National College of Ireland

BSHCSD4

Software Development

Academic Year 2023/2024

Joshua Arejola

x20357266

x20357266@student.ncirl.ie

MusicaLead

Technical Report

Contents

Executive Summary

The purpose delves into an evaluation of the whole project that is undertaken for the final year project. It is to assess the overall structure of the project, the aims, the technology that will be used, the requirement specifications and conclusions at the end. Practical recommendations are also recommended at the end to further enhance the project should it be continued in the future. The major points of the report are the system requirements and my own conclusions along with further recommendations I would like to do in the future of the project.

1.0 Introduction

1.1. Background

As a kid, I have always loved listening to music and loved playing video games. Growing up I have developed the skills to learn how to play multiple musical instruments and still have enjoyed playing video games. As I grew older, the games that I have enjoyed became more and more in-depth, and I have later realized that I started to enjoy the "How" the games played out rather than the games themselves. Thus inducing this overall goal and challenge to combine my two hobbies into one and creating a game centred around music and 'rhythm'. I have no prior experience in music or game creation, so this project is to challenge me in ways I have never experienced. Wanting to be a game developer in the future, this project opens up all the aspects of game creation that one must experience to become a game developer. I will be handling the music, the modelling and the game creation all by myself.

1.2. Aims

The project's aim is to develop and create a rhythm-based game in the Unreal Engine framework. The purpose of this is to experience the different subdivisions of game development, ranging from the music industry to the modelling department and lastly, the game development part. This game is to contain self-created music and have 2.5D side-scrolling functionality that implements a rhythm-based game structure. It aims to have good functionality and incredible timing-based input gameplay.

1.3. Technology

The technologies I will use to create the game is Unreal Engine 5.3 from Epic Games Studios as well as FL Studio as the Digital Audio Workstation (DAW) and Blender 4.0 as the modelling, rigging and animating software. From Unreal Engine, I will use the 'Blueprint' schemas to be able to create the game and to create and implement all of the functionality. FL Studio will be the application that creates all of the music and audio that the game needs to function to be rhythmic and Blender studio as a means to develop the models that will be used for the character in the game.

1.4. Structure

The document contains: The introduction which contains the details of the whole document and what is addressed in each section. The second is the Systems requirements, which goes over all of the requirements of the project and the different

diagrams like use case diagrams and other use cases that are used. Next is the conclusion, followed by the further recommendations and finally references.

2.0 System

2.1. Requirements

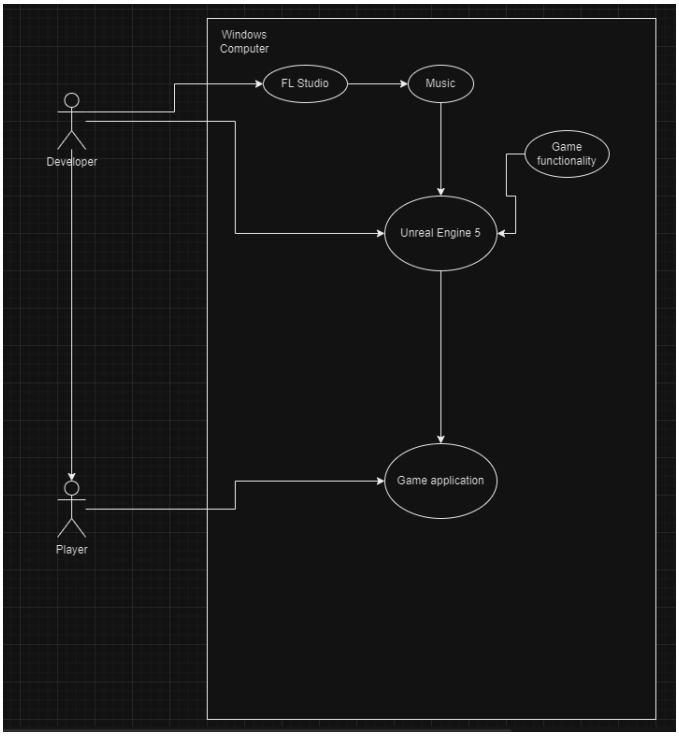
2.1.1. Functional Requirements

There are multiple functional and non-functional requirements for the project.

