National College of Ireland

Bachelor of Science (Honours) in Computing Information

Software Development

2023/2024

James McGrath

X19441824

X19441824@student.ncirl.ie

Impulse

Technical Report

Contents

Executive Summary

Max 300 words. Summarise the key points of the report. Restate the purpose of the report, highlight the major points of the report, and describe any results, conclusions, or recommendations from the report.

1.0 Introduction

1.1. Background

I decided to make a game for my computing project because, Games are the reason that I became a computer science student. Up until this point in my course I haven't had a large number of opportunities to work with games, game engines, and game systems. I felt like it would be a good challenge for me to develop my own game for my computing project as it's not something that I've done before.

I feel like the game mechanics present a good opportunity for me to work around problems. The design of the game is also good for me to work on as UI, front end development and overall look are not something I usually work with. Up until this point I've mainly focused on back-end functionality. This computing project is a good opportunity for me to work on elements of computer science that I haven't previously been very focused on, I also get the opportunity to make a game which I've wanted to do since I was a child and also, I get to experience game development which is the career that I would like to get into after college.

1.2. Aims

This project aims to create a game using Unreal Engine 5. At its core I would like people to have fun playing my game, but for me it also gives me an opportunity to improve how I develop software going forward as I get to focus on design based elements, I get to work with a new piece of software being the game engine and I get experience for what I want to do in the future.

For my game I would like to create a 2.5D momentum-based parkour game. I would like the game to be able to run on as many pieces of hardware as possible including mobile, PC and consoles. Having the software available to as many people as possible aims to encourage a competitive environment as the game is all about how quickly one can get through the levels. To encourage this competitive environment, I would also like to implement an online leaderboard where rankings can be distributed at some point in the future. In the future possibly reward players with new skins or cosmetics as updates and new levels come out.

1.3. Technology

The main piece of technology I plan to use is Unreal Engine 5. Unreal Engine 5 is basically the entire package when it comes to game development. For graphics I can use Unreal Engine 5s nanite system, which was only introduced when Unreal Engine five was released. I can also find many examples of assets available to use for free or for purchase online.

For lighting Unreal Engine 5 features the lumen lighting system, this is a fully dynamic global illumination solution that reacts immediately to scene changes and dynamic actors like characters or vehicles. Unreal Engine 5 also offers the ability to compile for all platforms, making it easy for me to ship the game to all of the platforms available.

In Unreal Engine 5, I can design the mechanics of the game, the levels in the game, the UI and menus for the game, the animations for the game using flip books for my character and the other characters in the game. The only thing I can't do in Unreal Engine 5 model is the 2D assets I'd like to use. To get these I can find them on one of the many asset libraries offered by Unreal Engine 5.

1.4. Structure

Provide a brief overview of the structure of the document and what is addressed in each section.

2.0 System

2.1. Requirements

2.1.1. Functional Requirements

Functional requirements detail the necessary effects of the software system, dictating what the system must achieve. These requirements are distinct from other types such as interface, performance, or reliability requirements, which describe how the system meets its functional requirements.

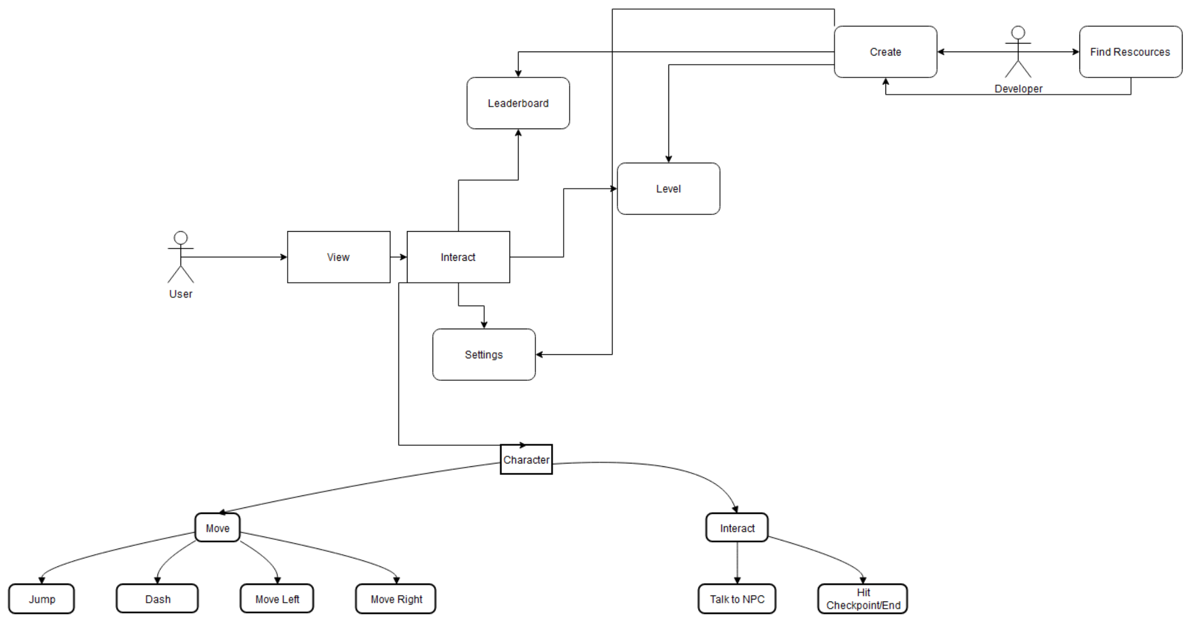
Movement Mechanics

Leaderboard Integration

Level Design Flexibility

Cyberpunk Aesthetics Implementation

2.1.1.1. Use Case Diagram



2.1.1.2. Requirement 1: Movement Mechanics

2.1.1.3. Description & Priority

Critical to gameplay, facilitating player interaction with the game world through movement, jumping, dashing, and wall jumping.

