National College of Ireland

BSc Computing

Software Development

2020/2021

Keith Mahony

x17521139

x17521139@student.ncirl.ie

“ABSENT”

Technical Report

Updated – 15/05/22

Contents

Note: Sections that have been adjusted or added since the Midpoint Presentation have been labelled *Updated* or *New*.

Executive Summary *Updated*

ABSENT is an action-adventure game set in a grounded but fictional world, with players taking on the role of a detective tracking down a missing person. The game is inspired by others that are usually referred to as immersive simulations or "immersive sims". Examples include Thief and Deus Ex in the late 1990's and more recently Dishonoured and 2021's Deathloop. These games feature open-ended level design, sophisticated enemy AI and many, many ways for the player to interact with those enemies and the worlds they inhabit. Thief and Deus ex were also two of the first games to offer non-violent solutions to video game objectives –an important step for a medium which was viewed at the time (in some ways fairly) as an ultra-violent toy for (male) teenagers.

It would be impossible to match or exceed these works of art, but this project is an attempt to showcase an understanding of the design philosophies behind them, implement them on a smaller scale, and add some new ideas into the mix. I believe it's possible to create a game with the sensibilities of Deus Ex or Dishonoured, without ever feeling the need to explain to your audience what that means. These tools will instead be used to immerse players in this fictional environment, provide them with tools to interact with it any way they see fit, and allow them to experience the story of Detective Hays in their own way.

The game has been designed from the ground up for mature audiences and will run on current and next generation consoles, plus higher end PC's. A mobile version is technically possible but is not considered a priority.

Final Report Update: A mobile version is no longer considered a possibility.

Overview

- **Genre:** Adventure / Action / RPG
- **Target audience:**
  - **Age:** 16-45
  - **Gender:** Male/Female
- **Monetization:** Episodic / Premium
- **Platforms:**
  - PlayStation 5
  - Xbox Series Consoles (later)

1.0 Introduction

1.1. Background

For as long as I can remember I have had an interest in *games*, but this eventually grew into an interest in game *design,* and how good game design can be used to tell stories not possible in a passive medium like cinema or literature. For me, the ultimate example of this style of storytelling can be found in games like the ones above, but I believe many of their philosophies are relevant to every game or even every *virtual experience*. In a world that is becoming more and more obsessed with gamification, these design principles will be crucial, even outside of the 'video game industry', which will itself become difficult to identify as more and more entertainment experiences will be gamified. For example, it won't be long before most in-person events can be attended in virtual reality; concerts, movie premieres and world cup finals accessible to anyone with a headset. But anyone who wants to provide this kind of experience will need to understand the fundamentals of how to help a user or **'player'** navigate a 3D environment, how to introduce them to complicated control schemes, or how to create systems that react to them in ways other than "You can't go over that way", "Please stay in your assigned seat", or some other variation of "no".

1.2. Aims

Saying **YES** to the player is a cornerstone of great game design, but it also presents a much bigger challenge than a more linear experience. The good news is, 'Thief' solved some of these problems in 1998, and 'Deus Ex' had created an even better template by 2000.

By understanding what made those games great, and how their spiritual successors iterated on the formula more recently, I will in theory have the tools I need to make **ABSENT** an enjoyable experience and be more prepared for what I believe is an imminent, gamified future. With all of that in mind, this project is an attempt to showcase that there is no immersive sim *genre*, but that there are tried and true game design methods and philosophies found in these kinds of games that translate across the entire industry, and maybe even beyond it.

1.3. Technology

1.3.1 Unreal Engine *Updated*

ABSENT will be developed using Epic Game's Unreal Engine v4.27 and make use of the open-source Advanced Locomotion System v4, developed by LongmireLocomotion [1].

I mentioned previously that ABSENT would feature sophisticated artificial intelligence systems and this plays into my choice of engine as well. It's possible to achieve excellent results in using UE's Behaviour Tree System, developing complex scenarios using Blackboard and Behaviour Tree editors. It's not impossible to get similar results using Unity (another game engine which was considered), but it would likely take much more time and effort. Unreal Engine will streamline the development of AI systems which, as mentioned, are crucial and make up a sizeable portion of ABSENT's development effort.

Unreal Engine also provides direct access to the Unreal Marketplace, where I can make use of open source and paid-for community assets, which will facilitate prototyping and development of the final version of ABSENT. This will be covered in more detail in section 6, Special Resources Required. With Unreal Engine 5 now available in early access (boasting

exciting new features for the next generation of games and consoles), there's no doubt this community will continue to grow over the next 5-10 years as well.

Unreal also provides opportunities to create new virtual reality experiences, which makes it possible to implement some of these features into the core game or maybe as an additive experience, but for now, with how hit and miss the success of current generation of VR/AR headsets has been, this is not yet considered a priority.

Final Report Update: Absent will no longer make use of ALS v4. This decision was made midway through this development cycle and will be detailed further in the relevant sections. Primarily, the switch was made to move the development of Absent over the Unreal Engine v5.1 and make use of the latest technologies available (Metahumans, Lumen, Nanite, etc). This would prepare Absent, a product which is likely several years away from completion, for the next generation of consoles. It would also make it possible to take advantage of tools and technologies that streamline development, crucial for the success of independent game development.

1.3.2 Blender *Updated*

While I won't be creating any complex 3D models from scratch, Blender is an open-source 3D modelling program which will allow me to tweak and adjust models as required. These models can then be exported to Unreal Engine and placed into the game.

Final Report Update: No, I haven't become a 3D artist since the midpoint presentation. However, the transition to Unreal Engine 5 and away from the stylised characters being used previously necessitated the development of some 3D models. In particular, the police jacket which plays a part in Absent's core gameplay loop. In combination with a program called **Marvellous Designer** (detailed below) Blender was used to prepare 3D models for use in Unreal Engine and was part of the workflow that transferred models to and from **Marvelous Designer**.

1.3.3 Marvelous Designer *New*

Marvelous Designer is a best-in-class 3D modelling tool which specialises in the modelling and simulation of cloth. This software was used to create a unique look for the player character, all of which is detailed in the Detailed Models section.

2.0 System

2.1. Requirements

2.1.1. Functional Requirements

2.1.1.1. Main Menu

The main menu acts as a hub for all the games content and settings options. This is the first screen every user will see, and it is therefore crucial that their options are presented to them in a clear and concise fashion. Having said that, I would hope to present an interesting and engaging user interface. Although not a priority, this is the first thing players will see and it always important to make a good first impression. This menu will also act as a hub for all future content updates and 'episodes', so it will need to be clear to all user what they do and don't have access to, and when future will become available to purchase.

This menu allows players to start a “New Game”, continue from where they left off with “Load Game”, browse the “Episodes” screen, or adjust “Settings”. Players can return to this screen from almost any other state of the application.

2.1.1.2. Adjust Settings

From this menu (accessed via the Main Menu) players can adjust graphical / sound settings and controller options. Options will be somewhat limited during development, and the extent of customisation provided to users in the final product is still under consideration. At a minimum, players will be able to adjust image brightness, invert their X/Y axis, and adjust volume levels. The addition of more specific settings will be somewhat dependant on feedback received during playtests.

2.1.1.3. Episode Select

From this menu (accessed via the Main Menu) players can select from the available Episodes. Only Episode 1 will be developed during the Software Project, but the systems and mechanics created during this phase can be expanded and enhanced for additional episodes beyond the deadline of this module. When that content becomes available, it will appear here with the option to purchase/download.

2.1.1.4. Play Game

Once a player has started a new game, selected an episode, or loaded a previous save, they will begin playing the game. Episode 1 is currently being developed and will be the centrepiece of the project, and where most of my time and effort will be invested during the Software Project module. As a result, this should also be where the end user spends most of their time.

2.1.1.4.1. Immersive, Authentic Environment

Players can explore the environment at their own discretion and there is very little restrictions placed upon them. Environments will be authentic, highly detailed and display a consistent art style. There will be no ‘invisible walls’, the player’s own sense of architecture and movement will help them understand where they have the potential to reach. Game design will be integrated into the environments, with locked doors or electrified puddles preventing immediate progress or requiring creative solutions.

2.1.1.4.2. Sophisticated AI Systems

Housed within the 3D environment is a network of AI systems. All civilians in these environments will act in ways which feel authentic to the world they’re supposed to be inhabiting. They will move around the world on their own without any input from the player but will also react to the players behaviour when it makes sense to do so. Players will also be accompanied by their partner, who will follow them around and assist them in combat if necessary.

2.1.1.4.3. Player Tools

The player will have multiple ways of interacting with the environment beyond exploring it.

Explicit: Press the action button near an interactive object like a switch or a door to interact with it. In these cases the objects in question will have 1-2 state changes which the player is cycling between. Door open, door closed. Power on, power off.

Implicit: The player also has access to more dynamic tools. The player's weapons, for example, can be equipped at any time. The equipping of the weapon alone has an impact on the behaviour of civilian's and their partner's behaviour. Firing the weapon has the potential to damage or kill other characters and so these objects can inflict an almost unlimited number of state changes.

2.1.1.4.4. Clear, Open-Ended Objectives

Large environments, complex AI systems, and a long list of tools at the players disposal combine to present the possibility of overwhelming the player. To mitigate this issue, the objects of the game will be quite simple. Get inside this building. Find this person. Escape. Furthermore, the objectives will make use of the player's predisposed understanding of problem solving. If you want to find someone in a town you're not from, maybe ask around. If the front door is locked, maybe the back door isn't. If that's locked as well, maybe someone has a key… Combining this with the large solution space encourages the player to explore and become immersed in this environment and story.

2.1.2. Data Requirements

2.1.2.1. Save States

Players can save their game at almost any time by pausing the game and selecting the "Save" button. The game will also 'autosave' at specific points to prevent players from losing significant amounts of progress due to system crashes or human error. The system will keep track of this progress and allow players to quit the game and resume from where they left off later. There are some circumstances under which the player will not be allowed save, such as if they have initiated combat. This is to prevent the player abusing the save feature.

2.1.3. User Requirements

2.1.3.1. Memory Capacity

Users will need to store the game on their local drive-in order to begin playing. The game will likely require 15-25GB of free space. A comparatively small amount of space will also be required to store their save games and maintain their progress between sessions.

Final Report Update: The game will likely require 25-50Gb of free space, thanks in large part to higher detailed characters and environments.

2.1.3.2. Hardware Specifications

The game has been designed with the latest generation of consoles in mind and will cater specifically to that hardware. Platforms and release timings are still under consideration, but users who purchase the game from the PlayStation Store, for example, will experience the game as it was intended. It is possible to make the game available to users on PC but in those cases a list of 'system requirements' will be shared with players before they purchase the game, to ensure their hardware will be capable of providing a good experience. A PC version of the game will likely take much longer to develop and release, as more graphical settings and adjustments would be required to ensure every player has a good experience, on what is a more dynamic platform. With this in mind, a PC version is not currently a priority.

2.1.3.3. Age Restrictions

ABSENT has been designed with a mature audience in mind, which is reflected in its gameplay and the themes of the story. Users on both major console platforms require age verification before purchasing content with a mature rating and therefore some younger players will not be able to experience this game.

2.2. Design & Architecture

2.2.1. Game

2.2.1.1. Levels / Episodes

The environment of each level/mission will be sectioned off from the rest of the fictional (and non-existent) world by blocked roads, locked doors, and other obstacles the player cannot traverse. The playable spaces themselves will be rendered in 3D and high detail. A fixed time of day will allow predetermined, cinematic lighting conditions, though weather will change if certain conditions are met. Key landmarks and thoughtful level design will ease navigation of these spaces.

2.2.1.2. Objectives *Updated*

Each level has one key objective, and they are always clear and understandable. For example, the first level's objective is to 'Find Henry Purcell'. Player will be given some information to help them complete their objective, in this case, 'He called 119 from the Blu Bar'. The player will then be set free within the playable space to discover their own path towards this objective. Objectives can be completed in many ways. Non-violent ways are heavily encouraged but not required.

Final Report Update: The structure of some missions has changed since the previous report and will likely continue to change and evolve over time. However, the overall structure of the game remains intact. For example, the "Find Henry Purcell" objective will likely occur later than initially planned to accommodate a tutorial-focused introduction sequence.

2.2.1.3. Rating System

To encourage 'good' police work, a rating system is under consideration. This would give points to a player for avoiding combat and finding more passive ways to complete their objectives.

2.2.1.4. Rating System

The game will have a simple but intuitive user interface. The main menu will allow the player to:

- Start a new game
- Load a previous game
- Adjust some in-game settings (brightness, camera inversion, etc.)

Player can also pause the game at any time by pressing the pause button. This menu allows players to:

- Review objectives and view additional information
- Resume game

- Load another saved game
- Exit to the main menu

2.2.2. Environment

Each environment will consist of:

- worldbuilding
- level boundaries
- enterable interiors
- dynamic hazards

Environments will look and feel like real places that exist with or without the player. The environments are not static. Player actions can add/remove/alter the elements mentioned.

2.2.2.1. Worldbuilding

Environmental storytelling will reveal details about the world of Ertha: advertisement boards, notes found by the player, and more. A dialogue system, allowing the player to communicate with other characters, will also provide opportunities to tell the player about the world they're inhabiting.

2.2.2.2. Boundaries

Levels will be 'grayboxed' initially (a rough block-out version of the environment), a level design practice which helps designers to iterate and test the layout as soon as possible. Afterwards, these 'blocks' will be replaced with a combination of textures and 3D models, creating the illusion of a living world beyond the boundaries of the playable space.

2.2.2.3. Interiors *Updated*

Helping create the illusion that there is activity behind every door, is the inclusion of activity behind some doors. For the first level specifically, there are 3 key locations which the player can explore the interiors of. A bar (first and second floor), an 'abandoned' toy store, and an apartment building (corridor and one apartment). These areas will be highly detailed, and each offers unique gameplay scenarios and opportunities.

Final Report Update: The first level has been adjusted. There are now only 2 interiors that the player can explore. However, some of the ideas presented initially will likely find their way into the next iteration of the project, and/or in a separate level. More details on this can be found in the Implementation/Level Design section.

2.2.2.4. Hazards

Hazards are harmful to the player and encourage thoughtful exploration or prevent direct access to points of interest. Hazards are often dynamic, however. For example, an electrified puddle could temporarily prevent access, but the player could find the power source and disable it. A more creative (and cruel) player could also find a way to use hazards like this against their enemies.

2.2.3. Characters

2.2.3.1. Controls

The player will control Detective Isaiah Hays from a third person perspective. The camera will rest over the shoulder of the character. All inputs will be mapped to a console controller and will not require any other device (such as a mouse or keyboard), and will allow the following:

- walk / jog / sprint
- jump / mantle (if near a climbable surface)
- interact (talk, open, unlock, use, pickup, etc.)
- crouch
- equip weapon / unequip weapon
- focus / aim weapon (if holding a weapon)
- kick / fire weapon (if holding a weapon)
- reload weapon (if equipped weapon ammo count is < maximum)

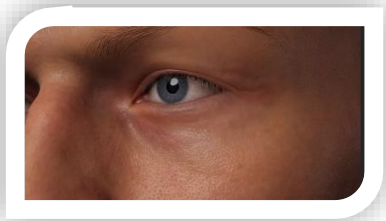
An inventory management system is still being considered and may be deemed necessary at a later stage.

Final Report Update: An inventory system is now considered a priority and a prototype is featured in the current iteration. The current version makes it possible to pick up objects, adding them to the characters and removing them from the world. One example, the backpack, is detailed further in the Gameplay Framework section.

2.2.3.2. Player Character Model *Updated*

Detective Hays is the most important character in the game and as such has a highly detailed (but stylized) character model. This stylized design will ensure target platforms can render the player character and multiple non-player characters on screen at the same time. The jacket can be equipped/unequipped, and this choice will be reflected on the character model. Initial Prototype (with jacket) is pictured on the right.

Final Report Update: The transition to Unreal Engine 5 provided the opportunity to focus on the next generation of console hardware and make use of the 'MetaHuman' creator, explained further in the Implementation section.



