Final Report for Algorithm and Programming

Lecturer:
Jude Joseph Lamug Martinez, MCS

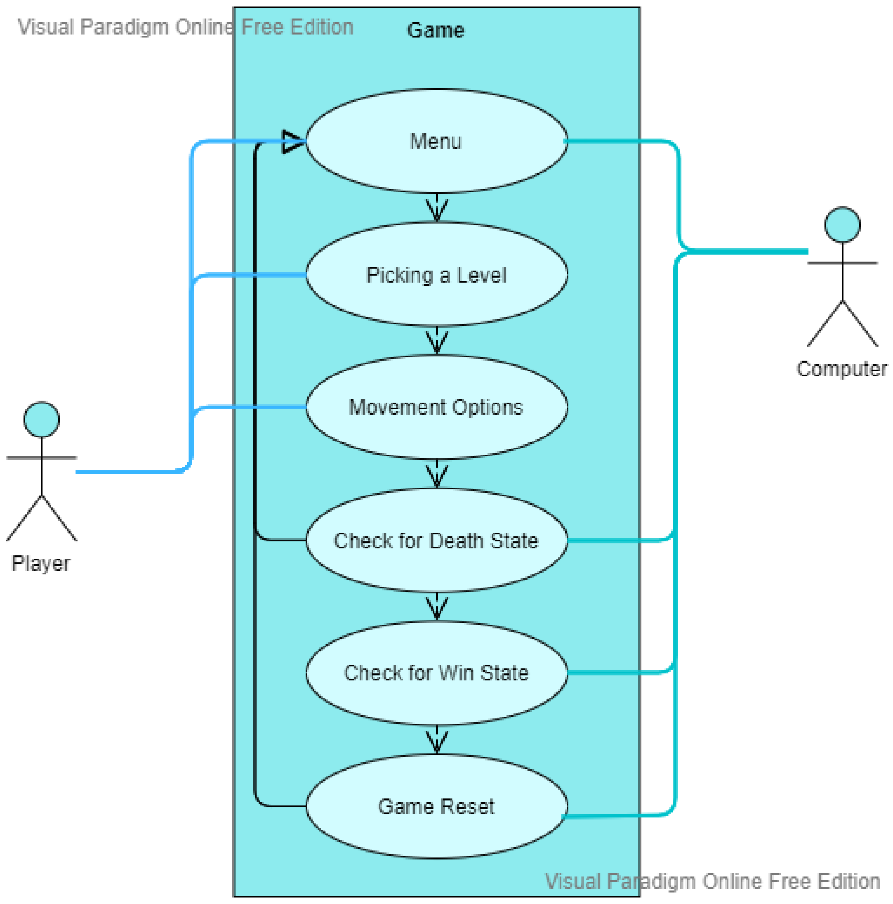
Made By:
Michael Bengawan 2602184290

Binus School of Computer Science Undergraduate Program
Universitas Bina Nusantara
Jakarta
2023