2.1.1.4. Use Case

- **Scope**: To provide dynamic and responsive movement controls.
- **Description**: This use case describes the interaction of the player with the game controls to navigate the game environment.
- **Flow Description**:
  - **Precondition**: Game level loaded and active.
  - **Activation**: Player uses control inputs.
  - **Main Flow**:
    1. System recognizes input for left/right movement, jump, dash, wall jump.
    2. Player character executes the action in the game.
    3. System calculates and displays the result of the action.
  - **Alternate Flow**:
    - A1: Incorrect Input Handling
      1. System identifies incorrect or impossible movement input.
      2. Displays feedback to the player.
      3. Returns to the main flow for new input.
  - **Exceptional Flow**:
    - E1: Control Failure
      1. System fails to recognize input.
      2. Player receives error message.
      3. Issue resolution process initiated.
  - **Termination**: Level completion or player character demise.
  - **Post Condition**: System returns to level selection or restarts level.

2.1.1.5. Requirement 2: Attack function

2.1.1.6. Description & Priority

High. Essential for combat and Movement

2.1.1.7. Use Case

- **Scope**: Implementation of multiple attacking mechanics
- **Description**: This use case describes how the player can interact with an enemy in the game.
- **Flow Description**:
    - **Precondition**: Player is not “Wall sliding”
    - **Activation**: Player Uses an Attack Input.
    - **Main Flow**:
        1. Input is recognised.
        2. Attack corresponding to the input is executed.
        3. Attack animation plays and enemy is damaged.
    - **Alternate Flow**:
        - A1: Projectile Attack misses
            1. .5 Second delay before next attack
            2. “Bullet” actor is destroyed after collision with a surface.
            3. Enemy is not Damaged.
    - **Exceptional Flow**:
        - E1: Attack input is not recognised.
            1. No animation is played.
            2. No line trace is performed.
            3. Enemy is not damaged.
            4. Event is logged.
    - **Termination**: Animation is complete
    - **Post Condition**: Return to idle animation.

2.1.1.8. Requirement 3: Level Design Flexibility

2.1.1.9. Description & Priority

Medium. Allows for multiple routes within each level, enhancing gameplay variety and replay value.

2.1.1.10. Use Case

- **Scope**: To enable diverse and engaging level designs.

- **Description**: This use case involves the design and implementation of levels with multiple routes.
- **Flow Description**:
    - **Precondition**: Level design phase.
    - **Activation**: Level is played by the user.
    - **Main Flow**:
        1. The system presents various routes within a level.
        2. Player character executes the action in the game.
        3. System calculates and displays the result of the action.
    - **Alternate Flow**:
        - A1: Incorrect Input Handling
            1. System identifies incorrect or impossible movement input.
            2. Player chooses a route and navigates through it.
    - **Exceptional Flow**:
        - E1: Route Blockage or Inaccessibility
            1. A route is blocked or inaccessible during gameplay.
            2. Player finds alternative routes or solutions.
            3. The gameplay continues with the new route.
    - **Termination**: Level Completion
    - **Post Condition**: Feedback on the multiple routes.

2.1.1.11. Requirement 4: Cyberpunk Aesthetics

2.1.1.12. Description & Priority

Medium. To create an immersive game environment in line with the cyberpunk theme

2.1.1.13. Use Case

- **Scope**: To provide a visually engaging and thematic game environment.
- **Description**: This use case describes the incorporation of cyberpunk elements like neon lights and dark alleys into the game's design.
- **Flow Description**:
    - **Precondition**: Game development phase.
    - **Activation**: Game is played by the user.

- **Main Flow**:
    1. System renders cyberpunk-themed environments with neon lights and dark alleys.
    2. Player navigates these environments and interacts during gameplay.
    3. Environment aesthetics contribute to the overall gameplay experience.
- **Exceptional Flow**:
    - E1: Game fails to load environment.
        1. System displays error message.
        2. System closes.

Termination: Game is closed

Post Condition: Enhanced player Immersion.

2.1.2. Data Requirements

Player Profile Management: Secure storage and management of player profiles, including game progress, settings, and preferences.

Game Save Functionality: Reliable game saves mechanisms to ensure player progress is accurately recorded and retrievable.

Data Security and Privacy: Strong security measures to protect player data and comply with privacy regulations.

2.1.3. User Requirements

Settings Feature: Inclusion of options like adjustable Resolution and

Multiple Control Layouts: Allowing players to use multiple control layouts. For example, enabling players to play the game using a controller.

2.1.4. Environmental Requirements

Hardware Compatibility: Specifying minimum and recommended hardware requirements for optimal game performance.

Operating System Compatibility: Ensuring the game runs smoothly on various operating systems.

Network Requirements: Detailing network requirements for online functionalities.

