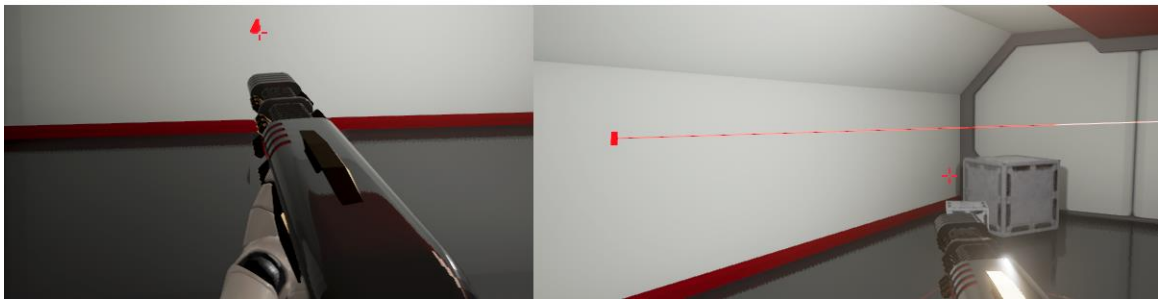
The game uses a popular type of combat system used in most first-person shooter games. This type of combat system is called “hitscan” or “line trace system” in the gaming industry.

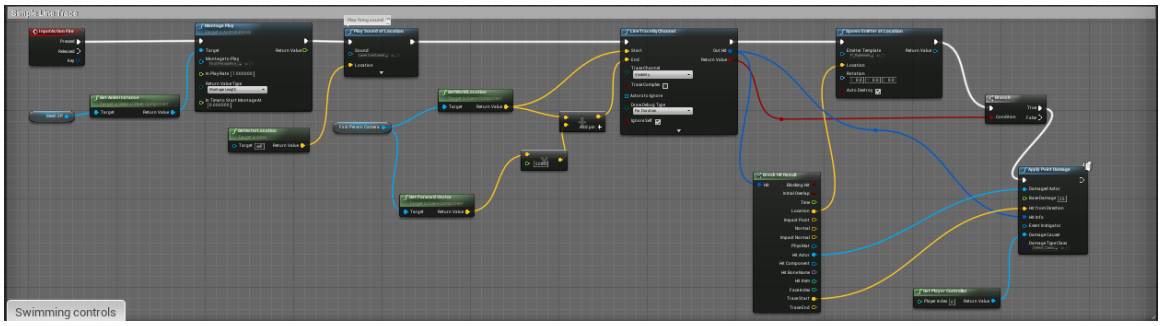
The weapon may it be a gun or melee will hit whatever the player is looking at instantly there is no travel time. This method works in a straight-line. Popular games such as Call of Duty and Counter Strike use this type of combat system.

Hitscan is a type of system where a calculation is performed on a straight-line to find a point where it intersects an object

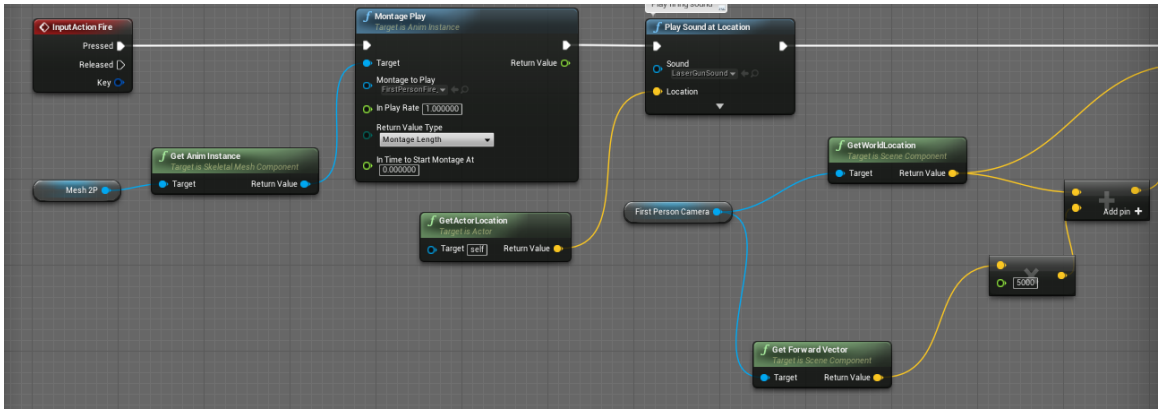
User Prespective

Line trace seen from another angle using debugging

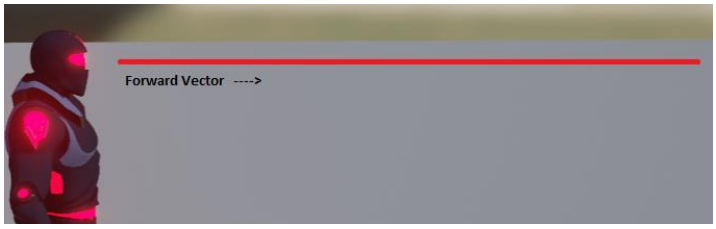




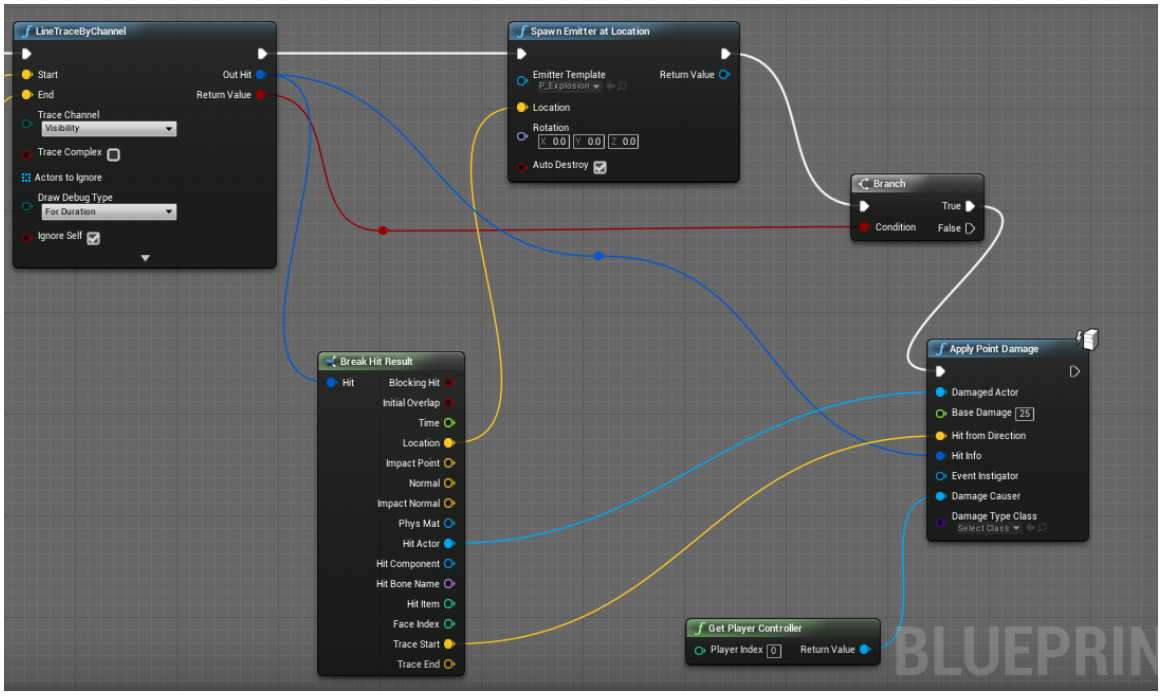
The line trace is triggered when an input is activated in this case the input is the action “fire”. After the fire input has been triggered we can see from the diagram below that an animation montage and sound is played whenever this event is triggered. This is just for aesthetics effects showing the player that the weapon is firing.



We can see an instance of the player camera which is called “First person camera” and we get its world location and get the forward vector (forward vector is the horizontal axis that the player sees) and we connect it to the line trace node to get what the player is looking at.



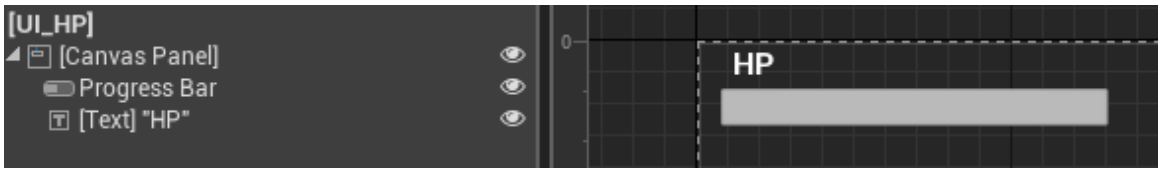
The Spawn Emitter Node allows us to spawn a particle effect so the player can see that they have hit an object in game.



The apply damage node is used to damage certain actors in the game such as enemies and objects.

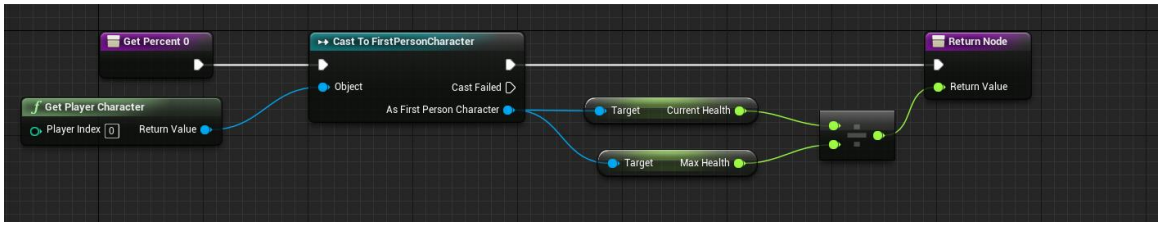
Health

The health bar is implemented using a user interface widget. A progress bar is used to visualize the health in game.

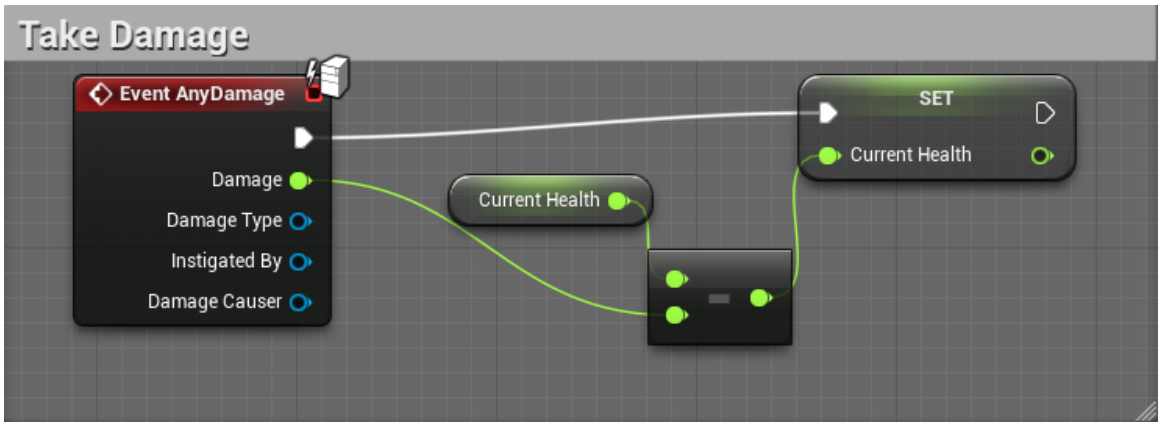


The health amount is casted to the first-person character which is the players character.

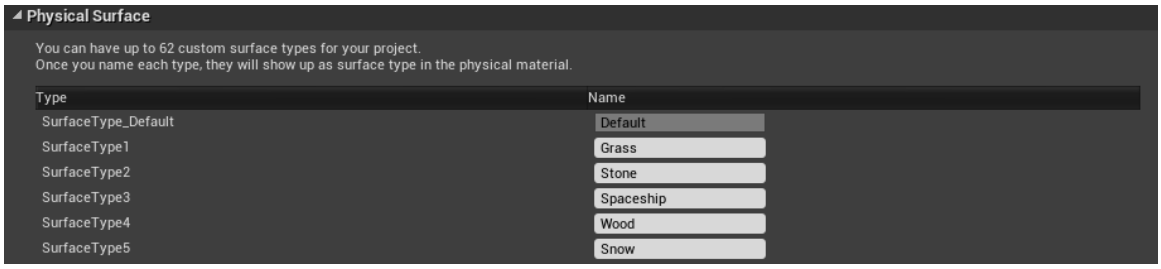
To calculate the player's health, we use a simple subtraction node to subtract the current health to the max health.



We also have an event anydamage inside our character's blueprint. This simple blueprint logic subtracts the incoming damage to our current health. After the computation, we use a Set current health node to set our current health.



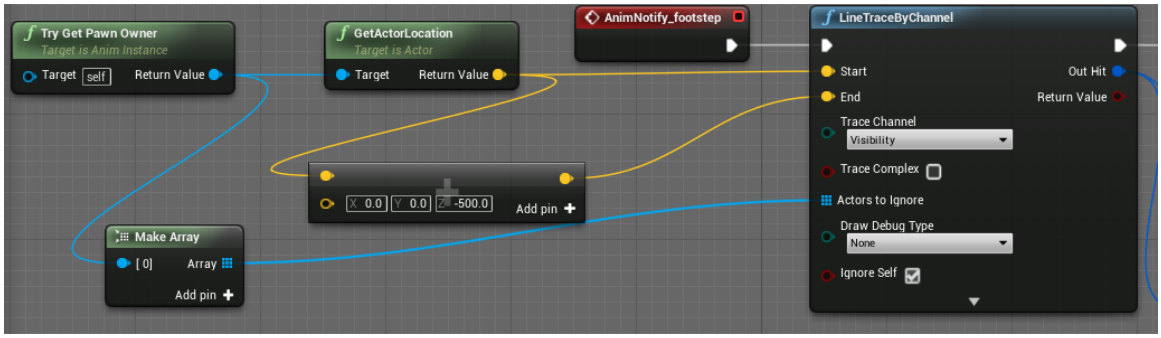
3.2.3 Footstep Implementation



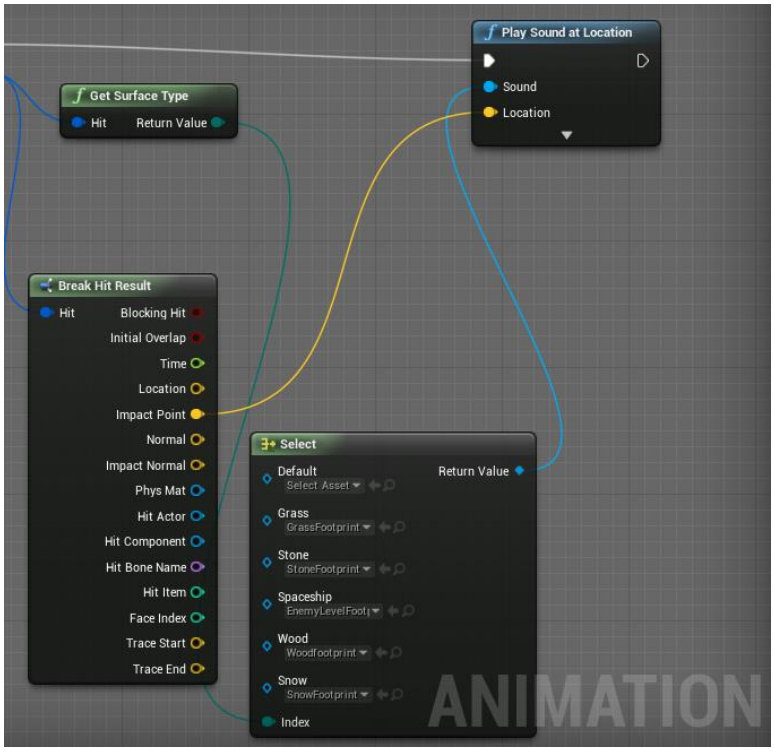
The footsteps in the game are implemented within the animations and animation blueprint.

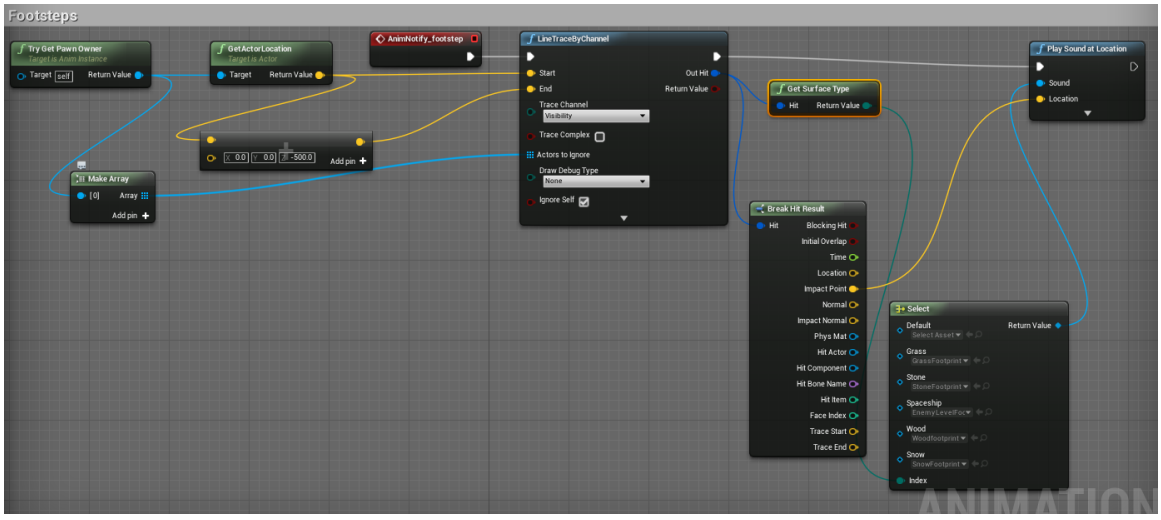
We have a notify in our animations. The notify alerts our blueprint to play a sound whenever it is triggered in the animation.

We use a simple line trace node so we can get our characters relative location and where the character is stepping on the landscape.



We use a vector addition node along with the line trace to calculate our player's Z axis. The line tracer looks along the positive and negative Z axis and calculates where our character meets the ground.





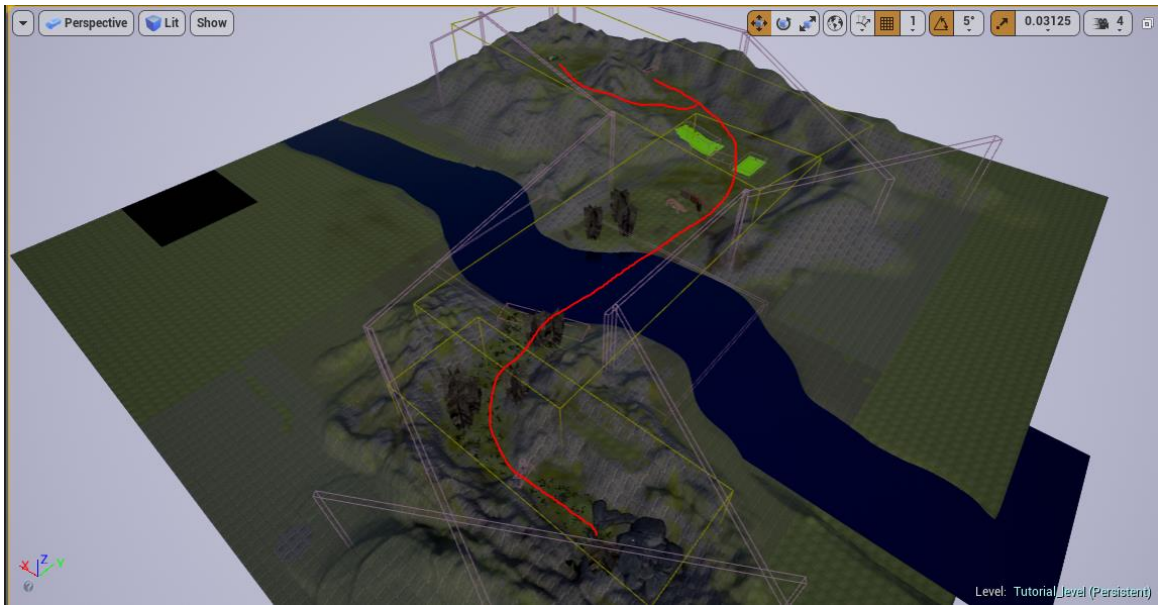
To set what sound clips needs to be played we can assign different physical materials to meshes or texture materials within unreal engine. The diagram below shows a physical material being

applied to an object. This will play the sound clip associated with the Spaceship material whenever the player is moving

3.2.4 Maps Implementation

The maps will give a sense of freedom an open world feeling that gives the players a chance to explore the map and discover hidden secrets and challenges. The map design will also include linear sections so that the player can be guided to certain objective. Below you will see a diagram of a linear section connected to an open area.

