Algorithm and Programming Class -

Final Project Report

Ali Samy Juffry

Binus University International Program

Computer Science - 2902736682

Jude Joseph Lamug Martinez, MCS

6 January 2026

Project Specifications

General Descriptions

Title : Samy's Parkour
Type : Game
Description : A fun 2D platformer game that has 5 levels, featuring blobs and lava.

Project Link

The GitHub repository for the project can be accessed using the link provided below:
https://github.com/Samyjuffry/samys_parkour

Target Audience

The game has a wide range target audience due to its simple mechanics of just using arrow keys to play, trying to progress through the obstacles. Therefore, the game is suitable for everyone, without restrictions.

Gameplay Mechanics and Features

The game is controlled using a cursor and keyboards. For the cursor, the player only needs it to start the game by clicking the "START" button as well as to close the game by clicking the "EXIT" button. For the keyboards, the player only needs to use the arrow keys to navigate the character. The objective of the game is to collect as many coins you can and reach the exit of each level while avoiding hazards.

To play, click the "START" button, and the game will start at the first level. The player character spawns and you move across platforms, jump over gaps, avoid the lava, avoid the blobs, and collect as many coins as you can. Every level is completed by reaching the exit door.

The game progresses through multiple levels. Each new level proposes a different layout and challenges. If the player dies, the **current** level restarts.

Win and Lose Mechanics

- You win a level when you reach the exit point.
- The game is completed when you complete level 1 to 5.
- You lost all of your coins when you hit the blob or lava.
- The level reset and the player respawns when you hit the blob or lava.

Level and Difficulty Systems

- Difficulty increases from each level through more lava, blobs, and complex layouts.
- Progression is linear from level 1 to 5.

Core Gameplay Systems

- Gravity based movement system (Ex: when you jump, the sprite will fall/you will fall when you walk on air)
- Collision detection with platforms, blobs, lava, and walls.
- Level reset system after death.

Dependencies

- Python 3.14 : The main programming language used for implementing the algorithms.
- Pygame : Python module that helps in creation of simple 2D games using the Python language.

Built in Python Modules

- os : Python module that provides a portable way to interact with the underlying operating system.
- pickle : Python module that saves and loads Python objects as binary data

Algorithm Implementation

Game Menu

The game starts by initializing the pygame, mixer, window, fonts, images, and sounds. After initializing, enter the main loop with main_menu = True. In the menu, the UI displays a background forest and two buttons “START” and “EXIT”.

- If the user clicks “EXIT”, the main loop ends and the game closes.
- If the user clicks “START”, the main_menu will become false and move onto the gameplay.

Main Game

As soon as the gameplay starts, the program will draw the first level and update the tiles, hazards, and sprite every frame. The world is built from a 2D grid loaded from a pickled file named level(1-5)_data. The world constructor reads each grid value and creates tiles and sprites. The game runs at a fixed 60 fps using the clock.tick(fps).

Event Handling

The game uses two input systems, which was :

- **Global events :** it checks pygame.QUIT to close the window and stop the loop
- **Player controls :** it reads pygame.key.get_pressed() inside Player.update() to move left, right, and jump
  - Arrow up triggers jumping
  - Arrow left and right to move horizontally.

Player Movement and Physics

Player.update() manages the character movement and physics each frame :

- It sets dx and dy from key input.
- It applies gravity by increasing the vel_y and adding vel_y into dy.

- It updates animation frames using counter and index.
- It updates the player's invisible rectangle position after collision resolution.

Collision Algorithms

Tile collision

The player checks collision against every tile in world.tile_list.

- X collision : If a tile blocks the next x position, dx = 0
- Y collision : If a tile blocks the next y position,
  - When moving upward, it pushes the player down and sets vel_y = 0
  - When falling, it places the player on top of the tile, and sets vel_y = 0, and sets in_air = false.

Sprite collision

The character checks sprite collisions using pygame.sprite.spritecollide().

- Blob collision sets game_over = -1 and plays the game over sound.
- Lava collision sets game_over = -1 and plays the game over sound.
- Win door collision sets game_over = 1

Moving Platform collision

The character checks collisions with every platform sprite.

- X collision blocks horizontal movement by setting dx = 0
- Y collision moves the character above or below the platform using a collision tolerance value.
- When the platform moves sideways, the character follows it.

Coin Collection algorithm

Coins are sprites in coin_group.

- If the character collides with a coin, the coin disappears and the coin sound plays.
- The program draws a "COINS COLLECTED=" text at top left.

Game Over

When game_over = -1,

- Player.update() switches the player sprite into a ghost.
- The program draws “GAME OVER!” text.
- The restart button displays.
- If the restart button is clicked, reset_level(level) reloads the current level and clears all sprite groups, game_over = 0, and coin_collect = 0.

Win and Level progression

When game_over = 1,

- The program goes on to the next level
- If level is still in max_levels range, it loads the next level using reset_level(level) and sets game_over = 0.
- If the level exceeds the max_levels range, it draws “YOU WIN !” text and waits for the restart button to be clicked.
- If the restart button is clicked, it will restart to level 1 and reload level 1.

Reset Level function

reset_level(level) starts a clean reload.

- It resets the player position
- It empties the enemy, platforms, lava, and win door.
- It loads the world_data from level(1-5)_data using pickle.
- It creates and returns a new world instance.

Classes

Button

- Stores images and rectangles.
- Detects when a mouse hovering the button and left click to return action boolean.
- Prevents repeated click triggers by tracking a clicked flag.

Player

- Loads animation frames and a ghost image in Reset().
- Handles keyboard input, movement, gravity, collisions, and rendering.
- Returns game_over.

World

- Builds the level from grid data.
- Creates tile rectangles.
- Spawns blob, lava, coins, and win doors into sprite groups based on tile codes.
- Draws solid tiles in every frame.

Enemy

- The blob moves back and forth left and right by changing direction after a move counter limit.

Lava

- A stand still hazard sprite.

Win

- A win exit door sprite.