Power Consumption Optimization: Optimizing the game to be energy efficient, particularly for mobile or handheld devices.

2.1.5. Usability Requirements

Intuitive User Interface: Designing a user-friendly interface that is easy to navigate and understand.

Responsive Control System: Ensuring game controls are responsive and consistent, providing a smooth gameplay experience.

Clear Instructions and Tutorials: Providing comprehensive tutorials and instructions to guide new players.

2.2. Design & Architecture

The architecture of my game is designed to take full advantage of unreal engines powerful features while still ensuring scalability and maintainability. Unreal Engine 5's framework handles most the architecture but here are some key components:

- gameplay framework
- user interface framework
- physics and collision systems
- asset management

Gameplay framework:

This framework is responsible for managing the game flow, player interactions and the overall game state. This framework defines the game mode, the game mode dictates the default pawn for the game, usually the player character but on occasion like on the main menu it is a camera. It is responsible for receiving inputs from the player and translating those inputs into actions for the character. This framework is also responsible for most pawns and actors in the game so it handles interactions with other enemies like drones or turrets as well as actors like spikes or moving platforms.

User interface framework

The UI framework uses unreal motion graphics to create and manage the user interface. The main components are widget blueprints. These widget blueprints define the layout and behavior of the UI elements. In my game they're used to making the main menu which provides the navigation to options like starting the game and adjusting settings. They also provide the in-game heads up display which displays real time information like player health, their current amount of lives and their level timer.

Physics and collision systems:

The engines physics system is responsible for realistic movement and interaction within the game world. It handles some of the collision detection for the player character as well as allows the player and other objects in the environment to move realistically. Without their system the player would have no gravity, no ability to jump and other actors like enemies wouldn't be confined to their parameters.

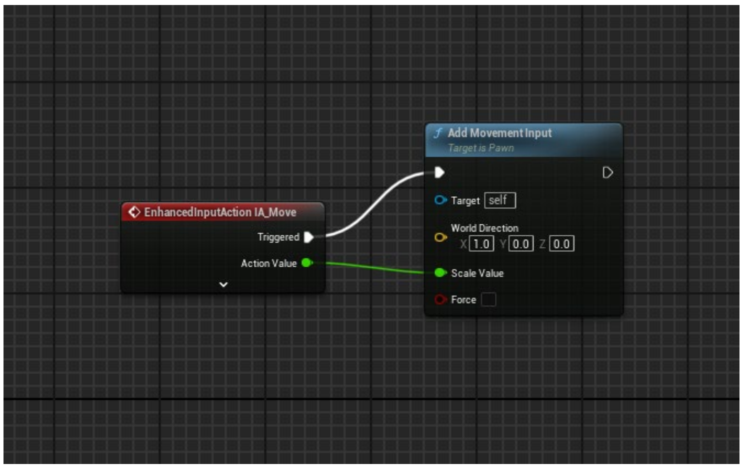
Asset management:

asset management Unreal Engine 5 and also organizing and utilizing the game assets like the textures, the models, sounds and animations. This is mainly done using the content browser in Unreal Engine 5. These game assets can be made yourself and imported where they can be bought online to be used. Unreal Engine 5 natively supports a feature that can create lower resolution or higher resolution versions of textures in order to help manage load times, performance, and storage space.

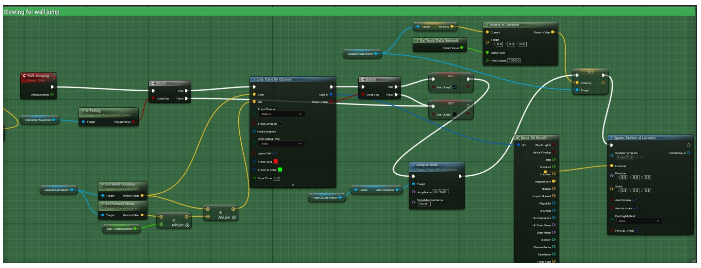
2.3. Implementation

Character movement:

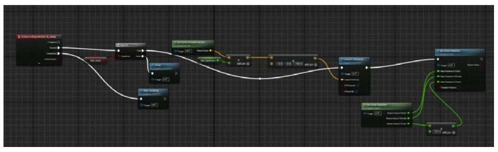
Below I've given some examples of other character movement for the player character. In the first image we can see that an enhanced input action has been mapped to a key or button on a controller. When this button is triggered or pressed the result is movement on the X axis for our player. Now this simple function is actually only one part of the system. There is also rotation to worry about to ensure that our player character is facing the right direction when they're moving in that direction. There are also the animations to worry about as when the player character is moving they also need to have their model move with them. For example, if somebody is standing still they're not going to have a running animation



Below is an example of the function I use to check if a player is on a wall. If they are on a wall the player is slowed down, their animation changes and smoke is placed at their location where they last touched the wall. Going through this function we can see that first a check is performed to see if the character is in a falling state. If this check returns true and they are in a falling state a line trace is sent from the character a set amount of distance away from the character. If this line trace detects an actor with the trace channel wall jump the wall jump Boolean will be set to true. This Boolean is referenced in other parts of the player character actor in order to prevent the character from attacking while on a wall. After this Boolean is set to true the animation we'll change the on wall animation the players velocity will be slowed down and smoke will be spawned on their location as they fall