Firstly, the game mechanics. The player must be able to jump when pressing space bar on their keyboard. The character must then jump when prompted. The next is the UI which should be easily read and navigated through. The audio of the game must be of high quality to ensure that the player may be able to hear the 'beat' or 'rhythm' of the game.

For the non-functional requirements, the game must be very responsive, achieve a playable frame rate and must have scalability to accommodate users on lower end systems.

2.1.1.1. Use Case Diagram



2.1.1.2. Use Case

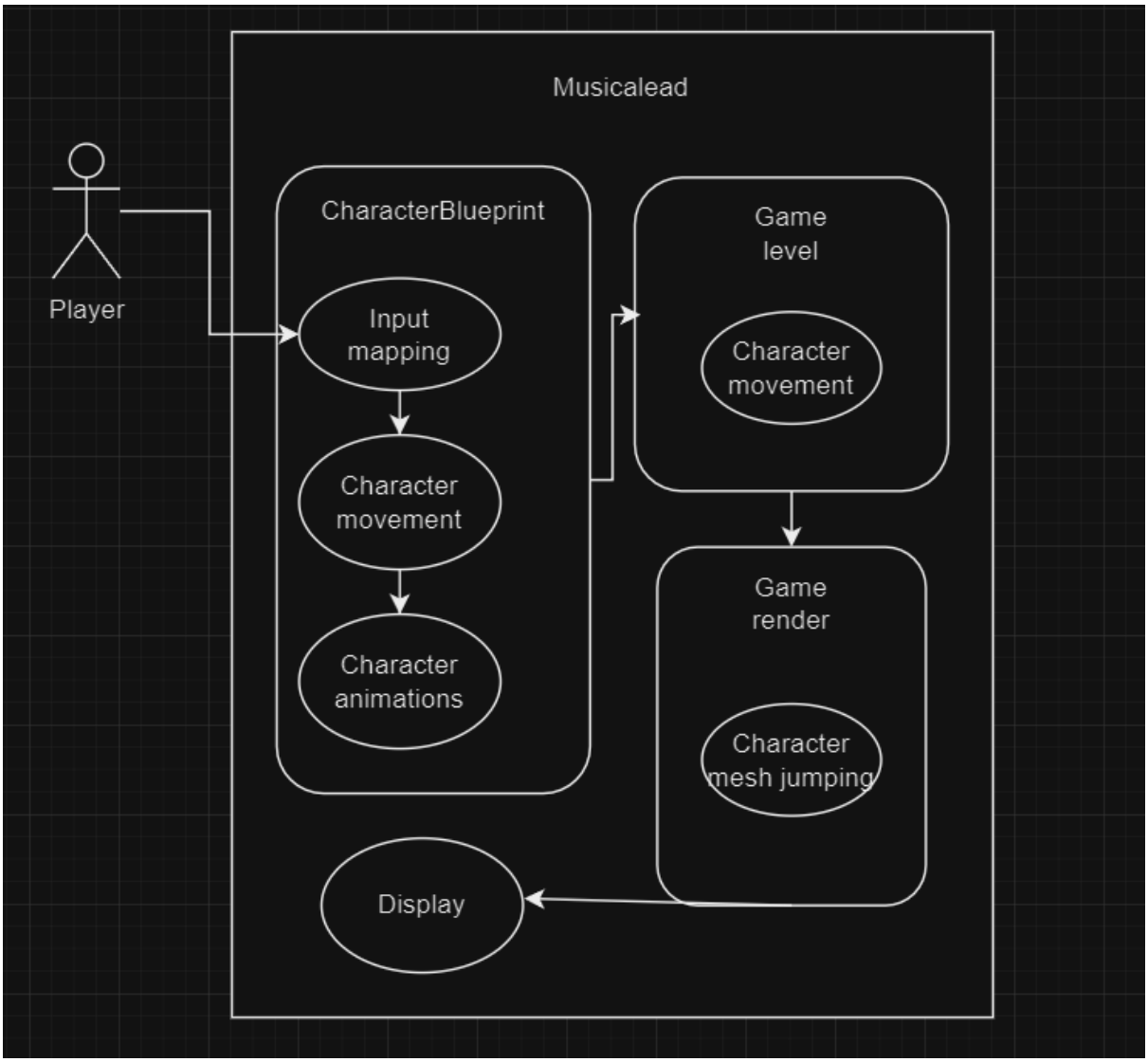
Scope

The scope of this use case is to let the player character in game to jump

Description

This use case describes the how the player input is taken to be able move the character in game by pressing space bar

Use Case Diagram



Flow Description

Precondition

The system is in initialisation mode when the user turns the game on

Activation

This use case starts when the player starts the game and presses space bar

Main flow

1. The system identifies the actor pressing space bar
2. The character blueprint input mapping acknowledges the space bar input
3. The system takes the information of player jumping and loads animations and loads the game level character movement
4. The game then renders the character jumping and displays it

Alternate flow

1. The system identifies the player pressing space bar
2. The player character is already jumping but still acknowledges space bar being pressed
3. The system keeps the current animation play
4. The game renders the character still jumping

Exceptional flow

5. The system identifies the player pressing space bar
