**Analysis**

Background

The Nintendo Entertainment System (NES) is an 8-bit console for home entertainment which was released in America in 1985 along with the first Super Mario Brothers and Tetris. These retro games provided a great challenge and are easy to understand and play however both the games and the consoles are no longer manufactured in their original state meaning that it is very difficult to play these games legally.

As a friend group it is difficult to find games that we all play. This is due to both hardware requirements and the varying skills of my friend group. Both the original Super Mario Bros and the original Tetris are games that we would like to be able to compete against each other and create strategies to complete the levels. However, it is difficult to obtain NES and other retro console games legally which can make it expensive. This often means that there is a barrier of entry due to the costs associated with the hardware and software requirements. As a friend group there is also an issue with the hardware that we use to play games. Therefore, we would like a game that we could all play on a device that everyone has that can run the game.

How I have Investigated the Problem

I have conducted multiple interviews with Noah, Tyler, Matthew, Elliott and Toby which detailed the exact specifications for my project. Toby only plays Rocket League, Elliott plays League of Legends, Noah plays Terraria, Tyler plays Terraria and Streetfighter and Super Meat Boy and Matthew plays Rocket League. They specified they wanted a game of the “platformer genre” where there are “multiple levels with increasing difficulty” and for the player to have “a run, a jump and a dash”.

After the first interview, I proposed a platformer game with multiple levels and a player with a run, jump and dash. This met their specifications at the time however Matthew wanted the game to have customisable controls due to him being left-handed. Tyler also suggested that the game should have a level selection screen which will display the levels that the user has unlocked. Elliott proposed the idea of multiple enemies in order to increase difficulty and improve the gameplay.

I have researched the problem by researching various games of the platform genre and how they have implemented player movement. To do this I have looked at how both Celeste and Super Meat Boy have created their player. I have found that Celeste’s approach to movement was very good, and I liked how the player felt responsive.

I have watched a video by Dawnosaur about improving a player’s movement by using forces. This was a very good approach to movement as it means that the player doesn’t immediately reach their top speed but instead accelerates towards it.

User Requirements

* A player who can run both left and right as well as jump, wall jump, and a dash
* Different types of enemies
* Multiple levels that generally increase in difficulty
* Customisable controls for maximum accessibility
* A GUI to allow the user to select the level they want to play

Objectives

1. The Game should provide a navigable GUI
   1. A Main Menu with buttons: Play Game, Settings, Quit
      1. When the user selects the Play Game button a level selection screen is displayed
         1. The Level Selection Screen shows which only allows the user to select the levels that the user has unlocked
      2. When the Settings Button is selected the user is presented with a settings menu allowing the user to customise their controls and change the audio output level
         1. Each action the player can execute will be listed with their respective key to execute that action with option to select a different key
         2. There will also be slider where the user can control the volume for the game music
         3. There will also be a slider where the user can control the volume for the game sound effects
      3. When the user selects the quit option the game will stop running
2. The player will be able to run left and right, jump, dash and wall jump
   1. When the run action is held it will allow the player to accelerate in the direction of the button press to the player’s top speed by applying a variable force depending on the speed difference of the player’s current speed and their target speed
   2. The jump will allow the player to travel vertically upwards
      1. Coyote time (this originates from the when Wile E. Coyote doesn’t fall immediately) will also be implemented which means that if the user presses the jump button a fraction too late and are not on the ground the player can still jump
      2. There will also be jump buffering so that if the user presses the jump button slightly too early the player will still jump the moment they touch the ground
   3. The player will be able to dash in the listed cardinal direction by pressing the movement keys and the dash button in various combinations DASH + (UP, UP+RIGHT, RIGHT, LEFT, UP+LEFT)
   4. The player will also be able to jump off walls when the player is holding the key towards the wall and the jump button is pressed.
3. There will be multiple levels that will increase in difficulty
   1. Each level will generally increase in difficulty, this would be by combining an increase in the length of the level or by requiring the user to have more precise control over the player by executing more difficult jumps and dashes
4. There will be one player on one machine that saves the levels that the player has unlocked
   1. As the user unlocks levels by completing the previous one, the levels that the user has unlocked are saved
   2. This game is saved locally onto the user’s machine

What is Out of Scope?

* An AI to show the player how to complete each level efficiently that learned by itself
* High resolution textures
* Random level generation with Wave Function Collapse
* Many worlds with multiple levels and boss fights
* Combat system
* Saving in the middle of a level
* Multiplayer across a LAN and Internet

Hierarchy Chart

Diagram

Description automatically generated

What is Needed to Create your Solution?

* A 3.11 Python interpreter
* The latest Pygame release, may need to be the dev build
* Internet access to view documentation for the Python Standard Library and Pygame

What is Needed to Run your Solution?

* A Windows machine will be needed to run the .exe file
* The minimum tested specifications are an i31005G1 with 8GB RAM