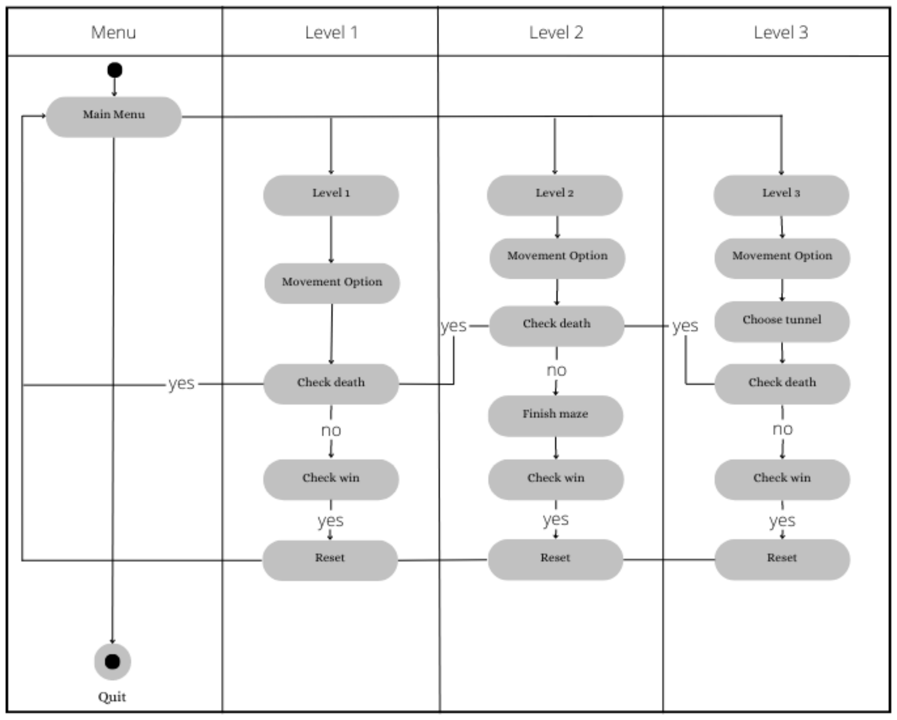
I. Brief Description

For my final project I made an unnamed game based on the popular game mario. Is a two dimensional platformer that is made in pygame with a few different added modules to aid in importing files. The game was first named RePixel but I ended up scrapping the idea due to pixel art is hard and I couldn't find suitable resources as replacements so with the pixel art I found on google it ended up more like tomb raider then what I imagined my game looking as, hence the reason its unnamed at the moment. The game is able to run with a standard start menu and 3 levels that are level 1, 2, and 3. The game features no enemies and the only way to die is to fall off the map, which resets your position back to the start menu. The game is more of a pixel art parkour then a 2d platformer but it does take likeness from mario games. The player character has an idle animation being pulling out her gun and putting in back and jumping, running, and also falling animations. There are also dust particles that spawn underneath the player as they fall, jump, or run.