6. The system does not acknowledge space bar being pressed
7. The character does not jump

Termination

The use case is terminated/completed when the actor has jumped in game

Post condition

The system will now wait for the next requirement specification

2.1.2. Data Requirements

The game needs multiple data requirements in order to function: Game state data, to track the current state of the game. Configuration data, to store the settings of the game and Asset data, to store the multimedia assets of the game.

2.1.3. User Requirements

User requirements include Game mechanics, the audio and video design of the game, Menu system and UI.

2.1.4. Environmental Requirements

The game's environmental requirements include: Platform compatibility, hardware requirements, peripheral devices, graphics and audio settings.

2.1.5. Usability Requirements

The game's usability requirements include: User feedback, navigation and interaction, User friendly interface and performance and responsiveness.

2.2. Design & Architecture

The game's design take was more of a simplistic and more calming cave exploration design using custom assets made in blender. The sound design is taken from a more melodic and calming music even for a rhythm game.

The game itself was more event-driven as certain animations only played when certain conditions are met like player input, velocity change and game events.

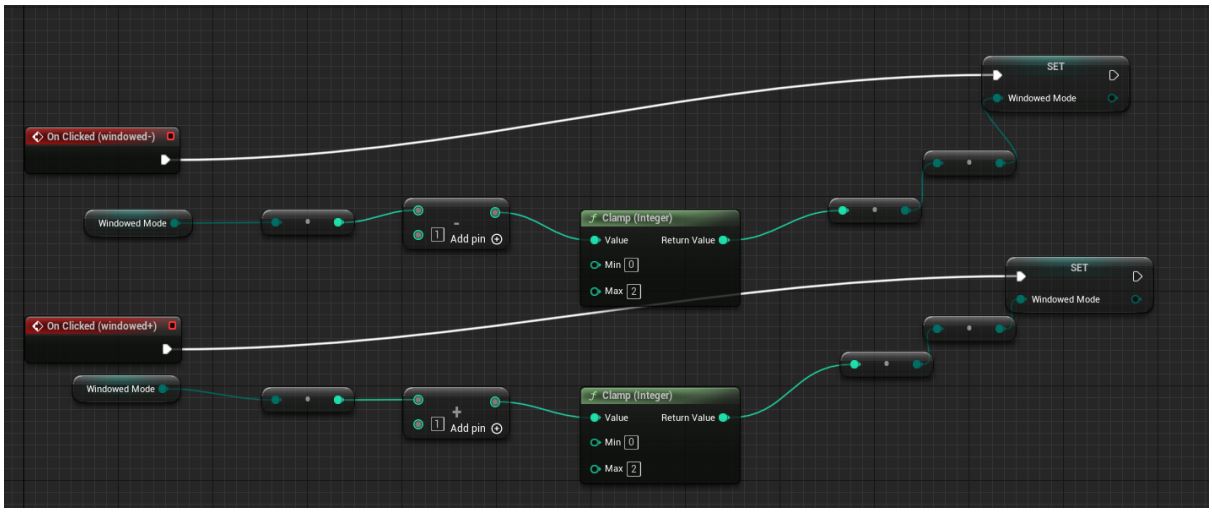
Blender was used as the main 3d modelling software, as well as rigging the meshes and animating them. Rigging of the armatures and animating said armatures were all created using blender.