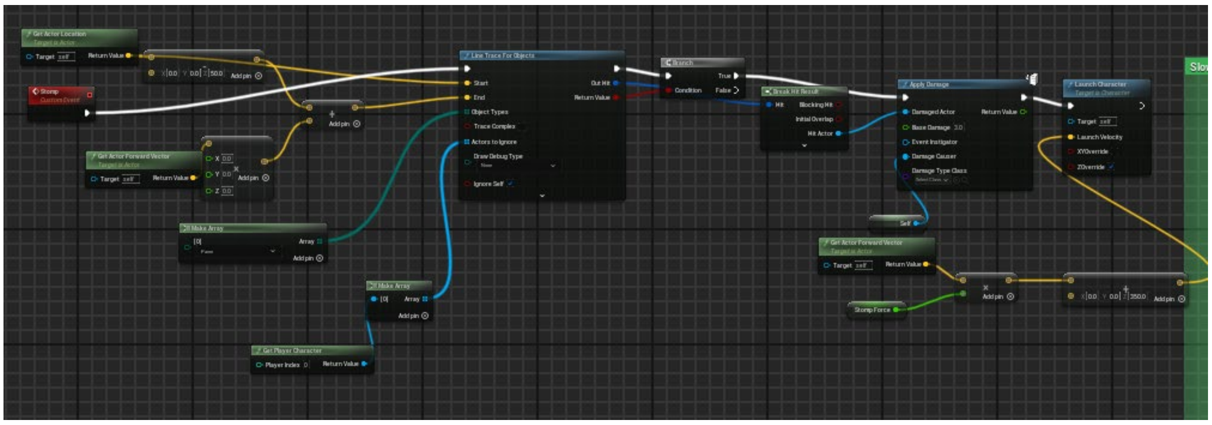
Next, we have the function for the player character to jump. We're in the enhanced input action is started the function checks to see if the wall jump state is active. If it isn't then the player character executes the jump function which propels the player asset distance in the air. However, if that Boolean is true the player character gets launched significantly higher upwards using the variable wall jump force and the actors rotation is changed so they are facing the right direction



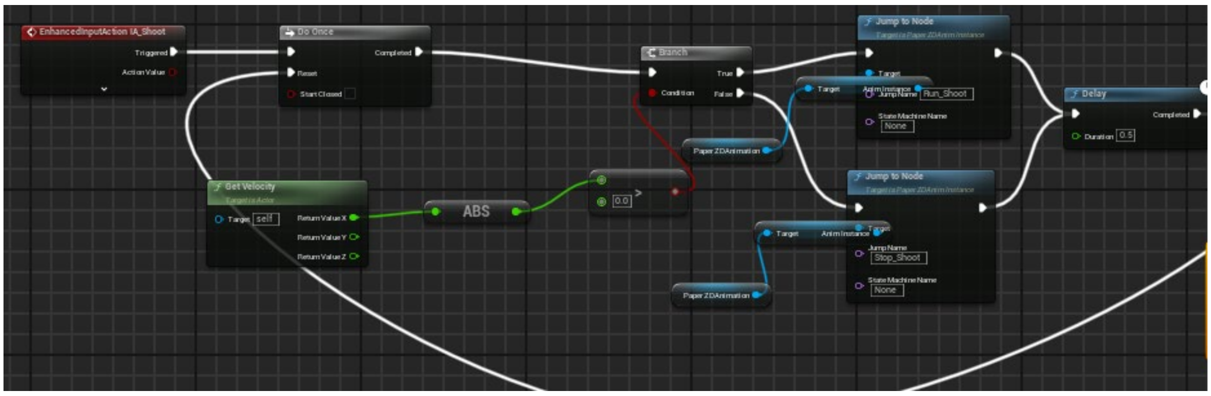
These are some key functions for the player character's movement though of course there are plenty that I haven't covered such as the player character's ability to dash.

Attack mechanics:

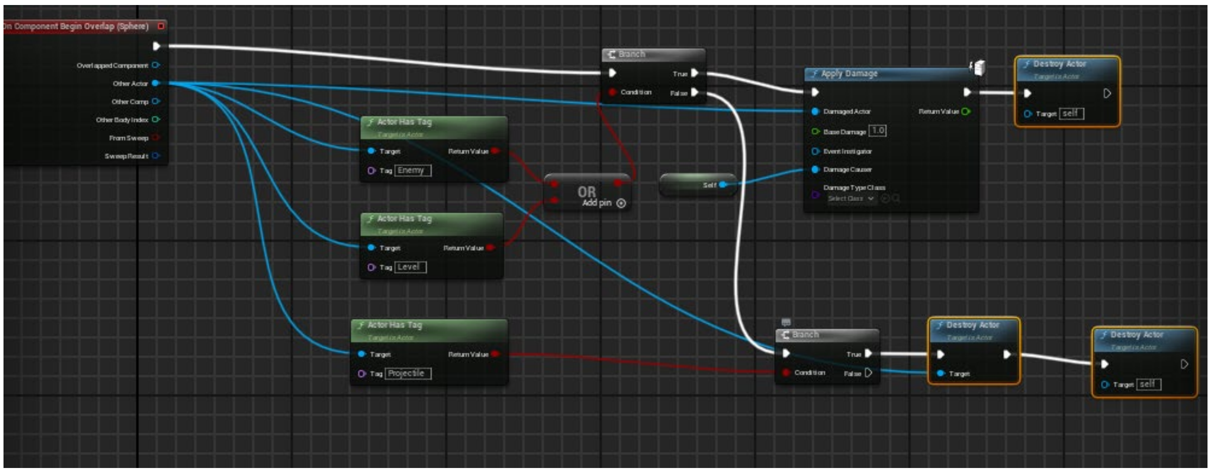
Here we have the function for the stomp attack, so working through this function we can see that a line trace is shot from the player character to 50 units below them. This line trace is looking for pawn objects and ignores the player character. If this line trace fines an object with the pawn class, then it will apply damage to that pawn using the applied damage function and then launch character set by the variable stomp force.