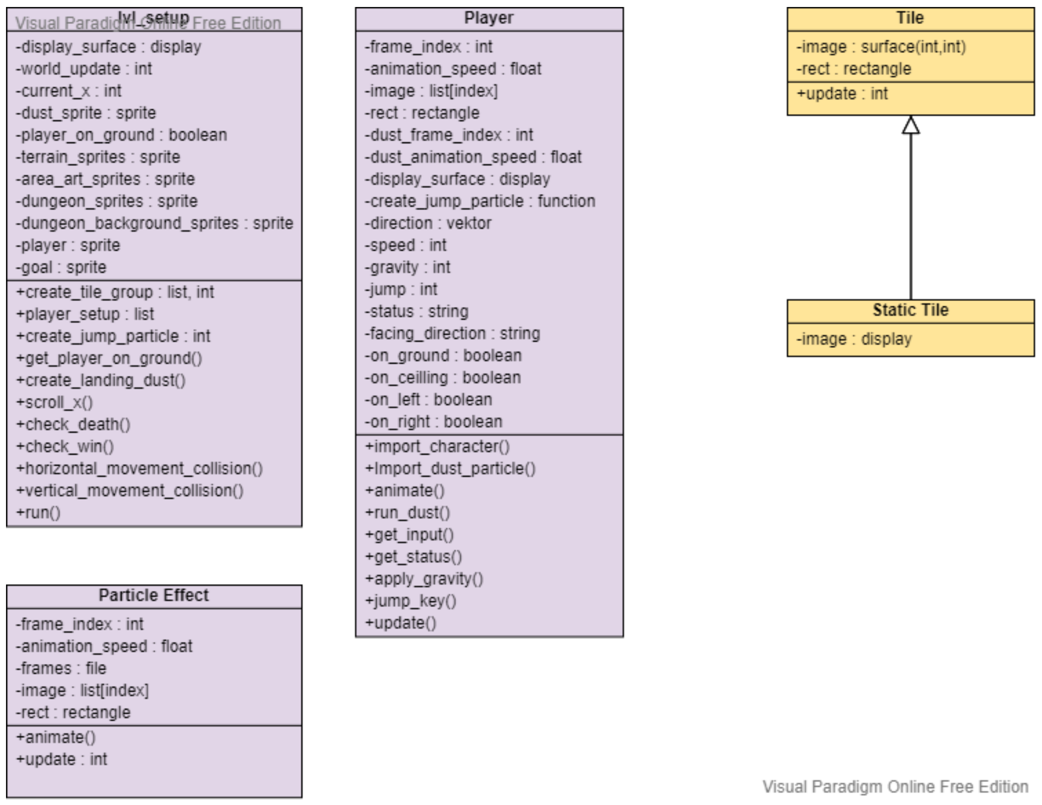
II. Use-Case Diagram



III. Activity Diagram



IV. Class Diagram



V. Modules

A. Pygame Module

Main module used in the creation of this game is pygame of course. Everything runs on pygame. From the window to loading each and every sprite pygame runs all of these. Pygame itself is a module of many many libraries worth of functions that are used to create primarily video games. Notable uses of pygame in this game are load images, transform scaling images, and getting inputs for each key pressed.

B. Sys Module

Sys isn’t really used much in this game. The only instance of it being used is for the sys.exit() function which is called for ending any existing python operation after pygame is done exiting. Sys is a system specific parameter which is a library of values and functions that interact heavily with the interpreter.

C. OS Module

OS is used in part of this project to gain all the files in a specific folder for animation. OS itself is a module in python that heavily interacts with the operating system. Hence why it is used as a path interpreter.

D. CSV Module

CSV is used in this project as a way to read a CSV file or a comma separated file. This is used using the modules reader function which is used to make a list that is filled with all the values for mapping the tiles in the terrain map.

VI. Essential Algorithms

A. Pygame While Loop Algorithm

Algorithm to keep pygame running in a window with the height and width that is being specified. Another thing in this loop is that it is used also to generate the background image. And to transform the image from a larger pixel size to the better scale of 1200 to 800 pixels. This loop runs up until the x button is pressed causing the sys.exit() and pygame.quit() function to start and end the program.

B. Collision Detection Algorithm

This algorithm is a collision detection that occurs when two objects or rectangles in the surface hit each other. This collision algorithm works by using the collision detection that is already in-built in pygame and using it to detect if the player is hitting a certain terrain. The algorithm itself is built in a way so that vertical and horizontal collision is separated. This is to ensure that not each one would take priority so that when an object hits an obstacle it checks its x position then its y position of collision separately. To ensure that the collision is current.

C. Tile Generator Algorithm

First the algorithm takes the path that specifies which image that needs to be generated. This algorithm also requires a few things to run which are the path that the image we want to place as a tile, a level map data which is a csv file, and the tile size. First it takes the image with the tile art then it uses the tile size to generate a list full of sliced images based on the tile size. This also takes into account that the image art is already sized according to the tile size. Next for the csv file it takes the value from it as a placement tile. For example in this project -1 is an empty tile so that will be one of our indicators. The tile generator also uses the index of the list as a way to generate each tile according to the csv tile map.

D. Animation Algorithm

The animation algorithm requires three additional values to run, which are the animation images location or path, the status of the player, and the frame index. The algorithm first takes the path of the image according to the status of the player. Then afterwards it cycles between each state of image with frame index which then gets increased per frame the program is running which is used to control the animation speed.

E. Player-Key Pressed Algorithm

A general key press detection algorithm which just uses pygame.keys method to take all the current key presses that the player is using. For if the player is pressing the space button it will activate the if statement which is dedicated to it and also for the other if statement.

F. General Overworld Algorithm

A compilation of small algorithms that are used in the overworld of the game which are a bit basic but works to create the game.