FL studio was the primary software used in music composition. All of the music from the game was created and all of the audio design used was composed within the software. Most of the music notes were created using the software's "Piano roll" feature.

Lastly, Unreal Engine 5 was the last software used in the creation of the game. This software handled all of the functions the game needed to run. All of the assets were exported from blender by .fbx files and the music from FL studio exported through .wav files. These are the primary files Unreal engine prefers when importing from different sources.

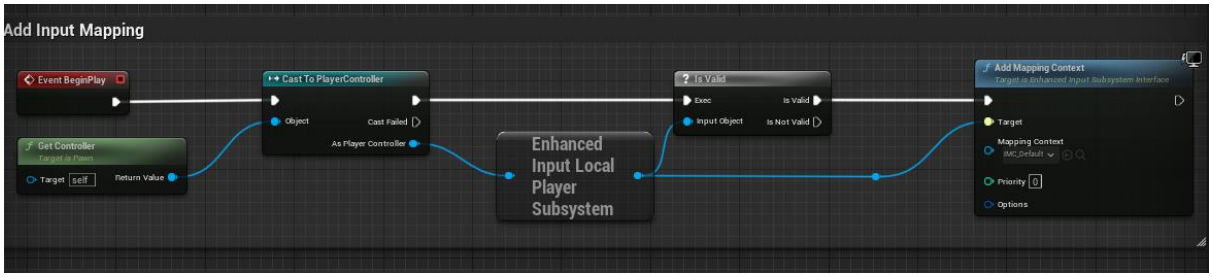
2.3. Implementation

All of the main classes and functions used in the game were handled by Unreal Engine's blueprint scripting. Most of these functions primarily originated as "OnClick" functions on buttons or Setters/Getters for getting the current value of information such as the value of the "ResolutionIndex" or "GraphicsIndex".