Tutorial Level

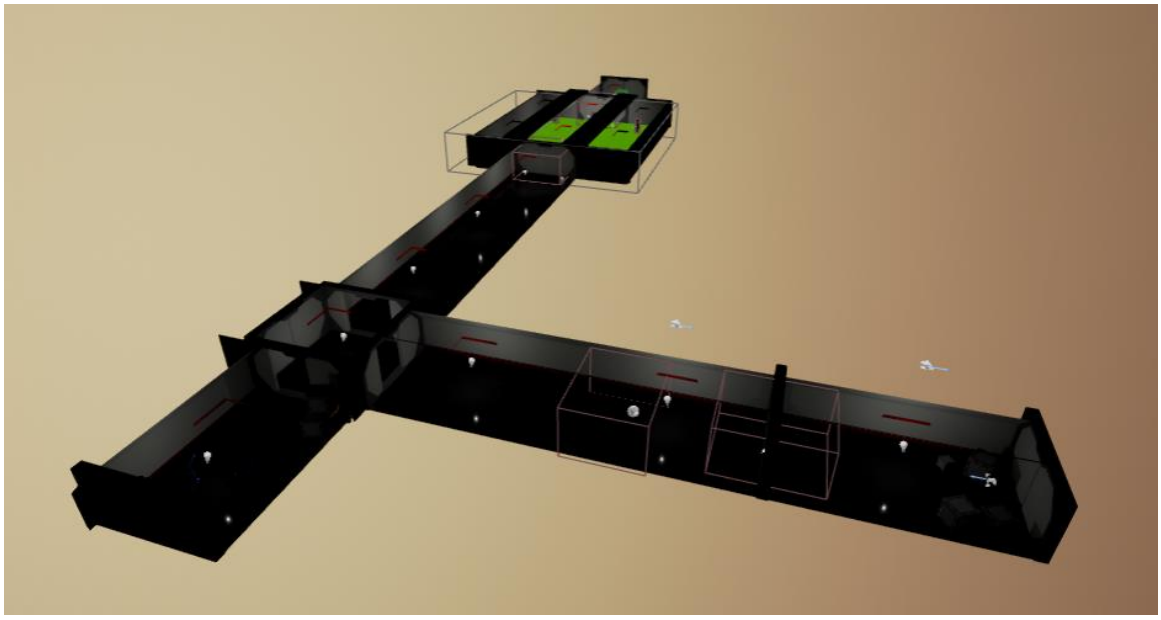


The tutorial level is implemented with a linear/corridor design where the player can only go forward to progress.

Transition Level

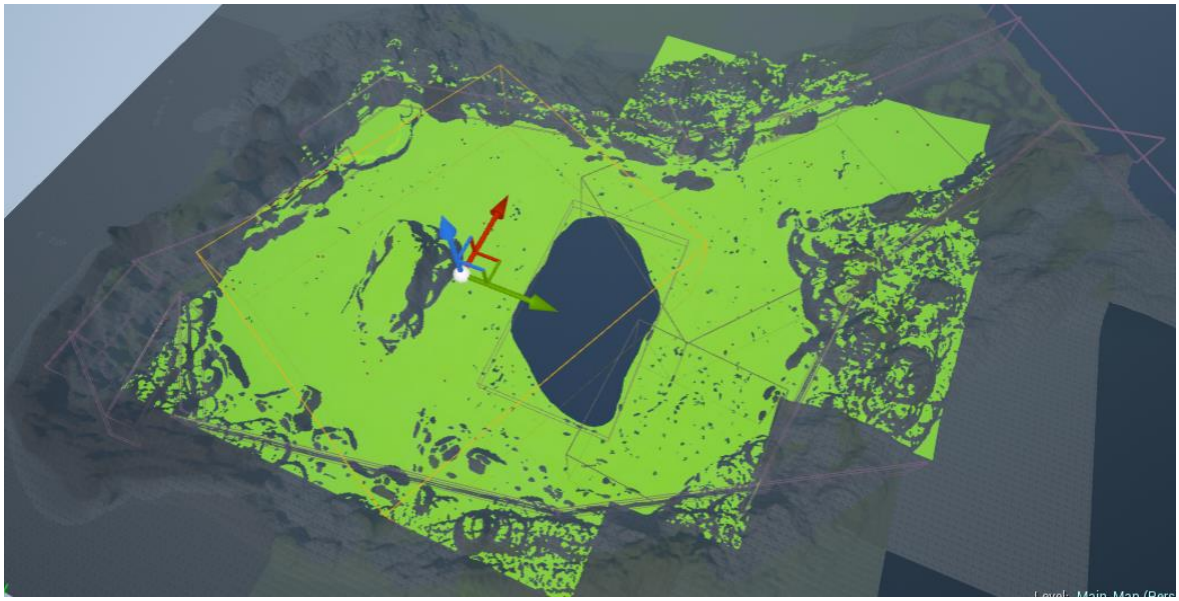
A more linear world level design. Designed to guide the player and teach the player about enemy types.

The level is set in the “enemies” spaceship which mean there will be small corridors and a futuristic design.



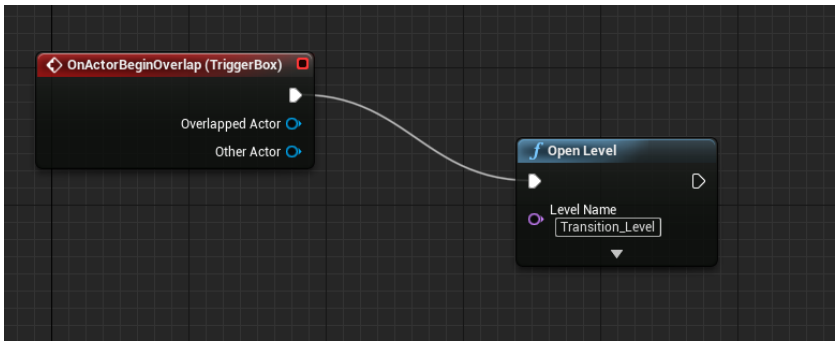
Main Level

The main level has a combination of linear and open world design.

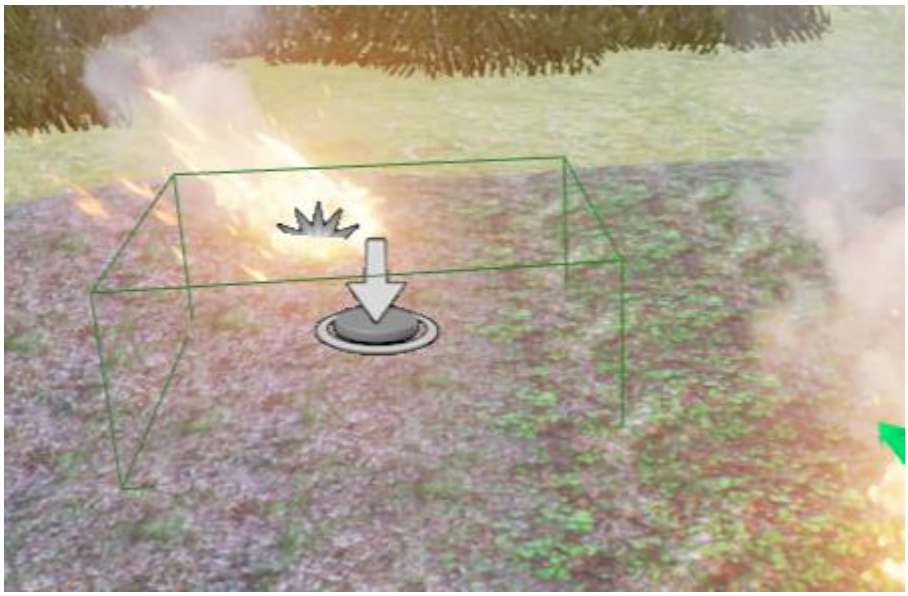


Level Transitions blueprint.

A simple overlap trigger is used to load the next level. Each level has a box trigger at the very end and will be triggered when the character steps into the portal.



Box trigger without a portal effect for clarity.

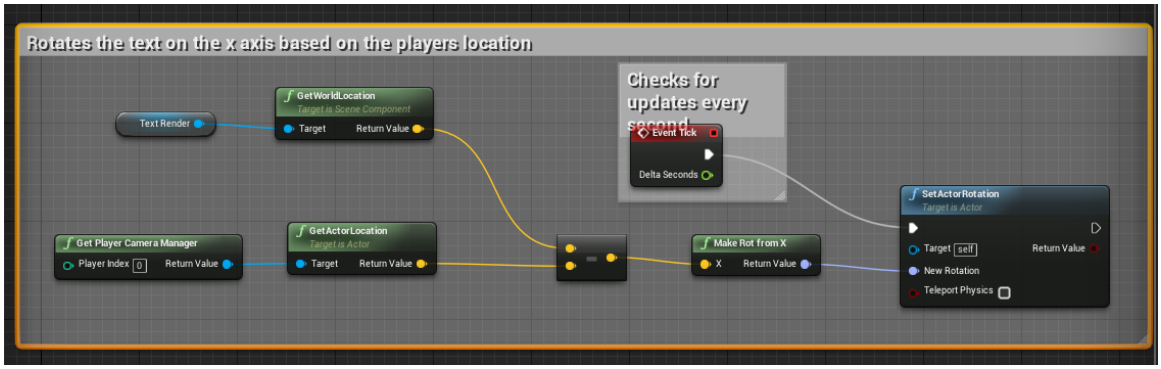


3.2.5 Text Pop Ups Implementation

In game the player will be show in game text pop ups for objectives and instructions.

Rotating Text

The in-game text rotates based on the player's location so it's always visible to player.



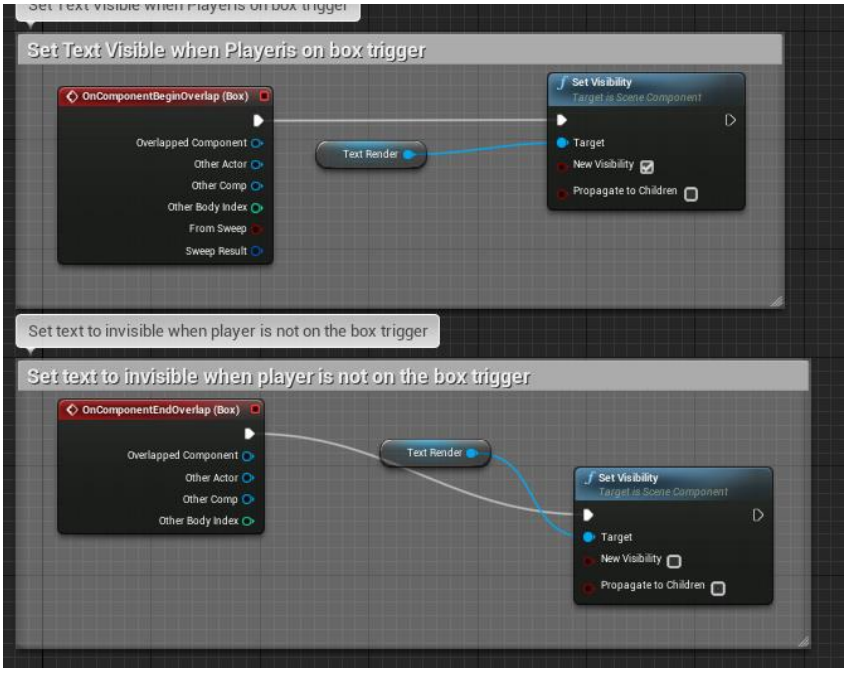
We get the text location named "**text renderer**" and we get the players camera and location by calling the nodes "**GetPlayerCameraManager**" and "**GetActorLocation**".

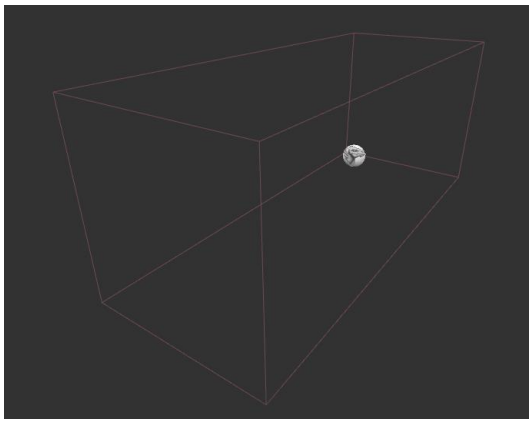
We also subtract the two vectors and make them rotate on the X axis and set the actors rotation with the **SetActorRotation** Node

The "Event Tick" node checks for changes every second.

Showing Text

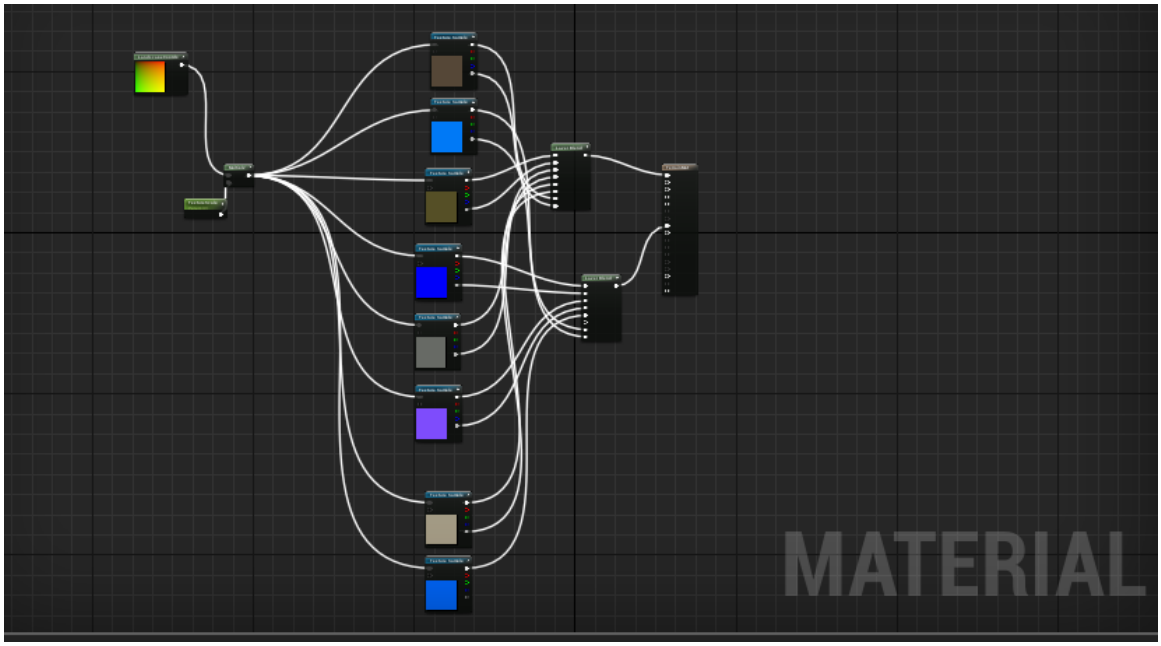
The text is only shown when the player stands on an invisible box trigger.



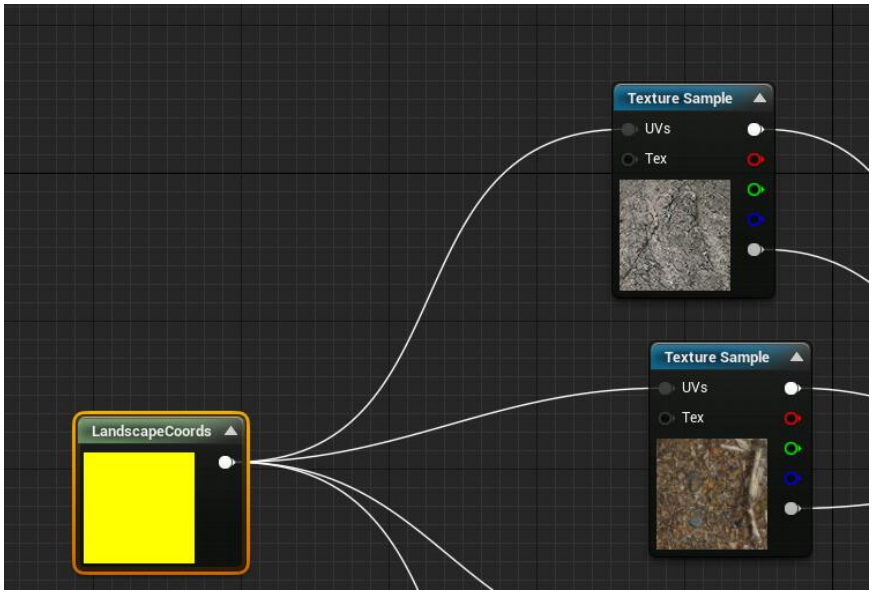


3.2.6 Texture/Materials Implementation

Textures are handled by Unreal Engine 4 and will be using Unreal's blueprint system. Below is an example of what a texture material looks like.

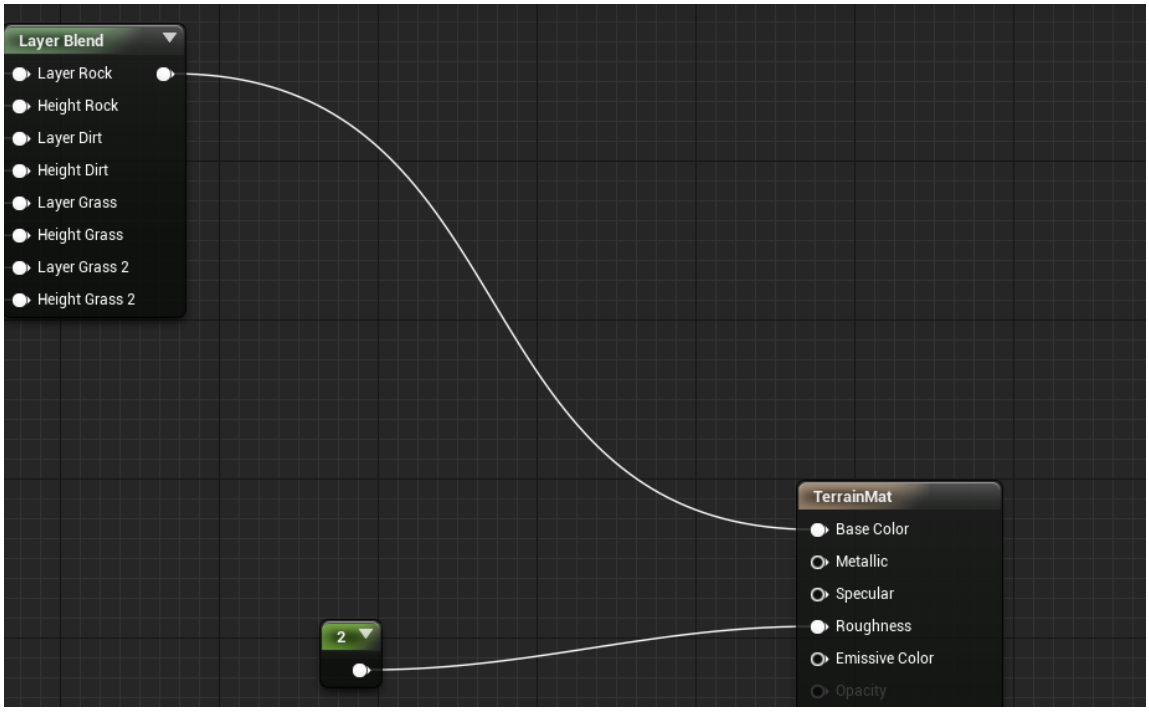


The landscapeCoordinate node let’s the developer choose the scaling of whatever texture its connected to. This can be used to rescale the developers texture if they are too big or too small.