2.2.3.3. Undercover System

In ABSENT, other characters will respond differently to you depending on whether you approach them as a police officer, or as a civilian. Players can return to their vehicle (usually located at the starting point of every level) and equip or unequip their police jacket. This is a clear and readable way to understand which 'mode' you are currently in. I had originally envisioned a police badge which you could hold, but this became cumbersome and frustrating. The player character controls identically in both modes, but they may have different options depending on how they approach a situation. For example, civilians in one an area of a level might be immediately hostile to police officers but overlook civilians. Returning here without your police identification might give you the opportunity to explore the previously hostile area while avoiding combat. By extension, some previously hostile civilians might now be spoken to, gathering potentially crucial information.

2.2.3.4. Health and Energy

Players will have to manage their health and energy levels.

Health represents Detective Hays's current condition and the number of injuries sustained. If their health drops to 0 the game will end, and progress will be lost. Health will decrease when hit by other characters in combat, by entering a hazard zone, or falling from too high a height. This encourages thoughtful exploration of the environment and avoidance of combat. Items found in the world can be used to restore any lost health.

Energy represents how energetic or tired Hays is. If energy drops below 0, the player will no longer be able to jump, climb or sprint like before. However, energy will regenerate slowly while not performing any of those actions. Combat is always discouraged, and restricting movement prevents players from engaging in combat and trying to run away to avoid the repercussions of their actions.

2.2.3.5. Weapon System *Updated*

The player starts most levels with a police-issued weapon. This pistol can be equipped and used at any time if the player is holding enough ammunition. To discourage overuse, initial ammunition counts will be low. Players will be able to find and pickup more ammunition by exploring the level. Some civilians will be carrying ammunition as well.

There are plans to include a weapon pickup system, which would allow the player to pick up weapons dropped by other characters, but this system might encourage combat more than I'd like to and is still under consideration. And another thing to consider, the effort required to develop a system like this might be better applied elsewhere.

Final Report Update: This ‘weapon pickup system’ could be integrated relatively easily into the inventory system, which is now considered a priority. However, this will not be included in this iteration of the project.

2.2.3.6. Stealth System

ABSENT will feature a relatively simplistic stealth system. Players can hide from civilians, avoiding combat in hostile areas, or eavesdropping on a conversation that wouldn’t occur within earshot otherwise.

Unreal Engine has some out of the box senses which can be applied to characters, and these will be used here. Primarily, characters will be able to see Hays if he is positioned within their vision cone (originating from their head) or moving (while not crouching) within a specific vicinity of the detecting character.

2.2.3.7. Dialogue System

Player can interact with civilians by talking to them. After pressing interact near a civilian, dialogue will be presented as text on screen and at some points, the player will have the option of choosing a response from 2-3 options. Sometimes the player will only have access to the ‘correct’ response when they’ve acquired essential information previously. For example, if they find evidence of a character committing a crime, (if they don’t arrest them immediately) they could further press them on the issue in dialogue and potentially receive some new information from them in return.

2.2.3.8. Civilians

Every level in ABSENT is filled with civilians that the player can interact with. These characters are not necessarily your enemy or friend. The players actions will dictate how they respond and behave. Civilians will have a mixture of random and predetermined paths around the environment, always finding the shortest path. The player can talk to civilians, which pauses their movements temporarily and allows them an opportunity to acquire important information. Civilians can also be ‘spooked’ if they witness violence (to themselves or others). This can prevent the player from interacting with them temporarily, or in some cases, permanently. Their behaviour will also change, and their movements may become more erratic. Depending on the character, they may also become violent towards the player.

Some civilians have generic movements and responses, while others have unique paths, behaviour trees, character models and ways for the player interact with them. Some Unique Civilians will be considered criminals and the player will have the option to arrest these characters, removing them from the world. However, it might be of interest to the player to ignore their crimes temporarily in exchange for more information (or maybe they just don’t want to arrest them). Some civilians will be immediately hostile towards the player and engage combat on sight.

2.2.3.9. Civilian Character Models *Updated*

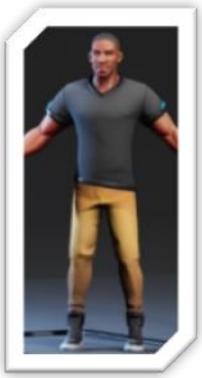
Civilians will have a range of designs with an art style matching the main character. As mentioned above, this art-style ensures multiple civilians can be rendered at the same time without a massive hit to performance.

Final Report Update: As mentioned previously, the transition to Unreal 5 made it possible to make use of the 'MetaHuman' creator and non-player characters will make use of this tool as well, drastically increasing fidelity and flexibility. This is explained further in the Implementation section.

2.2.3.10. Partner

With entirely different AI systems and behaviours, the players partner will stay relatively close to their location (within a certain vicinity) and act differently depending on the context. They might give their thoughts on the case, give hints as to how the player can progress (if enough time passes) or maybe critique the player's behaviour. Their behaviour will also change if the player enters combat, defending themselves and the player. The partner cannot be killed (by the player or other characters), but their accuracy is significantly lower than the players, so as not to become too useful. Their invincibility is a trade-off I do not make lightly. Every other character in the game can be killed and I intend to reward the player for thoughtful play (so not just killing everyone), but the partner is a key tool for critiquing the player and encouraging them to improve their behaviour. The possibility of them dying in the first 30 seconds was too big a risk.

2.2.3.11. Partner Character Model *Updated*



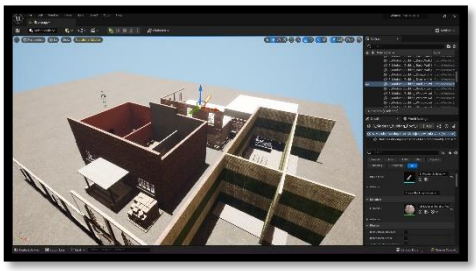
The player's partner features a highly detailed character model. Like the player character, the police jacket can be equipped and unequipped. Non-player characters will respond to the partner based on this choice, just like the player. The partner will always copy whatever the player is doing in this regard. First prototype is picture on the right (without jacket).

Final Report Update: Once again, the addition of the 'MetaHuman' creator made it possible to create a much higher fidelity version of this partner character. This is detailed later, in the Implementation section.

2.3. Implementation

2.3.1 Level Design *Updated*

The level is grayboxed to begin with, a level design technique which allows for an iterative creation process. The shape of any level can change drastically over the course of development as features are added and tested, difficulty is balanced, etc. Once a complete version of the level is finalised, it will then be possible to fill the level with detail and life, as seen in the concept image to the right.



Final Report Update: The level shown above evolved quite a lot over the course of development, with significant adjustments made since the first prototype, primarily in an effort to remove empty or 'dead' space. It is worth mentioning that levels did not receive the same attention as gameplay mechanics and animations, which is more likely a symptom of solo game development, as opposed to a frustration with this project specifically. I endeavoured to synchronise the development of all components, but some elements simply must come first, and I didn't have the option to delegate. For example, how wide should a street be if it isn't known how fast the player character can walk or run? These questions encourage a focus on mechanics at least initially, but what I found very effective was the development of a test level, of sorts, which acted as a sandbox for the testing of mechanics and systems. This test level began like the one picture above and evolved over the course of development alongside mechanical additions and changes.

Immediately apparent is the size differential. There was an effort to remove unused space, and this facilitated the desire to maintain a high level of detail across the board, something which has been stressed since the initial proposal. Focusing on a slightly smaller location and less interiors overall is

improving the quality of those elements, simply by allowing more time and effort be provided to what is present.

Level Design and Environment Design benefitted significantly from the transition to Unreal Engine 5, allowing levels to take advantage of both **Lumen** and **Nanite**, two technologies which streamline environment creation and optimisation. This is particularly crucial when developing a game solo, like I am here.

Firstly, **Lumen** is Unreal Engine's new take on **global illumination.** This allows the engine to light a level (or a massive world) entirely dynamically, from a combination of natural light (e.g. sun, moon, etc) and unnatural light (e.g. candles, light fixtures, etc). For context, this would previously involve placing what are called 'point lights' and faking the bouncing of light throughout a virtual space. Lumen not only bounces and fades light correctly, but it also manages to dynamically adjust the colour of light *after* it hits a surface. So the sun hitting a *red* wall will bounce *red* light around the immediate area. This is hard to identify as the cause of a game's unrealistic look, but once the correct form of illumination has been applied, it becomes clear just how *off* the previous version looked. Again, some of these effects are *technically* possible on older versions of Unreal but required an incredible amount of time and expertise to produce results even close to what Unreal Engine 5 is somehow producing in real-time.

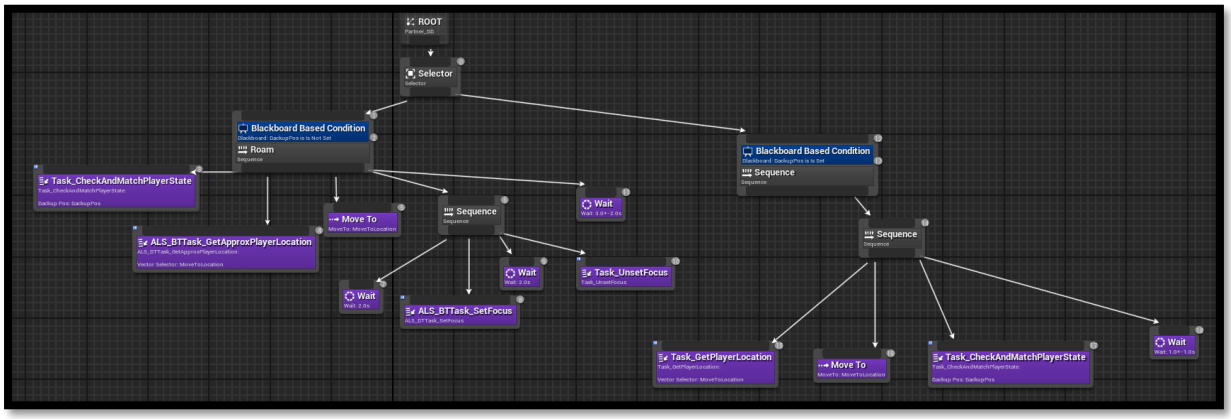
Nanite, on the other hand, is a revolutionary rendering technique that can "*handle orders of magnitude more triangles and instances than is possible for traditionally rendered geometry*" [2], by managing levels of detail automatically (primarily based on distance to player) and intelligently displaying details where they can be perceived and nowhere else. This is streamlining environment design/creation significantly and, alongside **Lumen**, is making Unreal Engine 5 the perfect tool for an independent game developer. Advanced

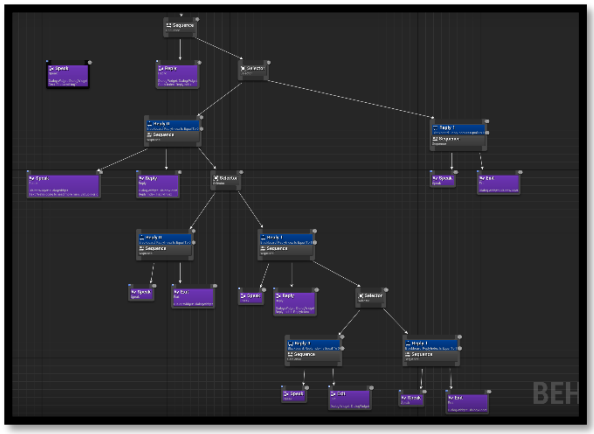
See here how the insane levels of detail present in the test level's interiors can be rendered alongside the high-quality character model without a massive hit to performance. Even running alongside the editor, I can walk around this interior at over 70 frames per second on a mid/high-end PC. This is without *any* time spent optimising. Lumen is quietly working away in the background too, bouncing light around the room realistically.

I hope the last couple of sections demonstrate the significant benefits of switching to Unreal Engine 5 and justify the increased focus on foundational developments.

2.3.2 Behaviour Trees *Updated*

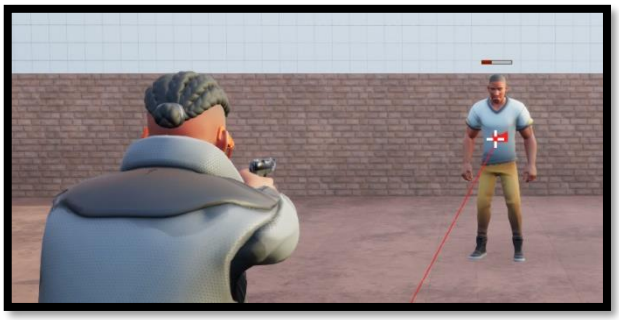
It's possible to achieve excellent results using UE's Behaviour Tree System, developing complex scenarios using Blackboard and Behaviour Tree editors. Below is a relatively simple example, which showcases an early draft of the Partner's Behaviour Tree (BT). In essence, this BT alternates between the default and combat states of the partner.





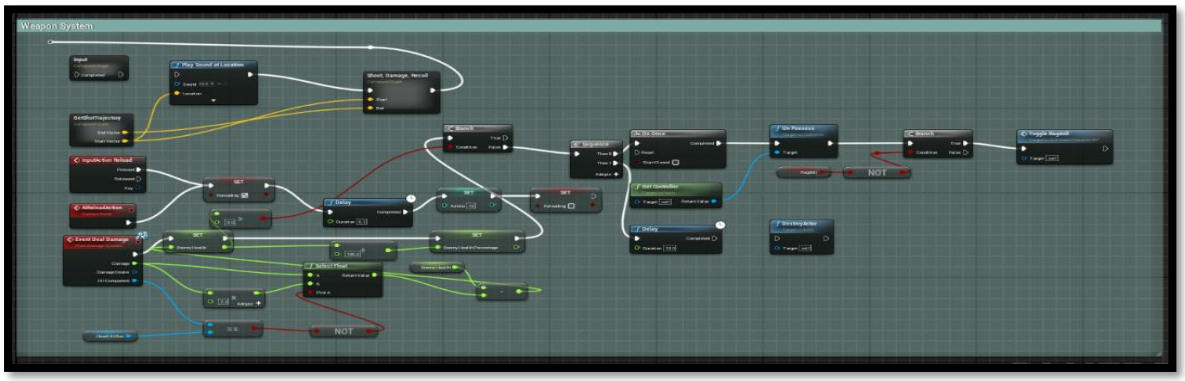
Final Report Update: There was less of a focus on artificial intelligence than I had originally planned, thanks in large part to the Unreal Engine 5 transition - and the ramifications of abandoning ALS v4. Unreal's Behaviour Trees have already been used in numerous ways, however. Most notably, the Conversation/Interaction System, discussed more in the Gameplay Framework section, is populated entirely using Behaviour Trees. The Parter Character's Dialog Tree is shown to the right and contains all the Player's options and the responses that will be received.

2.3.3 Weapon System

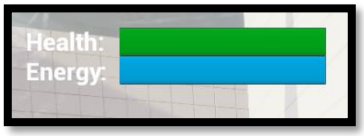


Players start the game with access to a police-issued weapon. Holding the "aim" button brings up a cursor (using widgets, which are detailed further in the GUI section) and clicking the "fire" button allows them to shoot, if they have enough ammunition. No 'bullet' moves through the space, however. Instead, a line is traced from the centre of the camera (as opposed to the weapon itself) out into the world at a 90-degree angle and its impact point is recorded. Every character in the world is 'listening' for these lines and if one impacts them, their health is reduced by a set amount, dependant on the location of the impact. For now, there are just 2 'hit-boxes', a sphere around each character's head and a rectangle that stretches from their shoulders down to their feet.

The player's partner and enemy weapons function similarly, and the player can be damaged in the same way. One section of the Weapon System blueprint is shown below. Just like BTs, these generate valid C++ classes when compiled.