1. Jumping Algorithm

   Taking the key pressed algorithm and checking if space is checked and the player is on ground then adding the players direction.y according to the jump height specified in settings.

2. World Shift Algorithm

   Taking the players position based on the screen width then moving the background by the player speed and setting the player speed to 0 to give the illusion of it moving.

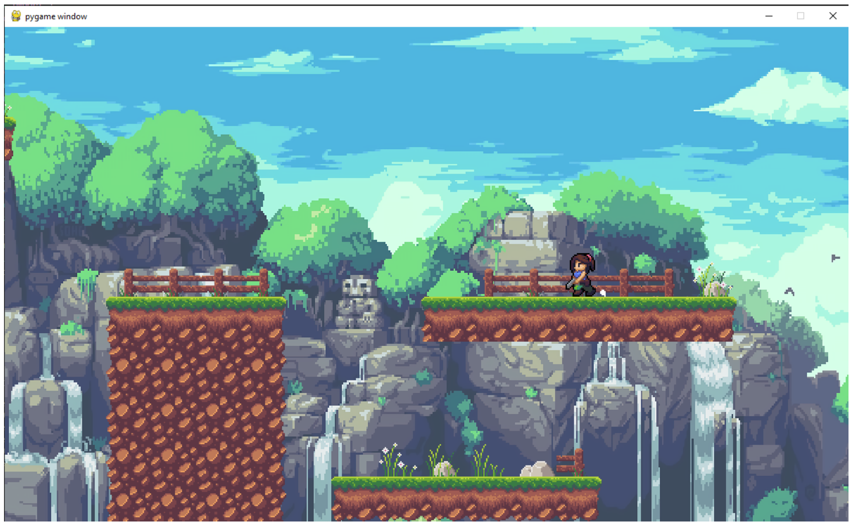
3. Gravity Algorithm

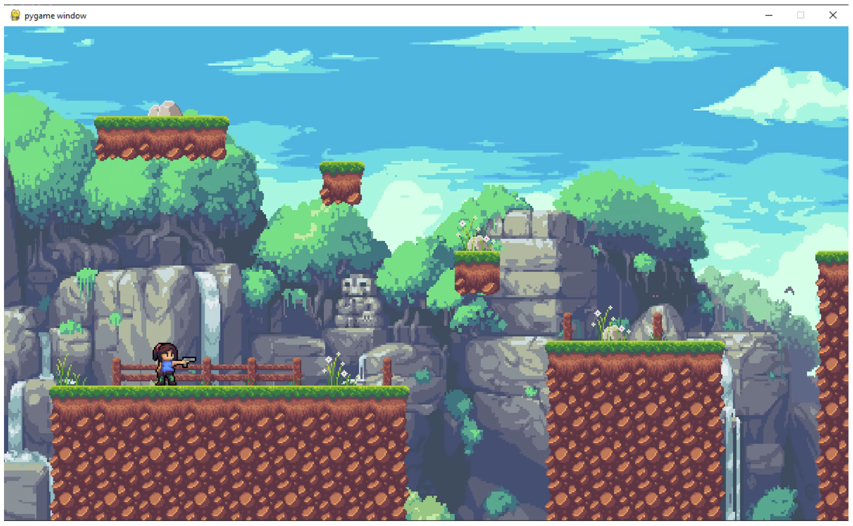
   Constantly reducing the players direction.y downwards by the specific modifier in settings.