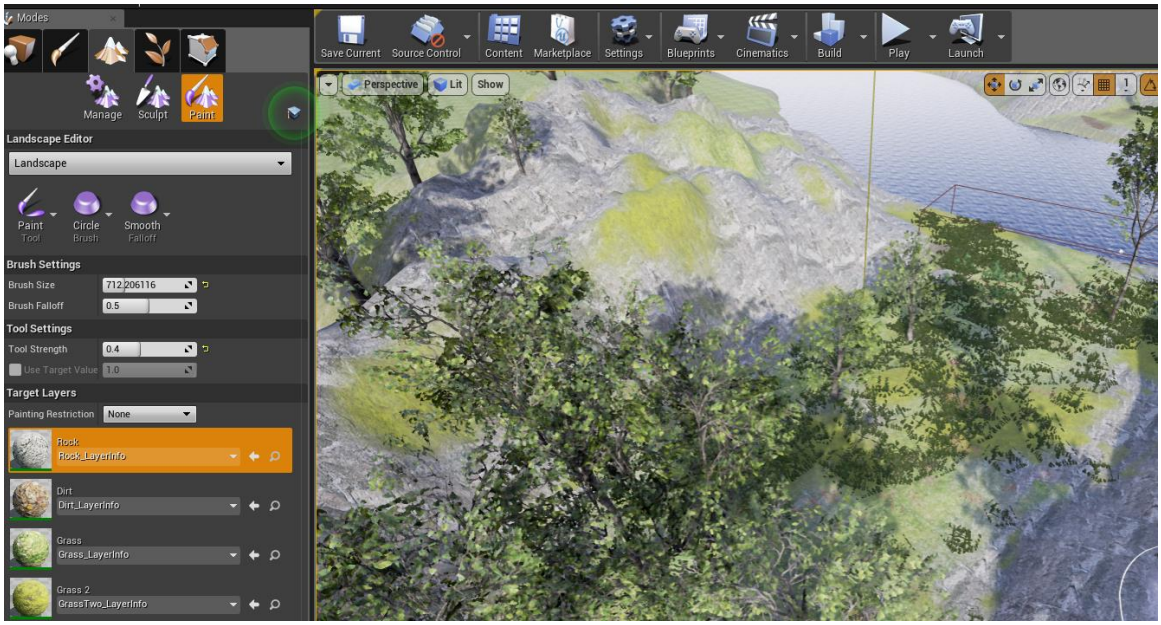


A layer blend node is used to combine textures for landscape. This way the developer can paint in multiple textures without creating multiple instances of the landscape material.

We also have a constant node connected to the roughness value in the terrain material, this is to stop the material from looking shiny.



Texture materials can be used to paint any assets in the game, for example we used the “terrainmat” material to pain the landscape of our game. We can see that we have 4 available textures to use in painting mode this is because of the layer blending node.



3.2.7 AI Implementation

Models/Types

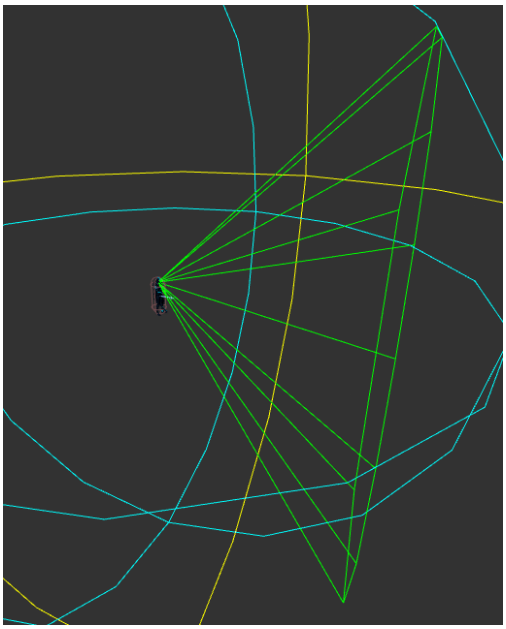
Hostile – Hostile enemies will attack the player once the AI comes in viewing distance of the player. The hostile enemies have a futuristic red glow to show that they are hostile.

Friendly – Friendly AI are also in the world the player can still kill them but they won't attack the player. The friendly AI have a futuristic blue glow to show that they are friendly.

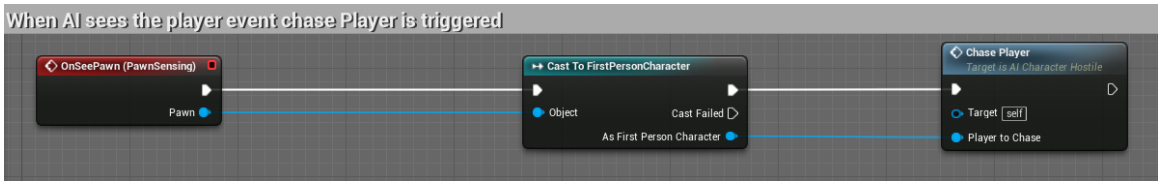
Diagram below shows the two types of enemies

AI PawnSensing

Pawn sensing is a component build in unreal engine 4 that aids the AI with sight and hearing detection. The default pawn sensing peripheral vision is 90 degrees but since this is a casual game, the AI has 45 degrees of peripheral vision giving the player a chance to sneak behind enemies.



Below is a blueprint class that tells the AI to chase the player's character whenever it inside the pawn sensing range of the AI.

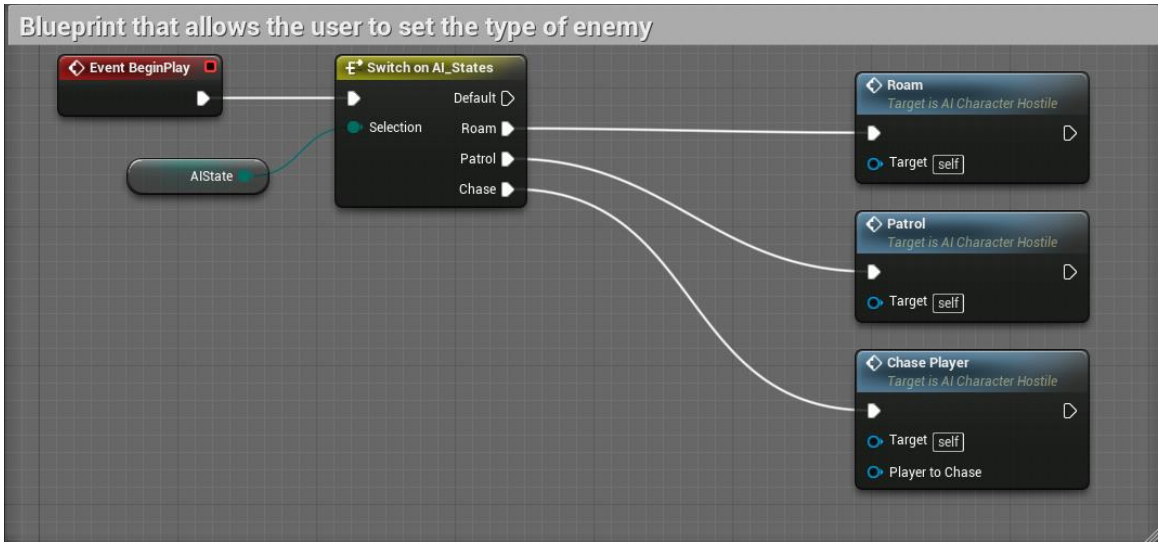


AI Navigation

The AI uses a navigation mesh or "NavMesh" in unreal engine 4, NavMesh is a data structure that aids the AI in unreal 4 engines in pathfinding.

The diagram below show where the AI can go or pathfind any area in green means that the AI can go or navigate to that area.

AI Blueprint

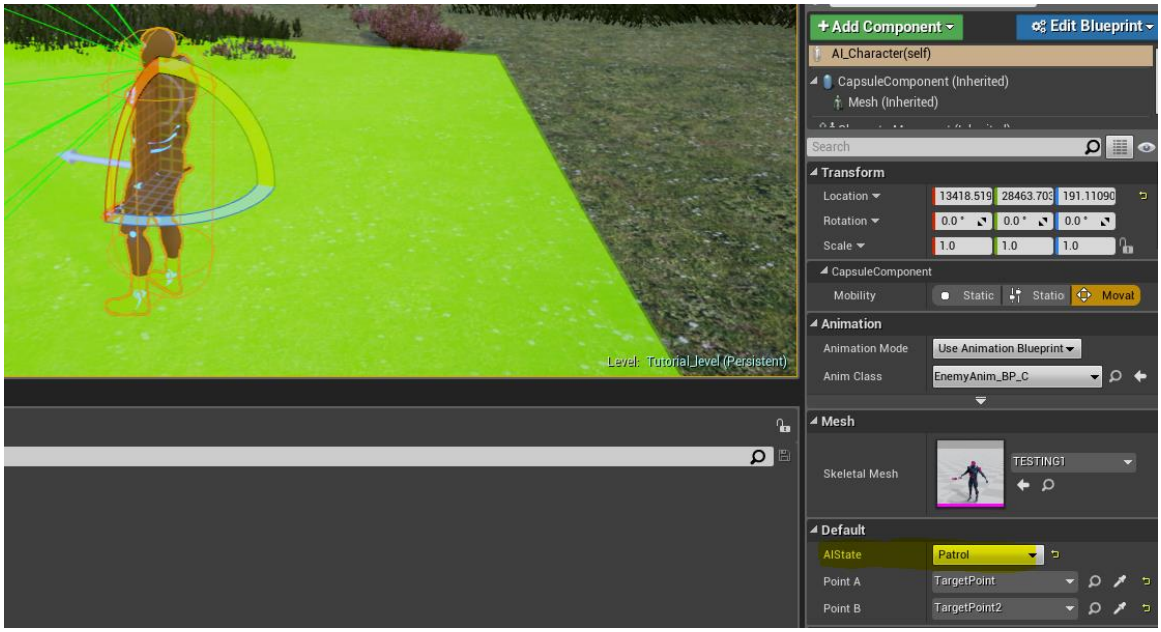


The event beginplay is always triggered whenever the game is loaded and running.

We have a switch so we can select what type of behavior our AI is going to do in our game, for this project we have four deferent states. We have Default, Roam, Patrol and Chase.

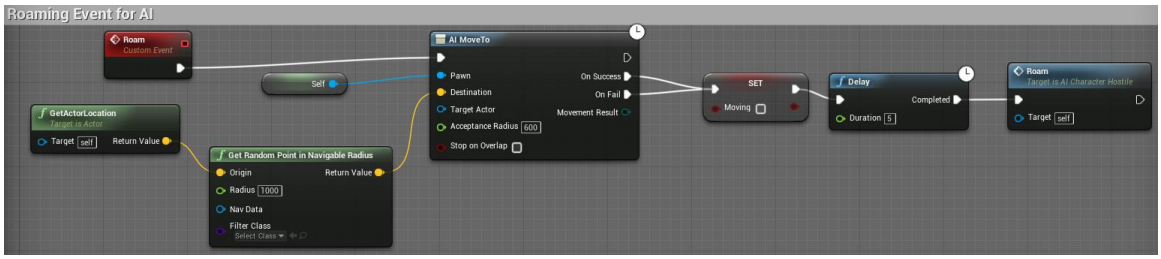
- Default – The AI will stand still until it sees the player.
- Roam – The AI will roam randomly on the map and when it sees the player it will chase the plater
- Patrol – The AI will patrol an area between two points.
- Chase – The player will chase the player until it dies or loses sight of the player.

We can set the AI states on an individual setting on the editor as show in the diagram below

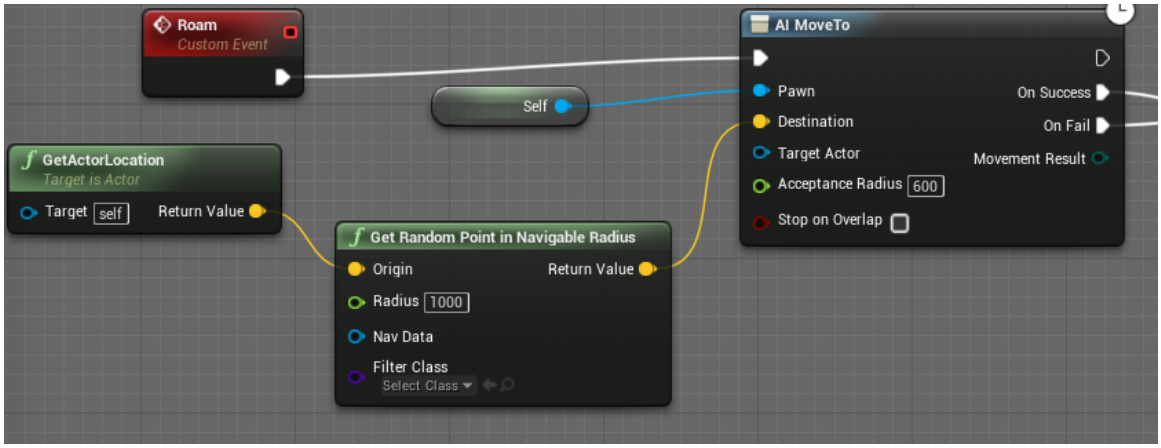


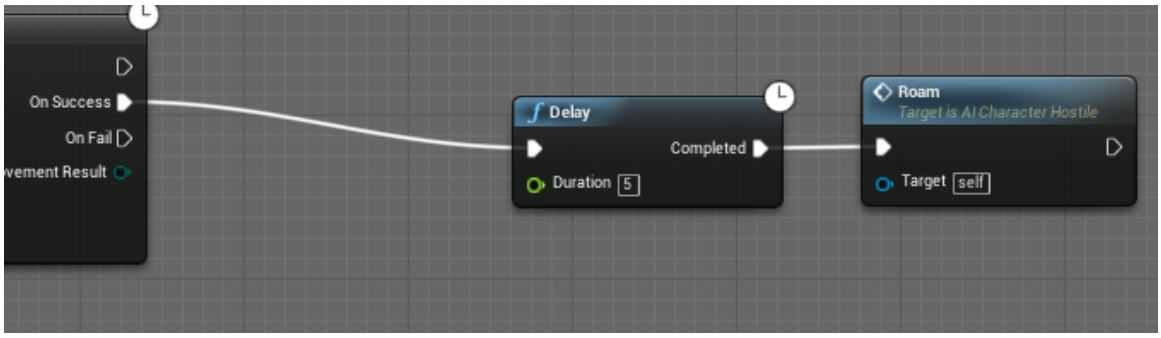
Roam Blueprint

Our roam blueprint allows our AI to roam around the map getting random points every 5 seconds.



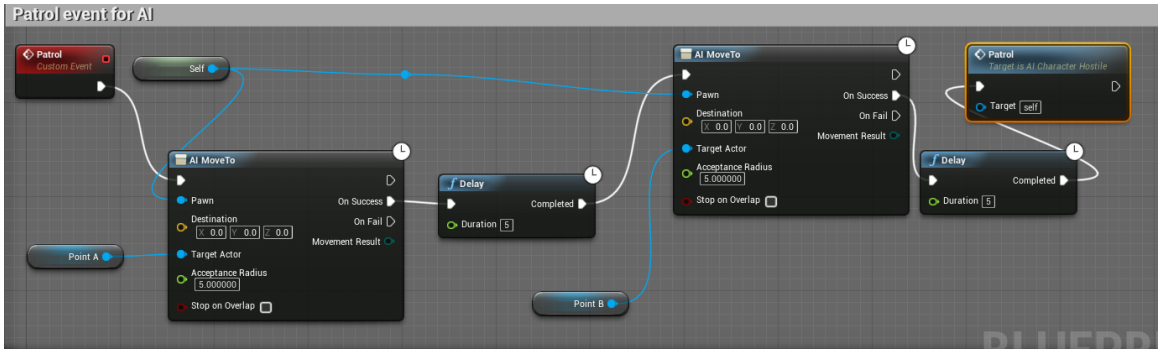
The “AI MoveTo” Node decides where the AI moves to and we use a “get random point” node to randomize the destination of our AI. There is also a delay of 5 second implemented before the AI moves again so that the player can see where the AI is heading.





Patrol Blueprint

We can use target points to set our AI’s route in game. Each AI can roam between two points on the map.

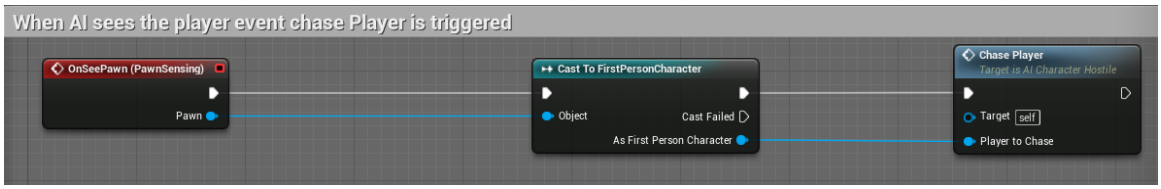


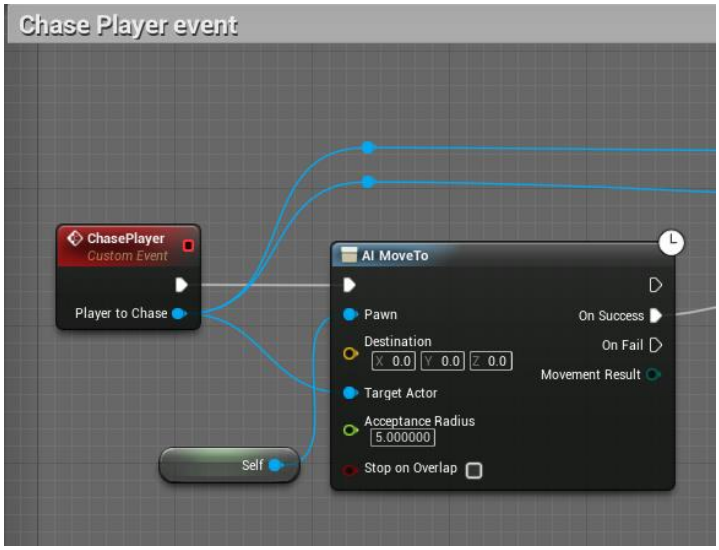
We have two “AI Move To” nodes one for Point A and one for Point B. When the AI has reached the target destination at point A the delay node gets activated and the AI waits for 5 Seconds till it moves to Point B and Vice versa.

Point A and Point B are Public Variables than can be set individually in game by the developer.