2.3.4 Health and Energy

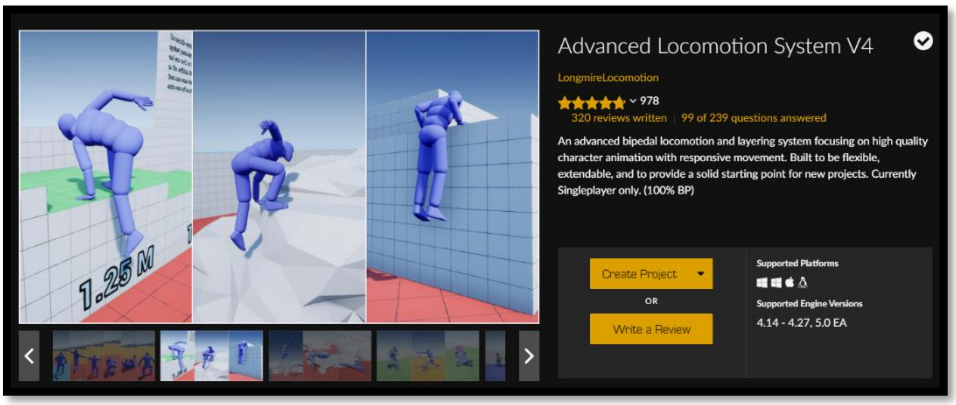


Every character in the game world can be damaged and if they sustain too much they will be killed. Their health is displayed using a Progress Bar which captures the current percentage of a variable called "health", representative of how much they have remaining. For NPC's, this bar only becomes visible when their health drops below 100, as seen here in the Weapon System section above.

The players health information on the other hand is always visible in corner of the screen, as seen in the image above. Just below the health bar is the energy bar. This will decrease whenever is performing strenuous actions (sprinting, jumping, mantling) and is stored in a variable like health. This stat is unique to the player character however, so NPCs do not need to account for this.

2.3.5 Advanced Locomotion *Updated*

ABSENT will make use of the open-source Advanced Locomotion System v4, developed by LongmireLocomotion [1]. This has made it possible to quickly scale up a third person character with highly detailed animations. This system is being used by the player character and all other characters in the game. ALS also provides a basic mantling system, ragdoll effects, sprinting, crouching and animations for all sorts of items including weapons.



Final Report Update: A decision was made to abandon the use of ALS v4. This decision was made as part of a strategic pivot towards the latest version of Unreal Engine and away from the no longer supported ALS framework (and its limited documentation). This decision was not made lightly, with significant work required to replace the framework provided by ALS. However, alongside the benefits of the newer engine, the system developed in ALSs places would be entirely custom and tailor-fitted to my project. This entire process is detailed thoroughly in the following sections.

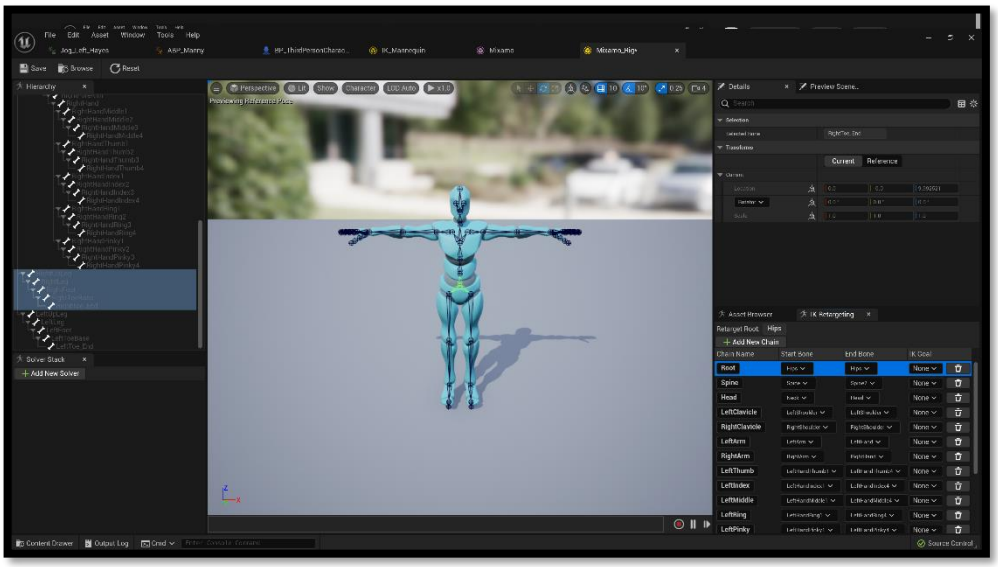
2.3.6 Custom Gameplay & Animation Framework *New*

What was initially the most terrifying point in Absent's development so far (the abandonment of ALS and development of my own gameplay framework) turned out to be the most exciting, interesting and rewarding aspect of the project so far. '*Exciting'* because I was working with cutting edge software with seemingly limitless possibilities. '*Interesting'* because I discovered a passion for the development of animation systems and gameplay mechanics. '*Rewarding'* because the end results will benefit this project now and long into the future, providing Absent with a unique game 'feel' and identity. I also have a thorough understanding of every inch of the project, unlike ALS, which was not designed to be used by beginners.

2.3.6.1 Animation Framework *New*

Firstly, it was necessary to acquire animations for use in the project. The 'final' implementation includes animations from a variety of sources, including sample Unreal Engine projects, the Unreal Engine Marketplace [3] and third-party animations* from Mixamo.com [4].

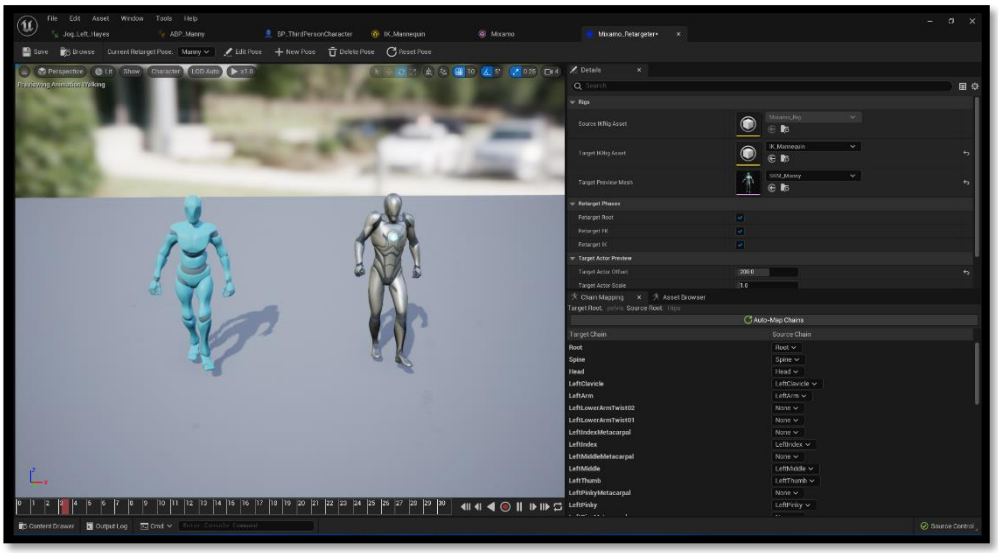
The image below showcases one part of the retargeting process required to use external animations. Since these animations were not created for my project or for Unreal's 'skeleton' specifically, they each required a form of retargeting. This procedure is baked into Unreal Engine and involves the labelling/matching of ***each and every*** bone, in both the source and the target skeletons. Some of this is automated, but most of it is not! To assist with this process, it is possible to create



an IK (inverse kinematics) Rig. This rig acts as a 'translator' of sorts and once plugged into an 'IK Retargeter', it is possible to convert an animation for use with any other type of skeleton, such as the one shown below.

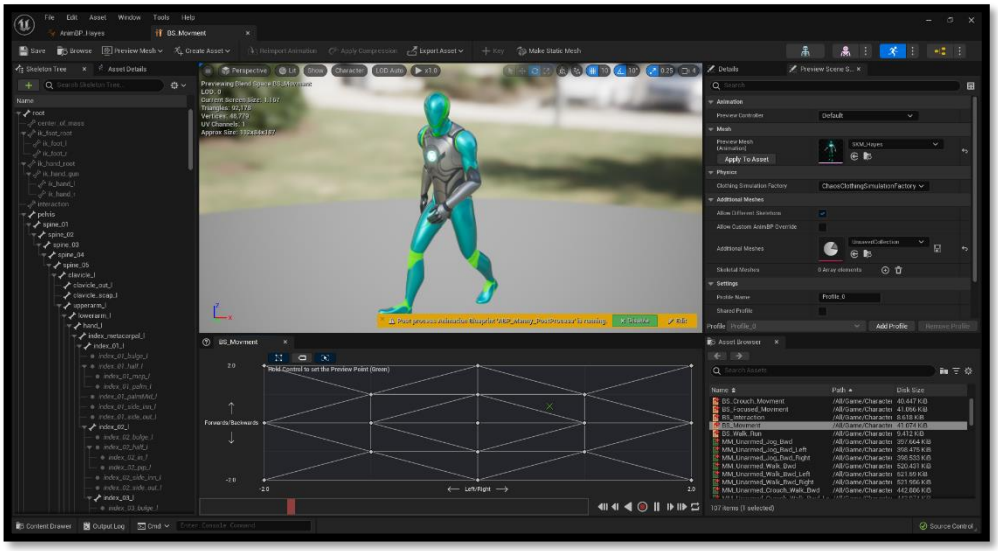
** These animations are licensed for use in commercial products.*

This makes it possible to animate my own custom character using these animations and make use of them in-game. The image below shows just one of these animations being successfully converted.



It might be apparent that this animation isn't lining up correctly (the shoulders on my custom character are too high) and this highlights how much work was required to bring all these disparate elements together. Each animation, even after the extensive retargeting process, was tweaked and adjusted to look and feel correct.

The 'final' implementation of the player character features movement animations for 8 directions (left, right, backwards, forward-left, backwards-right, etc) 2 speeds (run and walk) and crouching. That's a whole lot of unique animations. These are all implemented using a 'blend space', which can interpret the value(s) of a given input and output the resultant animation (or a combination of animations).

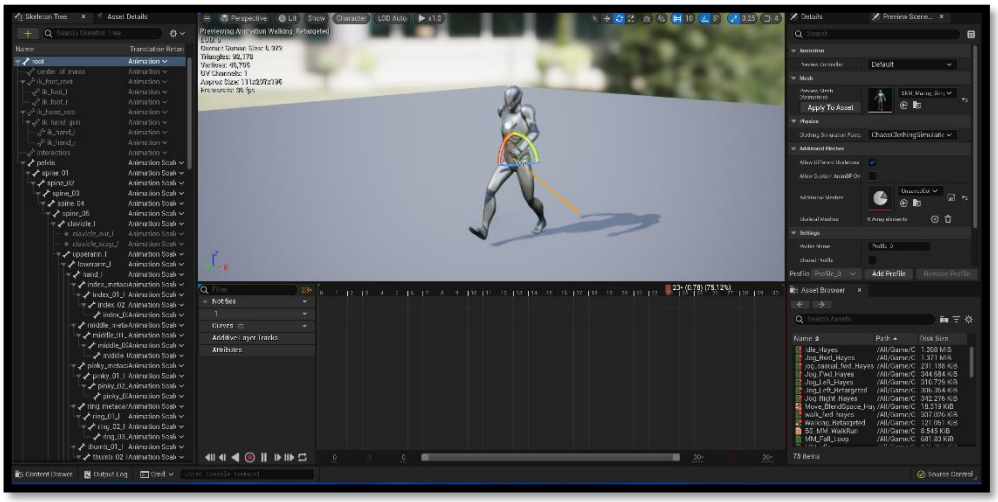


The image above demonstrates the current iteration. It is possible to input values such as character velocity or ground speed into a Blend Space like this one, but I've chosen to have this blend space accept the values 0, 1, 2, -1 and -2 on both the X

and Y axis (or a combination of 2 values on both axes for diagonal movement). This restricts movement to the 8 directions specified (from the player's point of view this is simply the commands forward, run-forward, right, left, etc) and this limitation prevents the sometimes-erratic blending of two or more different animations when the graph is hovering between values. Only accepting those fixed values did cause a jerkiness to movement, but the Blend Space settings make it possible to smooth the transitions between animation samples. Essentially, even if player input on a given axis changes from 1 to -1, the blend space will decrease the value slowly, which causes the animation to seamlessly transition between walking left and walking right, for example. There was some trial and error discovering a nice balance between the realistic simulation of weight and the responsiveness required for fun gameplay. I am pleased with the result for now, but I'm sure more tweaking will be required as more gameplay systems are added in the future. Thankfully, many of these settings can be changed relatively painlessly.

A very similar Blend Space was created for crouched movement, but different animations were plugged into the values seen above. Crouched movement also required adjustments to the player character 'collision sphere', which will be detailed in the Gameplay Framework section.

All these elements came together to create a believable character. However, it wasn't *perfect*. I once read that the problem for animation in games (and other mediums) is that everyone who sees it is an expert animator. Which is to say, we all know exactly what human movement looks like, and we're very, *very* good at spotting something that doesn't look human. In this case, there was something slightly 'floaty' about the movement, like the movement of the character's limbs weren't driving the locomotion. And that's essentially because they weren't. To alleviate this issue, **root-motion** was required.

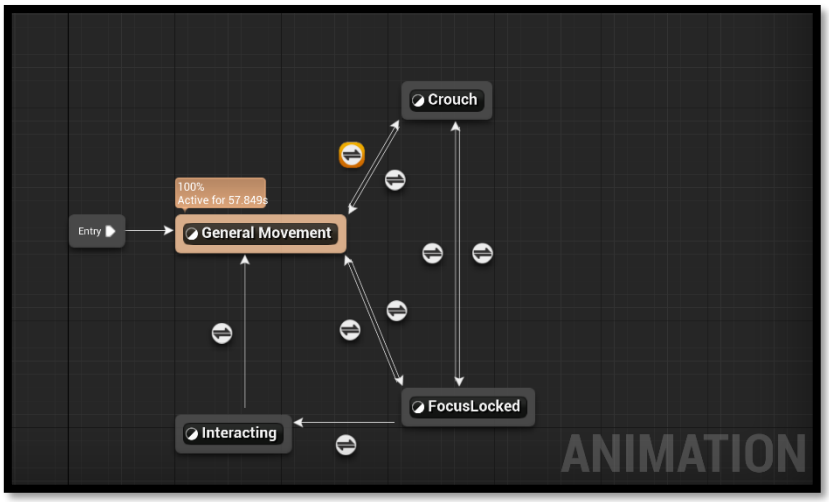


The image above shows the walk-forward animation (before the shoulders were fixed!). The red line demonstrates where the characters is in relation to their 'root'. As you can see, as the animation plays, the character is moving further and further away from their 'root', before resetting to its 'root' or '0' position when the animation loops. Because root motion wasn't being employed (and animations were NOT driving locomotion) within my current Blend Space (the character was moving

at a set speed in an X or Y direction strictly according to player input), this wasn't a problem, per se.

However, root-motion makes it possible for a character in the world to interpret an animations movement and use that to drive locomotion and much more complex movements throughout the virtual world. For example, a jumping animation that starts at position A and lands at position B will allow the player to continue playing from B instead of resetting them to A once the animation ends. Most people have seen root motion work correctly at some point but didn't even notice because, well 'of course the jump starts *here* and ends *there'*. The methods by which the engine converts the animation into movement would require an entirely separate report, but for the purposes of this project, a surface level understanding is enough (at least for now).

By activating root-motion at an animation level in each of their respective settings pages (this forces the character to stick to their 'root' and is clearly activated when that red line seen previously is no longer visible) and then forcing the character blueprint to interpret root-motion from all animations within the character's



settings page, it was possible to make use of root-motion effectively. Essentially, instead of the player's input teleporting the character across the X or Y axis while an animation plays, the player's input plays a walking animation, and the animation drives locomotion throughout the virtual world. This took quite a while to figure out, but it's very cool stuff and dramatically increases the fidelity of movement. One caveat, not all movement is currently using root-motion, but this is something I would work on in the future.

The character's animation blueprint also includes an 'AnimGraph' which controls all animation output, demonstrated above. Each of the nodes above represent a 'state' that the character using this blueprint can transition in and out of. The arrows indicate these transitions and settings can be applied to each. For example, the highlighted transition between the **General Movement** and **Crouch** states becomes active when a Boolean called 'IsCrouching' is true. It's worth noting that once a transition becomes active, it will be used. The transition settings also make it possible to set the 'Smoothing Time' between states, so animations blend nicely. For context, the states above simply include the Blend Spaces discussed previously.