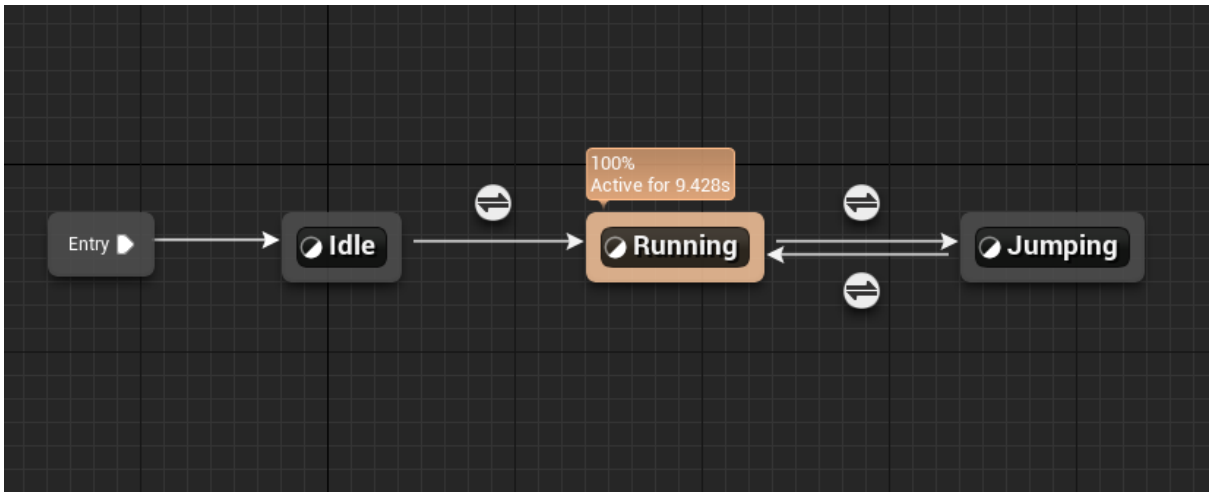


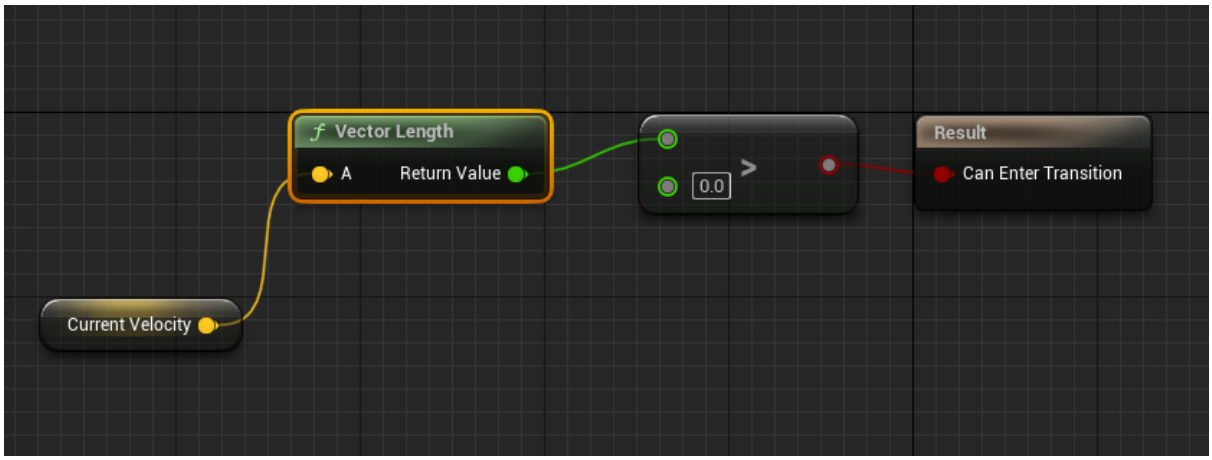
Getters and setters handling the state of the "Windowed" mode

The character blueprint consisted of an "Input Mapping" Schema which handled all of the games inputs from the keyboard, but primarily only the space bar as that was the only key mainly used within the game.