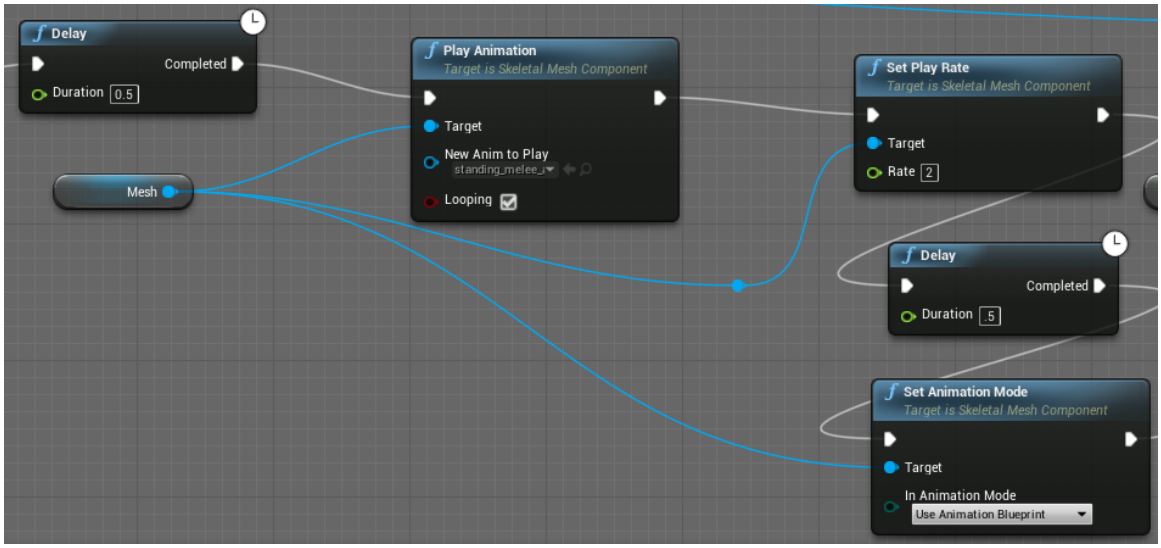
Chase Player

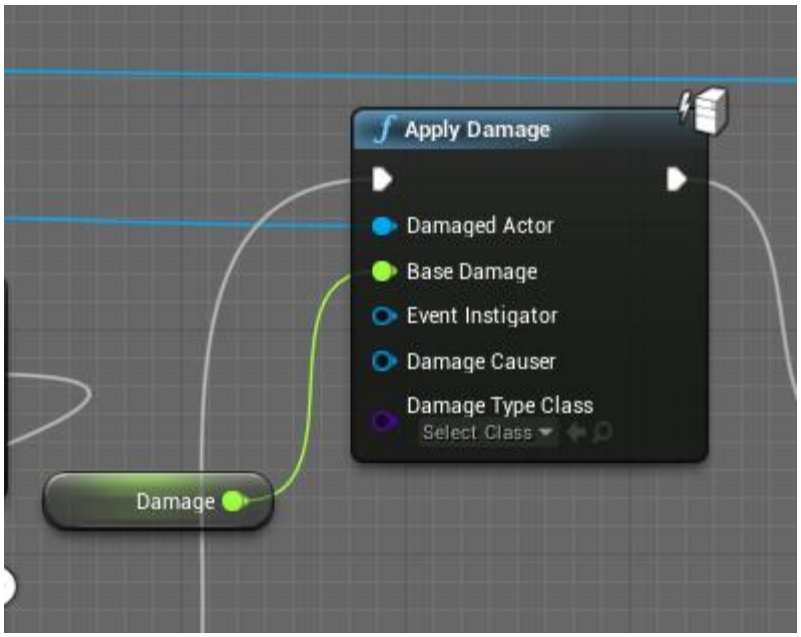
The chase blueprint for the AI is a bit more complicated than the other AI states. The Chase player is only triggered when the player is seen by the AI.





The AI will travel to the player's location whenever the "ChaseEvent" is triggered and when the AI reaches the player, an attack animation and damage event will be triggered.

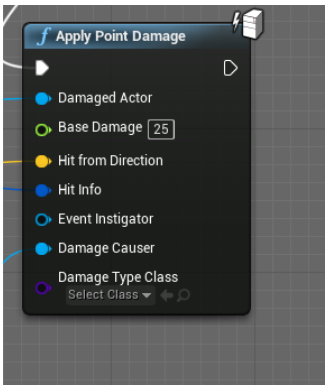




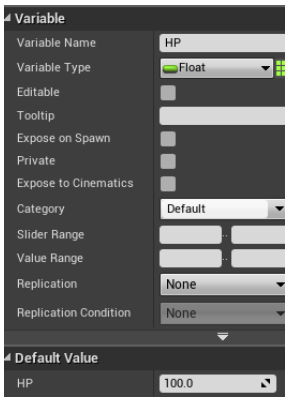
AI Take Damage

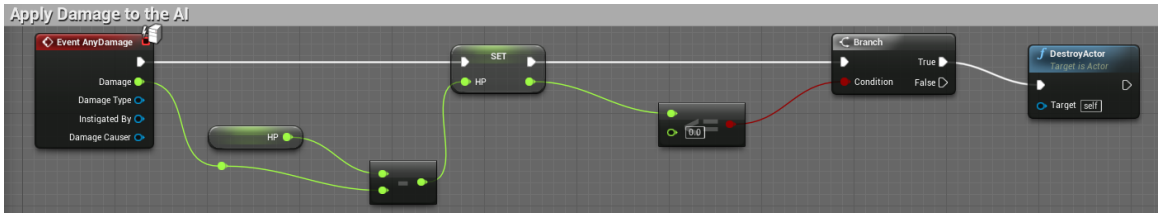
Since the AI can damage the player a system for damaging and destroying the AI is also implemented.

Diagram below is taken from the character controller for the player to show the base damage.



We also have an integer variable called HP that is set to 100.



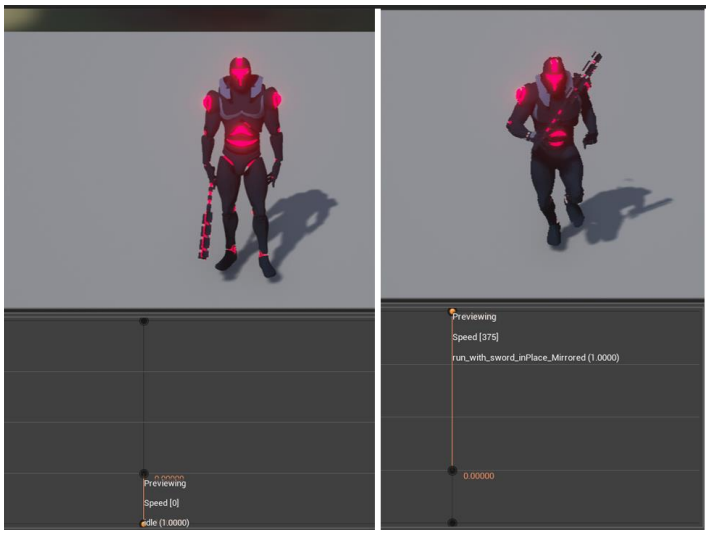


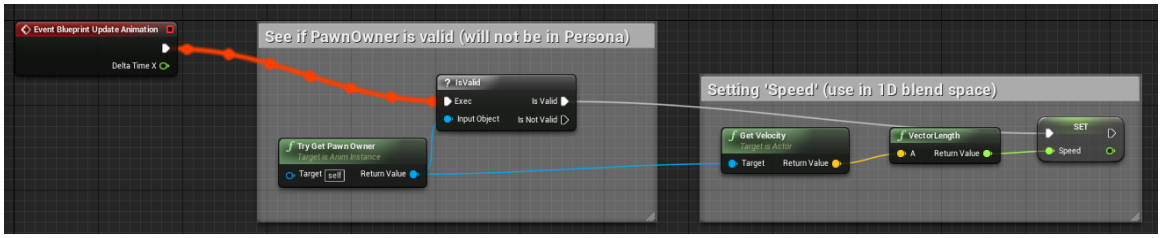
The “Event AnyDamage” node is activated whenever the AI is hit by the player. We calculate our players HP by subtracting the incoming damage to our HP variable which is an integer. We also have a condition where if the HP is less than or equals to Zero the Actor is destroyed killing off out AI.

AI Animations

Blendspace - Enables blending of animation for smooth transitions.

The enemy AI uses a blend space for its animation. The AI can smoothly transition from running to idle easily.

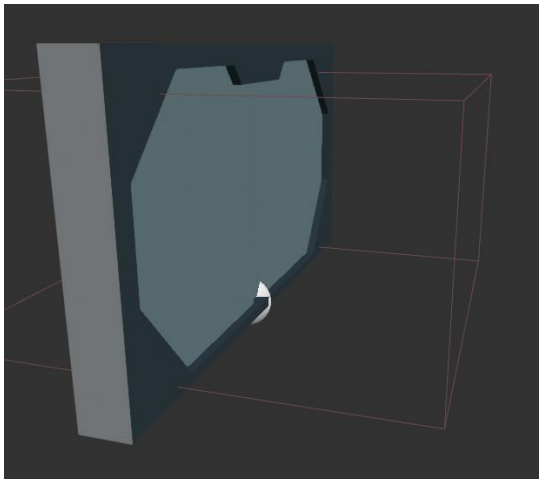




Blueprint for setting the speed setting for the AI

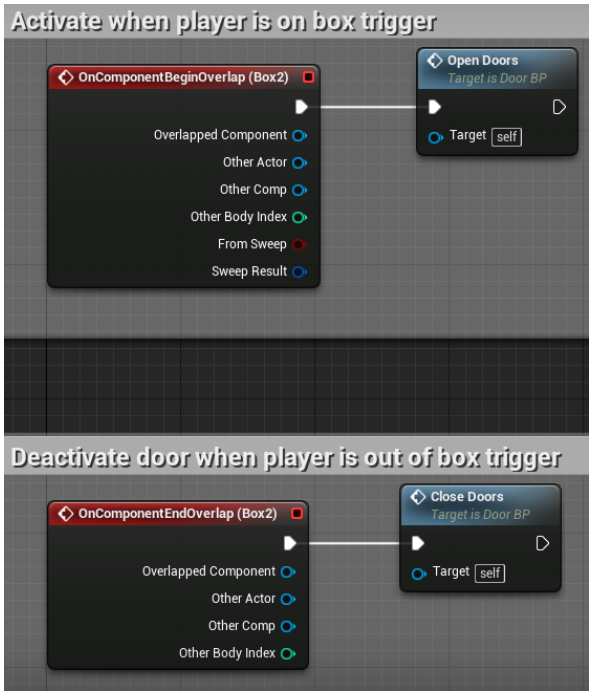
3.2.8 Door implementation

Door blueprint actor:

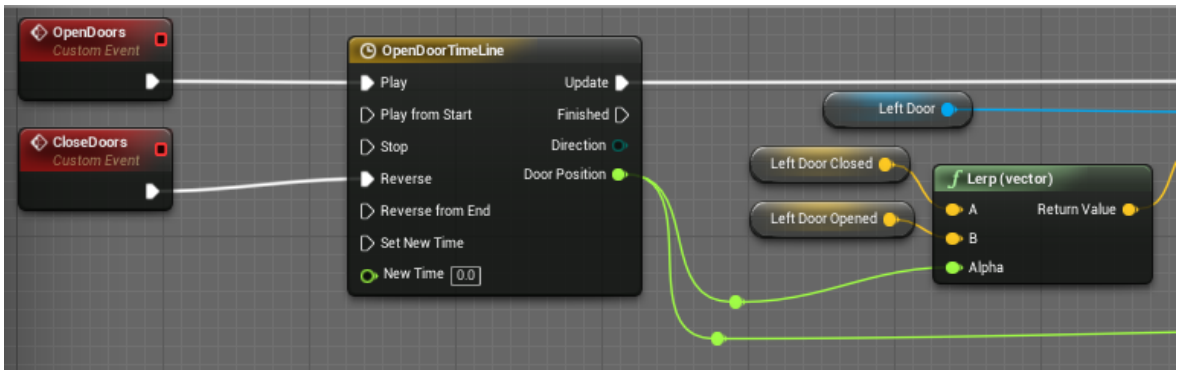


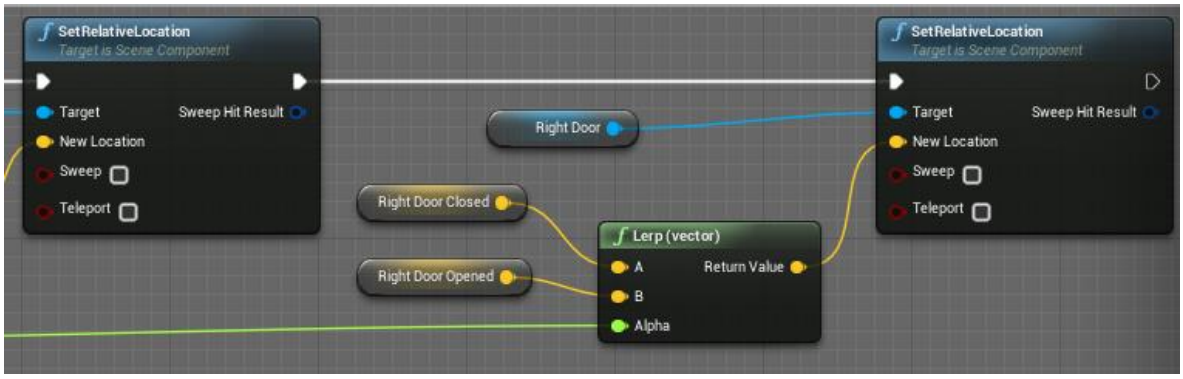
The door opens whenever the player steps inside the box trigger show above.

Below is the overlap event that senses when the player has stepped inside the area.



The open and close door events are custom events, these events use a timeline node. This timeline node sets the door position by using the vector values “Left/Right Door Close” and “Left/Right Door Open”. Depending on whether the timeline is revered or played the doors will open or close.





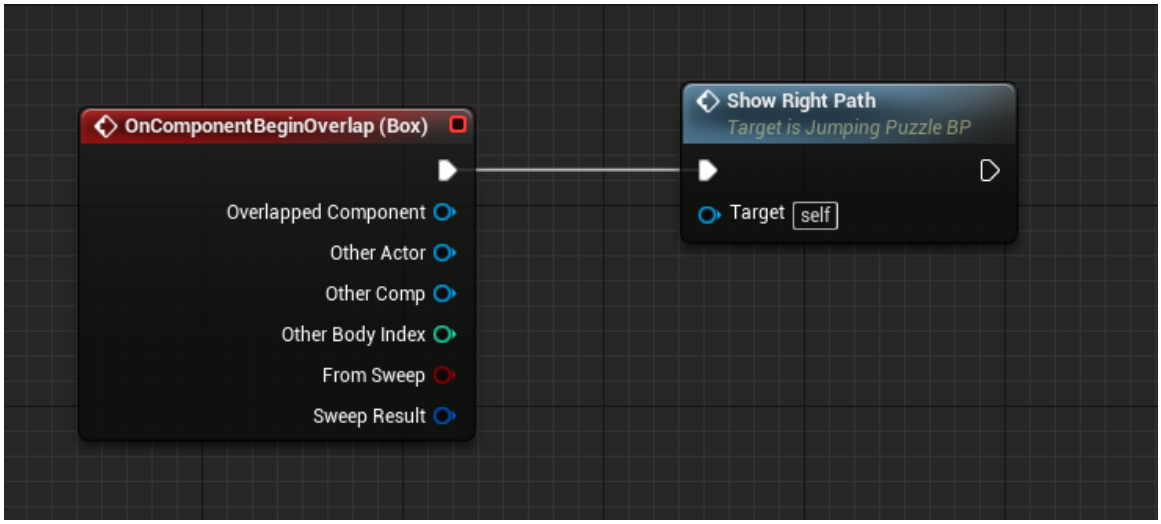
3.2.9 Platform Implementation

As seen in the screenshot an easy platform is implemented to challenge the player.

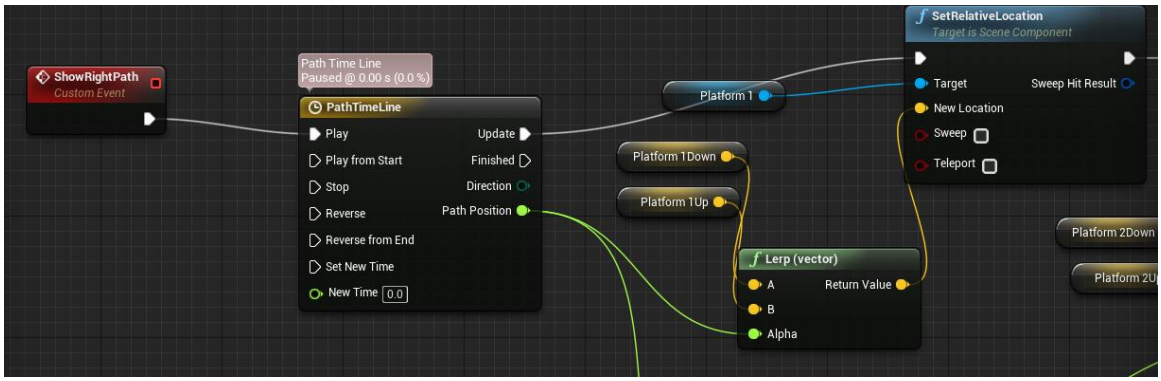
The platform is a simple puzzle that allows the player to cross a big gap.

Box trigger is show on the left-hand side of the screenshot above

The current implementation of the puzzle is very simple to keep it stream lined, the platform is activated when the player overlaps a box trigger by either standing on it or placing an object on it.



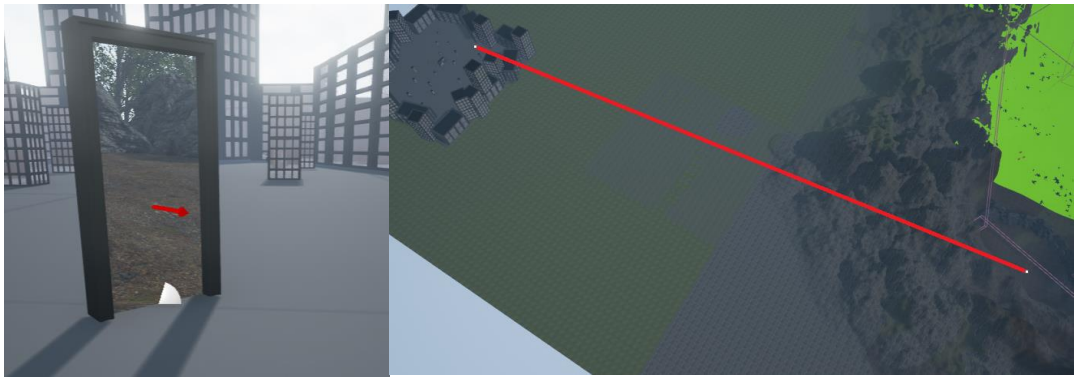
The logic is simple when the box trigger is activated a timeline node will be activated changing the relative location of the platforms. In this case the platforms will move on the Z axis.



3.2.10 Portal Implementation

There are two portals in game which teleports the player into two different zones. This is an Easter egg and is only available when the player find it.

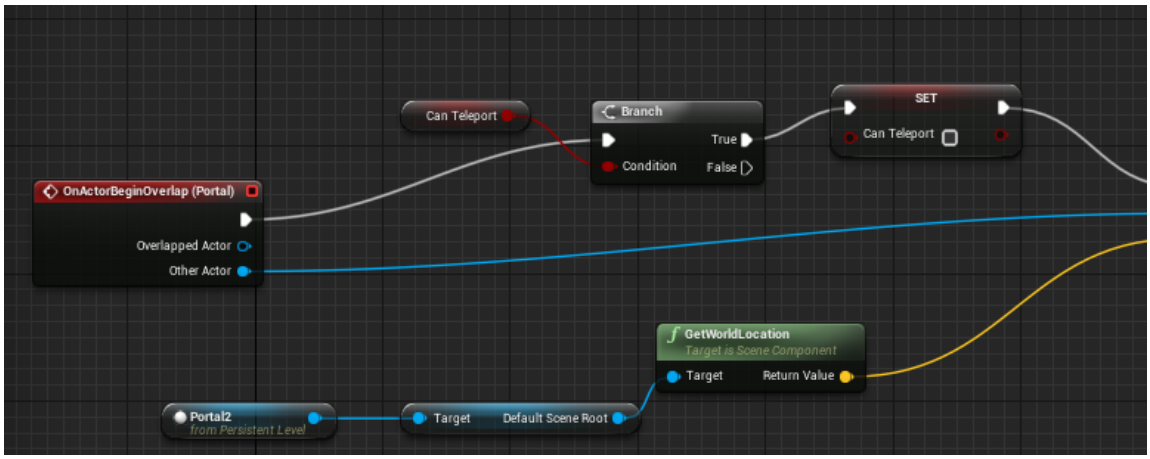
The teleportation is instant between the two portals. A live preview of the location of the door way is also shown so that the player can see what the area looks like before they teleport.