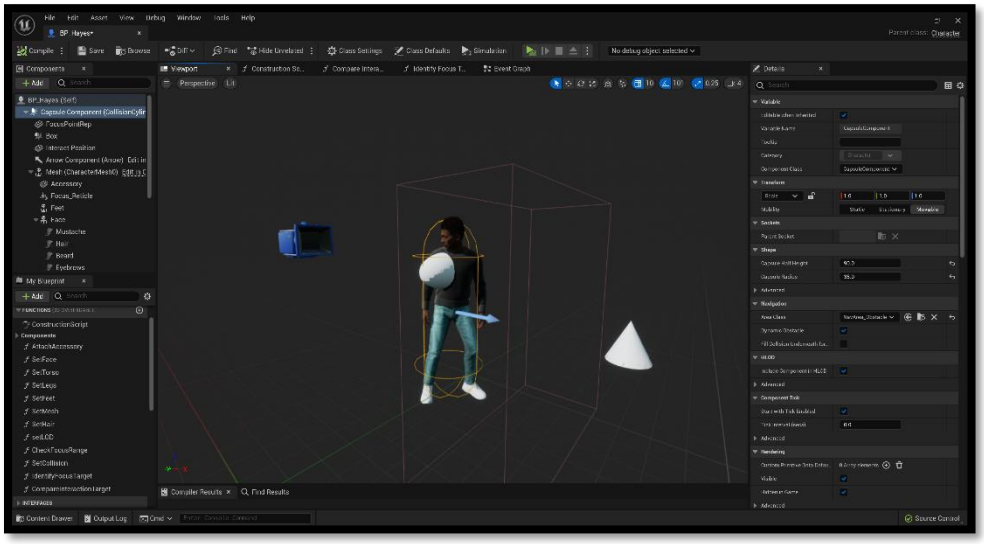
Also demonstrated is the Focus Locked state, which includes a similar Blend Space to the General Movement state but prevents the 'look around' idle animation from playing, so the character seems focused on the character they are talking to, or the object they are interacting with. The integration of the animation blueprint above and the character blueprint is essential, with animations *showing* the player what they are doing and what is happening, as opposed to *telling* them. See above the transitions between moving, focusing on an object and then interacting with it. The functionality behind this is detailed in the Gameplay Framework section, which details the character blueprint, but these animation states respond to those actions and give life to them.

2.3.6.2 Gameplay Framework *New*

Losing ALS also meant recreating much the gameplay framework from scratch. The image below shows the Player Character blueprint (This represents the parent class, which non-player characters inherit from). The base Unreal Engine Third Person Character template comes with some basic input and movement controls, but this would, of course, need to be extended upon significantly.

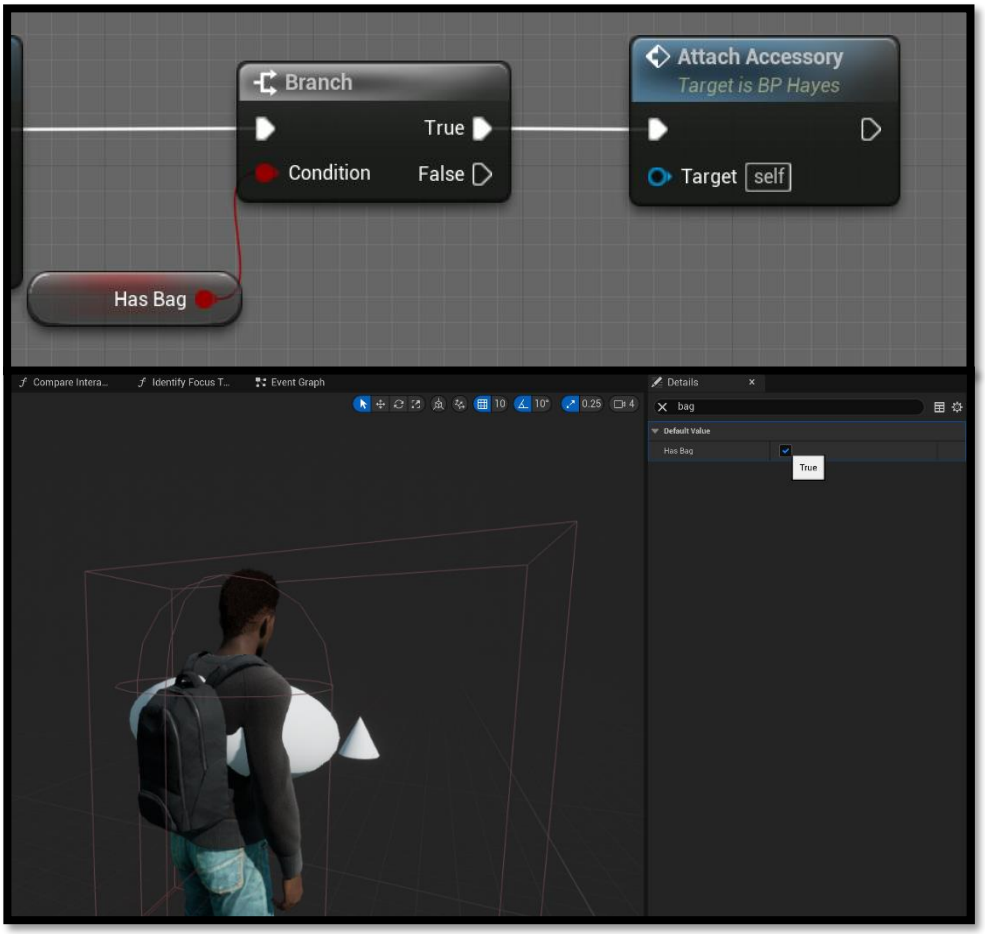
The dark boxes above include nodes and functions which come packaged with the base third-person character template. The lighter, grey boxes include nodes and functions that have been added during the development of this project. This includes the management of input, movement system, crouching, character customisation, interactable item detection, conversation system, and lots more. It also calls many sub-functions (not shown explicitly here) which I will include in the appendix. After the movement/animation system, the **focus system** could be the most deceptively sophisticated aspect of the final implementation. It allows the player to lock the camera to other characters / important objects and then interact with them in some way, such as initiating a conversation.

This blueprint also includes a viewport, which makes it possible to view and adjust how the object (or in this case the character) is represented in the world by adjusting their skeletal mesh (their 3D models), tweak more granular settings such as walk/run speed, scale, camera position relative to player, and lots, *lots* more. Since non-player characters also inherit from this class, it was challenging to allow all components to function without getting in each other's way. At some point, two characters from this blueprint will have to talk to each in a virtual environment and understand each other's position, status, etc. The conversation system is a great example of this and showcases how all these elements began to come together in a tangible, meaningful way.



In one laser-focused example of this blueprint's complexity, the white cone hovering in front of the character is not visible in-game. However, it's location is crucial and can be used to orient the player during conversations with other characters. When the player locks onto another character (using the **focus system** detailed earlier) and presses the action button, a conversation begins. The screen fades to black momentarily (with a User interface Widget like the ones detailed in the Graphical User Interface section) during which the player character is teleported to that white cone. But not *their* instance of the cone. They are teleported to the cone of the *non*-player character. But not *any* non-player character, it must be the player character they were focused on as they pressed the action button! Which means they will now be standing within a socially acceptable distance of the appropriate character and appear to be conversing. The player character is also rotated appropriately too, and their animation is changed to reflect their 'focus' on the conversation at hand. This system is almost entirely contained within this blueprint and one other, called Interaction Component, and this component can be 'attached' to any character in the world, streamlining future development and uses of this as a conversation and/or general interaction system.

The viewport also features components for the inventory system, such as the accessory component. An item, such as a backpack, can then be spawned into the world on this location when the right conditions are met. This is all a long-winded and technical way of saying; the player can pick up and equip a backpack. This is done here by creating a Boolean variable, in this case named ‘hasBag’, and enabling it's ‘instance variable’ setting. This makes it possible to edit the variable on an instance-by-instance basis (of the character class).



Once implemented correctly, it is then possible to adjust the current instance by using the character’s ‘Details’ panel on the right, or doing it programmatically.

It is also from the viewport that the player character is assigned their ‘MetaHuman’ skeletal meshes (more on this in Detailed Models section). In this iteration, on creation of this object in the world, the blueprint detects if a player controller (human) is present and assigns the Detective Hayes model if so. Otherwise, it assigns the relevant non-player character model, such as the player’s partner, Sam. This is a multi-step process, setting everything from the characters hair, face, torso, legs and feet to their style of eyebrow and eyelashes! Limited to 3 or 4 characters as I am in this iteration, that isn’t too big of a concern, but I would hope to streamline this process in the future.

2.3.6.3 Bring them to Life *New*

At this point, most of the components were in place to allow a player to move around a game world and perform some basic interactions. However, it is the smaller details that help bring these characters to life. The current iteration features ‘idle’ animations for characters who are still for a certain amount of time and every animation has been adjusted and tweaked to better fit the world of the game and its characters. As well as the tweaking of existing animations, some custom ones were created to enhance the experience, such as a ‘blinking’ animation which plays every couple of seconds to simulate blinking.

Unreal Engine has an animation editor baked in, which allows the manipulation of animations across a graph as seen above. In this example, the eyelid ‘bone’ (every part of the skeleton is referred to as a bone in Unreal) is selected and rotated 90 degrees. A ‘key’ is then set with the bone in this position. This key is then placed on the animation graph and its curve can be seen above. At roughly the 1 second mark, the key reaches a value of 1 (the curve up/down makes it rotate/close gradually), before returning to 0 a second later. There is then a small delay before this process repeats. I would like to introduce a small element of variation to the graph, but this aspect was not considered a priority and would likely be introduced in a future iteration.

2.3.7 Detailed Models *Updated*

Unreal Engine provides direct access to the Unreal Marketplace, where I can make use of open source and premium community assets, which will facilitate prototyping and development of the final version of ABSENT. Beyond the scope of this module, I would like to include completely original artwork and 3D models.

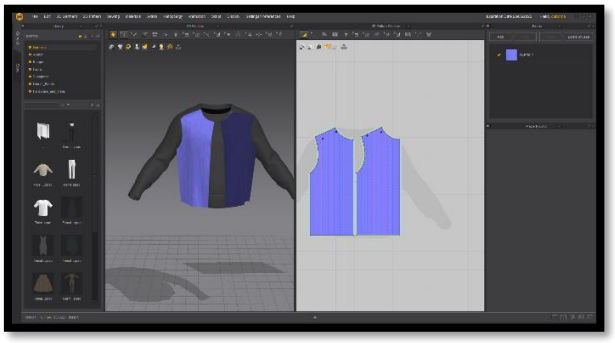
Final Report Update: The transition to Unreal Engine 5 provided the opportunity to focus on the next generation of console hardware and make use of the ‘MetaHuman’ creator,

available from any internet browser [5], which allows the development of highly detailed human characters, all of which can be imported directly into Unreal Engine. More importantly, it made it possible to create multiple, entirely unique characters very easily without the need for expensive assets or a professional 3D artist.

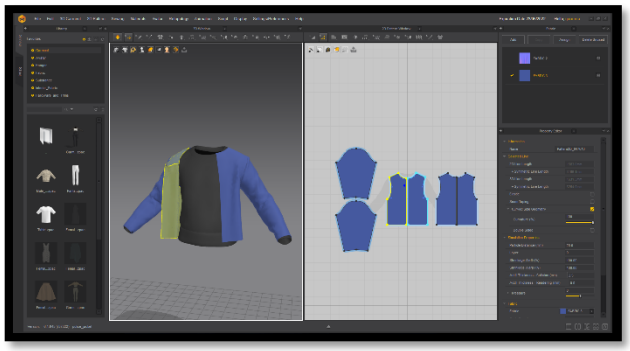
Although incredibly beneficial to the player character, it is even more crucial for non-player characters to make use of this tool. MetaHuman character models can increase and decrease levels of detail (LOD) as required. An exported MetaHuman includes up to 8 versions of the 'same' model, ranging from LOD1 (higher detail model with reduced performance) to LOD8 (lower detail model with increased performance and kind of creepy up close). It's possible to set the current LOD programmatically, such as increasing it when a character is being interacted with, or allowing the engine to determine the LOD procedurally, such as using the character's current distance from the player. This process can increase performance dramatically, without sacrificing graphical fidelity entirely. I mention this here, because its particularly relevant to non-player characters. The player character can likely sit comfortably at an LOD of 2 or 3 during gameplay, ensuring the character at front and centre 99% of the time looks great. However, I hope to, at some point in the future, render 4, 5 or 6 other characters on screen at the same time (during a combat sequence, for example) and this will make that possible without turning the game into a slideshow. Similarly, I had plans to feature a large crowd in a separate level, and this would be relatively easy thanks to the much lower LODs available. I will just need to make sure the player never gets too close *_shudders_*.

One side-effect of the move away from stylized characters, however, is the absence (look, a pun) of character models available publicly, such as the stylised police jacket featured in the prototype version of my player character. To overcome this issue, I could no longer ignore (or at least avoid) the development of custom 3D models. Thankfully, after some research on that front, I discovered a program called **Marvelous Designer** [6], which specialises in the development of cloth-based 3D models. I was aware that my skills as a 3D modeller were not going to improve enough in 2 months to create a work of art, but I assigned 1 week in March to the research and development of a single 3D-modelled jacket for use in my project. If it didn't come together in that timeframe, I conceded, it would be abandoned for the time being.

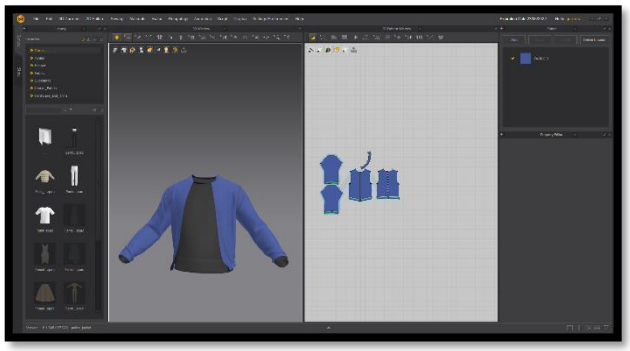
To begin with, I made use of an excellent 'Marvelous Designer to Unreal Engine workflow' tutorial [7]. This involved the export of my player character's torso into Blender, removing all the 'skeleton' elements associated with Unreal and saving it as an FBX file, before importing it into Marvelous. From there, I followed



some Marvelous Designer documentation available online, which helped me create a simple T-Shirt that draped over the torso object of my player character's torso object. Marvelous Designer is a really fun tool to use, and I can't believe I've never heard of it before. It's possible to create scraps of fabric using the 2D pane (see below) then click 'simulate' to see it portrayed in real-time in the 3D pane. For most, myself included, the first scrap of cloth will just fall to, and gather (surprisingly realistically), on the ground. But with time, and understanding of the tools available, some progress was made.



I managed to extend the sleeves of my new virtual T-Shirt and open the front, as seen in the image to the right. This image also demonstrates the ability to grab and pull a piece of cloth while in simulation mode, making it possible to review the garment in a tangible way. It might go without saying already, but I delved much deeper into the design of clothing during this week of development than I ever expected to (can't believe I just referred to the jacket as a garment…), this made it possible to create masculine silhouettes by tightening or loosening certain pieces of fabric, producing a masculine T-Shape, as opposed to a feminine hourglass shape. This project only involved the development of a single 3D model, as mentioned, but I have no doubt the skills and knowledge acquired while working on this section of the project will benefit its development in the future, when more characters and 3D models will be required. The workflow mentioned previously will also translate almost 1:1 when working with legs, feet, etc.