Data Structures Used

Lists

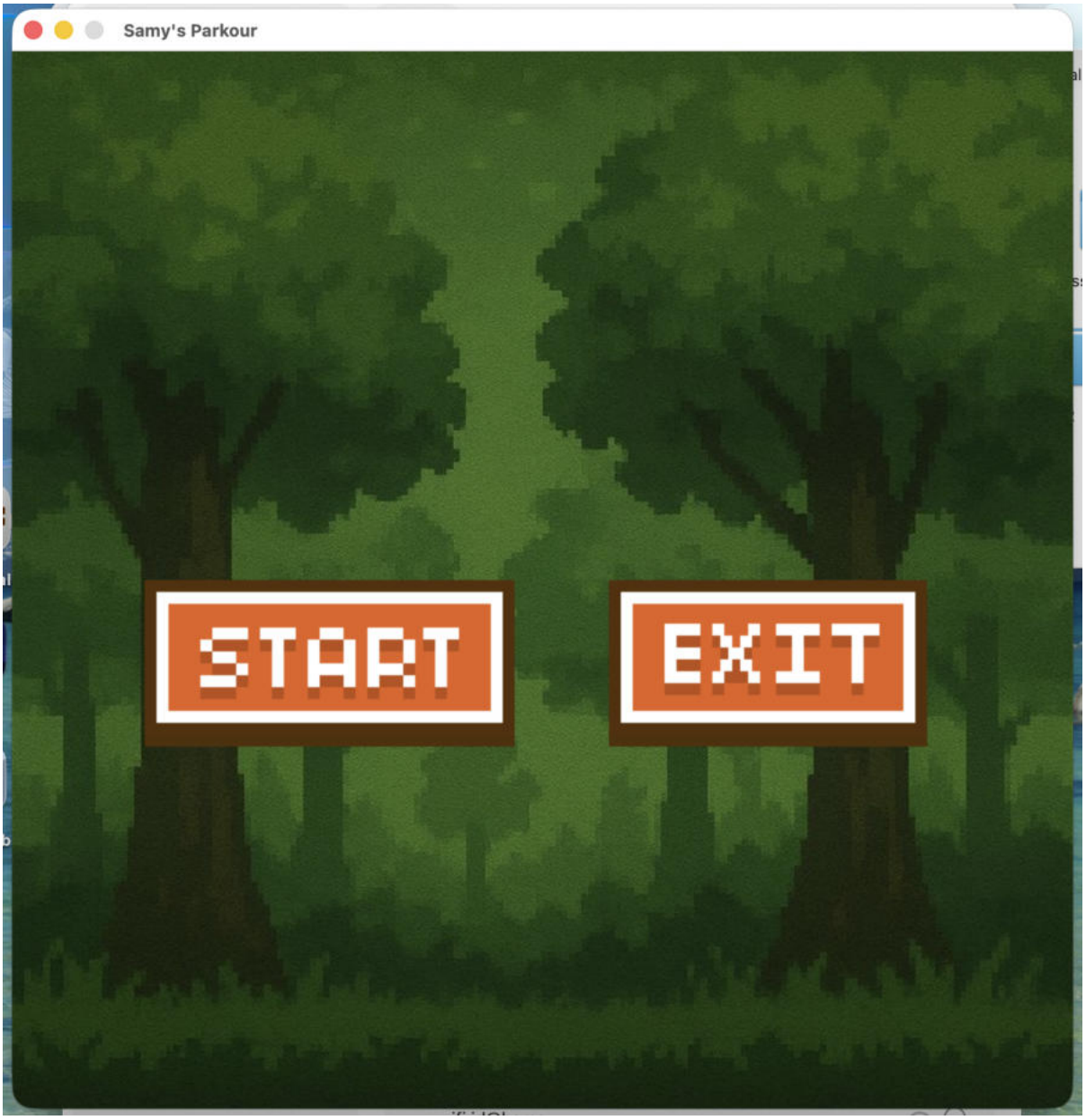
- world.tile_list stores tuples of image and rect for solid tiles.
- The level grid loaded from pickle is a 2D list of integers.

Tuples

- Every tile entry in tile_list is a tuple (surface, rect).
- The coin rect center uses a tuple (x, y).