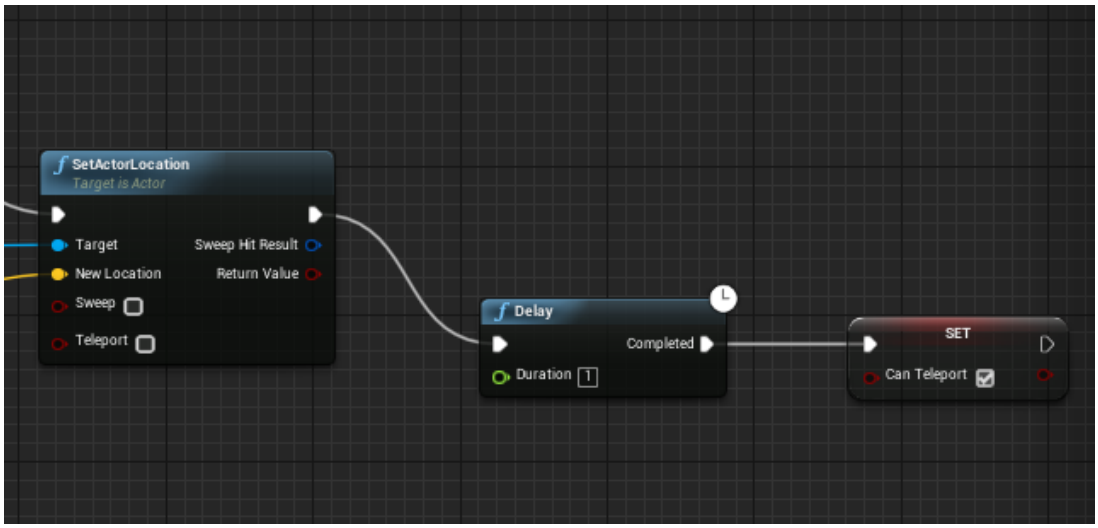


Blueprint for teleporting

The player is transported to the other portal whenever the player touches the doorway portal. To do this we need an overlap event so when the player touches the portal it will fire off an event.

The first thing we need is to get the location of portal we want to teleport to. In this case, the name of the portal is portal 2. We use a “get world location” node to get the portals location and then we need to connect it to the “new location” value in the “set actor location” node to set our character’s new location in game.





We just copy this function for Portal one.

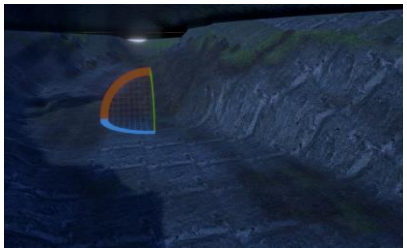
But this will result into an infinite loop and unreal 4 is smart enough and will not compile. Therefore, we have the delay node and can teleport condition. This stop the infinite loop and gives the player time to get out of the portal without being transported back.

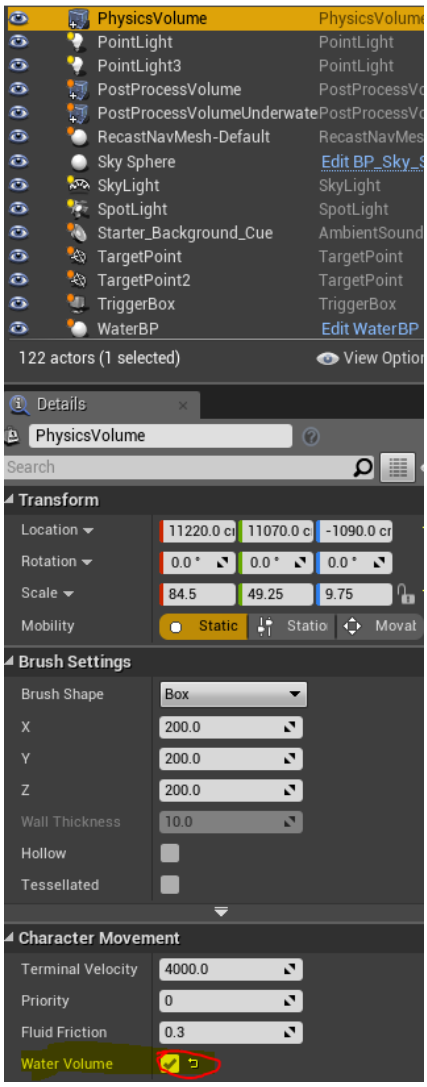
3.2.11 Swimmable water implementation

The swimmable water is implemented using a physics volume. Physics volumes are implemented in unreal engine and can be used to manipulate the world physics in games.

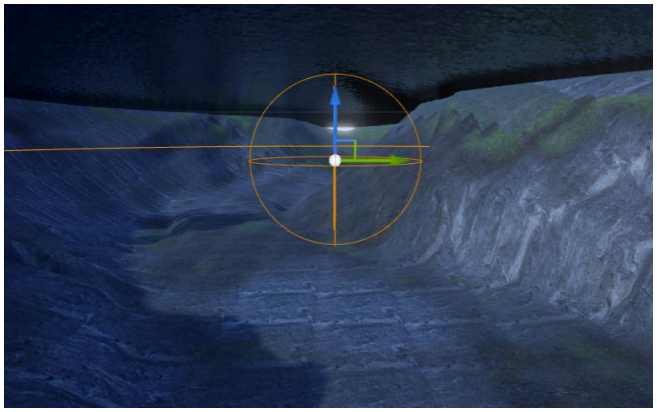
There is an option for a water volume in the physics volume in unreal engine. This option simulates water physics as accurately as possible the user can also change the physics values such as friction velocity etc.

To give the illusion of underwater a post process volume is used to simulate what it looks like under water. Post processing changes colors and lighting within a game giving the developer freedom to change what the game looks like.





An audio cue is also present used so that the player can hear underwater sounds. This audio cue is placed within the physics volume and is limited by range.



Lastly, we just use a simple water texture to cover the top of the physic volume.

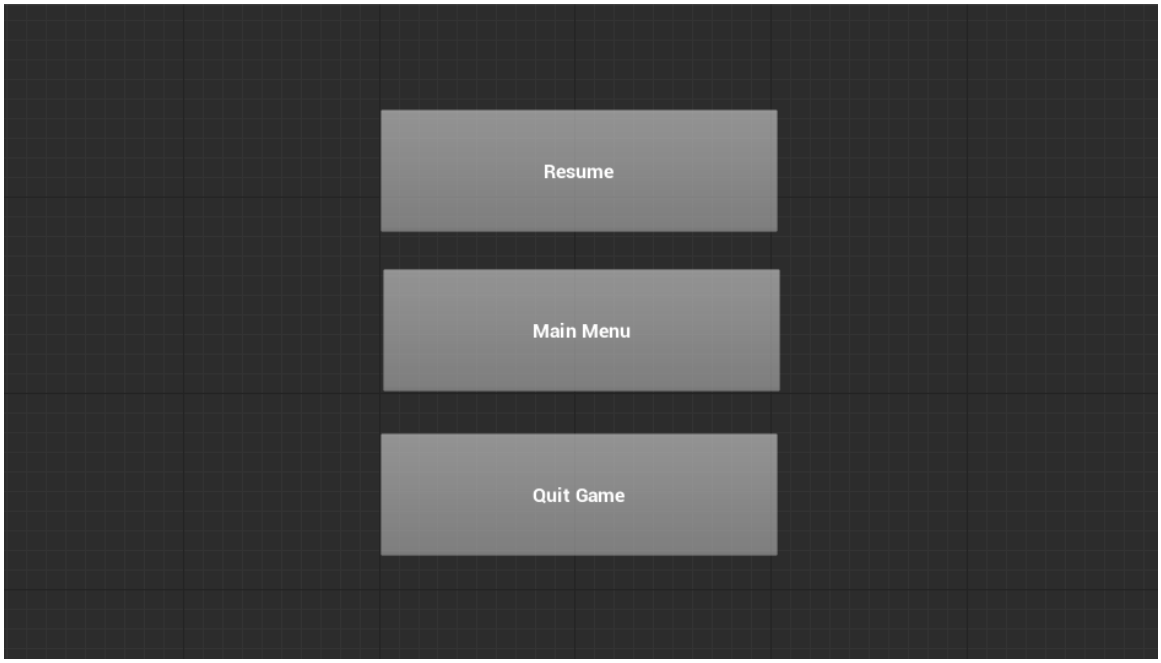
3.2.12 Graphical User Interface (GUI) Implementation

Main GUI

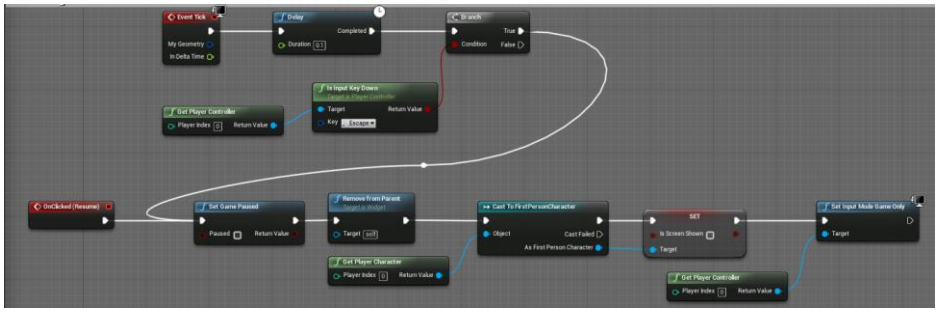
The main GUI shows a health bar for the health amount and a crosshair to show where the players bullet will hit. The GUI is very minimal to reduce the clutter.

Pause Menu

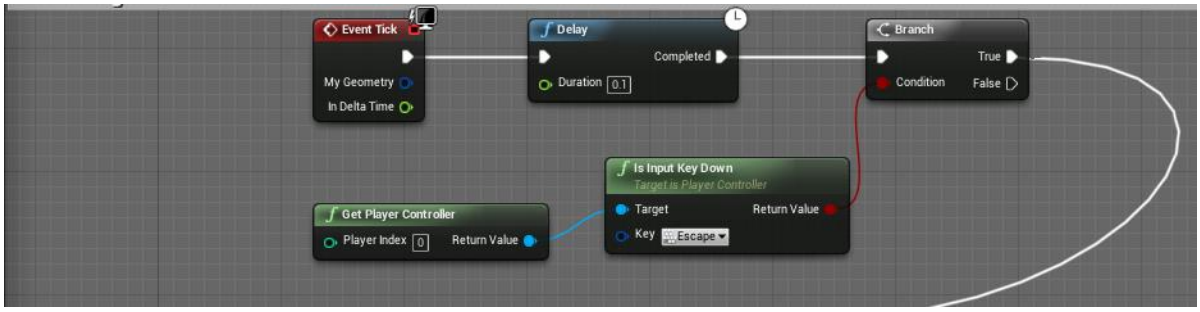
The pause menu consists of three buttons resume, main menu and quit game.



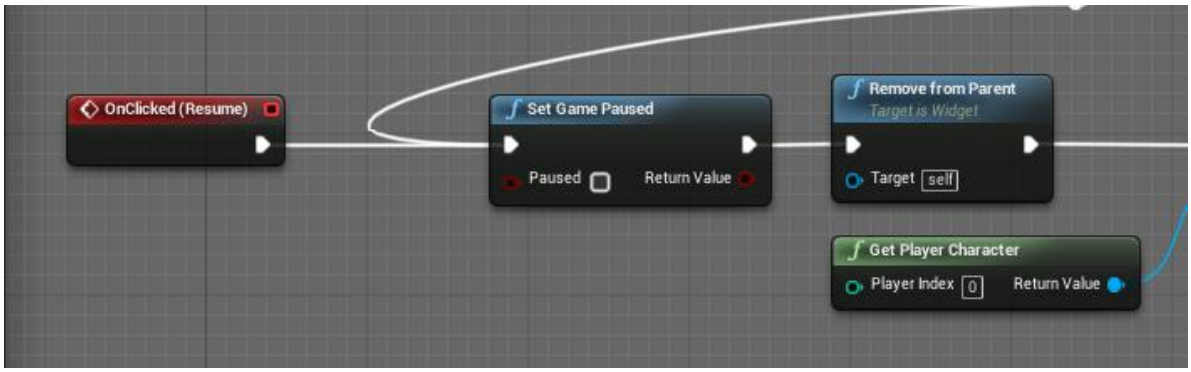
Full View of the blueprint

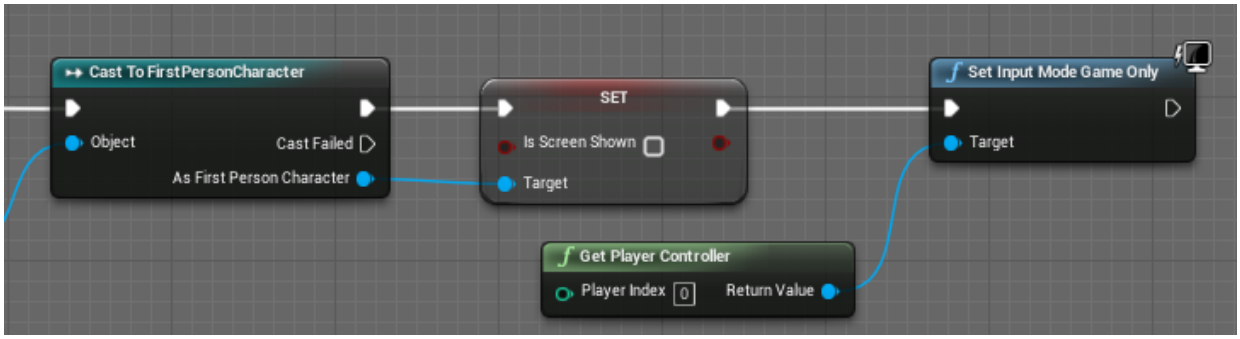


We have an event tick to check if the button is pressed to set the game in an unpaused state.



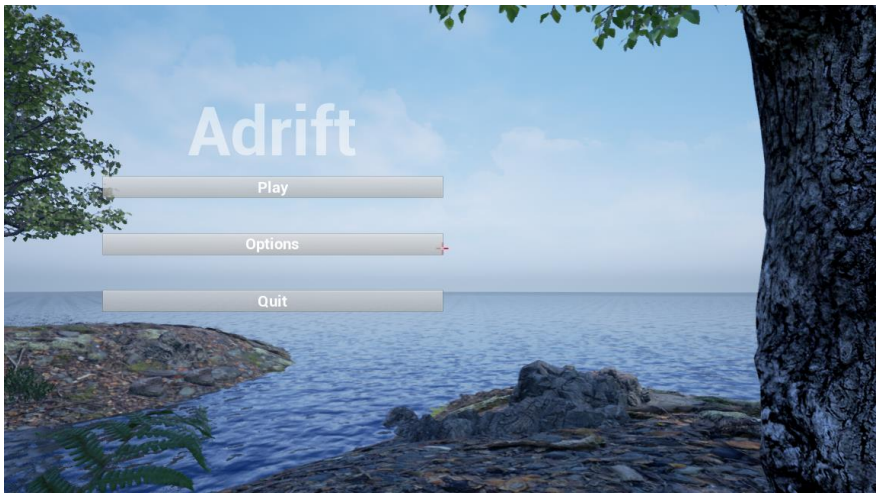
When the “Resume” event is triggered the game is unpaused and vice versa.





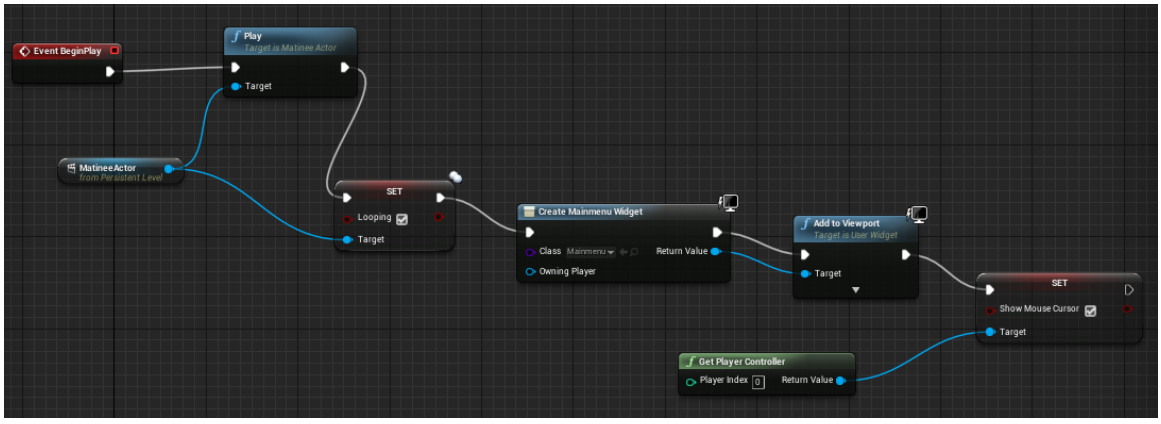
Main Menu

The project uses a live main menu which means that the background move. This is done by using unreal matinee cinematics.

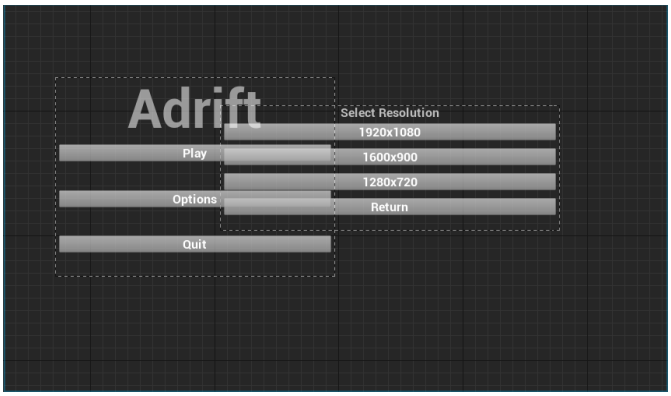


Matinee in unreal 4 is used for cinematics and moving actors in the game without the need for AI.

The Matinee actor is called when the game starts this overrides the player controller so that the mouse is only show and the player can move. This function is looped until the player selects or starts up a level.



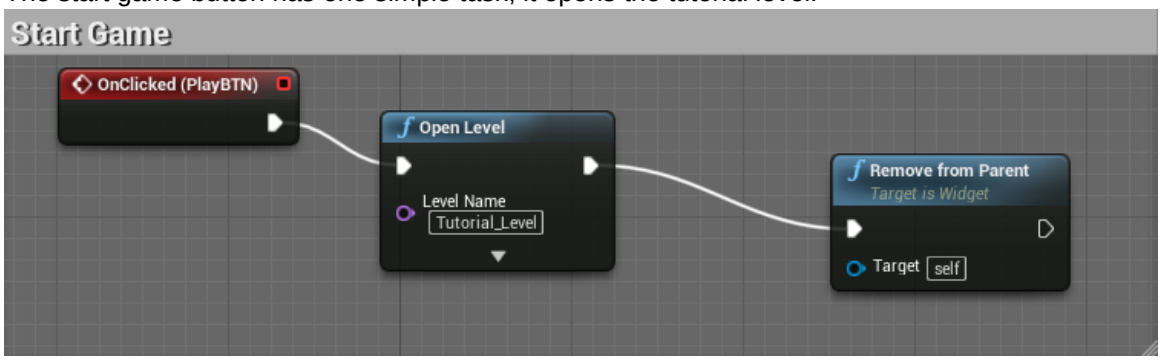
The main menu consists of three main buttons Play, Options and Quit. The Option has four children buttons as seen in the picture below.