With the jacket's shape coming together, it was time to add the smaller details, such as a zip at the front, a collar, cuffs, and some logos made using Canva [8] and imported into Marvelous. The exported model is 'baked' as it is at that time, so it's possible to use the simulation mode and grabbing functions, mentioned above, to position the jacket correctly, giving it as natural a look as possible. It is theoretically possible to run cloth simulations in Unreal Engine, which would be very cool, but was not considered a priority for this project. With that in mind,

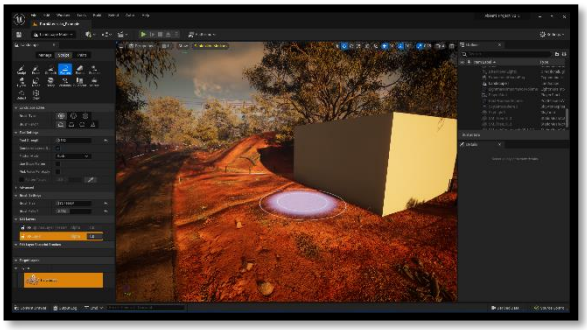
I returned to the workflow mentioned previously and moved the exported FBX file back into Blender and then imported it back into Unreal. This description of events brushes over the significant amount of trial and error required, during the learning and use of Marvelous Designer, not to mention the process of importing the file back into Unreal, which resulted in strange materials and missing logos/textures more often than not.

However, the end-result is more than good enough for this iteration of the project and can be improved in the future with Unreal's native cloth simulation and better textures/materials, though that would require other 3D modelling tools.

2.3.8 Special Mentions *New*

Many of the most significant developments since the midpoint presentation have now been discussed at length. However, there are several topics that I wanted to bring up, which I had the opportunity to learn about during this process, but may not have had the time to integrate in a major way for this milestone.

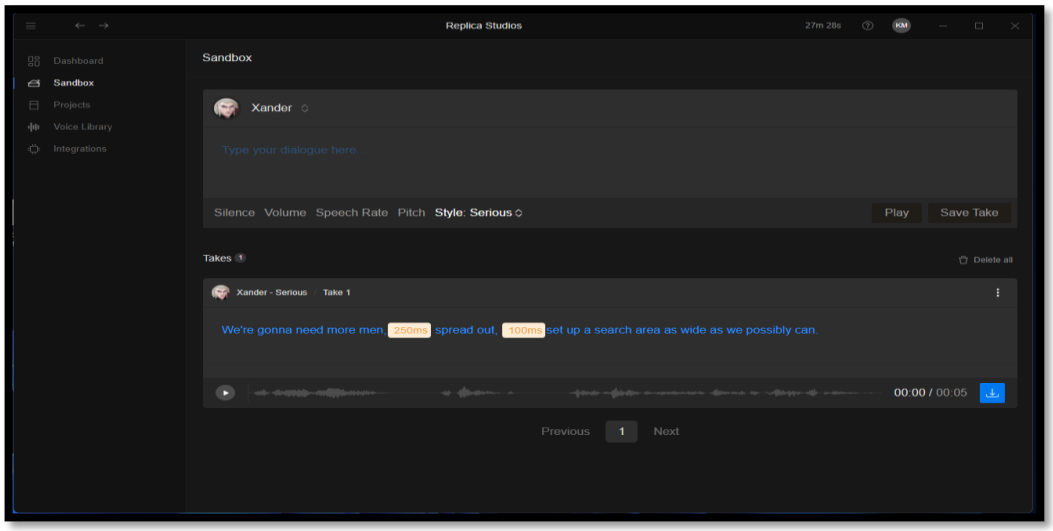
Unreal Engine offers sophisticated **Landscaping** tools, which allow terrain to be manipulated and adjusted quickly and easily. The image to the right shows an area being levelled out for a house (represented by white box). The tools form/deform the terrain and automatically adjust the textures and foliage present. These tools were explored but weren't considered a priority, considering the first level being developed is a flat, urban environment. These tools would be revisited in future iterations.



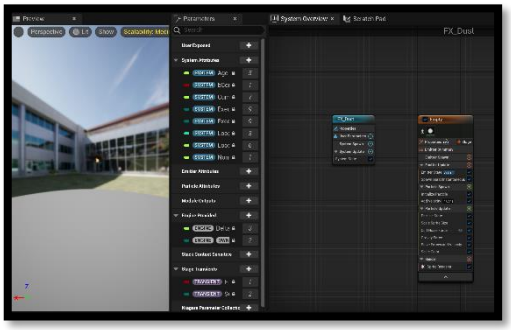
Briefly mentioned above, Unreal's **Foliage** tool makes it possible to scatter models around the environment. Once a mesh (model) is plugged into a foliage type class, this class can be used from the tool seen on the right. The class itself has settings (density, scale, etc) which can be adjusted too. The benefits for natural environments are clear, but there are *no* limitations to the models being selected. For example, debris or dirt you might see in a street could be used to add life to an urban environment.

There is no **Voice Over** included in this version of the project, for obvious reasons. But I did explore some potential alternatives, such as artificially generated voice over. See below a program called Replica Studio. This is a third-party software tool [9] but can be integrated directly into Unreal Engine. The results were *mixed*. There aren't many voices available, and those that are there were clearly designed for a more fantastical game.

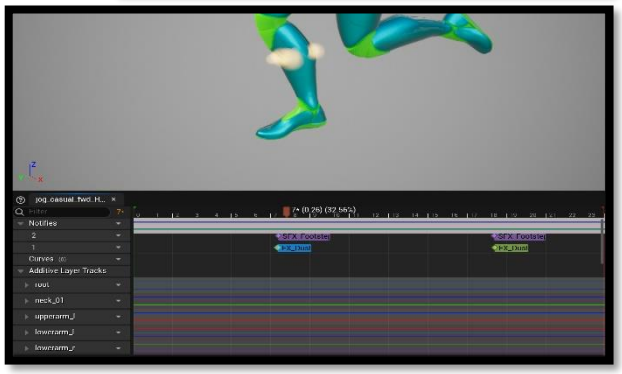
Some of the results were *kind of okay though,* so I explored some other of the other tools available online, such as Play.ht [10]. The voices were much more natural there, but a limited free trial encouraged me to park this for now.



Special Effects can be created in Unreal Engine using Niagara. I managed to create 1 (yes, 1) single special effect for use in this version of the game. Effects work somewhat similarly to animations, with a graph near the end that can be used to adjust how the effect looks. In my case, a simple dust effect was created to simulate the impact of footsteps.

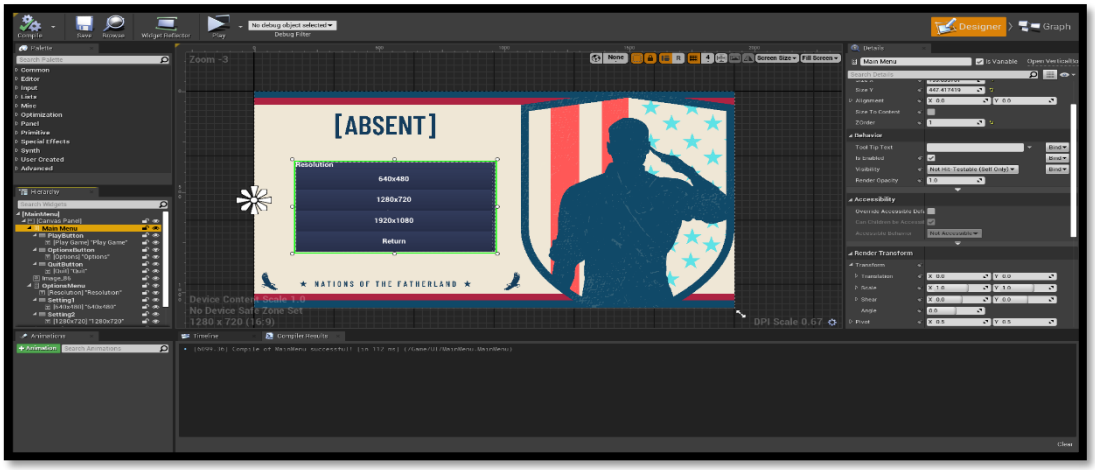


These effects can then be inserted into a level directly or plugged into specific animations. You can hopefully just about make out the blue tab on the graph to the right, which is taken from the run-forward animation graph. Notice how the dust cloud animations is playing and can be seen moving past the character's calf.

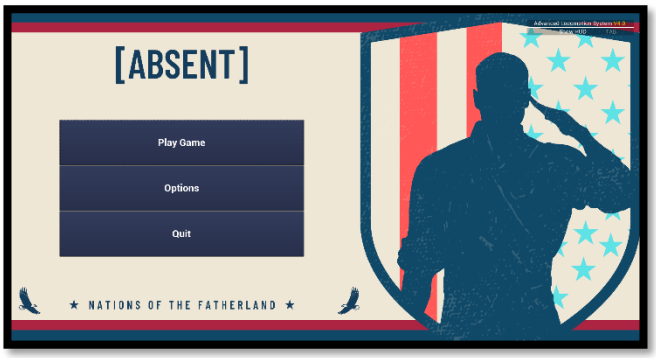


Sound Effects work similarly. The pink tab in the image above represents a simple 'thud' sound, simulating the impact of the foot hitting the ground. Again, I would need to dig deeper into these areas in the future, but these tools could be, for example, combined with voice over to create realistic facial animations that sync up with dialog.

2.4 Graphical User Interface (GUI) *Updated*



The user interface is controlled using 'Widget Blueprints'. These widgets make it possible to design each user interface and adjust individual elements and positioning as required. An example of this is the Main Menu, which is shown in these two images. Like Blueprints, compiling these widgets generates a valid C++ class for use within the game.



These widgets are flexible, and the "Persistent HUD" widget is shown below being used in-game, as opposed to a menu screen. It works similarly to the menu widget, but this information updates in real-time to inform the player of their status. Details such as their **Health** and **Energy** are pulled from the relevant variables and displayed on-screen.

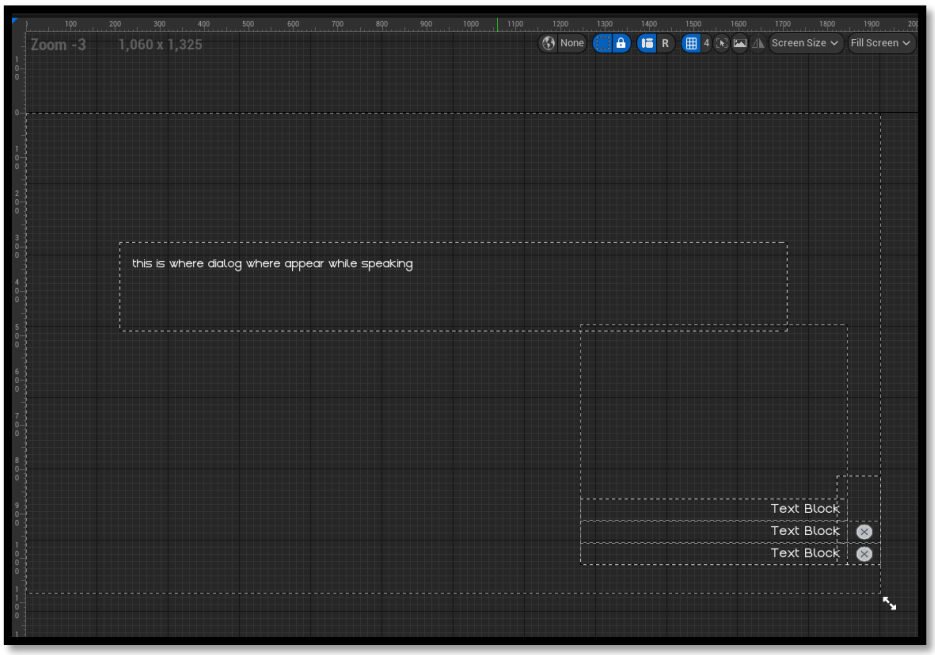
Additional UI elements will be needed to accommodate conversations with NPCs and general interactions. These will make use of widgets like above and will be showcased later.

Final Report Update: Some additions and changes have been made to the GUI to accommodate the dialog and interaction systems, using a combination of widgets, like the ones mentioned previously, and custom images created using Canva [8].

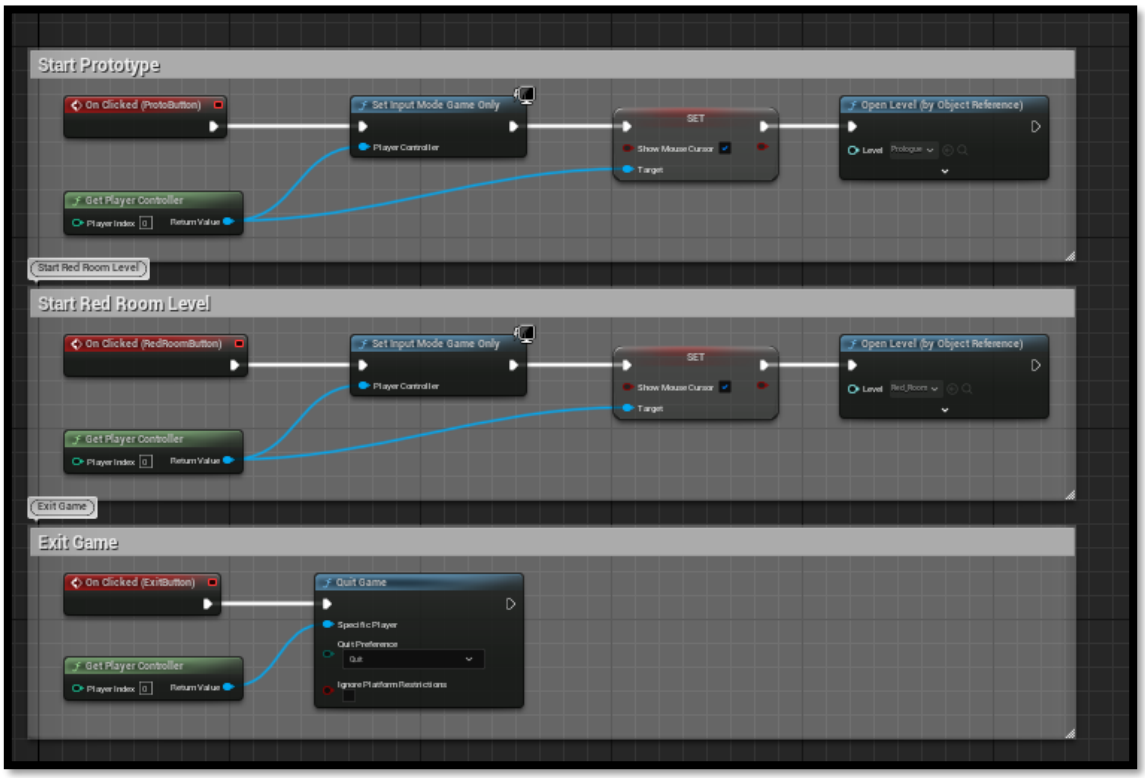
Notice to the right how the game's in-game GUI adjusts dynamically to suggest actions for the player. In this case, the very bottom right of the screen highlights the currently 'focused' object. This required a sophisticated tracing function to not only lock the players onto the object in question, but also label the object correctly, as seen here. This system had to be modular, accommodating a range of potential options, such as the character name above, or key interactive elements like doors or pieces of evidence that the player can view/collect. The prompts presented are entirely dynamic as well, in this case taken from the Partner Character's Behaviour Tree discussed earlier. Not shown here is the responses received (think subtitles), retrieved from the Partner's Dialog Behaviour Tree, as well.

All this modularity makes it possible to create and apply new dialog trees to any character/object in a scene on an instance-by-instance basis. The GUI widget shown here can then retrieve the relevant data and display that on the player's screen.

It is difficult to demonstrate all elements of the HUD concisely, as separate widgets are used to draw their own respective components. For example, the image below shows the primary 'DialogWidget' just discussed. It's clear how this corresponds to the image seen previously, but this doesn't include the small 'eye' icon, which is included in a separate widget class. I plan to consolidate these GUI elements in the future, where possible, but there are some benefits to maintaining independent components for development purposes.