Now the next attack that I'll be going over is the shoot attack. So, when the enhanced import action is triggered, the two once function is called. 1st I perform a check to see if the character is moving or not. If the character is moving, then we jump to the run shoot node and if they aren't we go to the stop shoot node. Then we trigger a delay before you can shoot again. In the animations that the function calls there are specific frames in which we call the shot blueprint that can be found below this image.

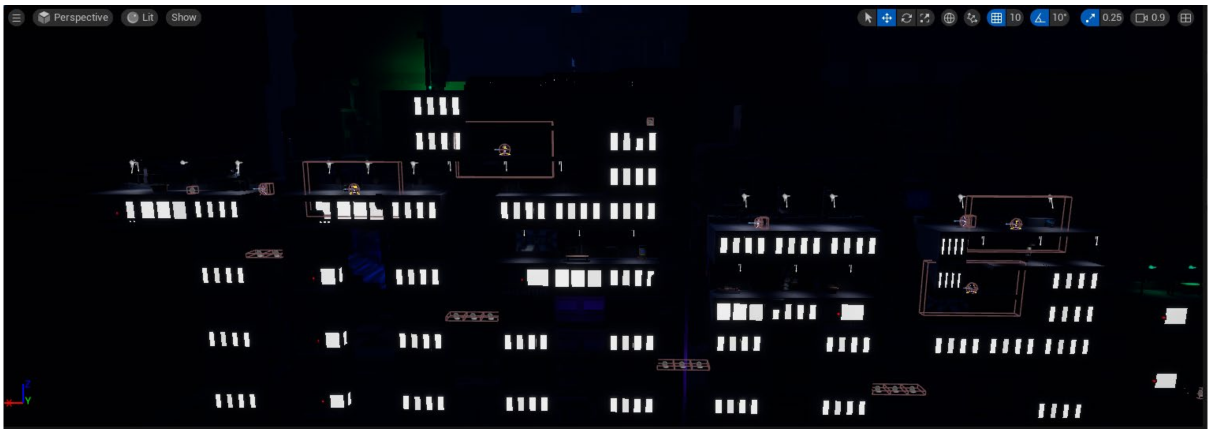


When this blueprint overlaps with another actor that has the tag enemy or a level, it will apply damage to it and then destroy the actor. However, if it comes into contact with something that has the tag projectile it'll destroy the actor that it comes in contact with and then it will destroy itself. This makes it so enemies can deny your ranged attack and you can deny an enemy.



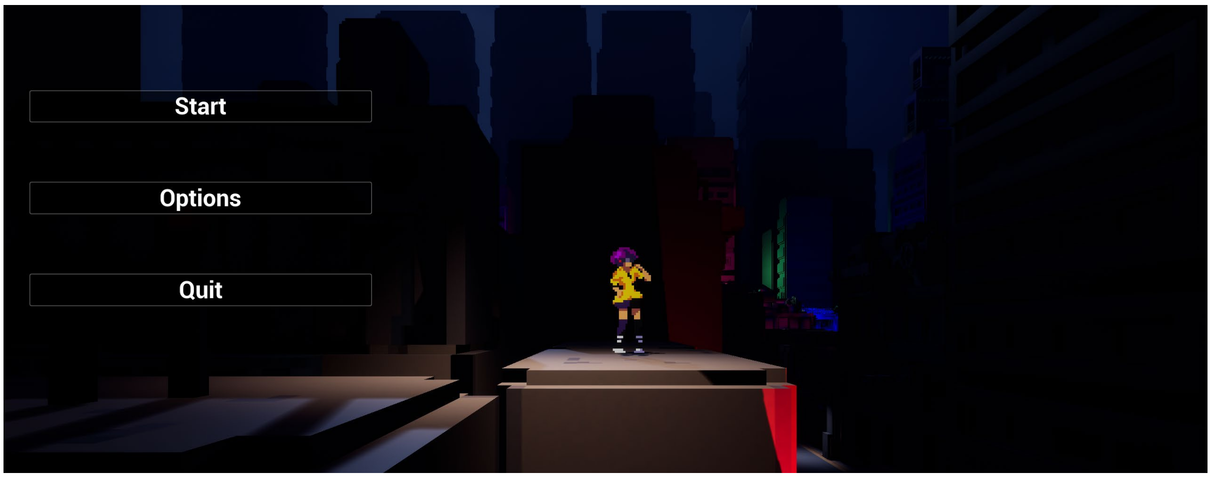
level design:

This part of my implementation was less about blueprints and models and more about how the level felt and how the level looked. In my game I wanted to have a cyberpunk theme, so I had to search and make assets to support this theme. The backgrounds of the game are filled with large skyscrapers neon lights and small alleyways often found with the theme. Another thing I thought was key to implement for the game where alternative routes throughout the level. And every level there is an alternative route the player can take to get to the end and some result in a much quicker clear time than others. May also add some replayability to the game the fact that users can go back to a level multiple times and try and perfect a route to get the fastest possible time.