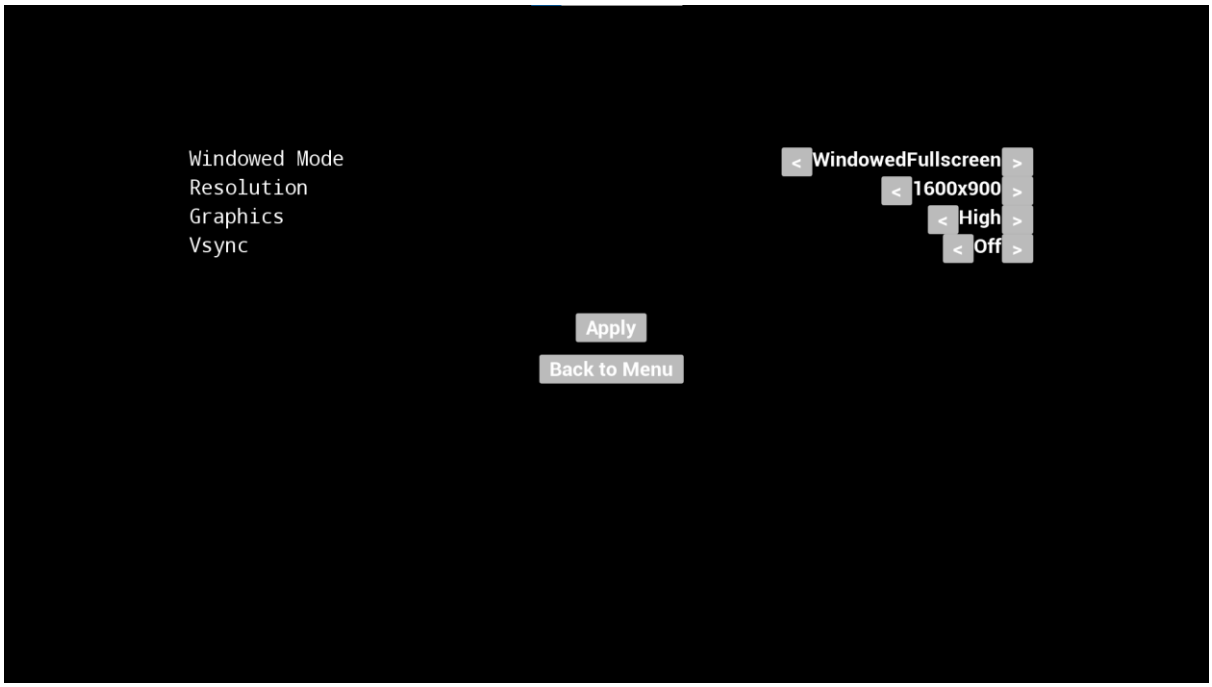
The animations of the player character were handled by a “State machine”. The state machine reads the current information off of the player character such as velocity and such determines which animation it would play. For example, if the player’s “Z” velocity is more than 0, it will play the jumping animation as it would read that the player is moving upwards. It would have a “Getter” for the player velocity to read this which is being called as a function in the player blueprint. If it the player is only moving on the “X” axis then it is only in a running state and stays that way as long as the velocity of the character remains above 0.





2.4. Graphical User Interface (GUI)

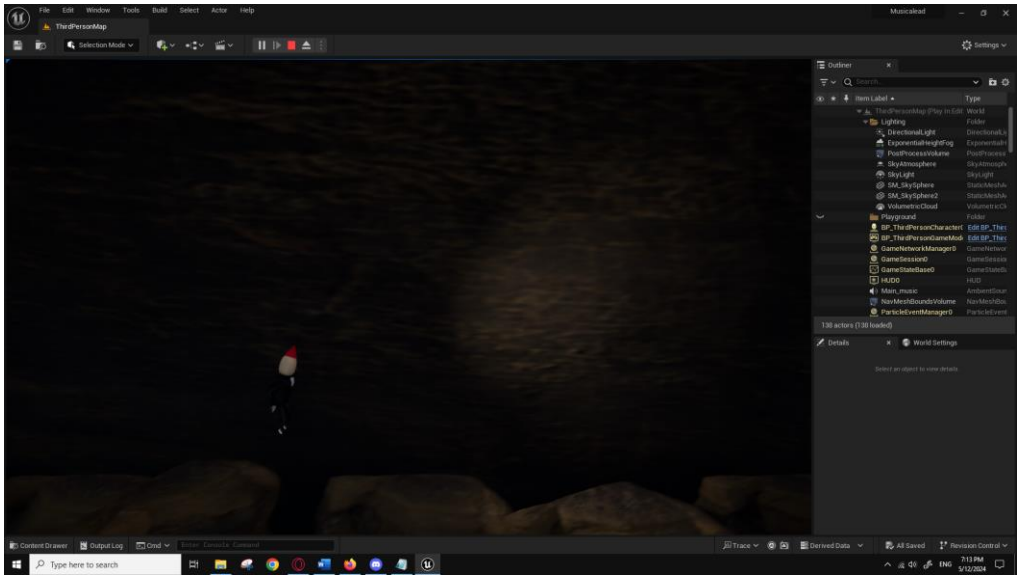
This is the main menu that you use to be able to play the game, go through the settings and exit the game. There are 3 buttons: ‘Play game’ ‘Settings’ and ‘Exit game’. The play game button starts the game, the settings button goes into the options menu and the exit game button exits the application.