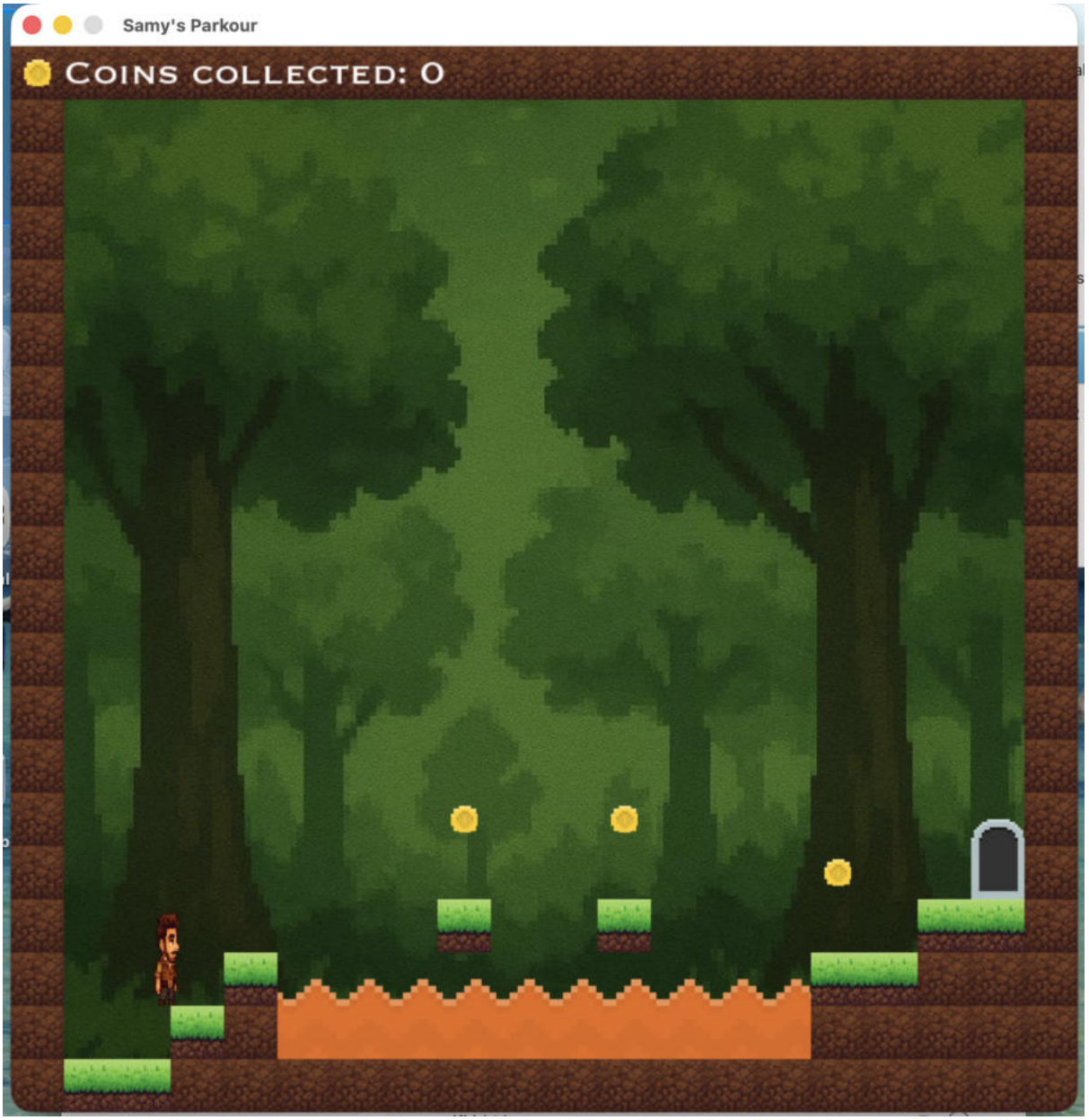
Evidence of a Working Program

Main Menu



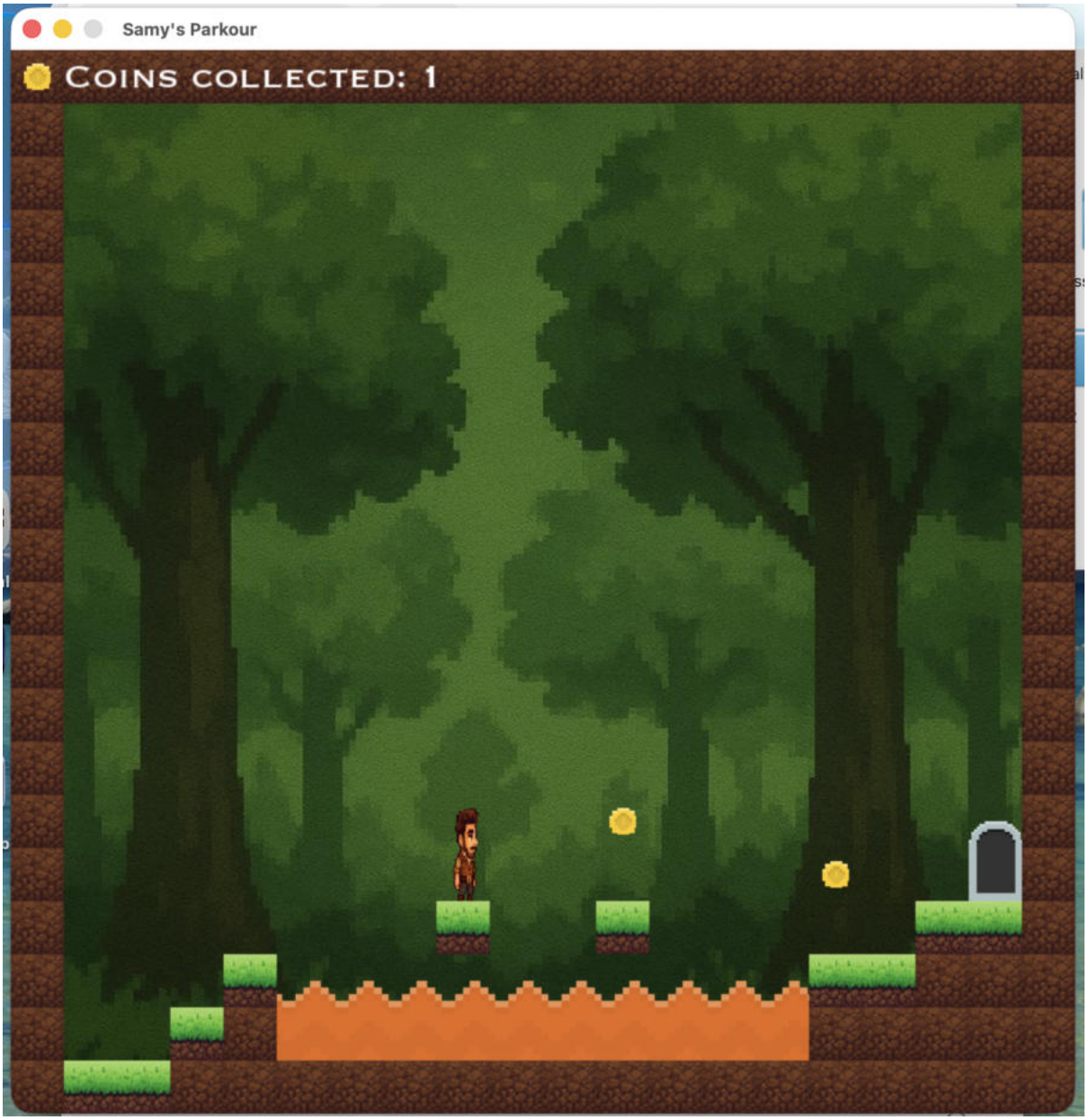
This is how the game looks when you first run the game/platformer.py file. The program would display two functional buttons to start the game and exit the game.

Gameplay



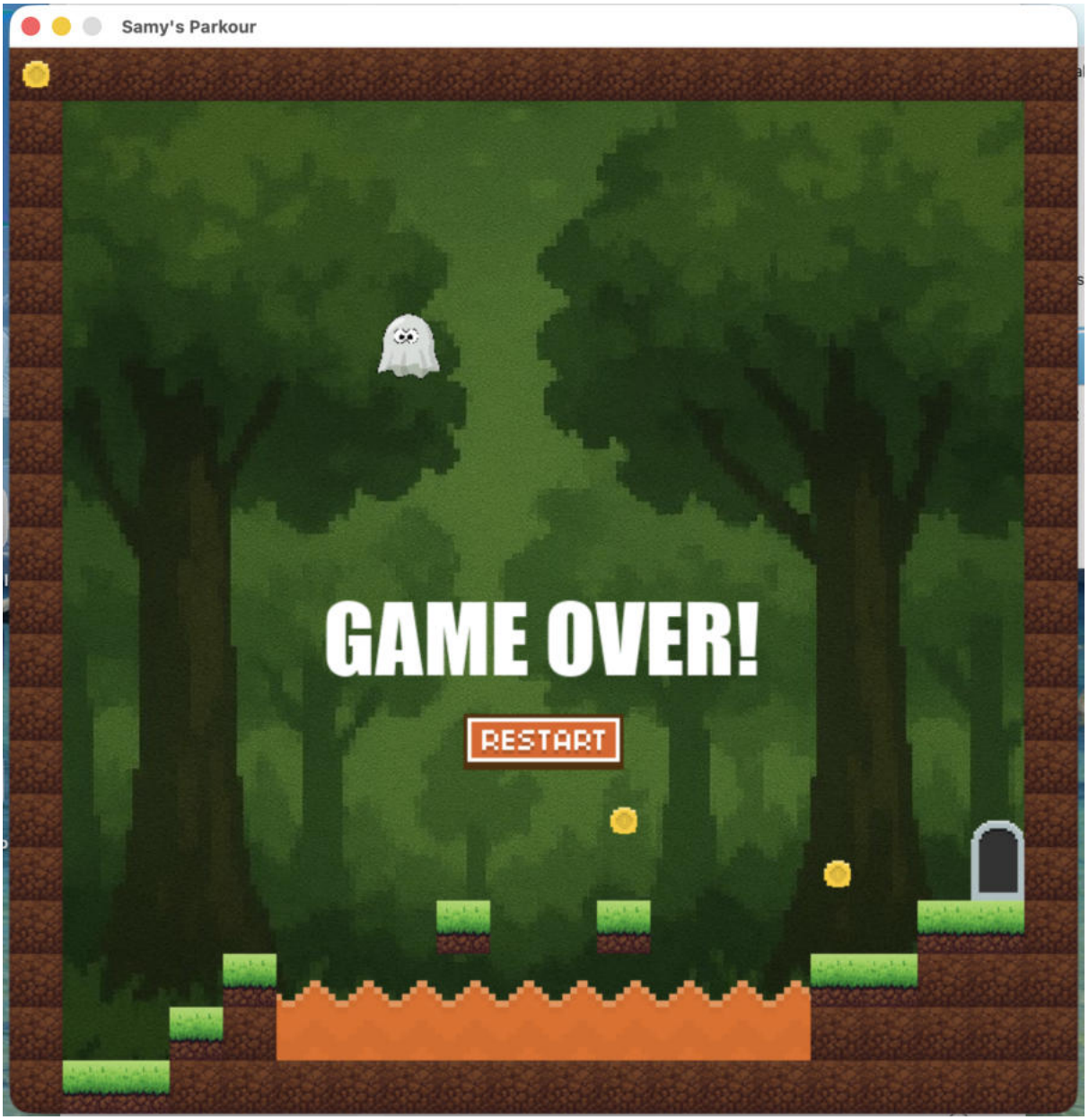
Once you click the “START” button, you will jump right onto the gameplay. Each level poses different challenges. Whether it’s a different coin placement, moving platforms, lava, and blobs.