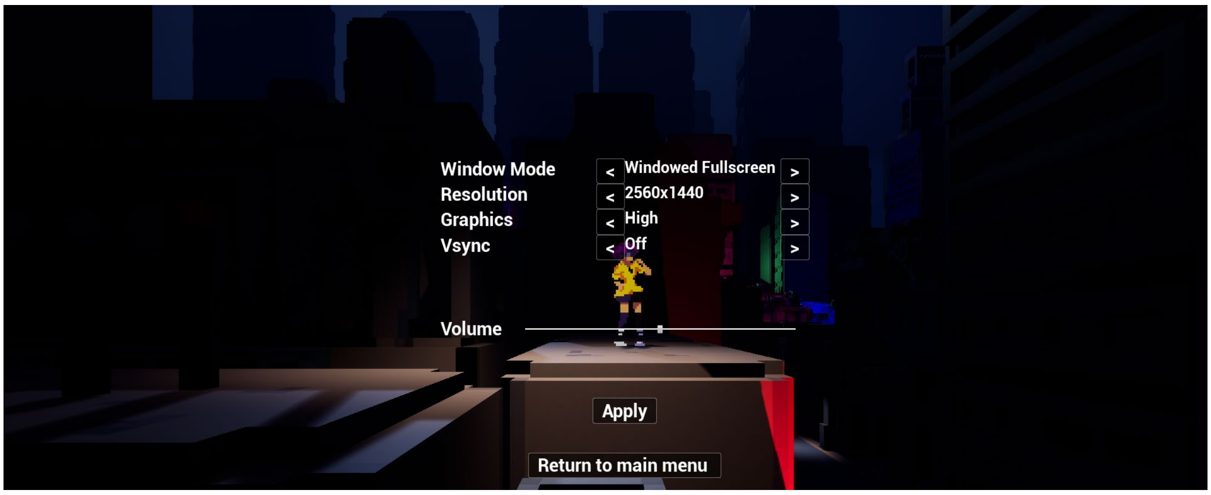


2.4. Graphical User Interface (GUI)

Main Menu: this is the main menu, in this menu a user can select the start button which will bring them to an option of the levels in the game or they can go to the option menu which is displayed below to change their settings if their system can't run the game or if they'd like to change the resolution of the game etcetera. They can also select the quit option which will close the application and bring them back to their desktop.



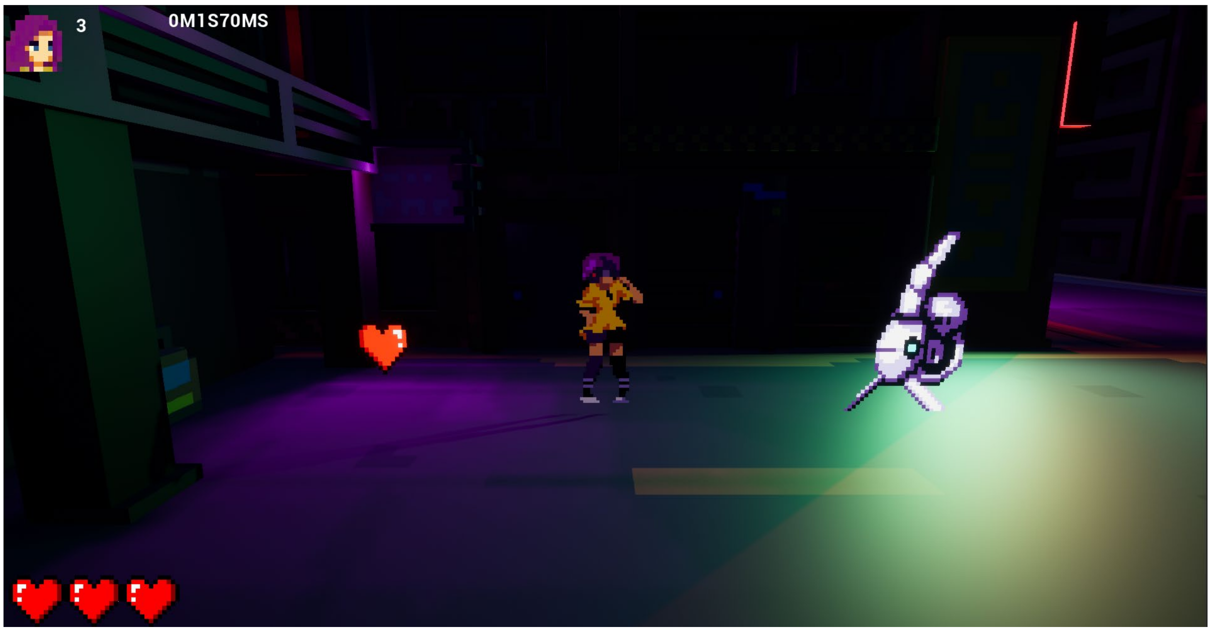
Options Menu: here we have the options menu here the user can decide whether they want to play the game in windowed mode windowed full screen or full screen. They can also decide the resolution of the game. They can choose their graphic settings car this will usually depend on what the user can run on their device. For example, on lower end hardware like the switch or a mobile phone you would probably be playing on the lowest graphics option. We also have a vsync option this helps prevent screen tearing when playing the game and finally we have a volume slider in case the background music or sound effects are too loud. When the user presses the apply button all of their, option settings are saved and loaded the next time they opened the game.