Blueprints for the main menu

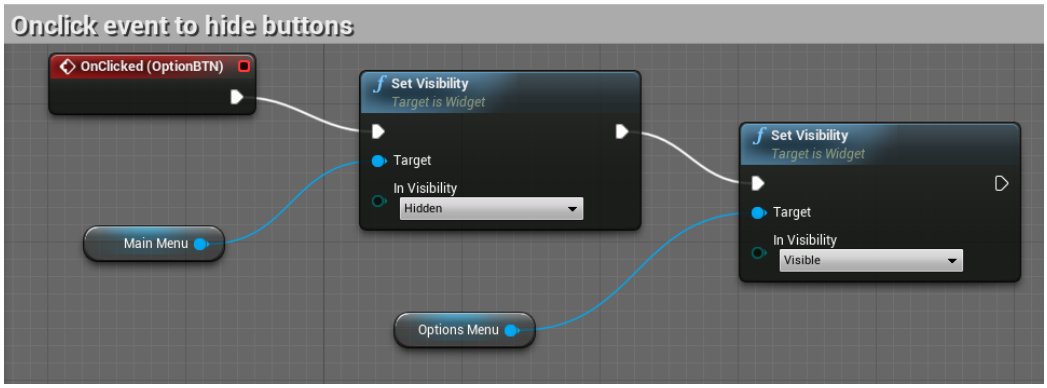
Start Game

The start game button has one simple task, it opens the tutorial level.

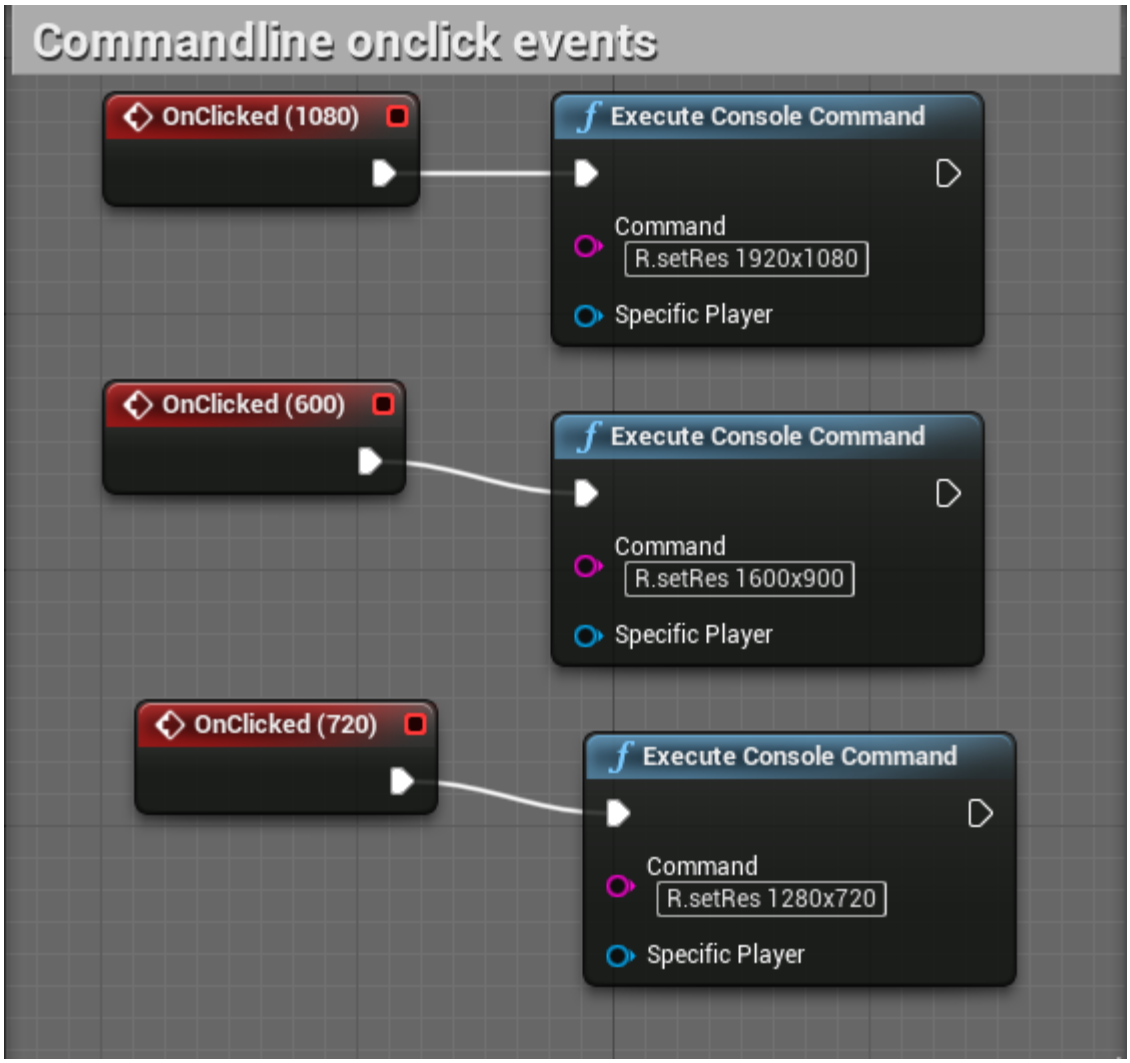


Option

When the option button is clicked a “set visibility” trigger is activated showing extra hidden buttons.

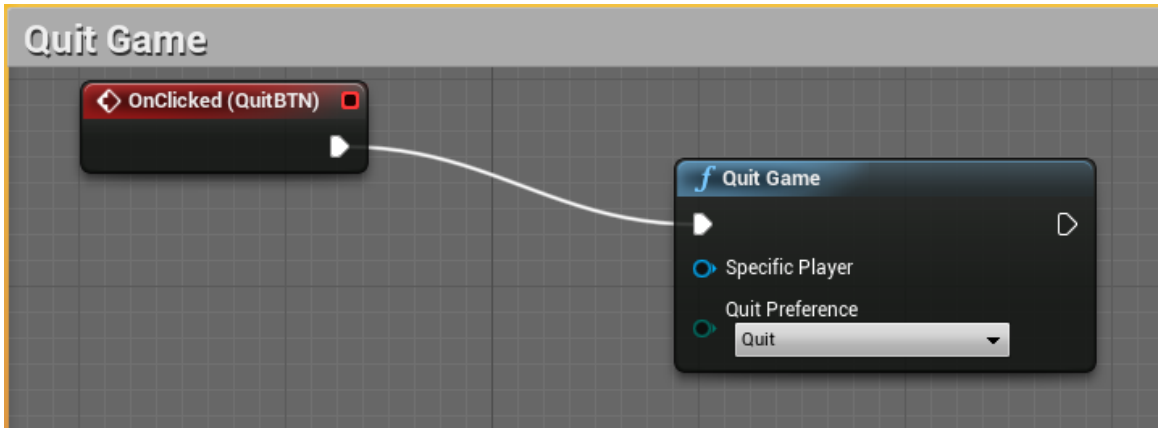


The extra buttons are resolutions options that the player can select. A console command node is used to set the game resolution.



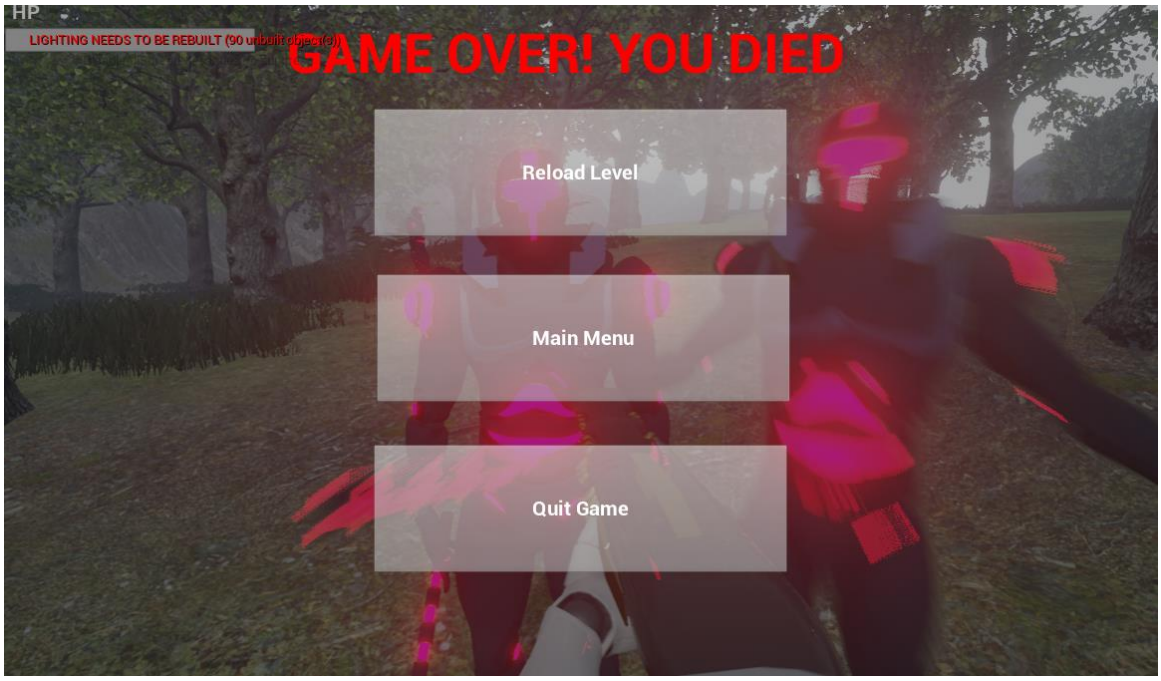
Quit Game

The quit button only terminates the game. This function is already inside unreal and only needs to be called.



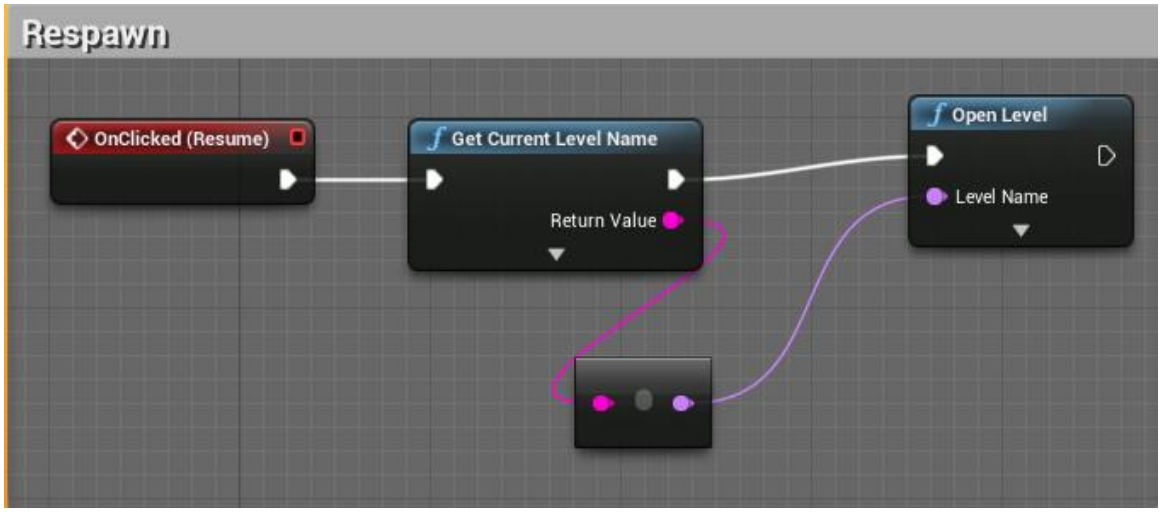
Game Over

Game Over Screen in game



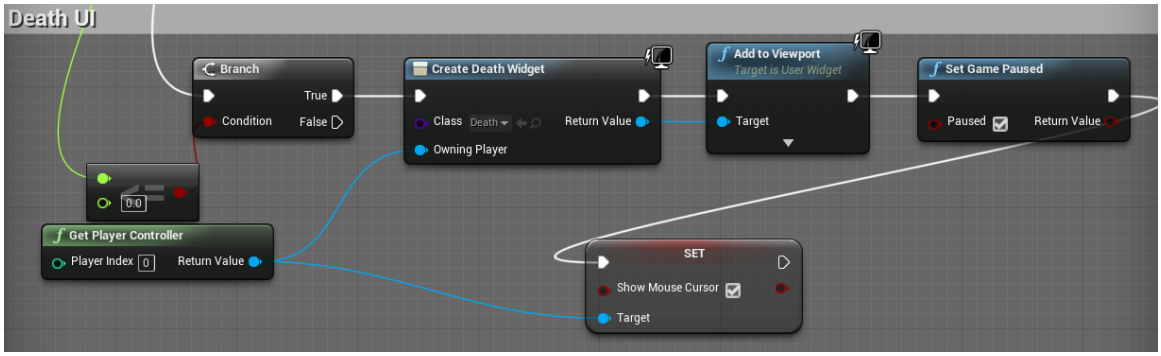
Widget Blueprint for reloading the current level the character is on

The “Get current level name” as the name suggests collects the current levels name and whenever the reload level button is clicked the current level is opened and reloaded.



Taken from first person blueprint

The logic is simple when the player health is zero, a new widget containing the game over UI will be shown to the players view and the game will be paused until the player selects an option.



3.3 Testing

Testing is important in game development, testing is done to find bugs and to see if the implemented features are working correctly.

For this project, I've elected to use three main methods of testing.

- Usability Testing
- Unit Testing
- Customer/Public Testing

Tools used in the testing process of Adrift:

- Unreal Engine 4 Preview/Simulation: Unreal engine 4 Editor provides a simulation mode to test in game logic.
- Unreal Engine 4 Test Levels/Projects
- Alpha/Beta Build
- Online Surveys

3.3.1 Usability Testing

Insuring that a product is working as intended is very important a usability test study has been conducted for the project to identify problems, gameplay glitches and user experiences.

There were two main questioners/survey used for the usability study, the first one was intended for the early game development phase and the second survey was for the alpha/beta build and was only answer after playing the game.

Questionnaire 1 During Early Development

Online Survey done during early game development

Question 1)

Do you prefer a big open map design (Example: Crysis) or small sectioned map design (Example: Call of Duty)?

- Open Design
- Small sectioned Design
- Mix of Both

Question 2)

Did you feel easily feel lost when the game doesn't help you or guide you to a certain objective?

- Yes
- No

Question 3)

Do you prefer a hardcore or casual gaming experience?

- Casual
- Hardcode

Question 4)

Do you prefer third person or first-person games?

- First Person
- Third person
- Mix of both

Question 5)

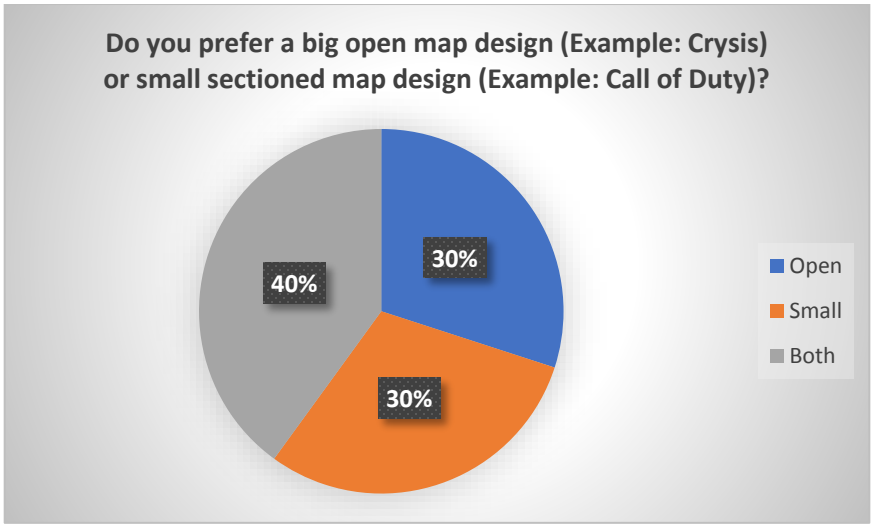
Would you want to play a virtual reality game?

- Interested
- Not interested

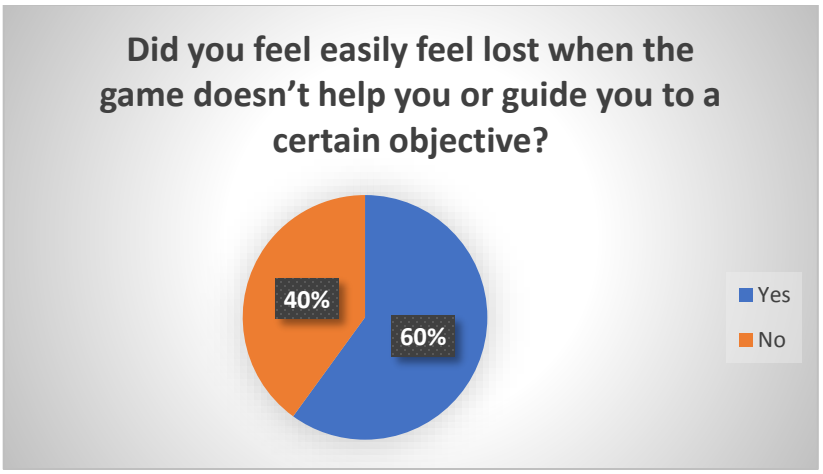
Results of the online survey

10 responses were received for the online survey

Question 1 Results

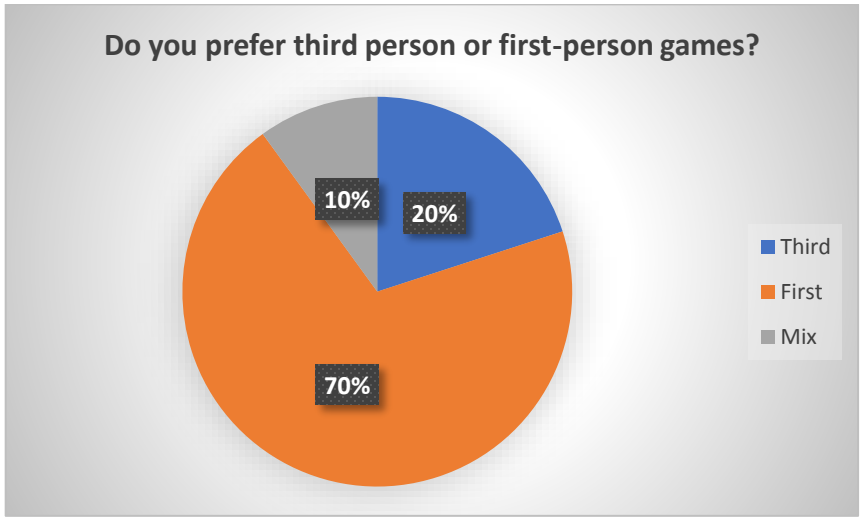


Question 2 Results



Question 3 Results

Question 4 Results



Question 5 Results

Results of online survey

Based on the results of the online survey users were split between an open worlds design and small level based design. The users also need objectives so they don't feel lost and the users also prefer playing first person games over third person games.

Furthermore, users prefer a casual experience over a hardcore experience and are split on VR games.

After the online survey and a tiny bit of development a think aloud test was done for a prototype build.

Feedback received after a small think aloud testing with a prototype.

Only two participants took part aged from 20-22

The testers were asked a couple of questions during the testing

Question 1

How did you like the first level?

"The map design is great but I felt lost"

"I felt lost during the open section of the map"

Question 2

How was the controls?

"The controls are okay very familiar"

"Controls is what you expect from a modern game"

Question 3 (Another level was I loaded to compare the first level)

How does this map compare to the first one?

"The second map was better than the first one, there were tiny sections that lead me into open areas of the map I still felt lost for the most part but I had somewhere to go"

“I like that it started with a small area and benched out unlike the first map”

Results of the prototype think aloud analysis

Based on the results we can safely assume that the second level with small sections was preferred over the completely open level and the controls were also familiar.