Coin System



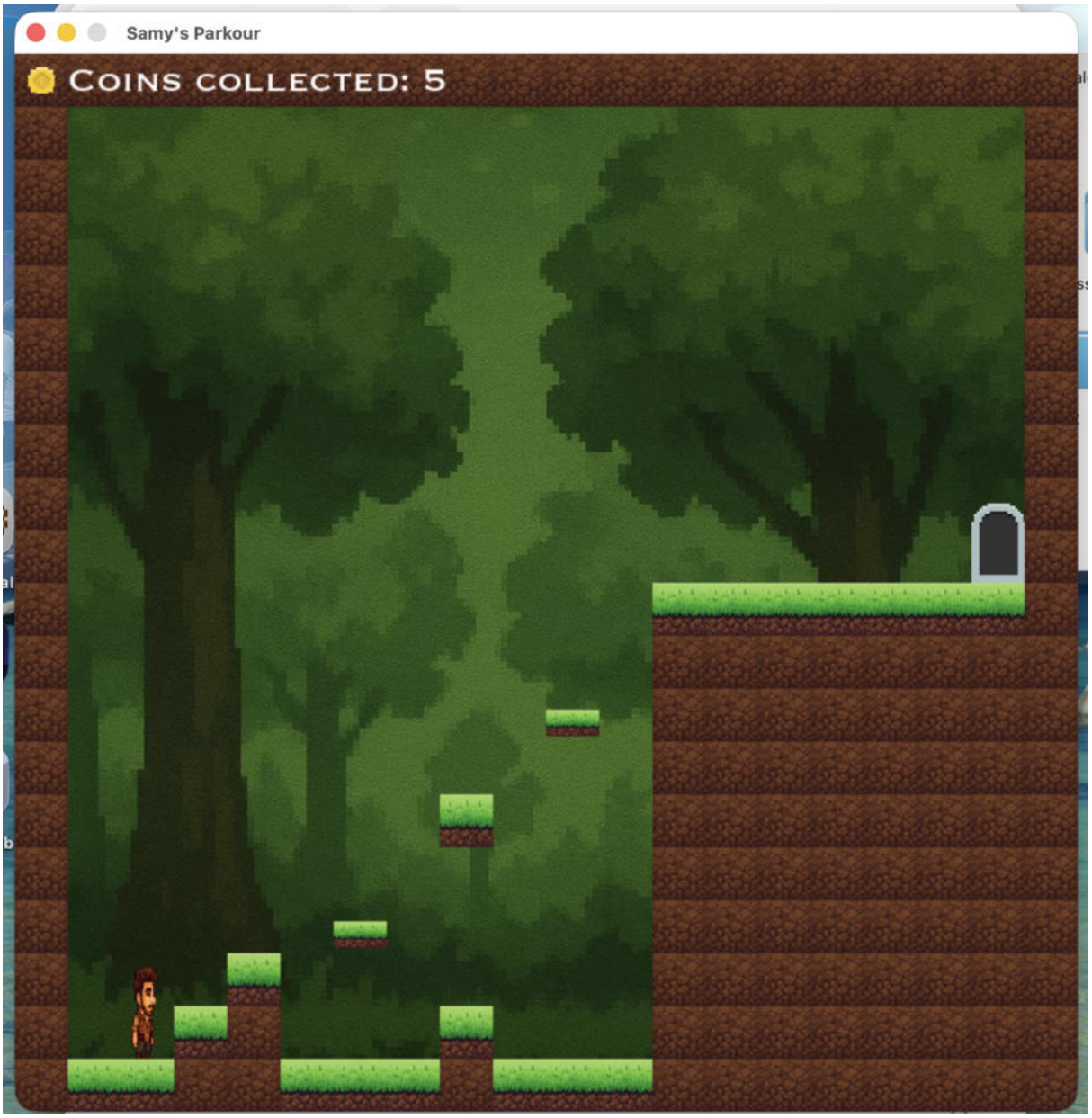
Once you collide with the coin, the coin disappears, and will play the coin sound effect. The text on the top left would also update and increase by one for each coin you collect.

Game Over



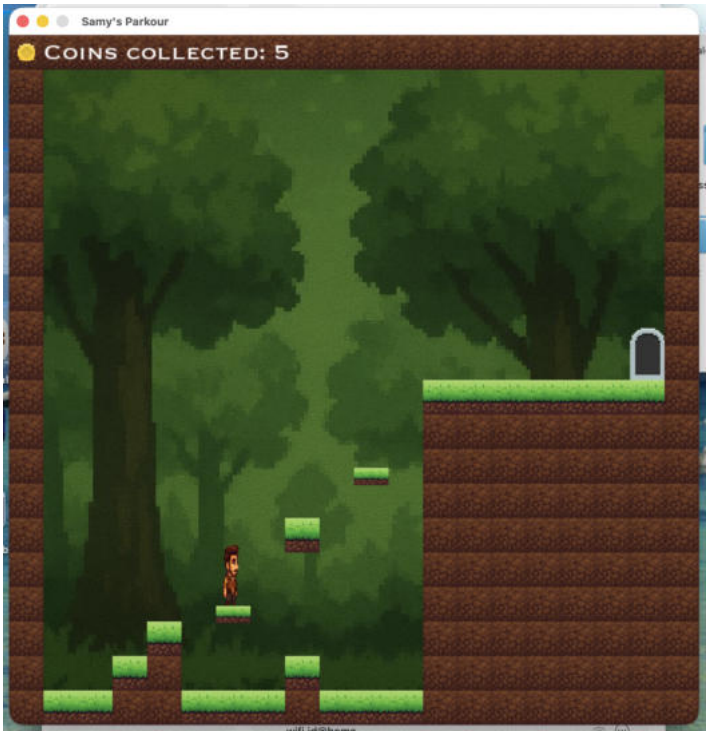
Once you collide with the lava or blobs, you will get a death animation, in which the character becomes a ghost and flies up. It would also draw a text that says “GAME OVER!” and a “RESTART” button to restart the current level.