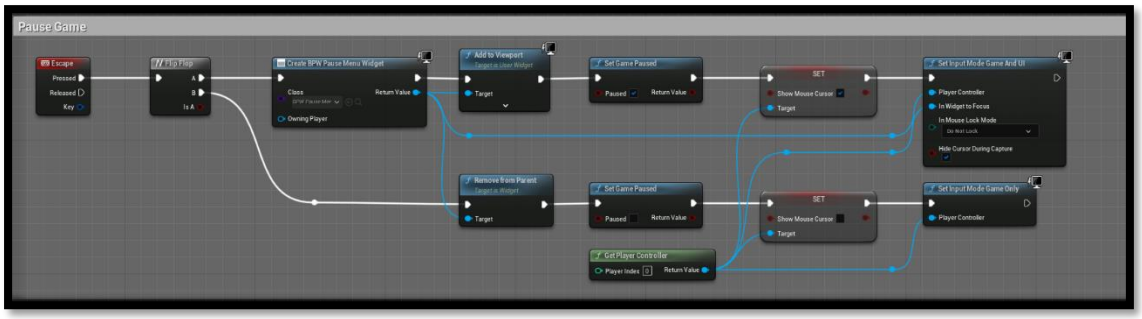


As well as the **Design** tab shown above (which acts somewhat like the viewport of an object class), each UI widget includes a **Graph** (which acts like the event graph of an object class) and looks a lot like the blueprints seen before.

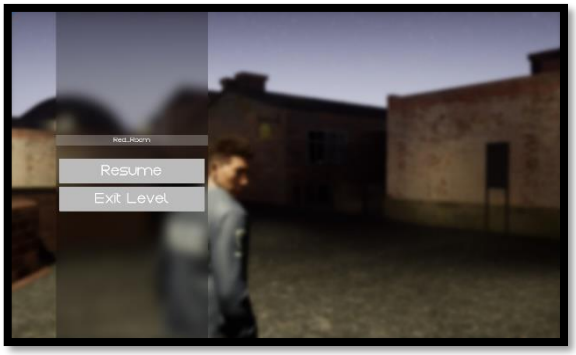


The main menu's event graph is picture above and shows what happens when each button is clicked. The first two buttons deactivate UI Mode (preventing the use of the mouse cursor) and open their respective levels with the built-in 'Open Level' function, which are both provided with the relevant Level assets. The exit button calls another native Unreal Engine function, 'Quit Game', that simply ends the current instance of the application. See below how this appears in-game. For context, the 'Red Room' option is being highlighted by the player.

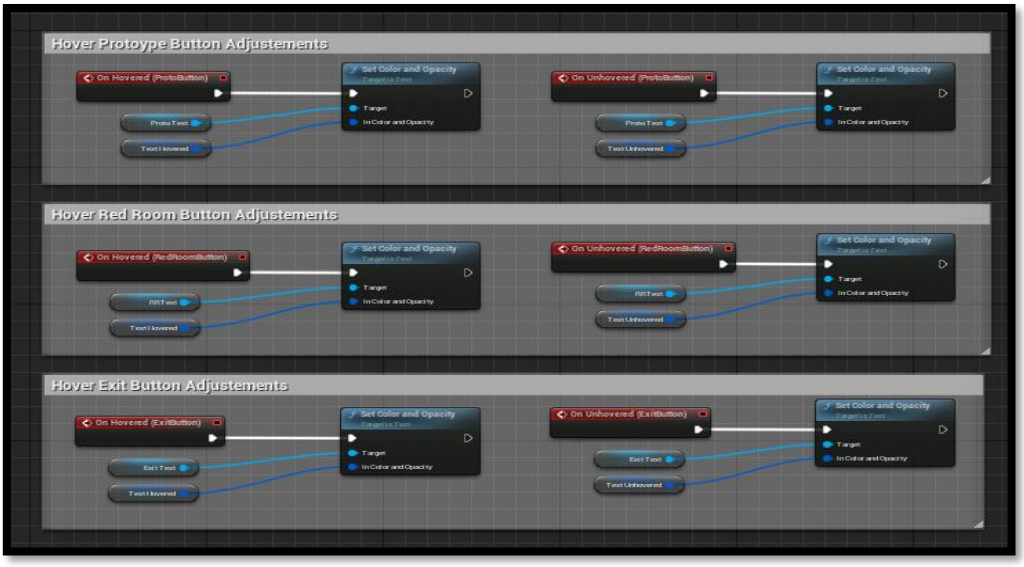
The **Pause Menu** required some integration with the character class discussed earlier. The blueprint below (taken from the character event graph) shows what happens on any press of the Escape key (which will be mapped to a controller button later).



The 'flip flop' node switches between the A and B outputs, creating or destroying the Pause Menu widget (created much like the Main Menu widget shown before), activates/deactivates UI mode, shows/hides the mouse cursor, and sets the native 'Set Game Paused' function to true/false, which pauses/resumes the level currently running. The widget also calls a native Unreal Engine function, 'Get Current Level' to display whatever level is currently running.



It's worth noting that these menus were created to act as a foundation for future developments and to streamline demonstrations, etc. Much more work would be required to create an enjoyable user experience. However, some small tweaks and adjustments were made to the widget's respective event graphs to ensure readability and usability. See below how each button is passed as a variable into the graph and an 'OnHover' function adjusts the appearance of the text held within. The text variable, which is a child of the button, is passed into the 'set colour and opacity' alongside a variable of type 'slate colour', which makes it possible to have any number of text colours/designs prepared. In my case, two slate colour variables, one for



both ‘hovered’ and ‘unhovered’, was enough. Now, selected menu items were ‘highlighted’ appropriately.

2.5 Testing

2.5.1 Test-Driven Testing

Games are difficult to test. It’s clear from the outset that a player can interact with an open world game in more ways than an iPhone user will interact with a music app, for example. However, my work with Mastercard this year showcased the benefits of automated testing, and I would really like to include some elements of that in my project. But even so, game development is an iterative process, much more so than the applications I worked on during my internship. Significant time is spent developing features that might fit initial requirements but end up not being used if they aren’t as fun as expected or are changed so significantly that they no longer even remotely resemble their initial iterations. Using any sort of automated testing at this point would be harmful to my development schedule or almost impossible to implement. Large teams often omit testing completely at this point, to ensure they are playtesting features as quickly as possible. In my case, I think it would make more sense to stick to Test-driven Development (TTD), which will streamline experimentation, and development of new features and level designs, while also ensuring these additions are built with quality and are designed with the player’s perspective in mind.

As I enter later stages of development, I might reconsider this option and create additional forms of testing. One of these options is Unreal Engine’s out of the box testing features, with hundreds of tests that will make sure all base functions and classes are working as anticipated. This includes Unit Tests and more importantly Integration Tests. The latter are used to evaluate larger pieces of software like gameplay mechanics or specific actions the player can take. It can also be used to test AI behaviour, which would be very useful in this project. These tests can be implemented in C++, but are even easier to implements using Unreal’s Blueprints, which I intended to use anyway.

2.5.2 Play Testing

Playtests will also be crucial to the success of the project. I intend to playtest an Alpha build of the game starting 15/04/2022 (as mentioned in my timeline) with Family and Friends and potentially advertise it online prior. Some players aware of my previous projects might be interested in participating and providing feedback. These playtests are invaluable, particularly if I can watch someone play live. This can showcase pitfalls in the design of levels, imbalances of difficulty and much more. What might seem obvious to me (playtesting the game for months) could be unclear to a new player. For players who I cannot watch live, I will create a feedback survey that they can complete online. I would like these tests to involve a mixture of player skill levels and preferences, to ensure the game scales well across a diverse range of players. The story of ABSENT is intended for mature audiences and some story elements might be unsettling for some players. I intend to submit an Ethics Form and discuss this in more details with my supervisor, prior to carrying out any playtests or gathering feedback.

2.6 Evaluation *New*

2.6.1 Technical *New*

The project must be continually evaluated from both a **design** and **technical** point of view. Firstly, as more and more systems and models were added into each scene, it was crucial that one eye was kept on the potential performance ramifications. Much has been documented in this report about the power of Unreal Engine's new **nanite** rendering technology regarding performance improvements, but there are still limitations. Too much detail in one area, too many AI systems or particle effects being fired off at the same time, for example, can cause significant slowdown or crashes. More important than running at a high of frames per second (FPS), is running consistently across all scenes. I didn't discover any sure-fire way of identifying these issues, besides thorough and consistent testing. My target FPS throughout development was 60. Any scenes or elements that caused a drop below that would be reworked or dropped entirely, determined on a case-by-case basis.

To review this consistently, a significant number of formal and informal tests were performed throughout development. This included running through the test level as additions were made or performing specific stress tests, like adding multiple NPCs characters into a scene to find the preferable level of detail range across the board. There may be more sophisticated tools available, but for my purposes the FPS counter switched on almost always made it clear what (if anything) was causing an issue.

Notice above how the FPS dropped below target as the 6th NPC entered the level. These tests help identify potential performance concerns, as mentioned, and sometimes inform design decisions, such as the number of enemies per level, the size of the levels themselves, and more.

2.6.2 Design *New*

I had plans to perform some playtests last month, but the project was unfortunately not in a playable enough to state to gain anything useful from those tests. For now, the project continues, and will continue, to follow a Test-Driven Development methodology, which is streamlining experimentation, the development of new features, and level designs. This will of course change in the future, as the opportunity to expose these game mechanics and level designs to potential players (and learn where they falter) will be invaluable.

3 Conclusions *New*

In some ways this project began in 2019 when I started learning and using a much simpler Game Development Kit call Dreams [11] to make my very first game. Dreams somehow managed to run on a PlayStation 4 console, which is incredible in its own right, but has its limitations. I eventually reached a point at which I had learned all I could from that tool, and shortly before September 2021 I began to toy around with more sophisticated 3D game engines, such as Unity, and eventually Unreal Engine.

This project was an attempt to translate the skills I had acquired over to a professional GDK, and an attempt to produce a game that would buck a trend often associated with independent game development: low fidelity. My initial prototype tried to find a comfortable middle ground, with stylised characters and environments, but my project lacked the 3D artist required to really make that strategy work, and so I decided to double down on high fidelity, making use of some of the most advanced 3D rendering technology publicly available.

I knew the lack of documentation available online (UE5 left Early Access in March!) would make this transition complex, but considering how ambitious the project has been since the beginning, it seemed appropriate to at least consider the possibility. This would eventually become a great decision, but I did not reach that point without some *issues* along the way.

I was understandably excited about ALS v4, which is an incredible (if unapproachable) framework to build a game from, I *underestimated* how easily the skills I acquired from similar GDKs would translate over to Unreal, and I *overestimated* how accessible 3D assets would be. It's clear I made a lot of mistakes (most of them not even listed here), but I learned just enough along the way to pivot. The move towards Unreal Engine 5 was risky, but I had learned enough about the benefits to know it would significantly benefit the projects development, now and in the future. Unlike something like ALS v4, Unreal was created and would be supported by Epic long into the future. Even 1 month after the release of its new engine, there is a significant amount of supporting documentation online, which allowed me to make use of all the new features available and streamlined development.

The **MetaHuman Creator** helped the project overcome a significant portion of the 3D model issue. **Lumen** and **Nanite** streamlined, optimised and improved the appearance of Level Designs across the board. The **Gameplay Framework** built in place of ALS v4 forced me to dive deeper into the guts of the project and the engine than I had originally planned to, but the benefits of this, from a project *and* educational perspective, were significant. Finally, the character animation systems were built almost entirely from scratch – I now have a genuine understanding of those systems, as well as a gameplay 'feel' that is entirely unique to my project.

There is something incredibly satisfying about creating systems that are now beginning to *click* into place; conversing with another character, pressing a button and seeing the door swing open, *simply* moving around the environment using the custom movement system. More important than that, however, the components created so far are *modular* and *scalable*, which means they have the power to eventually become much more than the sum of their parts.

4 Further Development or Research *New*

My intent with this report was to visualise the current trajectory of the project. It's likely clear already that I'm not finished yet, and my hope is that the current trajectory, visible from September 2021 through May 2022, will persist over the next 2-3 years, or longer. Unfortunately, this is simply not possible due to the financial requirements/restraints of the project, and the amount of time required. Essentially, there will be more than one lull in this project's likely turbulent future. Having said all that, I have plans in place for gameplay mechanics, characters, levels, the overall theme, and the plot (something which wasn't relevant to this technical report).

The **Gameplay Framework** already developed and detailed in this report represents a foundation upon which more sophisticated mechanics and levels could be introduced. I intend to expand upon the weapon systems, artificial intelligence of NPCs, stealth mechanics, inventory system and conversation/interaction system.

Level Design would be updated to reflect and enhance these new features and mechanics, expanding upon ideas and concepts not completed before this milestone.

New Characters have already been written and designed, to be included in updated versions of the featured (and subsequent) levels. They include allies, enemies and everything in between.

3D Models will be required for these characters designs and the levels they inhabit. This issue was avoided up until now, but the project will not be capable of overcoming this hurdle without assistance from an expert in the area.

The experience would benefit greatly from **Voice Over, Original Music** and additional sound effects. This would be an expensive but transformative addition.

All these disparate elements will hopefully come together in way that is cohesive, engaging and enjoyable for the player, telling what I believe is an interesting story. Having said that, the story of the game's development is likely to have just as many twists and turns.

5 References

[1]"Advanced Locomotion System V4 in Blueprints - UE Marketplace", Unreal Engine, 2021. [Online]. Available: https://www.unrealengine.com/marketplace/en-US/product/advanced-locomotion-system-v1. [Accessed: 07- Nov- 2021].

[2]"Nanite Virtualized Geometry", Docs.unrealengine.com, 2022. [Online]. Available: https://docs.unrealengine.com/5.0/en-US/nanite-virtualized-geometry-in-unreal-engine/. [Accessed: 12- May- 2022].

[3]"Marketplace - UE Marketplace", Unreal Engine, 2022. [Online]. Available: https://www.unrealengine.com/marketplace/en-US/store. [Accessed: 12- May- 2022].

[4]"Mixamo", Mixamo.com, 2022. [Online]. Available: https://www.mixamo.com/#/. [Accessed: 12- May- 2022].

[5]"Epic Games", Metahuman.unrealengine.com, 2022. [Online]. Available: https://metahuman.unrealengine.com/. [Accessed: 12- May- 2022].

[6]"Marvelous Designer", Marvelous Designer Official Site, 2022. [Online]. Available: https://www.marvelousdesigner.com/. [Accessed: 12- May- 2022].

[7]"Custom Clothes for Metahumans (Blender, Marvelous Designer, Ue5) | Community tutorial", Epic Developer Community, 2022. [Online]. Available: https://dev.epicgames.com/community/learning/tutorials/MlLM/custom-clothes-for-metahumans-blender-3-marvelousdesigner-ue5. [Accessed: 12- May- 2022].

[8] Canva, 2022. [Online]. Available: https://www.canva.com/. [Accessed: 12- May- 2022].

[9] R. Studios, "Synthesize Voice AI and Natural Sounding Text-to-Speech — Replica", Replica Studios, 2022. [Online]. Available: https://replicastudios.com/?gclid=Cj0KCQjw4PKTBhD8ARIsAHChzRImYpSdJEIcJmwrzRNOHRd02WOoegA20_kdxYjUNapBtGREDBbFFKsaAjm6EALw_wcB. [Accessed: 12- May- 2022].

[10]"AI Voice Generator & Realistic Text to Speech Online", Play.ht, 2022. [Online]. Available: https://play.ht/. [Accessed: 12- May- 2022].

[11]"Dreams", Media Molecule, 2022. [Online]. Available: https://www.mediamolecule.com/games/dreams. [Accessed: 12- May- 2022].