Questionnaire 2 alpha beta built

A second questionnaire/think aloud was conducted based on an alpha build of the game. The survey was focused on getting feedback on the usability of the controls and the quality of the map design

Questionnaire

Please answer the questioner below (5 being the best Rating and 1 being the worst)

Overall how would you rate the level design of the first level?

- 1
- 2
- 3
- 4
- 5

Overall how would you rate the design of the second level?

- 1
- 2
- 3
- 4
- 5

Overall how would you rate the design of the third level?

- 1
- 2
- 3
- 4
- 5

Overall how would you rate the controls of the game (Mouse and Keyboard)?

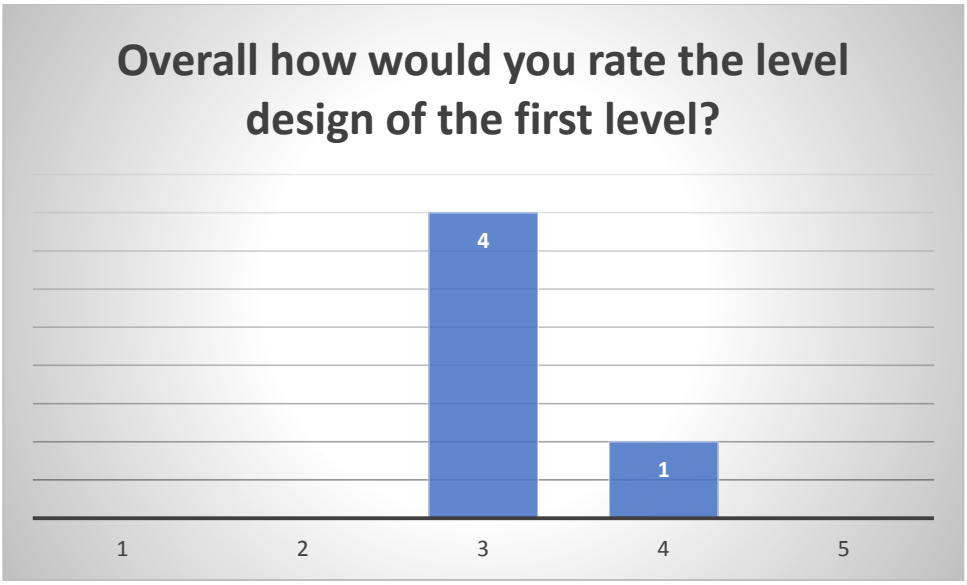
- 1
- 2
- 3
- 4
- 5

Overall how would you rate the controls of the game (Controller)?

- 1
- 2
- 3
- 4
- 5

Question 1

Overall how would you rate the level design of the first level? (5 Good – 1 Worst)

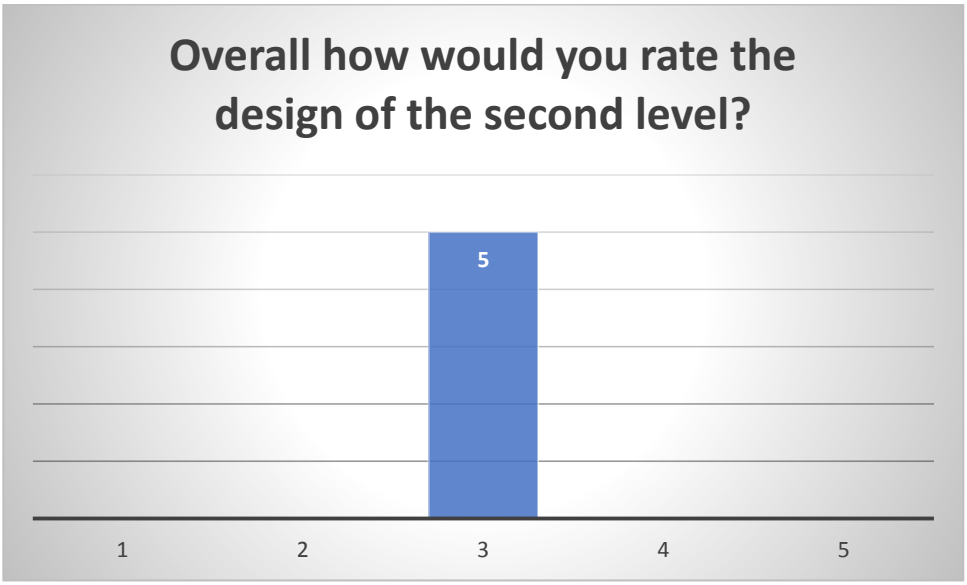


Analysis: The map design seems to be decent or good based on the gathered results

No problem has been reported.

Question 2

Overall how would you rate the design of the second level? (5 Good – 1 Worst)



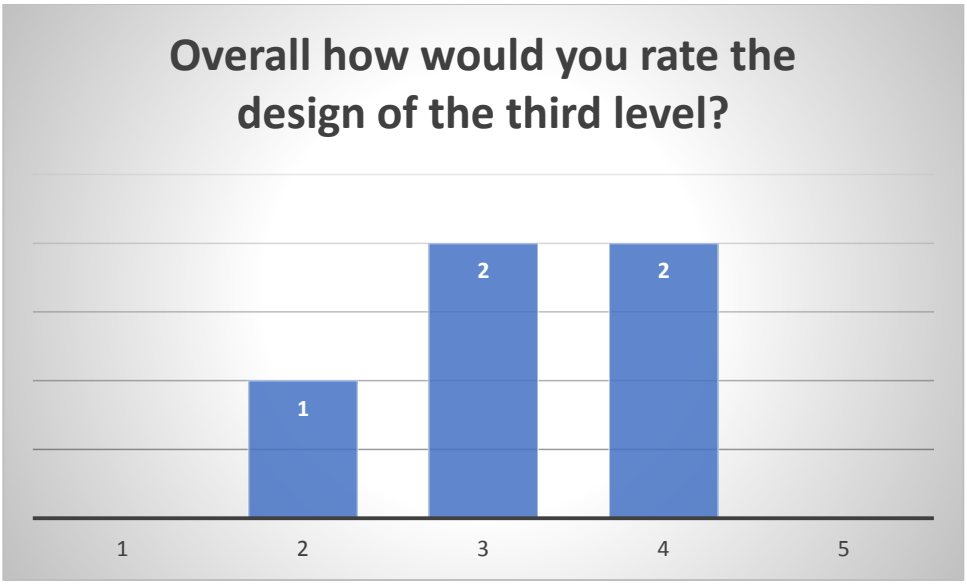
Analysis: The second level is neither great or bad based on the results.

Response taken during the think a loud

“The level looks good, but it was too short for me”

Question 3

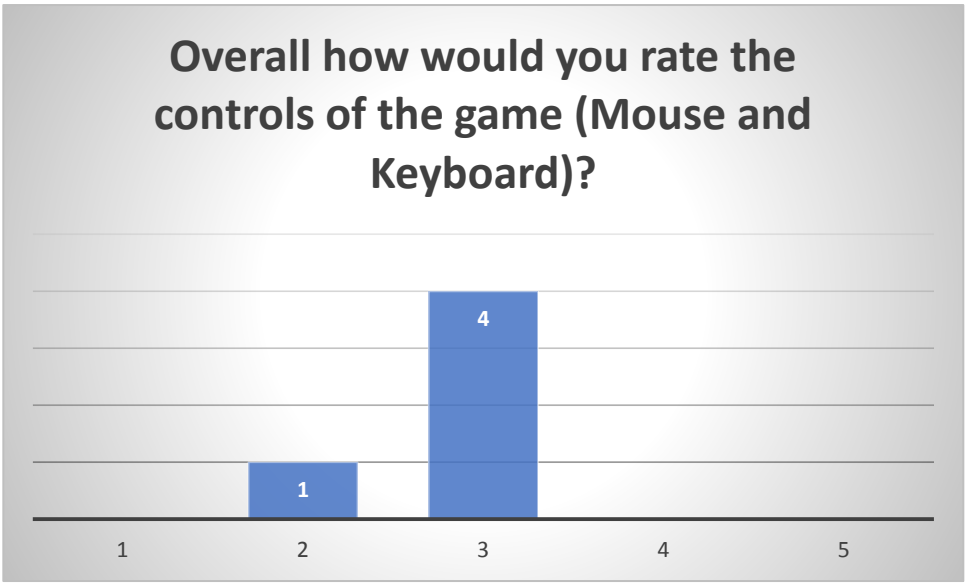
Overall how would you rate the design of the third level? (5 Good – 1 Worst)



Analysis: Most of the response agree that the level is good and we only received one bad rating

Question 4

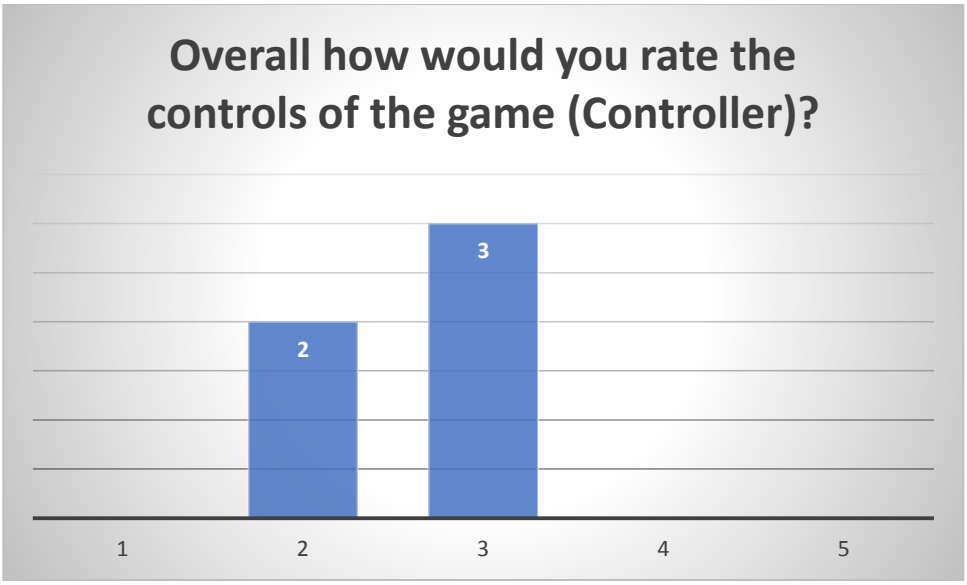
Overall how would you rate the controls of the game (Mouse and Keyboard)? (5 Good – 1 Worst)



Analysis: Most the testers thought that the controls were sufficient only one in five thought that the controls were not good.

Question 5

Overall how would you rate the controls of the game (Controller)? (5 Good – 1 Worst)



Analysis: at least two people didn't like the controls since it scored 2 out of five.

Response taken directly from the think aloud test

"The menu did not respond to the controller, I had to use the mouse."

Problem that needs a solution:

Menu needs controller support.

3.3.2 Unit Testing

Unit testing is a huge part of development. To ensure that certain functionalities were developed properly they have been built and tested in isolation.

Tools and methods used for the unit testing:

- Unreal Engine 4 Preview/Simulation: Unreal engine 4 Editor provides a simulation mode to test in game logic.
- Unreal Engine 4 Test Levels/Projects
- Game Testing on another Machine

Unit Test 1: Mesh and Skeletal Mesh quality control

Tools used: Extra windows machine and test project

Reason for unit test: To ensure that the meshes or skeletal meshes were safe to use and did not cause a problem. Importing bad or incorrectly formatted meshes can be damaging for the unreal engine since it can cause certain crashes in game.

Steps taken:

1. Create and launch identical project on another machine
2. Import the mesh into unreal engine
3. Inspect the mesh inside unreal engine
4. If the import was good and did not have a problem delete within the test project and use in main project.

5. If the import is bad and comes up with error messages. Re import the mesh with different settings from the 3D editor choice and retest.

Unit Test 2: Weapon/Combat System Check

Tools used: Unreal Engine Test project with Simulation Mode on another machine

Reason for unit test: Check to see if the combat system is working properly and the main player can damage and destroy AI or objects in game.

Steps Taken:

1. Create and launch identical project on another machine
2. Copy over the first-person blueprint from main project
3. Compile and Save the project
4. Import AI and blueprints from main project
5. Place AI in the level
6. Activate simulation mode
7. Shoot AI till its destroyed
8. If AI is destroyed remove from test project and use in main project.
9. If AI is not destroyed, check blueprints and test again.

Unit test 3: AI Check

Tools used: Unreal Engine Test Project and Simulation mode on another machine

Reason for unit test: AI is an important component of the game and needs to be tested. For the unit test, we check if the AI is doing the roam, patrol and chase behaviour.

Steps Taken:

1. Create and launch identical project on another machine
2. Copy the AI blueprint along with animations
3. Compile project and save project
4. Place 3 AI Agents on the map
5. Launch Unreal Engines simulation function
6. If the AI set to the roaming behavior = roams randomly on the level then the AI is working perfectly.
7. If the AI set to the roaming behavior stands still and does not roam the behavior needs to be fixed and retested again
8. If the AI set to the patrol behavior patrols two points on the map then the behavior is implemented correctly and is working.
9. If the AI set to the patrol behavior doesn't not patrol the two given points then the behavior is broken and needs to be fixed and retested.
10. If both AI agents Chase the player when the player walks in front of them the behavior is working as intended and can be used in the final project
11. If the AI agents does not chase the player then the behavior needs to be fixed and retested.

3.3.3 Customer testing

A customer testing testing/Study has been carried out towards the end of the development cycle to gather feedback and criticism of the final version of the game.

This study will determine the future systems and improvements for the project.

Number of Participants: Six (18 – 24)

Questionnaire:

Question 1

Overall how satisfied were you with the product?

- Satisfied
- Somewhat Satisfied
- Neither
- Somewhat Dissatisfied
- Dissatisfied

Question 2

How would you rate the quality of the product? (5 being good and 1 being bad)

Question 3

Would you play again if more levels are added and the game is developed more?

- Yes
- No
- Maybe

Question 4

How much would you pay for the product?

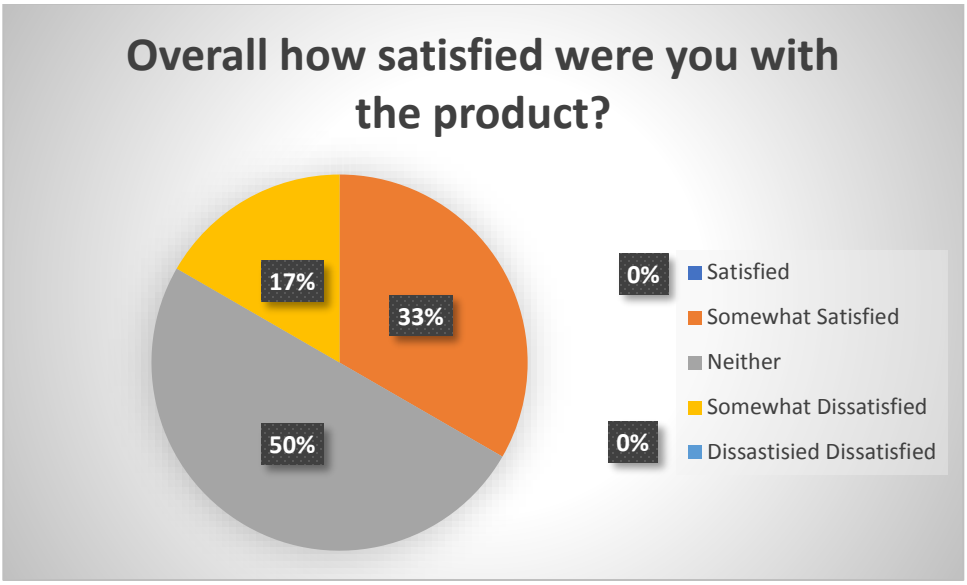
Question 5

What words would you describe the product with? Select all words that apply

- Fun
- Immersive
- Interesting
- Alright
- Needs more development
- Unfinished
- Poor Quality
- Unreliable

Results

Question 1 Overall how satisfied were you with the product?



Analysis:

Based on the feedback received users are in the middle ground and two users said somewhat satisfied, there was one somewhat dissatisfied user which means there is room for improving the game.

Overall the feedback was mainly positive and an indication that the project is headed into the right direction.

Question 2 How would you rate the quality of the product?

Analysis:

Based on the result the users believe that the product is in a good place and can be improved. We had one tester that was not happy because of the lack of controller support for the menus with a controller but that can be easily fixed in future patches of the game.

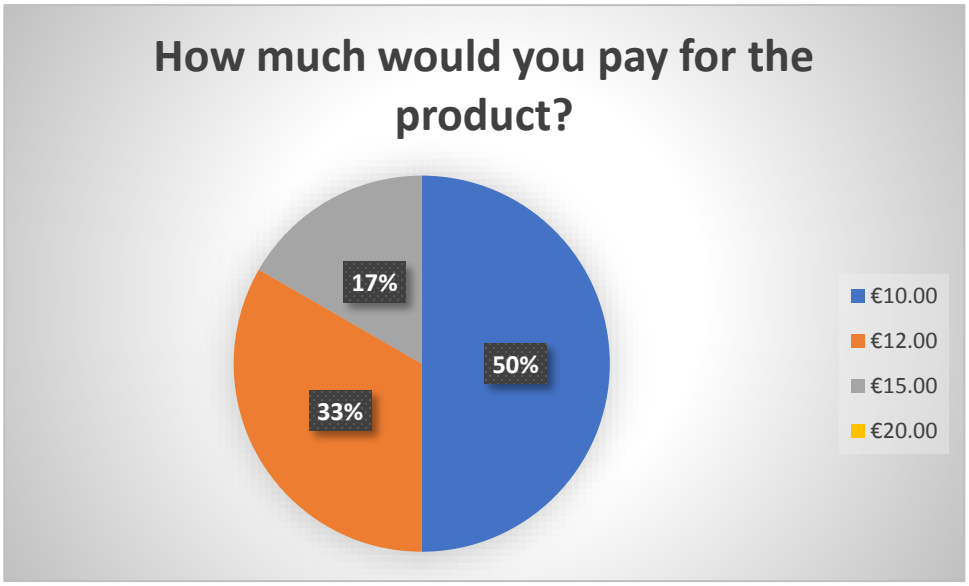
Question 3 Would you play again if more levels are added and the game is developed more?

Analysis:

Based on the results of the study the testers were split on whether to play the game again if the game is polished more and more levels are added.

This indicates that the game needs to add more functionality or gimmicks to attract more customers

Question 4 How much would you pay for the product?



Analysis

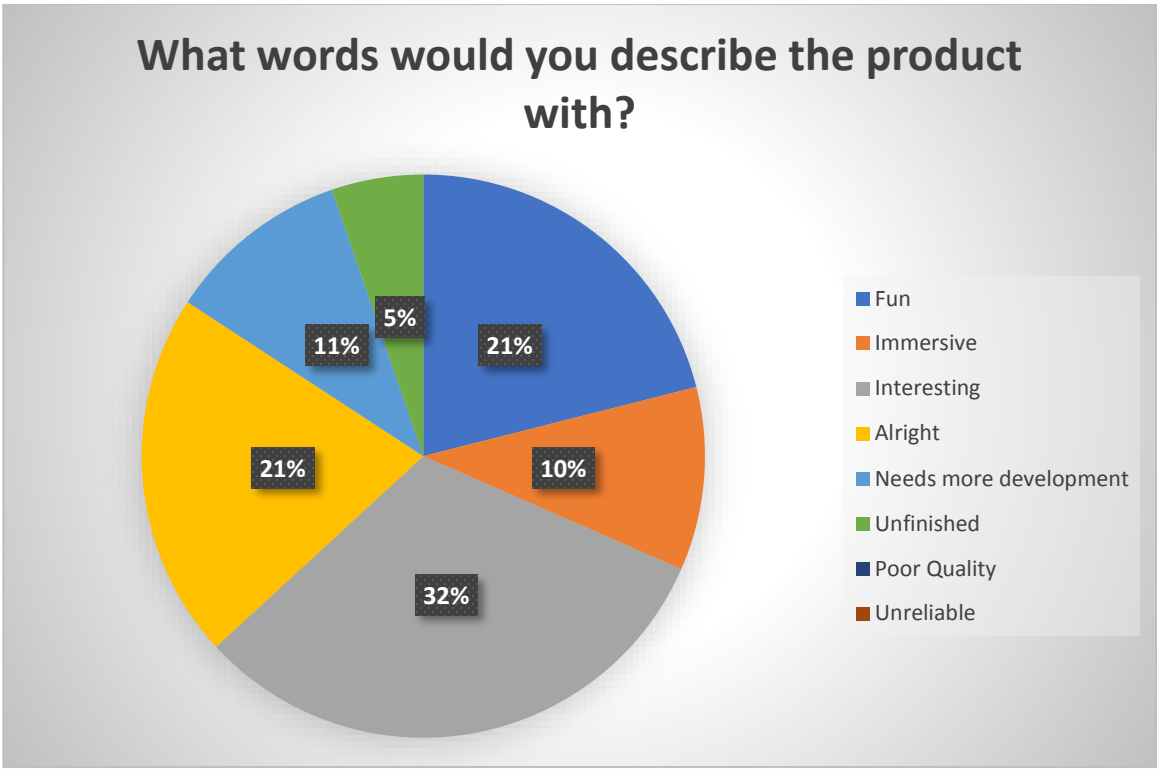
The result of this survey was expected coming from a PC gaming background and since the game is not from a huge development studio the price range of 10 to 15 Euro is understandable.