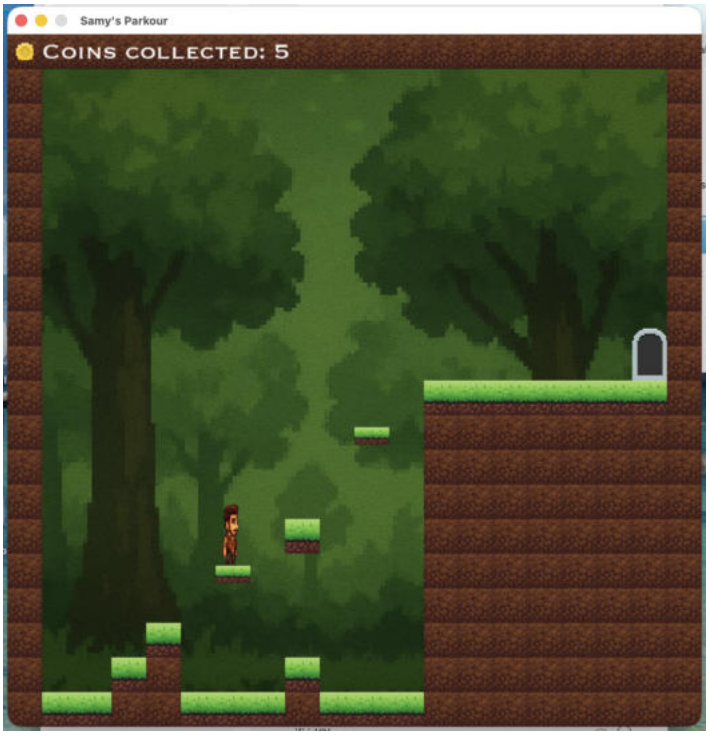
Exit Door



Once you collide with the exit door, the player would spawn in the next level.

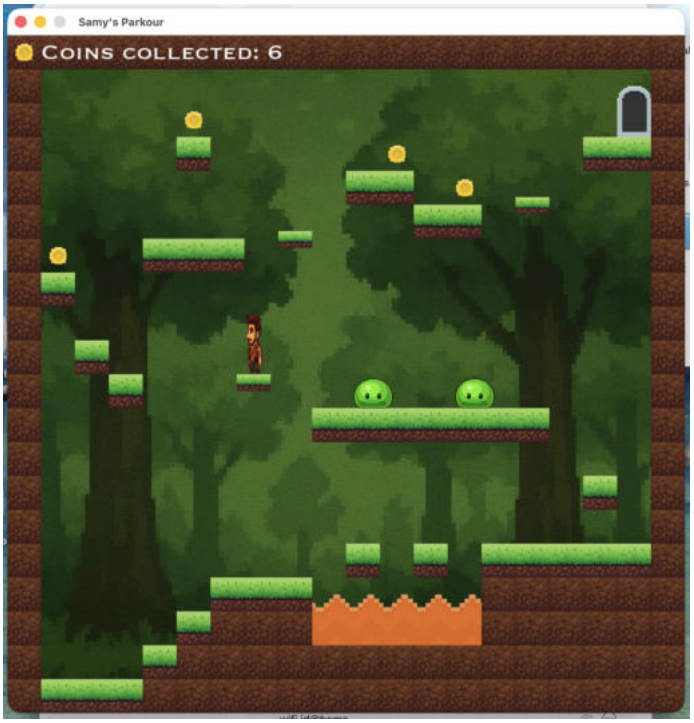
Vertical Moving Platform

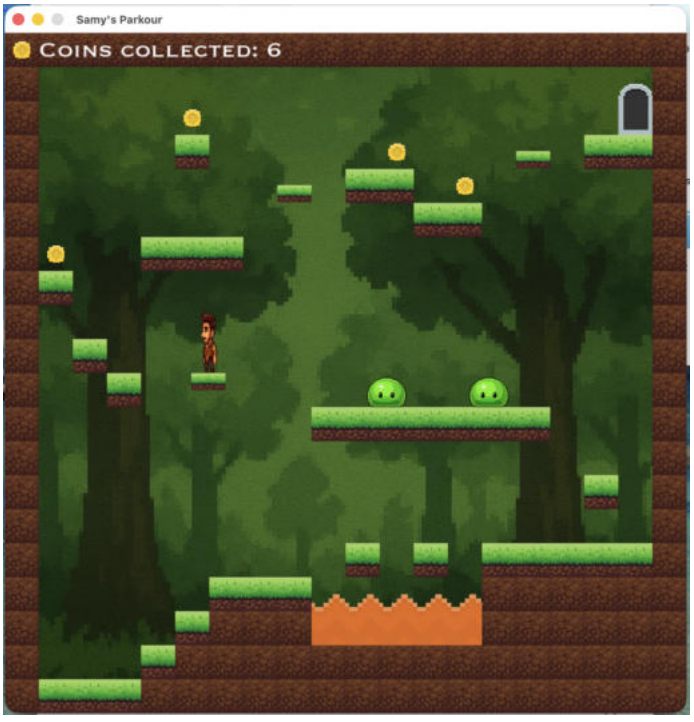




The moving platform moves vertically.