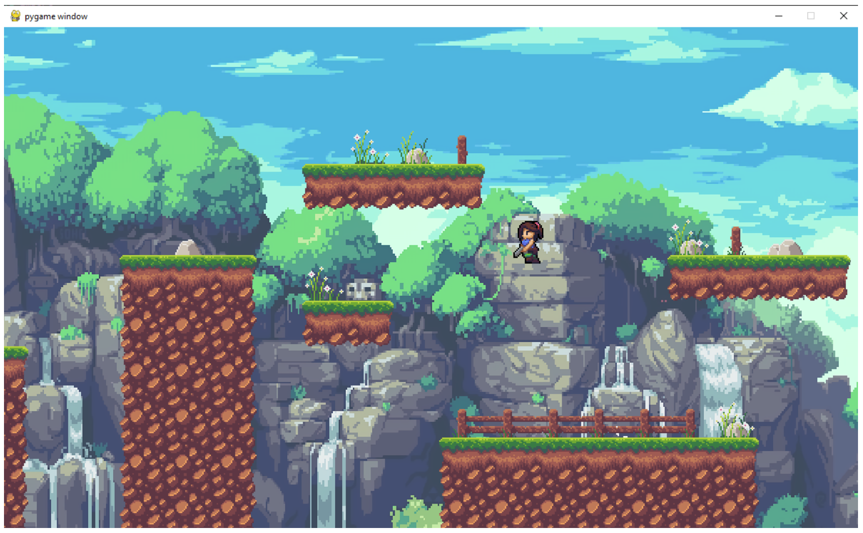
4. Status Algorithm

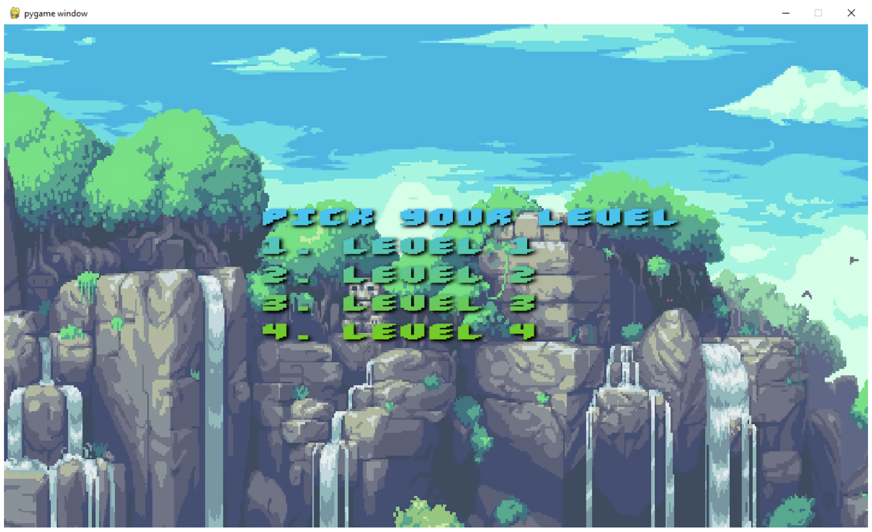
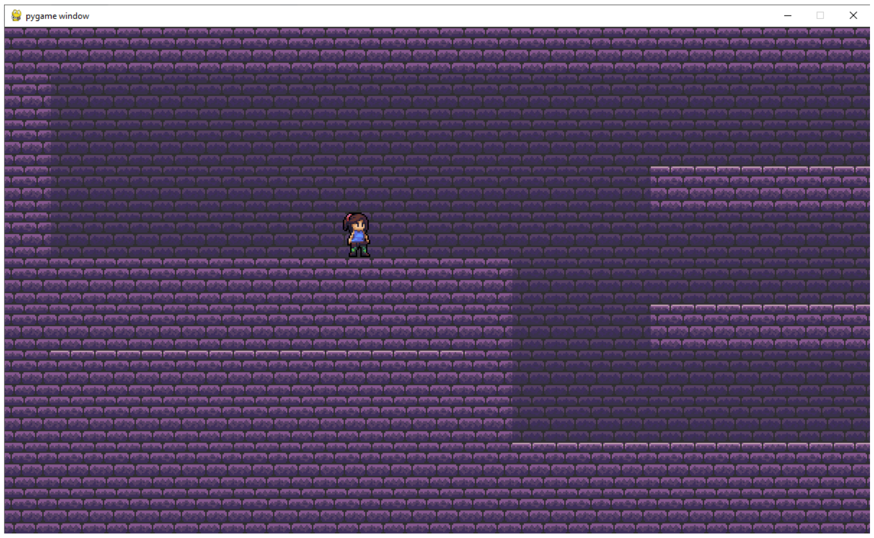
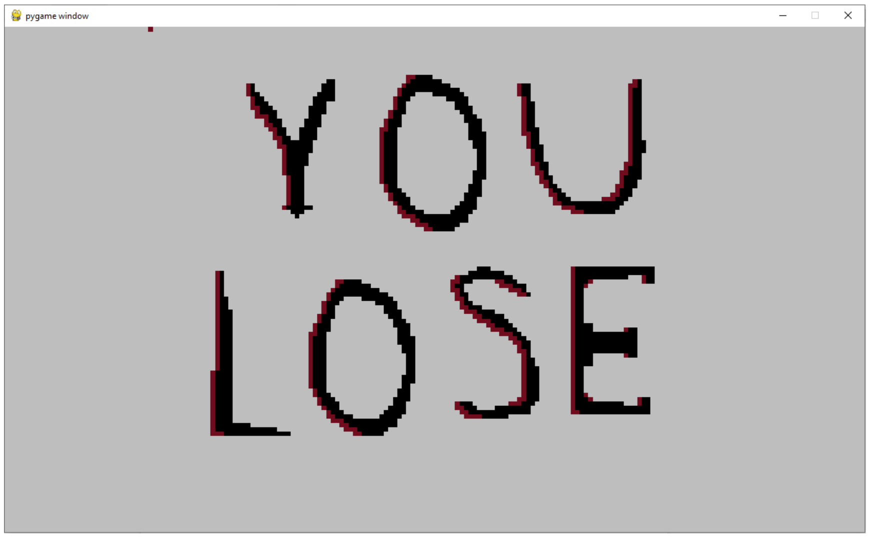
   Changes the players status depending on the current position of the player. There is status on.ground which happens if the player is colliding with the floor this algorithm is run parallel to the collision algorithm so that if collision occurs it changes the status.