6 Appendices

6.2 Reflective Journals

Supervision & Reflection Template

Student Name	Keith Mahony
Student Number	X17521139
Course	BSHCD4

Month: April

What? The transition to Unreal Engine 5 and away from ALS v4 was a daunting one, but the results, from a project and educational point of view, could not have been more positive. The animation system and much of the gameplay framework are now complete. I even managed to import both metahuman characters created last month into the project and have both the player and partner characters making use of their models. This was tricky, as the MetaHuman imports and my player character don't have identical skeletons, but it was possible to rework some of that and get everything hooked up.
So What? I'm entering the final stretch and things have been coming together quite nicely. There is quite a lot that won't make it in in time, such as the weapon system and much of the artificial intelligence systems I had proposed initially. But the foundational work completed so far will benefit the project in the long run and I'm grateful for that.
Now What? I hope to do some work this month on the first level design, which hasn't received much attention over the course of this project. I have been making small additions to the level throughout development, but those changes were mostly made for development purposes and not with a player in mind. I also plan to take full advantage of UE5s Lumen and Nanite technologies to streamline this development as much as possible.

Student Signature	Keith Mahony

Supervision & Reflection Template

Student Name	Keith Mahony
Student Number	X17521139
Course	BSHCD4

Month: March

What? After much consideration, I decided to transfer the project over the Unreal Engine 5. This move will also include the removal of ALS v4 from the project and the development of a custom movement and animation systems in its place. This is an on-going process, but the new system is showing promise. Meanwhile, the features developed up until now where not reliant on the ALS framework, and so much of that was relatively easy to transfer over too. The MetaHuman character creator has made it possible to create unique characters at a much higher level of fidelity and I now have some early versions of the player and partner characters created.
So What? Unreal Engine 5 has provided access to incredibly powerful tools, such as Lumen (a new take on Global Illumination), Nanite (a new rendering technology) and MetaHumans, which are technically possible to use on UE4, but were designed for this new engine. All of this combined has made it possible to overcome some the difficulties facing the project and this month has been incredibly exciting.
Now What? Some work remains to be done on the gameplay and animation framework being built. Animations require a form of retargeting if they're acquired from anywhere outside of the Unreal Sample projects and I intend to recreate the movement system of ALS v4, which allowed movement in 8 different directions and allowed from sprinting, crouching, etc. This is big undertaking, but initial results have been promising.

Student Signature	Keith Mahony

Supervision & Reflection Template

Student Name	Keith Mahony
Student Number	X17521139
Course	BSHCD4

Month: February

What? The focus system was worked on this month and proved to be a deceptively complex mechanic, with sphere tracing and several new functionalities of the engine required to get it working as it is now. I'm happy with the outcome, but many of the issues were caused by the incredibly complex ALS framework. The lack of official documentation available online has made this the Achilles heel of the project so far.
So What?

I'm considering a couple of major overhauls to the project. Firstly, abandoning ALS v4 (which has been troublesome) and secondly, merging the project over to Unreal Engine 5. These changes (if done correctly) would not undo any of the progress made so far. However, the potential benefits would be significant, including the MetaHuman character creator, Lumen and nanite. These tools would help overcome many of the issues plaguing the project in recent months.
Now What? A decision must be made soon. I'm relatively certain this would be good for the project, but some research and investigation will be carried out this month to weight up the pros and cons.

Student Signature	Keith Mahony

Supervision & Reflection Template

Student Name	Keith Mahony
Student Number	X17521139
Course	BSHCD4

Month: January

What? A lot of work has been done this month on the conversation system and it's in a good place now. Some GUI elements have been developed for it as well, though some of these aren't working perfectly yet. I have some concerns regarding the lack of 3D models available. My initial budget laid some relatively cheap options, but my circumstances have changed recently, and these may no longer be an option.
So What? Some considerations must be made to accommodate what will likely be a much smaller budget. Levels will need to be from scratch in almost all cases and character models will be limited.
Now What? I plan to move on to other game mechanics, such as the focus system, and hopefully integrate this into the conversation system mentioned above. I also intend to investigate possible solutions to the ones mentioned above.

Student Signature	Keith Mahony

Supervision & Reflection Template

Student Name	Keith Mahony

Student Number	X17521139
Course	BSHCD4

Month: December

What? This month I produced the midpoint presentation, which provided a significant update on the progress made so far and where I hope to go from here. Many of the mechanics and systems are now under development, such as the conversation and movement systems. Some simple menu systems have also been created for pausing the game and starting the first level.
So What? Milestones like this provide an opportunity to review the progress made so far and consider how much is possible to make before the next one.
Now What? I intended to have more of the project completed by this point, but a busy first semester prevented that. I'm hoping to make much bigger strides in the coming months, and from what I understand Semester 2 is designed to accommodate that.

Student Signature	Keith Mahony

Supervision & Reflection Template

Student Name	Keith Mahony
Student Number	X17521139
Course	BSHCD4

Month: November

What? This month I produced the project proposal, which details almost every aspect of this project. However, before that I created a Game Desigen Document. This document places more focus on the project from a game development point of view and provided an opportunity to document in extreme detail many of the mechanics and systems the game will consist of. It also included details of the third part assets I hope to use, and how they will be implemented.

So What? Development has become more focused. I have committed to the title “ABSENT” and narrowed the scope of this project down to what would be Episode 1 of a series (which could be continued after completion of this project).
Now What? With clear requirements and objectives in mind, development has begun, and I intend to have many of the core systems created in time for the mid-point presentation. This includes a first draft of the level; some core gameplay features, user interface/menu and AI systems.

Student Signature	Keith Mahony

Supervision & Reflection Template

Student Name	Keith Mahony
Student Number	X17521139
Course	BSHCD4

Month: October

What? A significant portion of my project time this month was dedicated to idea generation and early experiments with new technologies such as Unreal Engine and Unity. After some research, I’ve decided to learn and use Unreal for this project, with its well implemented and accessible AI systems (Behaviour Trees), increased graphical fidelity, out-of-the-box multiplayer features, and its reputation as a AAA studio and game development engine. I already had a vague outline and some ideas in mind for the type of game I wanted to make, but it was now time to be more specific. “VICE” / “[REDACTED]” / “Absent” (working titles!) is a story-driven third person, action-adventure game, designed in a similar manner to games which are usually called “immersive sims”, like Deus Ex or more recently Dishonoured and Deathloop.
So What?

As mentioned above, Unreal seems like the right engine for the game I'm creating, and with that decision made I could now start experimenting with the engine and generating new ideas for gameplay systems and mechanics. This part of development is fun, but it won't be long before this transitions into a requirements specification. This will be the next major challenge and I'm hoping to meet with my supervisor this month to discuss this. Also, while experimenting with Unreal, the story of the game was starting to come together (which led to the working titles above!). The story I have in mind is making me consider the need for an Ethics Declaration Form, which is something else I'll need to discuss with my supervisor.	
Now What? This month I'm planning to document my requirements specification for the project. I will try to have an early draft together before meeting with my supervisor to go over the fine details. Experimentation with Unreal has gone well so far and I'm already getting to grips with the software. This month I'm hoping to identify some third-party assets (models, animations, sound effects) which will help bring the world to life. On top of that, I have already built some custom functionality and plan to continue learning practically next month. An ethics declaration form may be required and I'm planning to meet with my supervisor to discuss the need for this in my project, with the story I'm planning to tell.	
Student Signature	Keith Mahony

6.6 Project Proposal

National College of Ireland

Project Proposal

"**ABSENT**"

01/11/2021

BSHC4

Software Development

2020/2021

Keith Mahony

X17521139

X17521139@student.ncirl.ie

Contents

7 Objectives

ABSENT is an action-adventure game set in a grounded but fictional world, with players taking on the role of a detective tracking down a missing person. The game is inspired by others that are usually referred to as immersive simulations or 'immersive sims'. Examples include Thief and Deus Ex in the late 1990's and more recently Dishonoured and 2021's Deathloop. These games feature open-ended level design, sophisticated enemy AI and many, many ways for the player to interact with those enemies and the worlds they inhabit. Thief and Deus ex were also two of the first games to offer non-violent solutions to video game objectives –an important step for a medium which was viewed at the time (in some ways fairly) as an ultra-violent toy for (male) teenagers.

It would be impossible to match or exceed these works of art, but this project is an attempt to showcase an understanding of the design philosophies behind them, implement them on a smaller scale, and add some new ideas into the mix.

Immersive sims don't necessarily need the fanciest graphics, the biggest worlds or make use of the latest technologies – but this has often made them infamously difficult to advertise. Most publishers seem to believe the solution to that problem is to showcase how *complicated* the games are, how many *systems* they have, or how many ways you can play them. This might seem like a good idea, but it very rarely seems to connect with the wider audience it seems designed for. I think it's possible to create game with the sensibilities of Deus Ex or Dishonoured, without feeling the need to explain to your audience what that means: Most Fortnite players don't know that game was inspired by an indie game called PlayerUnknown's Battlegrounds, but they also don't need to.

So this project is an attempt to showcase that there is no immersive sim *genre*, but there are tried and true game design methods and philosophies found in these kinds of games that translate across the entire industry, and maybe even beyond it.

8 Background

For as long as I can remember I have had an interest in *games*, but this eventually grew into an interest in game *design,* and how good game design can be used to tell stories not possible in a passive medium like cinema or literature. For me, the ultimate example of this style of storytelling can be found in games like the ones above, but I believe many of their philosophies are relevant to every game and every *virtual experience*. In a world that is becoming more and more obsessed with gamification, these design principles will be crucial, even outside of the 'video game industry', which will itself become difficult to identify as more and more entertainment experiences will be gamified. For example, it won't be long before most in-person events can be attended in virtual reality; concerts, movie premieres and world cup finals accessible to anyone with a headset. But anyone who wants to provide this kind of experience will need to understand the fundamentals of how to help a 'player' navigate a 3D environment, how to introduce them to complicated control schemes, or how to create systems that react to them in ways other than "You can't go over that way", "Please stay in your assigned seat", or some other variation of "no".

Saying **YES** to the 'player' is a cornerstone of great game design, but it also presents a much bigger challenge than a more linear experience. The good news is, Thief solved some of these problems in 1998, and Deus Ex had created an even better template by 2000.

By understanding what made those games great, and how their spiritual successors iterated on the formula more recently, I will in theory have the tools I need to make **ABSENT** an enjoyable experience, and be more prepared for an imminent, gamified future.

On a side note, I do not expect to commercialise ABSENT within the next 9 months, but I do intend to continue working on this project beyond May 2022 (with the possibility of preparing it for sale on consoles), which has also contributed to my choice of project.

9 State of the Art

As mentioned previously, ABSENT is inspired by many games; like Dishonoured, Metal Gear Solid, and one of 2021's game of the year contenders, Deathloop. I'm of course not competing with these projects, all of which cost millions to make. I'm instead attempting to understand what sets them apart and replicate those elements on a much smaller scale.

ABSENT is an entirely new experience however, with its own setting, characters, and gameplay systems. It isn't the first detective game ever, but this is a style of game that is yet to be perfected. Evidence gathering is something games can do well, whether it's exploring a crime scene or speaking to suspects, and there are plenty of games which have got these aspects right. However, deduction

is much more difficult. As a designer, how do you provide a player with the 'eureka' moment often seen in more passive mediums? I'm excited to showcase some of my solutions to this problem.

Another feature, which I believe is important to making the player feel like a detective, is a partner who accompanies you and *behaves* like your partner, aiding you in conflict, displaying insight during quieter moments, or maybe critiquing your behaviour. This will require sophisticated artificial intelligence systems that respond to all potential player actions, but it could be one of the projects defining features and I believe it will be worth the additional effort.

10 Technical Approach

A game design document (GDD) is a software design document that acts as a blueprint from which a game can be built. It helps define the scope of the game and sets the general direction for the project. In AAA development teams, it helps keep everyone on the same page, and in my case, it provides focus, prevents scope creep and aides the documentation of requirements specifications. However, in a medium that thrives on (or even requires) iteration, the trouble with GDDs is that they can become outdated as soon as you finish writing them. Most game developers follow the **agile** approach to documentation, so GDDs are now being adapted to support the creative, iterative process of game development. With this agile approach to development in mind, I've designed a lightweight, living GDD, allowing leeway for redirection in the future (whether that's the plot, gameplay systems or anything else), and will continually update and iterate on this initial draft as required throughout development.

The first draft of my GDD outlines the following:

- Summary (concept, genre, story, project scope, target platforms, etc)
- Gameplay (objectives, rules, progressions, AI, combat, user interface, etc)
- World Building (plot, characters, locations, level design, etc)
- Development System and Assets (Game engine, 3D Models, sound/visual effects, etc)
- Schedule (Start dates, milestones, etc)

Although this document might evolve over time, the initial draft will help to structure and organise my ideas for the project. Putting these ideas down on paper forces me to evaluate individual components, features, and concepts, hopefully determining what's right for the project's 'bigger picture'. And going forward, as initial plans change, this document will help me keep track of any changes being made and will assist in the documentation of any changes to requirement specifications. By extension, a good GDD will prevent me from making drastic, chaotic changes on a whim. Every change will be weighed, and its effects considered appropriately.

Even though ABSENT is a solo project for now, I will be reporting to my supervisor and this document will help showcase my plans for the project and receive feedback regarding requirements, timelines, and milestones, which will be invaluable.

Primarily, this document will allow me to identify requirements, break them down into tasks, and meet the key milestones as set out in the GDD. **The first draft is included in section 7**.

11 Technical Details

ABSENT will be developed using Epic Game's Unreal Engine v4.27 and make use of the open-source Advanced Locomotion System v4, developed by LongmireLocomotion [1].

Unreal Engine has become the choice for many AAA studios around the world – thanks to its efforts pushing the boundaries of graphical fidelity in 3D rendering. Unity, a similar game engine, is often recommended for college level projects, and Unreal is certainly more complex and difficult to learn, but the entire industry seems to be moving towards Epic's software (and some other industries too), so this seems like the perfect time to expand this skillset. Unreal projects can be written in C++, not exactly the easiest programming language to learn, but it is possible to use Blueprints to offset this issue and my initial experiments with this approach have been successful. Blueprints, a visual code editor, were created to help artists and designers develop content in UE without the need for a software engineer, but it also works very well in my case. Unity has a similar tool, called Prefabs, but there is a key difference between them. Blueprints, when by UE, generate valid C++ classes. As a result, an entire game can be made using Blueprints, since they can be considered valid chunks of code (once generated). Prefabs on the other hand are limited to linking separate scripts together.

I mentioned previously that ABSENT would feature sophisticated artificial intelligence systems and this plays into my choice of engine as well. It's possible to achieve excellent results in using UE's Behaviour Tree System, developing complex scenarios using Blackboard and Behaviour Tree editors. It's not impossible to get similar results using Unity, but it would likely take much more time and effort. Unreal Engine will streamline the development of AI systems which, as mentioned, are crucial and make up a sizeable portion of ABSENT's development effort.

Unreal Engine also provides direct access to the Unreal Marketplace, where I can make use of open source and paid-for community assets, which will facilitate prototyping and development of the final version of ABSENT. This will be covered in more detail in section 6, Special Resources Required. With Unreal Engine 5 now available in early access (boasting exciting new features for the next generation of games and consoles), there's no doubt this community will continue to grow over the next 5-10 years.

12 Special Resources Required

This project would not be possible without a high spec PC. Luckily, my internship with Mastercard provided an opportunity to invest in a machine capable of running the latest Unreal Editor adequately. In the future, if I decide to transition development onto Unreal Engine 5 and intend to use all the next generation features (like Raytracing), I will need to invest in a more capable graphics card, but for now I have the hardware I need.

As mentioned in Section 5, some third-party assets will be required for this project. This includes animations, sound effects, music, 3D models, textures, visual effects, and more. These requirements are further detailed, and sources are referenced, in the game design document included in section 7.

I would like to include voice acting in my project, but there are too many characters for this to be a realistic objective. If this becomes a possibility in the future, I have some of the equipment required to record high-quality audio, but I would not be able to afford to pay anyone. In the past I have worked with trainee voice actors, who recorded and sent their work over remotely, which would improve the quality of the game dramatically. However, the project is already adequately ambitious without this additional requirement.

13 Project Plan

ABSENT – Original Game Design Document

Version 1.0

05/11/2021

7.1 Working Title

ABSENT

7.2 Elevator Pitch

ABSENT is an action-adventure game set in a grounded but fictional world, with the player taking on the role of a detective tracking down a missing person. The game is inspired by others that are usually referred to as immersive simulations or 'immersive sims'. Unlike many of them, however, the protagonist in ABSENT will be controlled from a third person perspective. ABSENT is a story-driven game, taking inspiration from popular noir detective movies and crime dramas. The game encourages thoughtful and peaceful play but allows the player to choose what kind of police officer they would like to be.