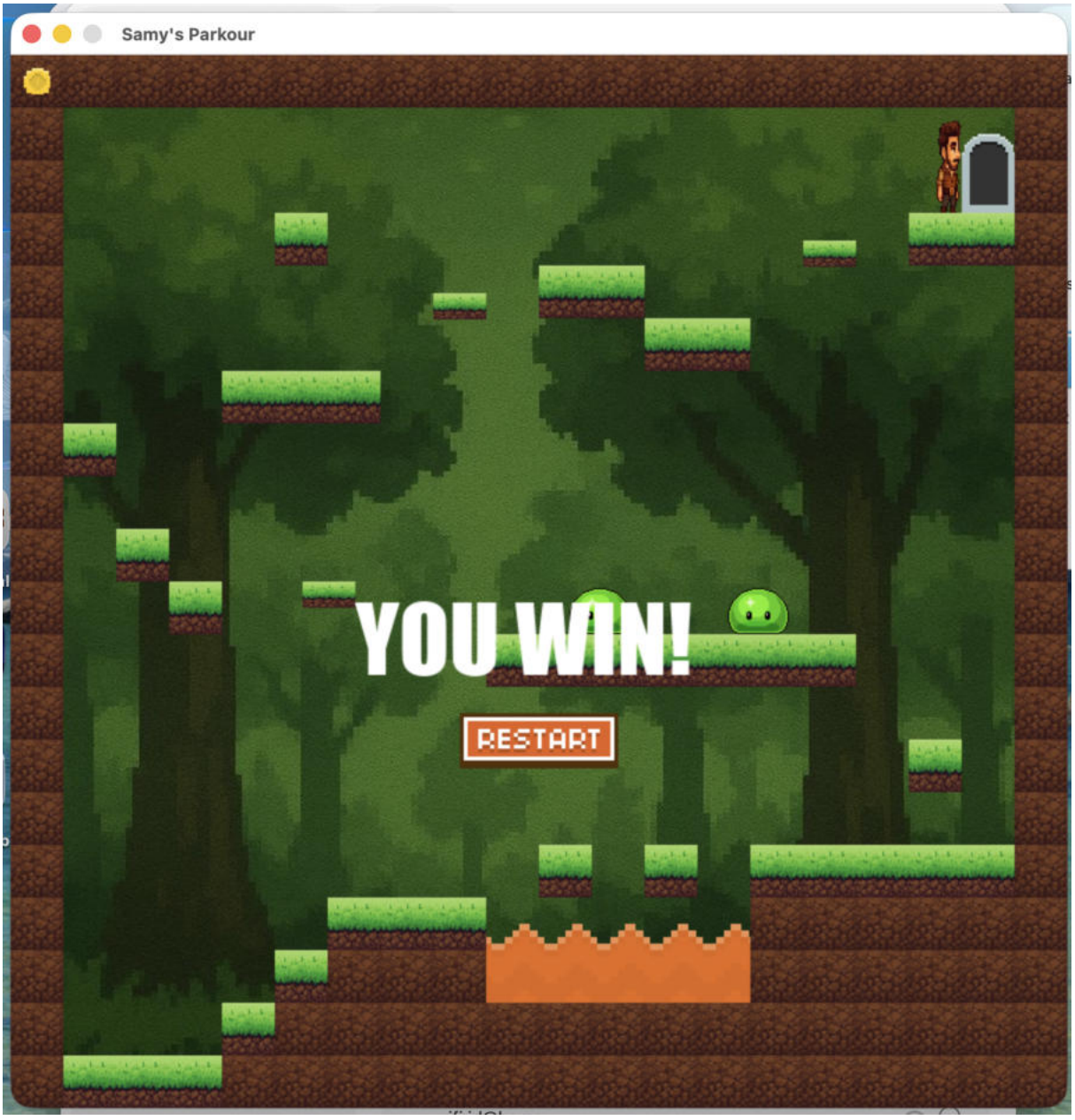
Horizontal Moving Platform





The moving platform moves horizontally.