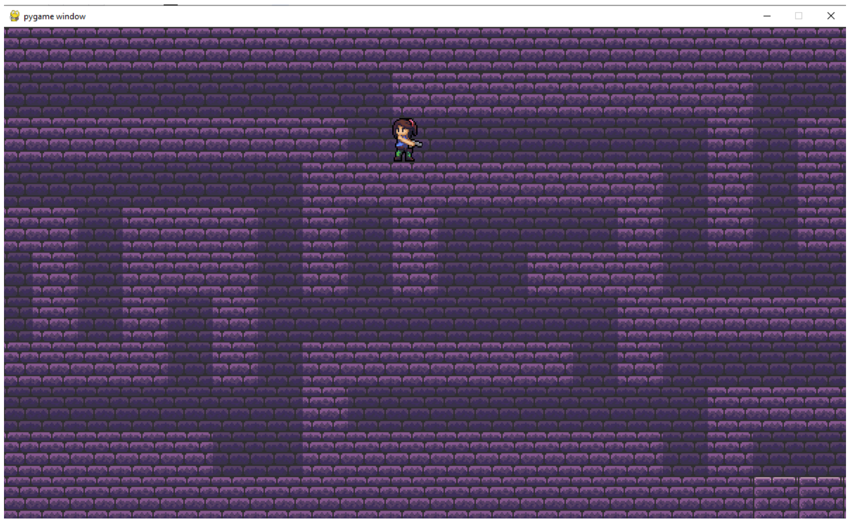
VII. Screenshot of your Application

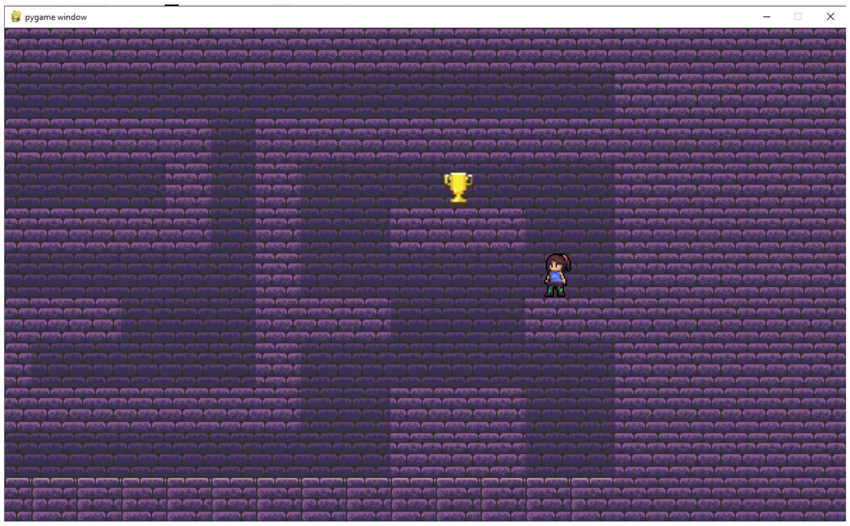


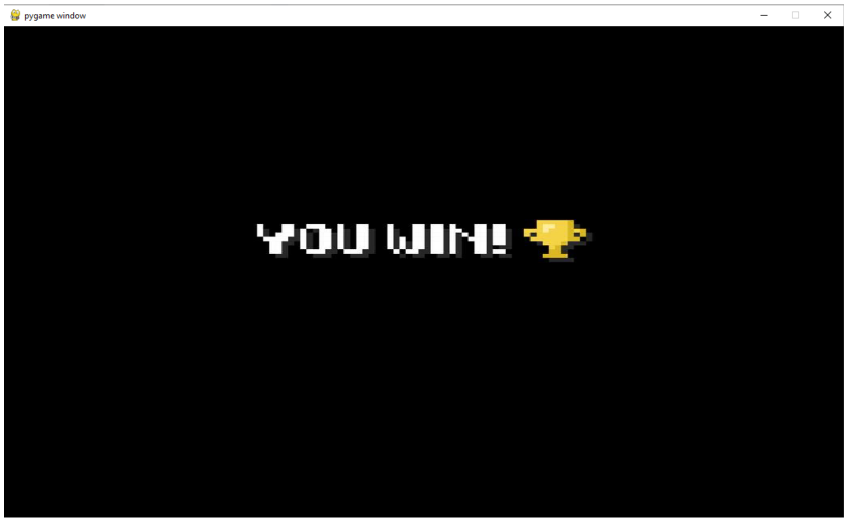





pygame window
PICK YOUR LEVEL
1. LEVEL 1
2. LEVEL 2
3. LEVEL 3
4. LEVEL 4

pygame window

pygame window
YOU
LOSE


pygame window


pygame window


pygame window
YOU WIN!

VIII. Lessons Learned

I have learned a lot since the start of making this project, most importantly time management. While I expect most to learn a lot about coding in they're projects, time management has probably been the thing that I have slowly come to understand as the most important thing in making something like this. I over estimated the amount of work that would be required to make something like this. That is the main result on why the unnamed game is as it is right now. If i had more time management the game could have been finished and would have been something I am proud of. But as it is currently more of a tech demo then a game. But regardless, lessons about coding were still learned, most of what I learned is importing files into data types in pygame or just python in general. For example i had to import all the files in a csv format that being a comma separated file which had the data map for all my tile blocks. I used the in-built csv module in pygame to extract this data and put it in a list. This data map is then used to create the tile map that it is now. Of course I also learn more about pygame as it is like how it is able to detect collisions from one rectangle to another rectangle. And using this detection as a way to be able to stop the player and put it in a different area according to the values that are there. All and all I did learn a lot from this experience giving more insight on how code works and how pygame more importantly python runs.