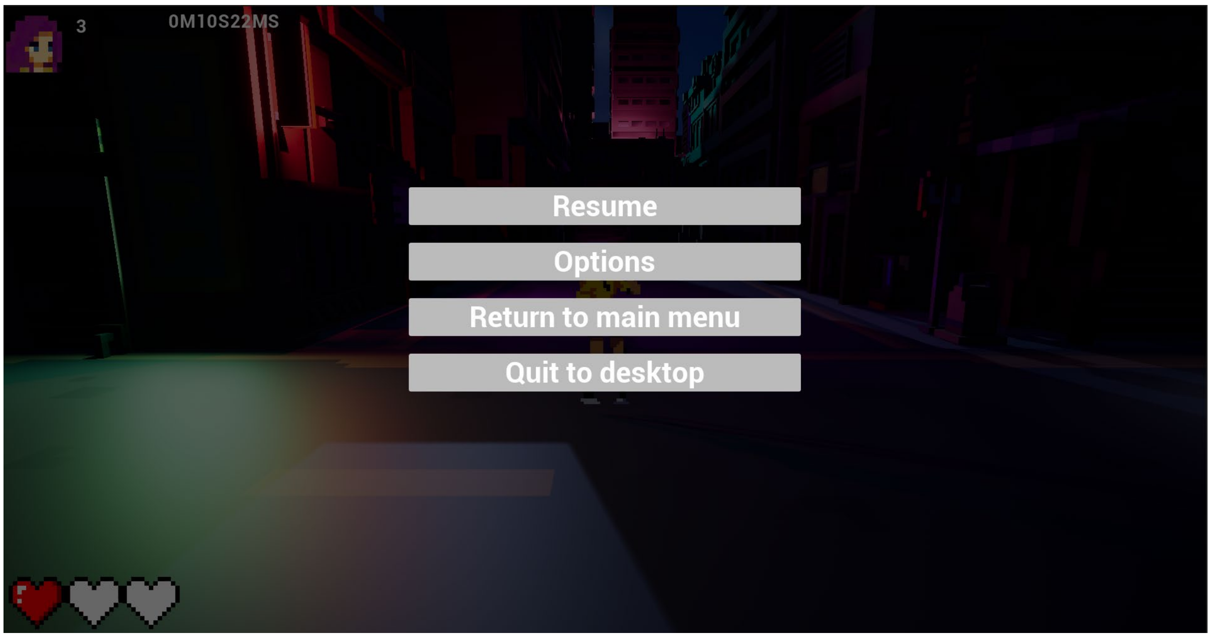
Window Mode < Windowed Fullscreen >
Resolution < 2560x1440 >
Graphics < High >
Vsync < Off >

Volume

User HUD: this is the heads-up display for the user. The main three things here are the users lives in the top left, the users' hearts in the bottom left and the timer there's also in the top left. These values all change dynamically on every frame if needed,



pause menu: we also have the pause menu which can be activated at any time when playing the game. This menu allows the user to pause the game if they need to take a break color, they can also change their settings in the options menu if their device is struggling to run the game or it is too loud, they can also return to the main menu of the game and quit to the desktop from the level.



2.5. Testing

The nature of game development comes with challenges particularly when it comes to testing. Games often involve tightly interwoven systems like physics and graphics and input handling which can make it difficult to isolate individual things for testing. Many functionalities rely on real time interactions and user input which are difficult to simulate accurately in a test environment and the effectiveness of many features like graphics and user experience especially when it comes to a game that's heavily themed like mine make them unsuitable for tests that focus on code logic. As a consequence of this the tests for my project were mainly focused on performance and user feedback of my game.

Performance testing:

Performance testing on my game was conducted across a variety of devices with different hardware specifications.

- Frame Rate Consistency: Ensuring a stable frame rate across devices to provide a smooth gameplay experience.
- Load Times: Minimizing the time taken to load levels and assets to enhance user experience.
- Resource Usage: Monitoring CPU, GPU, and memory usage to identify and optimize resource-intensive operations.

I did these tests on different devices at the end of my project’s lifetime to ensure that it would be able to run on a wide variety of devices. The only device that my game seemed to struggle on with the lowest settings was my old laptop from 2013. Considering that this laptop has less ram on a lower frequency than most modern phones I feel like the fact that I can get 30FPS on this ensures that in the future I would be able to launch on mobile devices. Using the midpoint laptop as a reference I would probably be able to put my game on most any console of the modern era without any issue. The one thing I wasn't able to test I was running the application on a hard disk drive as barring the laptop all devices I tested on had solid-state drives full stop.

User feedback:

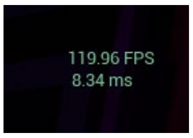
The other part of my testing stage that I had mentioned previously we're consistent alpha and beta tests throughout the production of my game. At this point I sent the game and it's alpha version to a few of my friends and also had a few family members play the game. I asked some questions in relation to the game such as how they felt about the environment, what they feel could be improved about the game, what different features they'd like to see implemented. For example, the stomp feature that we went over earlier was requested by a friend as he personally felt like it would have improved the flow of much of the levels and adds another way for players to kill enemies and it enhances the movement options.