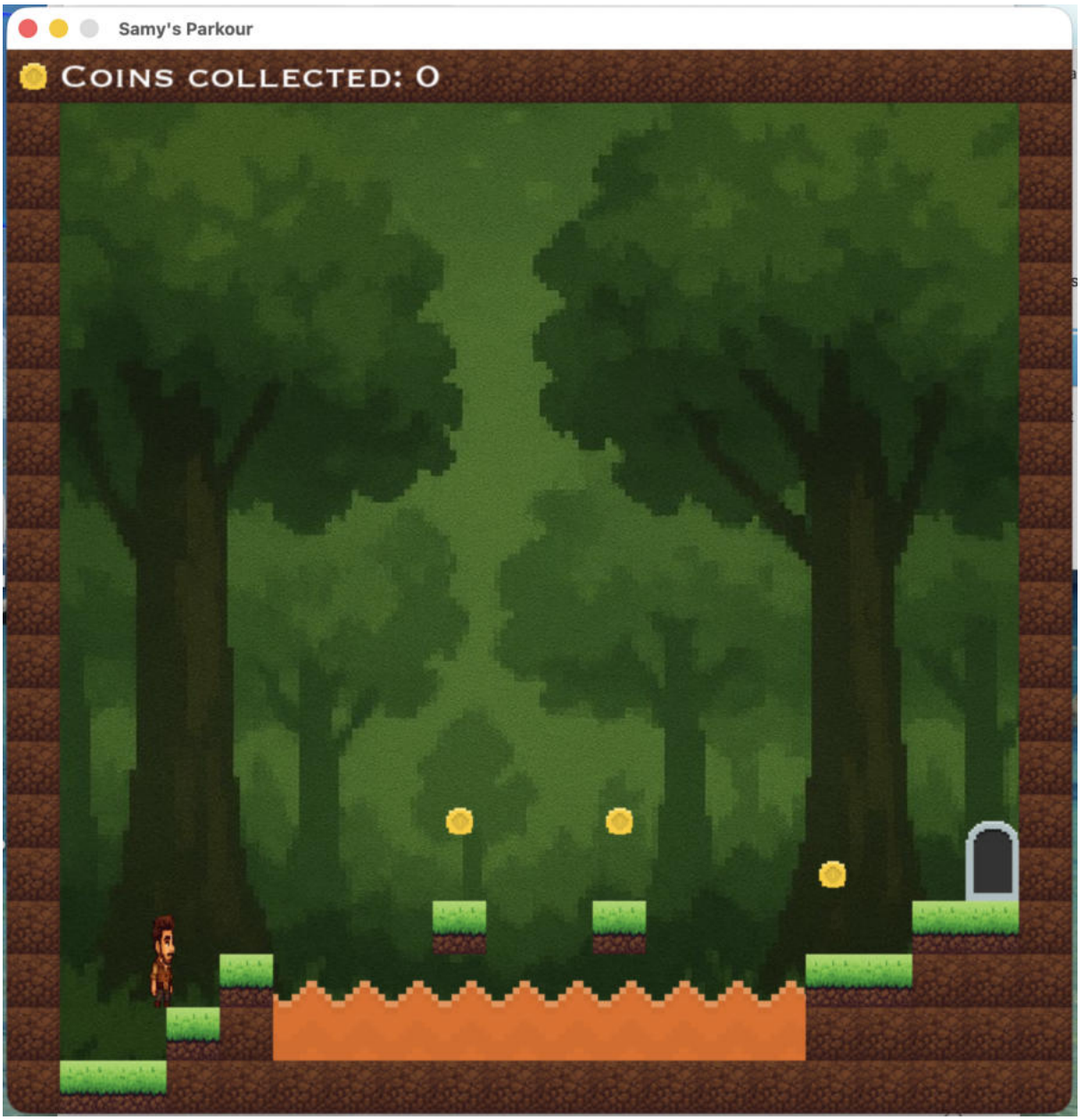
Win



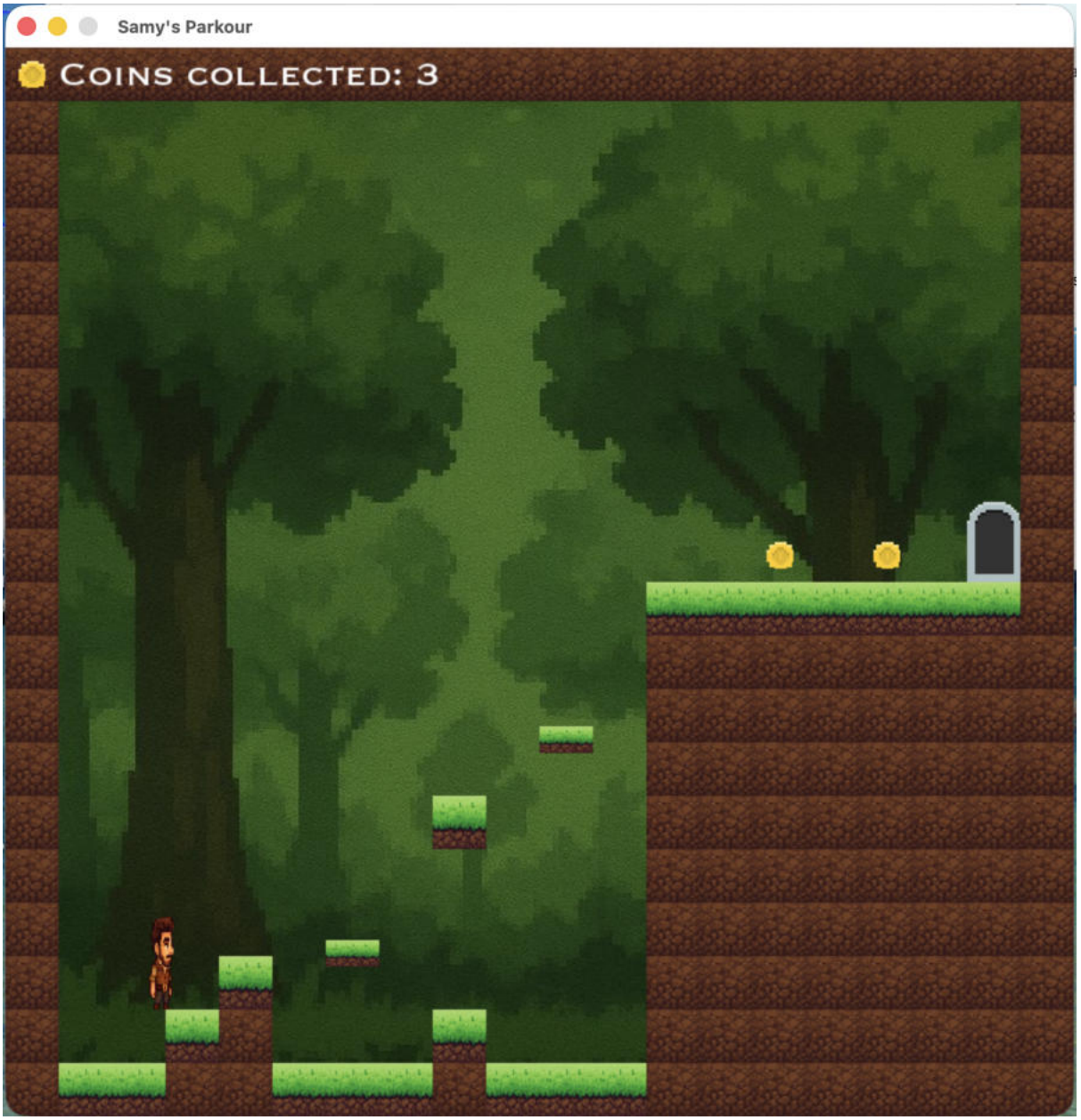
Once you made it to the last level and went on to the exit door, the game would freeze and there would be a pop up text saying “YOU WIN!” and a “RESTART” button if you would like to play again from the beginning.