To give more of a value for the game further development and free updates should be considered in the future.

Note: The product will not be sold to the public this question was only asked for research purposes.

The product also needs to be updated for retail because some unreal engine assets were used during development and cannot be sold.

Question 5 What words would you describe the product with?



Analysis

Based on the study most of the users seemed to be interested and had fun with the demo of the game. One of the testers selected the world unfinished and needs more development. This criticism is valid and will be taken in consideration during future development.

4 Conclusions

This project will combine a couple of different established genres to create an immersive experience and the main reason for selecting or combining genres was based on personal preference and a bit of research on what games are being made and released today.

The project was a fun experience learning blueprint/Visual coding was different and refreshing. It had its ups and downs but I learned a lot developing this project.

Small Hurdles:

Self-Learning how to use unreal 4's blueprint system, this is a new concept for me since I've been mainly writing code in college. Luckily there are a lot of resources online to help me learn how to use unreal 4's blueprint system.

Character creation for the enemy:

Unfortunately, blender was the primary modelling tool I was using for months to develop the enemy character and to develop meshes for my game.

After a couple of months working on the enemy character, I found that it was difficult to rig the character's skeleton into unreal engine 4 it took some time before I could find a solution for this hurdle. The solution I found was to import my character into Adobe's Mixamo. I had to import my character model into Mixamo and built the skeletal mesh inside Mixamo and the imported it into Unreal Engine 4.

Hardware Failure:

My main laptop had a couple of fan failure during the development cycle of the project luckily, I had my desktop at home with backups of my project from the unit testing phase.

The main hurdle: was understanding the limitations one developer can do in such a small amount of time, setting realistic expectations for the game. At the start of this project I set out to make a game where there are multiple large open world maps and in each of the levels there are going to be multiple mini games waiting to be discovered. I've solved this problem by asking people what they wanted to see in a video game and there was a resounding response to a more narrative or structured level design. This has also helped me to focus and lighten my work load as a developer since I have a sense of direction to follow and new simplified concepts to be developed.

5 Further development or research

5.1 Further development:

With unreal engine 4 being upgraded constantly the game can be upgraded so more functionalities can be added.

Smooth transition on enemy melee attacks can be implemented.

More AI behaviour can be added such as dialog chatter between AI characters.

5.2 More AI types

More AI types can be added to the game. For example, an AI that can shoot at the player and AI that can drive vehicles.

5.3 Improved Sound design

Since I have no sound design background the in-game sounds and sound effects can be improved.

5.4 Vehicle Implementation

Vehicles can be implemented with future levels allowing the level to traverse the map more effectively.

5.5 Virtual Reality Improvements/implementation:

Because of time constraint and hardware failure the oculus rift was not implemented in the final product.

6 References

Digi-capital.com. (2016). *Augmented/Virtual Reality revenue forecast revised to hit $120 billion by 2020 | NEWS >> Digi-Capital*. [online] Available at: http://www.digi-capital.com/news/2016/01/augmentedvirtual-reality-revenue-forecast-revised-to-hit-120-billion-by-2020/#.WE29XvmLSHt [Accessed 30 Nov. 2016].

Docs.unrealengine.com. (2016). *Blueprints Visual Scripting*. [online] Available at: https://docs.unrealengine.com/latest/INT/Engine/Blueprints/ [Accessed 2 Dec. 2016].

Statista. (2016). *U.S. most popular video game genres 2015 | Statista*. [online] Available at: https://www.statista.com/statistics/189592/breakdown-of-us-video-game-sales-2009-by-genre/ [Accessed 2 Dec. 2016].

Statista.com. (2016). *Cite a Website - Cite This For Me*. [online] Available at: http://www.statista.com/graphic/5/189592/breakdown-of-us-video-game-sales-2009-by-genre.jpg [Accessed 1 Dec. 2016].

YouTube. (2016). *Nvidia Pascal - GEFORCE® GTX 1080 Official Launch (Live) Release*. [online] Available at: https://www.youtube.com/watch?v=mHxV0Gt-HI4 [Accessed 2 Dec. 2016].

Answers.unrealengine.com. (2017). *Building Lighting for Instanced Static Meshes - UE4 AnswerHub*. [online] Available at: https://answers.unrealengine.com/questions/134916/building-lighting-for-instanced-static-meshes.html [Accessed 2 Mar. 2017].

Answers.unrealengine.com. (2017). *How to add footstep sounds?- UE4 AnswerHub*. [online] Available at: https://answers.unrealengine.com/questions/7437/question-how-to-add-footstep-sounds.html [Accessed 15 Mar. 2017].

Answers.unrealengine.com. (2017). *Line Trace not applying damage to an actor - UE4 AnswerHub*. [online] Available at: https://answers.unrealengine.com/questions/427546/line-trace-not-applying-damage-to-an-actor.html [Accessed 1 Apr. 2017].

Forums.unrealengine.com. (2017). *Blender to UE4 using UE4 animations?*. [online] Available at: https://forums.unrealengine.com/showthread.php?28882-Blender-to-UE4-using-UE4-animations [Accessed 1 Apr. 2017].

Unrealengine.com. (2017). *Open World Demo Collection by Epic Games in Environments - UE4 Marketplace*. [online] Available at: https://www.unrealengine.com/marketplace/open-world-demo-collection [Accessed 7 Nov. 2016].

YouTube. (2017). *UE4 - Portal Doors Tutorial - Part 1 [The Portal Effect]*. [online] Available at: https://www.youtube.com/watch?v=PQy7C1RowB0 [Accessed 7 Apr. 2017].

YouTube. (2017). *UE4 Third Person Cover Shooter - 33 Blueprint Door*. [online] Available at: https://www.youtube.com/watch?v=D4oah_JjJv4 [Accessed 12 Feb. 2017].

YouTube. (2017). *Unreal Engine 4 - Line Trace Basics*. [online] Available at: https://www.youtube.com/watch?v=yBmPxGHu3uo [Accessed 8 Feb. 2017].

YouTube. (2017). *Unreal Engine 4: POP UP TEXT (Interactive Painting)*. [online] Available at: https://www.youtube.com/watch?v=_FpM1snveWQ [Accessed 7 Jan. 2017].

YouTube. (2017). *Unreal Engine 4 - Line Trace Basics*. [online] Available at: https://www.youtube.com/watch?v=yBmPxGHu3uo [Accessed 8 Feb. 2017].

YouTube. (2017). *Unreal Engine 4 - Line Trace Basics*. [online] Available at: Tutorial Realtime background Main Menu - Unreal Engine 4 [Accessed 15 Feb. 2017].

6.1 Asset References

There were two asset pack used for this game, the asset packs are provided by Epic games and can be used if the product is not for sale and used to learn for learning purposes.

Unrealengine.com. (2017). *Kite Demo Assets*. [online] Available at: https://answers.unrealengine.com/questions/423943/use-kite-demo-assets-in-my-own-level.html [Accessed 8 Apr. 2017].

Unrealengine.com. (2017). *Open World Demo Collection by Epic Games in Environments - UE4 Marketplace*. [online] Available at: https://www.unrealengine.com/marketplace/open-world-demo-collection [Accessed 7 Nov. 2016].

Unrealengine.com. (2017). *Learn*. [online] Available at: https://www.unrealengine.com/blog?category=learning [Accessed 7 Nov. 2016].

Futuristic blaster (modified via blender)

Turbosquid.com. (2017). [online] Available at: https://www.turbosquid.com/3d-models/laser-rifle-energy-blend-free/699887 [Accessed 7 Feb. 2017].

7 Appendix

7.1 Project Proposal

7.1.1 Objectives

The main objective is to develop an open world game that supports virtual reality game. The game itself will run on both normal monitors or virtual reality gear such as the oculus rift.

The game will be a first person open world game; the game will have hidden secret scattered all throughout the map which the player needs to find. The open world will act as a hub for the player giving them an immersive world to explore and experience. Some secrets scattered across the maps will either teleport the player to another world or into a mini game. Each secret will act differently when activated.

In short the main objective of this game is to give the player the freedom of exploring a virtual world and being immersed in the game world.

The main target audience of the game are school children, children that are confined in hospitals etc… and can't experience going out on an adventure for a little while. This will give them the chance explore around the world without having to go out.

7.1.2 Background

VR technology has been around for a long time but up until recently the technology and timing was not right for VR technology but now there has been a big development in the VR technological scene which sparked the interest of developers and consumers. Even though Oculus rift is not the first VR gear to be developed, Oculus has generated a lot of interest in the are ever since it was announced this lead big companies to develop their own version of a virtual reality gear such as Sony and Steam (collaborated with HTC).

Most first party virtual reality game and early virtual reality game has the player standing in place and experiencing the world through set pieces or on rail events. There are a few open world games that fully support virtual reality and most of them need 3rd party software to work. This is where this game comes in the game will provide the player with an open world where they can be immersed in. It will give the player a chance to explore and experience different sceneries form the comfort of their own home.

7.1.3 Technical Approach

For this project, I will need to do extensive research on game engines specifically Unity and Unreal Engine 4 to see which engine will be more suitable for a virtual reality setting. I will also need to do research on how to make it work with traditional monitors as the game will have support for normal monitors and VR headsets.

Extensive research on VR technology and how to implement it will also be needed, most users of VR technologies tend to get motion sick when the game is not optimized and is poorly implemented.

The two-main technology that stands out right now are Unreal Engine 4 and Oculus Rift.

After the research is done, development of the map and character controls are top priority. Without the game world and player controlled character the game will not be a game and other features will be developed after the player has control of the character and can traverse the world. Optimization will also play an important role in the development.

Implementation of features will be done in small increments and will be prioritized by the requirements.

7.1.4 Special resources required

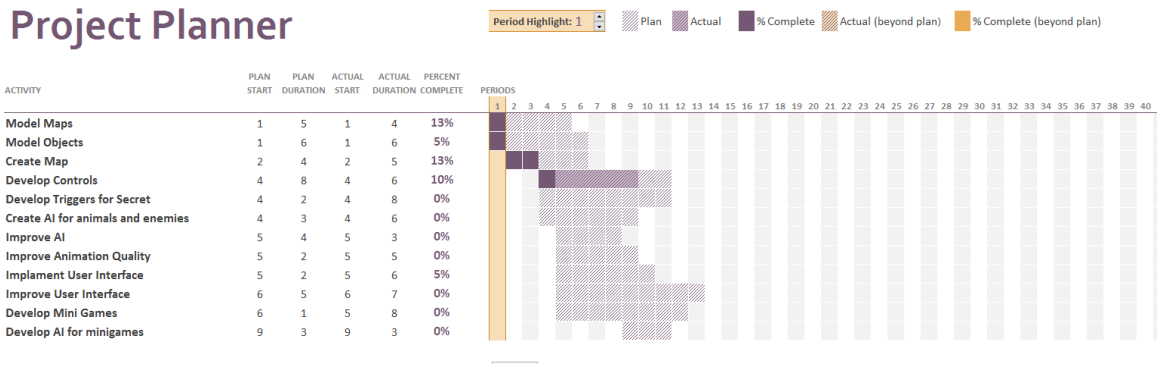
There are two major hardware resources required for this project; A strong Desktop or laptop and a virtual reality headset such as the OCULUS or HTC Vive.

The desktop/laptop should be powerful enough to support Unreal Engine 4 development and running the game. It should also have support for a virtual reality headset.

Software wise the project will need Unreal Engine 4 and 3D modeling software such as Maya or Mudbox.

The game will also need Audio and Images for its UI, these elements will be created via Photoshop and other editors.

7.1.5 Project Plan



ACTIVITY	PLAN START	PLAN DURATION	ACTUAL START	ACTUAL DURATION	PERCENT COMPLETE
Model Maps	1	5	1	4	13%
Model Objects	1	6	1	6	5%
Create Map	2	4	2	5	13%
Develop Controls	4	8	4	6	10%
Develop Triggers for Secret	4	2	4	8	0%
Create AI for animals and enemies	4	3	4	6	0%
Improve AI	5	4	5	3	0%
Improve Animation Quality	5	2	5	5	0%
Implament User Interface	5	2	5	6	5%
Improve User Interface	6	5	6	7	0%
Develop Mini Games	6	1	5	8	0%
Develop AI for minigames	9	3	9	3	0%

7.1.6 Technical Details

Unreal Unity supports C++ and it also has visual scripting system, this visual scripting systems can be used to make life easier since manual coding can be skipped but for the project I will be using both since things can be done better when you code the logic manual.

For the in-game assets, I will be providing my own models, sounds and I will also use assets included in Unreal Engine 4.

Implementation of VR technology will be important because VR needs specific conditions to have a smooth experience, for example the game needs to have framerates about 75 and have a wider field of view to prevent motion sickness.

7.1.7 Evaluation

The projects usability will be tested via surveys and beta testing by other people, the survey will be question based where participants will be asked how the game performed, how the interface can be improved and how the controls works. Unit testing will also be introduced while developing the game, to make sure all the functions work as intended. Further public testing will be needed when new features are added; the test will be carried out by other people and feedback will be given.

7.2 Monthly Journals

7.2.1 Reflective Journal Month 1

My Achievements

This month I finished making plans for my project and I also pitched my project to the lecturers. I've also considered what type of virtual reality gear I will use to make this game.

My contribution to the project included; Defining what the game is and defining its target audience. I also considered carefully what technology to use for the project and how I'm planning to implement it.

My Reflection

I felt that I come up with a viable project to make for 4th year, I've also considered carefully what different technologies I can use to make this project into reality.

I felt like I didn't do a good enough job explaining what my project is on the pitch this month. I could have explained it better but I was slightly nervous when I was doing the pitch.

Intended Changes

Next month I will try to finish designing the levels for my game. I will try to make three different maps for my game once I'm sure my idea for the project has been fully accepted If not I will talk to one of the lecturers and see if I can take the project they listed on board as my own project.

I realized that I need to do more research on how to make a virtual reality game, researching what engines works better with the oculus rift and what kind of engine supports it better. If my idea was rejected I also realized that I will need to talk to one of the lecturers for a project.

Supervisor Meetings

At the end of this month we had to pitch our idea to the lecturers that will be our supervisors there hasn't been any meetings since we only finished pitching our idea to them.

7.2.2 Reflective Journal Month 2

My Achievements

I continued to make sample maps and familiarizing myself with unreal engine. I also tried to make my own character models, objects, and other textures just in case I need them in the future.

Broad Research – Research on map design and layout. Research on game mechanics and what makes a game fun. I also, did research on what is required for games.

I also managed to meet with my supervisor and discuss my final year project and on how to best approach it.

My Reflection

I felt like I worked well on the research topics related to game mechanics, game engines, VR technologies and designing maps.

I could not start the main development yet haven't finished writing up the requirements. I also failed to manage my time because there was a lot of C. A's happening during the last week of this month and got overwhelmed with work to be done.

Intended Changes

Even though we only got assigned our supervisors, I would make more of an effort to arrange meetings with them to discuss my final year project.

I would also try to manage my time better, I got overwhelmed with the amount of work to be done because of other modules. There were a lot of class assessments and projects due on the same week.

Supervisor Meetings

Date of Meeting: Oct 27th, 2016

Items discussed: Discussed what the main premise of the game is and what technologies can be used for the game. IE Oculus rift, Unreal Engine 4 and Unity.

We also discussed on a potential cross platform accessibility for the application and compatibility for different devices.

Action Items: Write functional and non-functional requirements for the game.

7.2.3 Reflective Journal Month 3

My Achievements

This month, I completed my technical report and finalize the requirement specification for my project. I also worked on my prototype, adding small texture work for the landscape and edited the first-person camera perspective.

I was also able to meet with my supervisor about the prototype and we also discussed the technical report.

My contributions to the projects included texture work, A.I Research, research\ on level design and what elements are needed for a game. I also asked a tiny group of people to do a survey on what they would like to see in a game. I also did a few research on game engines to finalize what development type I will be using.

I was also able to develop a basic AI opponent that follows the player when they get close.

My Reflection

I felt, it worked well in terms of researching materials for my technical report

However, I was not successful in developing my prototype to its fullest because of the tight schedule I have with other project in college.

Intended Changes

Next month, I will try to start developing the prototype further adding in more assets, developing the AI further and making more levels for the game

Supervisor Meetings

Date of Meeting: 22th November 2016

Items discussed: Discussed what is needed for the prototype.

Action Items: Started developing a prototype level

Date of Meeting: 6th November 2016

Items discussed: Discussed what is needed for the technical report

Action Items: Research on key development in the gaming industry

Research genre of games

7.2.4 Reflective Journal Month 4

My Achievements

This month, I was able to develop my project prototype as best I could, I also finished my midpoint presentations slides and midpoint technical report.

I was also able to present my prototype to my supervisor, the prototype consisted of a section of the map implemented with textures and assets like sound and meshes. The prototype also included two simple implementations of an AI (Artificial Intelligence).

My contributions to the projects included finishing a prototype of my game with simple AI, Texture work, sound design and map design. I also presented my porotype to my supervisor at the midpoint presentations and my supervisor was happy about the progress made in the game.

My Reflection

I felt, it worked well on the prototype and I feel like I'm developing my game in a steady phase. I feel like I spent more time making a map than developing game mechanics.

I felt like I was still swamped with other course work and I need to manage my time better.

Intended Changes

Next month, I will try to develop more of the games map and implement a weapon system and implement UI elements to the game.

I will also try to develop more game mechanics and functionally of the game I also intended to implement oculus rift into the game.

Supervisor Meetings

Items discussed:

Had a supervisor meeting about the status of my midpoint technical report and prototype and what should be included in the midpoint presentation.

7.2.5 Reflective Journal Month 5

My Achievements

This month, I decided to make one big level or two separate smaller levels for my project the original 5 levels is impossible to complete with just a one man development team.

My contributions to the project included map designing, asset modelling, UI development and gameplay development.

My Reflection

I felt, it worked well on designing levels and creating assets for the game.

Intended Changes

Next month, I will try to cover some elements of the game with unit test and finalize the map design and work on game play element like intractable objects and develop a combat system with a more advance AI

Supervisor Meetings

Items discussed:

Discussed the size of the project on what I should change about it and also discussed on what type of platform to use when sharing the project to the supervisor.

7.2.6 Reflective Journal Month 6

7.2.7 Reflective Journal Month 7

My Achievements

This month I am finalizing the enemy AI and made some changes on the combat system. I have also finished making the 3 maps for the game (Tutorial, Transition and main level).

My Reflection

I have been busy with other projects and CA during the last few weeks and work for the project has been very slow.

Intended Changes

For the final, few weeks I will finalize the models for the enemies and test out any map bugs.

I will also finalize the player model.