7.3 Concept

7.3.1 Overview

- **Genre:** Adventure / Action / RPG
- **Target audience:**
  - **Age:** 15-40
  - **Gender:** Male/Female
- **Monetization:** Episodic / Premium
- **Platforms:**
  - PlayStation 4 / 5
  - Xbox Consoles (later)

7.3.2 Theme and setting

ABSENT is set on the fictional planet of *Ertha*. This world and its timeline of historical events are like our own, but there are distinct differences, allowing creative freedom when designing locations, technology, characters and more. On Ertha, there is a single, giant landmass which has been divided into 4 independent nations for over 250 years. Smaller, less significant islands, each with their cultures and customs, dot the oceans surrounding the landmass called the *Nations of the Fatherland*.

The game takes place in Republic of Cedar (1 of the 4 nations mentioned above) during the late 1980's. Player control Isaiah Hays, a detective in the Republic of Cedar Police Department (RCPD), who is investigating a missing persons case. While some aspects are historically accurate to our version of the 1980's, the game features some science fiction and subtle supernatural elements.

7.4 Gameplay

7.4.1 Game

7.4.1.1 Levels

The environment of each level/mission will be sectioned off from the rest of the fictional world by blocked roads, locked doors, and other obstacles the player cannot traverse. The playable spaces themselves will be rendered in 3D and high detail. A fixed time of day will allow predetermined, cinematic lighting conditions, though weather will change if certain conditions are met. Key landmarks and thoughtful level design will ease navigation of these spaces.

7.4.1.2 Objectives

Each level has one key objective, and they are always clear and understandable. For example, the first level's objective is to 'Find Henry Purcell'. Player will be given some information to help them complete their objective, in this case, 'He called 119 from the *Blu Bar*'. The player will then be set free within the playable space to discover their own path towards this objective. Objectives can be completed in many ways. Non-violent ways are heavily encouraged but not required.

7.4.1.3 Rating System

To encourage 'good' police work, a rating system is under consideration. This would give points to a player for avoiding combat and finding more passive ways to complete their objectives.

7.4.1.4 User Interface

The game will have a simple but intuitive user interface. The main menu will allow the player to:

- Start a new game
- Load a previous game
- Adjust some in-game settings (brightness, camera inversion, etc.)

Player can also pause the game at any time by pressing the pause button. This menu allows players to:

- Review objectives and view additional information
- Resume game
- Load another saved game

- Exit to the main menu

7.4.2 Environment

Each environment will consist of:

- worldbuilding
- level boundaries
- enterable interiors
- dynamic hazards

Environments will look and feel like real places that exist with or without the player. The environments are not static. Player actions can add/remove/alter the elements mentioned.

7.4.2.1 Worldbuilding

Environmental storytelling will reveal details about the world of Ertha: advertisement boards, notes found by the player, and more. A dialogue system, allowing the player to communicate with other characters, will also provide opportunities to tell the player about the world they're inhabiting.

7.4.2.2 Level boundaries

Levels will be 'grayboxed' initially (a rough block-out version of the environment), a level design practice which helps designers to iterate and test the layout as soon as possible. Afterwards, these 'blocks' will be replaced with a combination of textures and 3D models, creating the illusion of a living world beyond the boundaries of the playable space.

7.4.2.3 Interiors

Helping create the illusion that there is activity behind every door, is the inclusion of activity behind *some* doors. For the first level specifically, there are 3 key locations which the player can explore the interiors of. A bar (first and second floor), an 'abandoned' toy store, and an apartment building (corridor and one apartment). These areas will be highly detailed, and each offers unique gameplay scenarios and opportunities.

7.4.2.4 Hazards

Hazards are harmful to the player and encourage thoughtful exploration or prevent direct access to points of interest. Hazards are dynamic, however. For example, an electrified puddle could temporarily prevent access, but the player could find the power source and disable it. A more creative (and cruel) player could also find a way to use hazards like this against their enemies.

7.4.3 Player Character

7.4.3.1 Controls

The player will control Detective Isaiah Hays from a third person perspective. The camera will rest over the shoulder of the character. All inputs will be mapped to a console controller and will not require any other device (such as a mouse or keyboard), and will allow the following:

- walk / jog / sprint
- jump / mantle (if near a climbable surface)
- interact (talk, open, unlock, use, pickup, etc.)
- crouch
- equip weapon / unequip weapon
- focus / aim weapon (if holding a weapon)
- kick / fire weapon (if holding a weapon)
- reload weapon (if equipped weapon ammo count is < maximum)

An inventory management system is still being considered and may be deemed necessary at a later stage.

7.4.3.2 Character Model

Detective Hays is the most important character in the game and as such has a highly detailed (but stylized) character model. This stylized design will ensure target platforms can render the player character and multiple non-player characters on screen at the same time. The jacket can be equipped/unequipped, and this choice will be reflected on the character model. Initial Prototype (with jacket) is pictured on the right.

7.4.3.3 Undercover System

In ABSENT, other characters will respond differently to you depending on whether you approach them as a police officer, or as a civilian. Players can return to their vehicle (usually located at the starting point of every level) and equip or unequip their police jacket. This is a clear and readable way to understand which 'mode' you are currently in. I had originally envisioned a police badge which you could hold, but this became cumbersome and frustrating. The player character controls identically in both modes, but they may have different options depending on how they approach a situation. For example, civilians in one an area of a level might be immediately hostile to police officers but overlook civilians. Returning here without your police identification might give you the opportunity to explore the previously hostile area while avoiding combat. By extension, some previously hostile civilians might now be spoken to, gathering potentially crucial information.

7.4.3.4 Health and Energy

Players will have to manage their *health* and *energy* levels.

Health represents Detective Hays's current condition and the number of injuries sustained. If their health drops to 0 the game will end, and progress will be lost. Health will decrease when hit by other characters in combat, by entering a hazard zone, or falling from too high a height. This encourages

thoughtful exploration of the environment and avoidance of combat. Items found in the world can be used to restore any lost health.

Energy represent how energetic or tired Hays is. If energy drops below 0, the player will no longer be able to jump, climb or sprint like before. However, energy will regenerate slowly while not performing any of those actions. Combat is always discouraged, and restricting movement prevents players from engaging in combat and trying to run away in order to avoid the repercussions of their actions.

7.4.3.5 Weapon System

The player starts most levels with a police-issued weapon. This pistol can be equipped and used at any time if the player is holding enough ammunition. To discourage overuse, initial ammunition counts will be low. Players will be able to find and pickup more ammunition by exploring the level. Some civilians will be carrying ammunition as well.

There are plans to include a weapon pickup system, which would allow the player to pick up weapons dropped by other characters, but this system might encourage combat more than I'd like to and is still under consideration. And another thing to consider, the effort required to develop a system like this might be better applied elsewhere.

7.4.3.6 Stealth

ABSENT will feature a relatively simplistic stealth system. Players can hide from civilians, avoiding combat in hostile areas, or eavesdropping on a conversation that wouldn't occur within earshot otherwise.

Unreal Engine has some out of the box senses which can be applied to characters, and these will be used here. Primarily, characters will be able to see Hays if he is positioned within their vision cone (originating from their head) or moving (while not crouching) within a specific vicinity of the detecting character.

7.4.3.7 Dialogue

Player can interact with civilians by talking to them. After pressing interact near a civilian, dialogue will be presented as text on screen and at some points, the player will have the option of choosing a response from 2-3 options. Sometimes player will only have access to the 'correct' response when they've acquired essential information previously. For example, if they find evidence of a character committing a crime, (if they don't arrest them immediately) they could further press them on the issue in dialogue and potentially receive some new information from them in return.

7.4.4 Non-Player Characters (NPCs)

7.4.4.1 Civilians

Every level in ABSENT is filled with civilians that the player can interact with. These characters are not necessarily your enemy or friend. The players actions will dictate how they respond and behave. Civilians will have a mixture of random and predetermined paths around the environment, always finding the shortest path. The player can talk to civilians, which pauses their movements temporarily and allows them an opportunity to acquire important information. Civilians can also be 'spooked' if they witness violence (to themselves or others). This can prevent the player from interacting with them temporarily, or in some cases, permanently. Their behavior will also change, and their

movements may become more erratic. Depending on the character, they may also become violent towards the player.

Some civilians have generic movements and responses, while others have *unique* paths, behavior trees, character models and ways for the player interact with them. Some **Unique Civilians** will be considered criminals and the player will have the option to arrest these characters, removing them from the world. However, it might be of interest to the player to ignore their crimes temporarily in exchange for more information (or maybe they just don't want to arrest them). Some civilians will be immediately hostile towards the player and engage combat on sight.

7.4.4.2 Civilian Character Models

Civilians will have a range of designs with an art style matching the main character. As mentioned above, this art-style ensures multiple civilians can be rendered at the same time without a massive hit to performance.

7.4.4.3 Partner

With entirely different AI systems and behaviors, the players partner will stay relatively close to their location (within a certain vicinity) and act differently depending on the context. They might give their thoughts on the case, give hints as to how the player can progress (if enough time passes) or maybe critique the players behavior. Their behavior will also change if the player enters combat, defending themselves and the player. The partner cannot be killed (by the player or other characters), but their accuracy is significantly lower than the players, so as not to become *too* useful. Their invincibility is a tradeoff I do not make lightly. Every other character in the game can be killed and I intend to reward the player for thoughtful play (so *not* just killing everyone), but the partner is a key tool for critiquing the player and encouraging them to improve their behavior. The possibility of them dying in the first 30 seconds was too big a risk.

7.4.4.4 Partner Character Model

The player's partner features a highly detailed character model. Like the player character, the police jacket can be equipped and unequipped. Non-player characters will respond to the partner based on this choice, just like the player. The partner will always copy whatever the player is doing in this regard. First prototype is picture on the right (without jacket).

7.5 Project Scope

7.5.1 Software Project Module

During the Software Development module, all efforts will be focused on the development of Episode 1 (the first level).

7.5.2 Budget

Required: €200*	**Desirable: €174***
Stylized Modular Males and Females - €42 [2]	

Fuel Bar - €43 [3]

Fluorescence: Corridor and Apartment - €63 [4]

SciFi Trooper Set for Stylized Male and Female - €37 [5]

Bank Robber 01 - €15 [6]

Weapon Kit €53 [7]

Abandoned Room €21 [8]

OldWitch_BD - €47 [9]

Stylized Character Kit: Casual 02 - €53 [10]

***These details and valuations are restricted to Episode 1 and may change over time.**

I've assigned a budget, as detailed below, to this project. The project is entirely self-funded for now.

7.5.3 Timeline

- Milestone 1 – **Vertical Slice** (Level boundaries, Player Character features, some Non-Player Character features, some hazards, primitive user interface) : 21/12/2021
- Milestone 2 - **Look and Feel** (interiors, models, textures, complete hazards): 15/02/2022
- Milestone 3 – **Episode 1 Alpha** (worldbuilding, objectives, rating system, visual effects, Non-Player Character features, complete user interface) : 10/04/2022
- Milestone 4 – **Episode 1 Alpha Playtest** (Family and Friends / Online Invitations) : 15/04/2022
- Milestone 5 – **Version 1.0 and End of Project** : 08/05/2022

7.6 References

7.6.1 Third-Party Assets

[1]"Advanced Locomotion System V4 in Blueprints - UE Marketplace", *Unreal Engine*, 2021. [Online]. Available: https://www.unrealengine.com/marketplace/en-US/product/advanced-locomotion-system-v1. [Accessed: 07- Nov- 2021].

[2]"Stylized Modular Males and Females - assets only in Characters - UE Marketplace", *Unreal Engine*, 2021. [Online]. Available: https://www.unrealengine.com/marketplace/en-US/product/stylized-modular-males-and-females-assets-only. [Accessed: 07- Nov- 2021].

[3]"Fuel Bar in Environments - UE Marketplace", *Unreal Engine*, 2021. [Online]. Available: https://www.unrealengine.com/marketplace/en-US/product/fuel-bar. [Accessed: 07- Nov- 2021].

[4]"Fluorescence: Corridor and Apartment in Environments - UE Marketplace", *Unreal Engine*, 2021. [Online]. Available: https://www.unrealengine.com/marketplace/en-US/product/fluorescence-corridor-and-apartment. [Accessed: 07- Nov- 2021].

[5]"SciFi Trooper Set for Stylized Male and Female in Characters - UE Marketplace", *Unreal Engine*, 2021. [Online]. Available: https://www.unrealengine.com/marketplace/en-US/product/scifi-trooper-set-with-heads-for-stylized-male-and-female. [Accessed: 07- Nov- 2021].

[6]"Bank Robber 01 in Characters - UE Marketplace", *Unreal Engine*, 2021. [Online]. Available: https://www.unrealengine.com/marketplace/en-US/product/bank-robber. [Accessed: 07- Nov- 2021].

[7]"Weapon Kit in Weapons - UE Marketplace", *Unreal Engine*, 2021. [Online]. Available: https://www.unrealengine.com/marketplace/en-US/product/weapon-kit. [Accessed: 07- Nov- 2021].

[8]"Abandoned Room in Environments - UE Marketplace", *Unreal Engine*, 2021. [Online]. Available: https://www.unrealengine.com/marketplace/en-US/product/abandoned-room. [Accessed: 07- Nov- 2021].

[9]"OldWitch_BD in Characters - UE Marketplace", *Unreal Engine*, 2021. [Online]. Available: https://www.unrealengine.com/marketplace/en-US/product/oldwitch-bd. [Accessed: 07- Nov- 2021].

[10]"Stylized Character Kit: Casual 02 in Characters - UE Marketplace", *Unreal Engine*, 2021. [Online]. Available: https://www.unrealengine.com/marketplace/en-US/product/stylized-character-kit-casual. [Accessed: 07- Nov- 2021].

14 Testing

Games are difficult to test. It's clear from the outset that a player can interact with an open world game in more ways than an iPhone user will interact with a music app, for example. However, my work with Mastercard this year showcased the benefits of automated testing, and I would really like to include some elements of that in my project. But even so, game development is an iterative process, much more so than the applications I worked on during my internship. Significant time is spent developing features that might fit initial requirements but end up not being used if they aren't as fun as expected or are changed so significantly that they no longer even remotely resemble their initial iterations. Using any sort of automated testing at this point would be harmful to my development schedule or almost impossible to implement. Large teams often omit testing completely at this point, to ensure they are playtesting features as quickly as possible. In my case, I think it would make more sense to stick to Test-driven Development (TTD), which will streamline experimentation, and development of new features and level designs, while also ensuring these additions are built with quality and are designed with the players perspective in mind.

As I enter later stages of development, I might reconsider this option and create additional forms of testing. One of these options is Unreal Engine's out of the box testing features, with hundreds of tests that will make sure all base functions and classes are working as anticipated. This includes Unit Tests and more importantly Integration Tests. The latter are used to evaluate larger pieces of software like gameplay mechanics or specific actions the player can take. It can also be used to test AI behaviour, which would be very useful in this project. These tests can be implemented in C++, but are even easier to implements using Unreal's Blueprints, which I intended to use anyway.

Playtests will also be crucial to the success of the project. I intend to playtest an Alpha build of the game starting 15/04/2022 (as mentioned in my timeline) with Family and Friends and potentially advertise it online prior. Some players aware of my previous projects might be interested in participating and providing feedback. These playtests are invaluable, particularly if I can watch someone play live. This can showcase pitfalls in the design of levels, imbalances of difficulty and much more. What might seem obvious to me (playtesting the game for months) could be unclear to a new player. For players who I cannot watch live, I will create a feedback survey that they can

complete online. I would like these tests to involve a mixture of player skill levels and preferences, to ensure the game scales well across a diverse range of players. The story of ABSENT is intended for mature audiences and some story elements might be unsettling for some players. I intend to submit an Ethics Form and discuss this in more details with my supervisor, prior to carrying out any playtests or gathering feedback.