All Levels Documentation

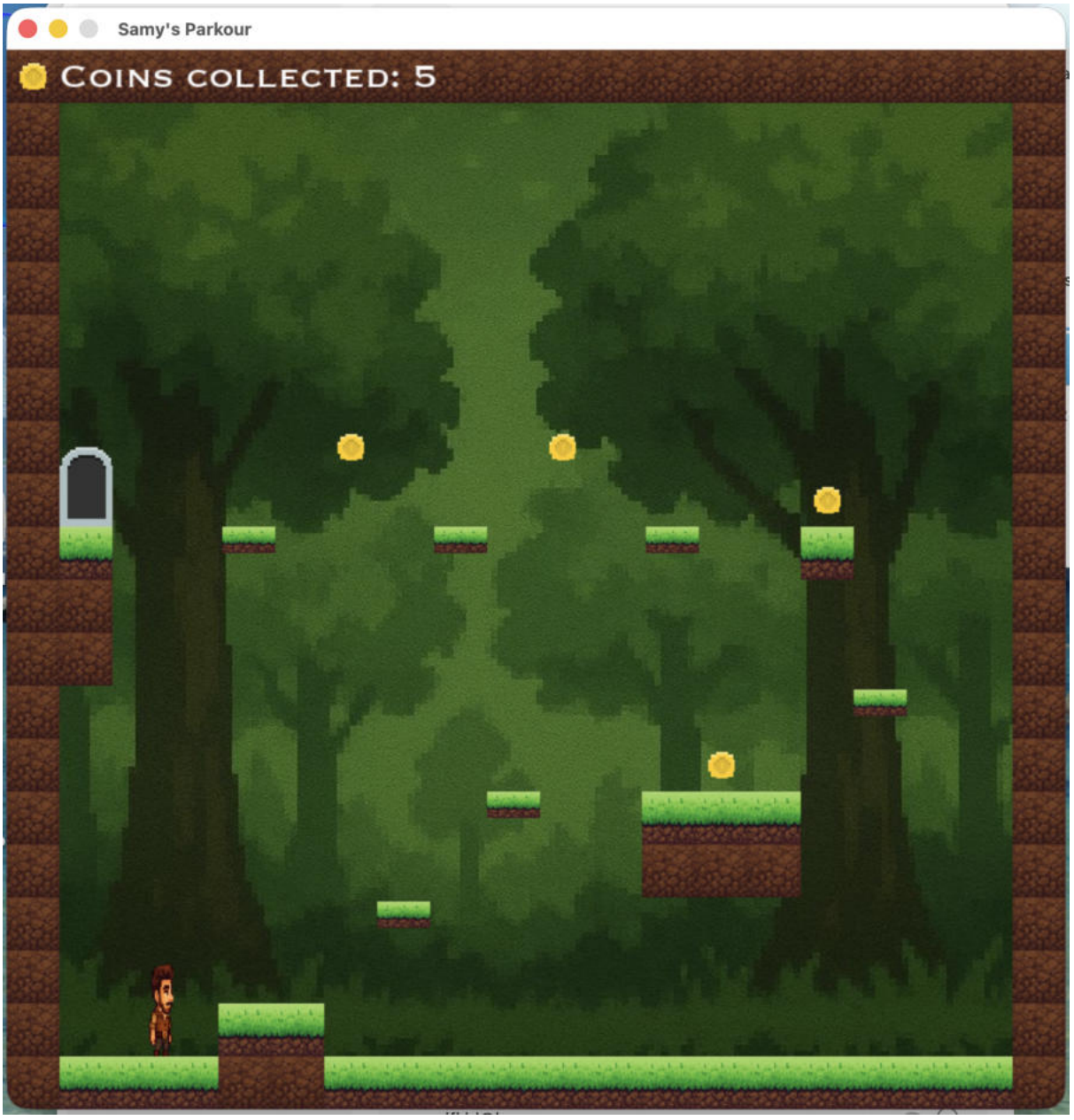
Level 1



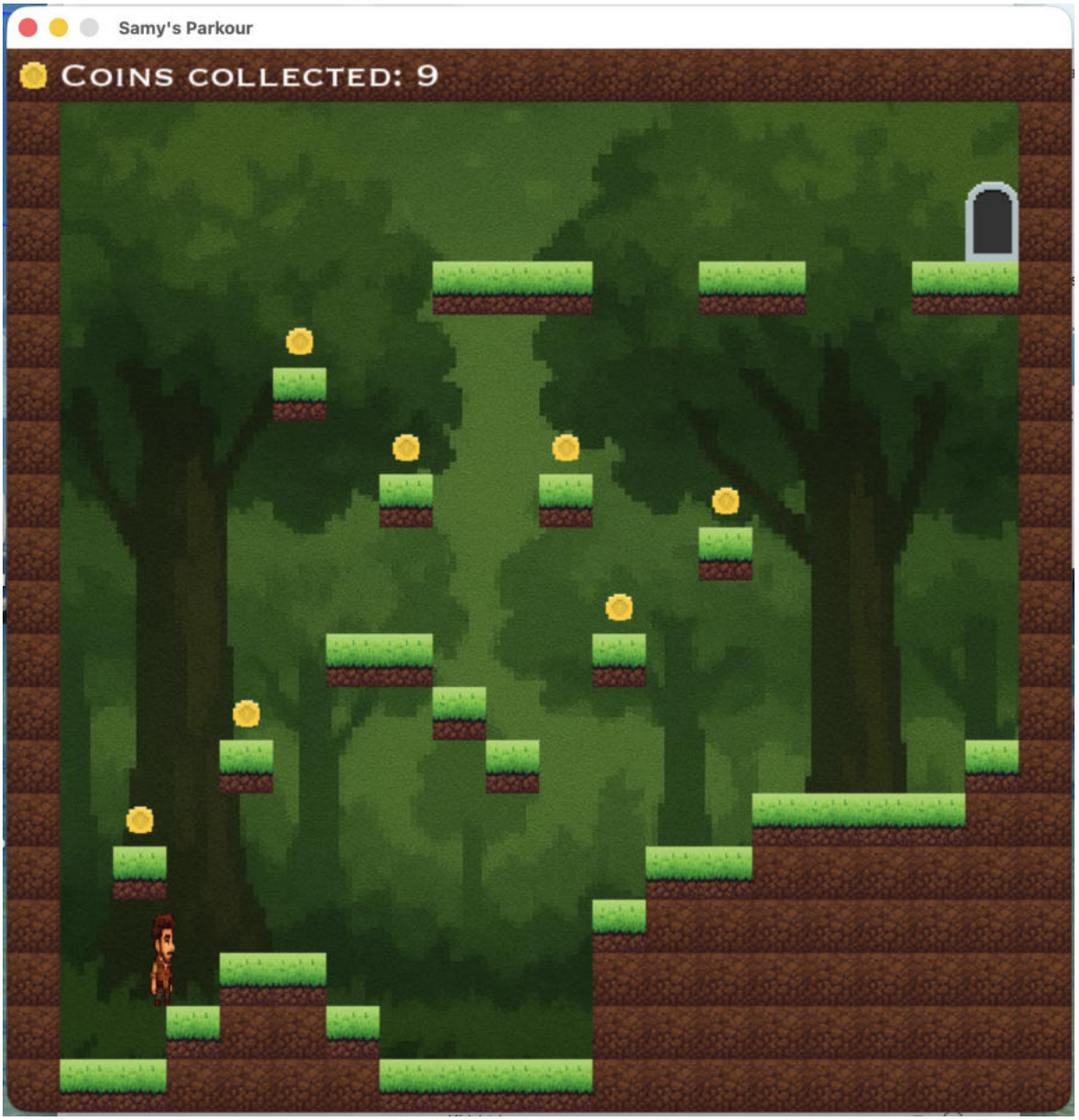
Level 2



Level 3



Level 4



Level 5

Reflection

What Works Well

- Clear separation between the menu logic, gameplay logic, and level data.
- Consistent physics for the character's jump and gravity.
- Consistent collision handling.
- Smooth animation handling with frame counters.

Challenges

- Collision handling became complicated with the moving platforms.
- Physics tuning for the perfect jump and gravity pull down.
- Managing the game states.

Future Improvements

- Killing blob mechanics by stepping on the blob's head.
- Add more variety of monsters.
- Add power ups to purchase with coins collected.
- More levels.

AI Usage

For this final project, I used AI solely for game designs.

Player Character

I benefit from AI when making the player model. What I did was attach the following picture into ChatGPT's image generator,

And then followed up with the prompt "Please make this guy into a pixelated form with adventure clothes on. Make it in the same format of the pixelated character I've attached". I also attach an example picture of what I want the model to look like. Which was like the following picture,

The result of the picture is the following

Later on, I use a background remover to remove the black background from the picture.

To create the animation image for the character, I also use the same AI with the following prompt : “Please make a walking animation from the picture you just generated without changing anything (example : hair, colour, clothes, etc.)”. It then results in the following picture.

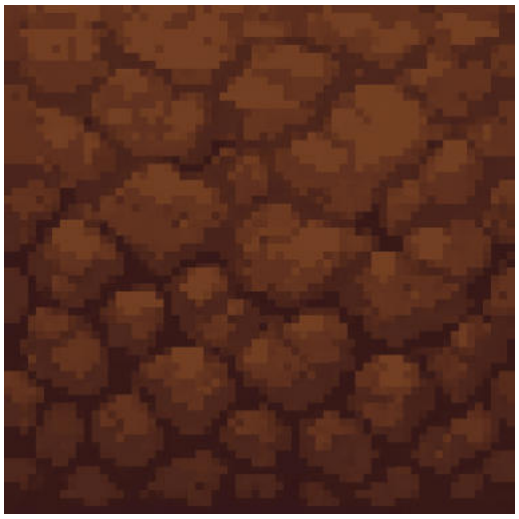
Next one, to create the walking animation with the left hand upfront, I tried with ChatGPT but somehow it just generated the same exact picture, so I moved onto another AI which was Gemini. With Gemini, I attached the walking picture I got recently, and put the following prompt : “Please make this guy walk with his left hand upfront and his left leg upfront. Keep in mind, don't change the model at all (example : hair, colour, clothes, etc.)”. It then results in the following picture.

Background and Dirt

I also benefit from AI when making the background and dirt tile for the game. What I did was attach the player character into ChatGPT's image generator, and proceeded with the following prompt : "I'm making a platformer game, can you please make a background for it that suits the character I've already attached. P.S. I prefer a forest type background but just make it somehow suits the character". It then results in the following picture.

After that, I also asked GPT for help to make the grass and dirt image. What I did was attach the background it just generated into ChatGPT's image generator, then followed up with the following prompt : "Please make a grass tile for a platformer game that suits the background I've attached.". It then results in the following picture.

After that, I went on to ask GPT for another help, which was a help to make the dirt image. What I did was attach the grass GPT just generated into ChatGPT's image generator, then followed up with the prompt : "Please make the dirt tile for a platformer game the same as the dirt in the grass image you just generated.". It then results in the following picture.



Conclusion

This project successfully demonstrates the development of a 2D platformer game using Python and Pygame. Samy's Parkour implements the core mechanics of platformer games, which are player movements, gravity, collision detection, enemy interaction, and also level progression.

References

Tutorial on how to make a platformer game
https://youtube.com/playlist?list=PLjcN1EyupaQnHM1I9SmiXfbT6aG4ezUvu&si=M7maxEA4aNM-v95j

Downloading assets

https://github.com/russs123/pygame-platformer-assets