Overall during these tests we're happy with the game and barring some complaints like the lack of sound and at the time the two levels that were available all feedback was positive.

Quality assurance in the game development field is heavily focused on player testing and user feedback. Most modern games come out with an early access version or a beta version before they release in order to get a gauge on how their players are going to interact with the game and what they'd like to see. I've got to get a taste of this and get a view on my game from another person's perspective.

2.6. Evaluation

My PC: Avg FPS:100Fps+ Avg Load Time: <1s



119.96 FPS
8.34 ms

CPU: Intel i7-14900K

GPU: Nvidia 4080 SUPER

RAM: 64GB DDR5 6000Mhz

High End Laptop: Avg FPS:60Fps Avg Load Time: <1s

CPU: AMD Ryzen 7 5800H

GPU: NVIDIA GeForce RTX 3070 Laptop GPU

RAM: 32 GB DDR4 3200Mhz

Mid Laptop: Avg FPS: 60Fps Avg Load Time: 1s+

CPU: Intel Core i7-9750H

GPU: GeForce GTX 1650

RAM:16GB DDR4 2777Mhz

Mid-PC: Avg FPS: 60Fps Avg Load Time:<1s

CPU: Ryzen 7 5800x

GPU: RX 5600xt

RAM: 32gb ddr4 ram

Low-end laptop: Avg FPS: -30Fps Avg Load Time:10s+

CPU: Core i3 6100U

GPU: Integrated CPU

RAM: 4 GB RAM

All tests were performed at the low setting from the games option menu, tests were also performed on the high setting in the games options menu and all devices bar the low-end laptop were able to achieve 60FPS average on this setting.

3.0 Conclusions

Overall I think that this project was an overwhelming success for me. I was able to develop a 2.5 D game in Unreal Engine That incorporated different features like combat enhanced movement and a boss. I went from knowing absolutely nothing about game engines to being confident in working with them and excited to develop my next project. Throughout this project I learned a great deal about the engine itself, the modeling that goes into the assets for the project and the sound design. As well as this I got to interact with different people to get their feedback on the game which I felt helped to improve it. I was able to implement two different control schemes for both mouse and keyboard and controller and I was able to build the game into an executable that I could put on any market right now to be sold.

From my experience on this game I planned to make other games myself as an indie developer and release them to the public. So hopefully the next game I make, is just a bit better than this one, I learn more from that experience and I can keep developing and learning until I have a product that a great many people can get enjoyment from.

There were however some great limitations in deciding to do a game for my project the main drawback from doing this type of project is testing. Game development does not lend itself well to 1classical unit tests like we were taught to do in class. I now greatly understand the reason, bet so many game development companies invest heavily need to play tests and beta versions of their game. Every single time I made a change to the game I would have to complete a full level of the game to ensure that all blueprints and other code still worked. This took a considerable amount of time compared to a regular project where I could have made automated tests for functions that would run with no issues.

Another drawback to this type of project is the file size, because this game is over git hub's limit for file size I have to use OneDrive as a remote repository for all of my code. This is the only solution I could find that would allow me to easily sync my files between multiple devices that I work from as well as download the game easily to different devices for play testing.

Sadly for me personally, I also was drawn back from achieving the full potential of this project as I spent time in hospital after being ill.

4.0 Further Development or Research

In the future I hope to expand the game with additional levels as well as include more enemies with enhanced AI for those enemies. I'd like this to just be the start of this project with plenty to improve upon. As previously mentioned, to enhance the competitive nature of the game i would love to at some point create online leaderboards for each level, so that users could see where they score against others. I'd also like to implement a local multiplayer feature perhaps something like split screen or a mode where both users race on a mirrored map to a finish line in the middle.

Further research would also include making more of my own assets and models to be used in the game. It would also be great to figure out Unreal Engine five more as I feel like I've barely scratched the surface with this project. I've yet to work in the third dimension and I feel like that's definitely something that I would like to do, be it on a future game or a later update to this game.

5.0 References

Epic Games, 2024. *Unreal Engine Forums*. [online] Available at: https://forums.unrealengine.com/

YouTube, 2024. *Unreal Engine Tutorials*. [online] Available at: https://www.youtube.com/results?search_query=unreal+engine+tutorials

Epic Games, 2024. *Unreal University*. [online] Available at: https://www.unreal-university.com/home

Uisco, 2024. *UISCO Blog*. [online] Available at: https://www.uisco.blog/

Itch.io, 2024. *itch.io*. [online] Available at: https://itch.io/

Epic Games, 2024. *Epic Games Store*. [online] Available at: https://store.epicgames.com/en-US/

Epic Games, 2024. *Unreal Engine Marketplace*. [online] Available at: https://www.unrealengine.com/marketplace/en-US/store

6.0 Appendices

6.1. Project Proposal

Project
Proposal.docx

6.2. Reflective Journals

Here are Project Proposals for October to December:

(November in with December)

9806_James_Mc_Gra 9806_James_Mc_Gra
th_James_McGrath_Ith_Nov_Dec_monthl

Jan-April:

Due to illness I wasn't able to complete many things during this time. Both Frances Sheridan (Year supervisor) and Anu Sahni (Project Supervisor) are aware of the full situation.