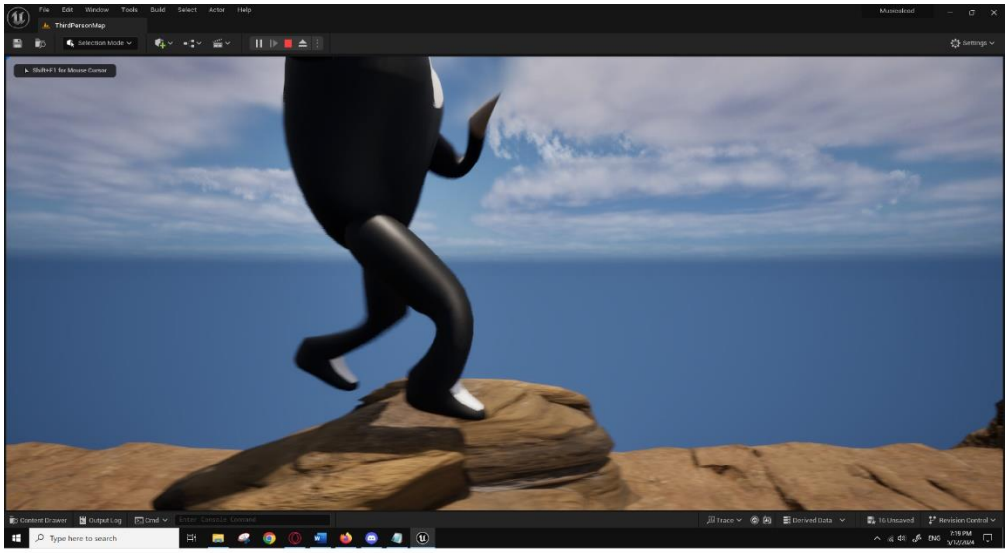
This is the settings menu that contains the different settings to be able to be customized per personal needs. All of the settings are available and will be applied once the apply button is pressed.

2.5. Testing

Since this is a game, there are limited ways I am able to conduct testing to it. The main testing used was Black-box testing as it is the only way to “Playtest” the current gameplay loop over and over. Automated testing is not possible with the current application. Unreal engine makes this possible as there is a level preview where you are able to play over the scene as if it is fully running.



Regression testing was also using during the development of the game as I was constantly adding, developing and playtesting the game. An example of this is shown here where after altering a character animation, a bug appeared which caused the character to exponentially increase in size.



3.0 Conclusions

The project has a variety of advantages and disadvantages.

The advantages and strengths of the project is that it is made on the 3d plane. There is more creativity and more freedom to move, create, model and animate in a 3d environment rather than a 2d plane. But this also comes at the cost of time, as modelling takes more effort as well as performance as the hardware may be a limiting factor to how well the project runs.

4.0 Further Development or Research

With additional time and resources, this project would have multiple levels, better models, better animations and much more music. I feel as though I could have improved every aspect of the project if given more time. More time could have been spent creating better models with more robust rigs and animations, as well as better and more fruitful music

5.0 References

6.0 Appendices

6.1. Project Proposal

Project proposal.docx

6.2. Reflective Journals

Reflection October x20357266.docx

December Reflection x2035726

January Reflection x20357266.docx

February Reflection x20357266.docx