Bhavin Baldota

**Super Mario Bros.**

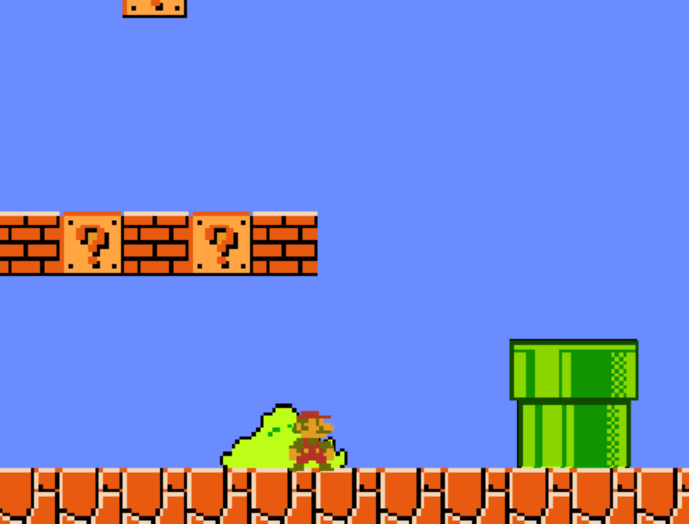
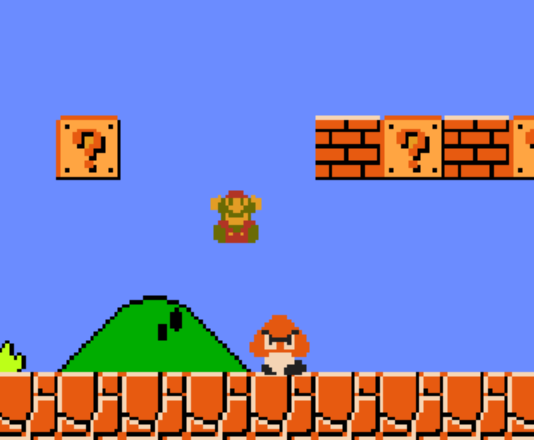


**Introduction:**

Super Mario Bros. is a classic video game that has captured the hearts and minds of gamers since its release in 1985. As a fan of the game, I decided to create my own remake using Python and Pygame. In this report, I will provide an overview of my remake, including its name, purpose, and unique features. I will also describe the development process, technical details, and testing and debugging procedures used in the creation of the game. Finally, I will discuss future improvements that can be made to the game.

**Development Process:**

The development of my Super Mario Bros. remake began with researching the original game mechanics and level design. I then began coding the game using Python and Pygame, starting with the game loop, player movement, and enemy behavior. I also created the game graphics using Adobe Photoshop.

One of the biggest challenges I faced during development was implementing the collision detection system. This required precise calculations and testing to ensure that the player could interact with the game environment in a realistic and intuitive way. To overcome this challenge, I consulted online resources and experimented with different collision detection algorithms until I found one that worked well for my game.

Another challenge was creating the levels for the game. I wanted to recreate the original levels from Super Mario Bros., but I also wanted to include some new elements to make the game more challenging and interesting. I spent a lot of time designing the levels, playtesting them, and adjusting them until they were just right.

Throughout the development process, I used version control software to keep track of my code changes and ensure that I could easily revert back to earlier versions if needed. I also frequently tested the game on different platforms to ensure that it was compatible with various operating systems.

Overall, the development process was a fun and rewarding experience, although it required a lot of hard work and problem-solving. I am proud of the final product and am excited to share it with others.

**Game Mechanics:**

My Super Mario Bros. remake currently includes the first four levels of the first world, and the objective of the player is to save the first princess. The player controls Mario, who can move left or right using the arrow keys, and can jump using the space bar. Mario must navigate through various obstacles and enemies to reach the end of each level.

The game features several types of enemies, including Goombas and Koopa Troopas. Goombas move back and forth on platforms and can be defeated by jumping on them. Koopa Troopas walk back and forth, and can be defeated by jumping on them or kicking their shells.

 A picture containing text, scoreboard

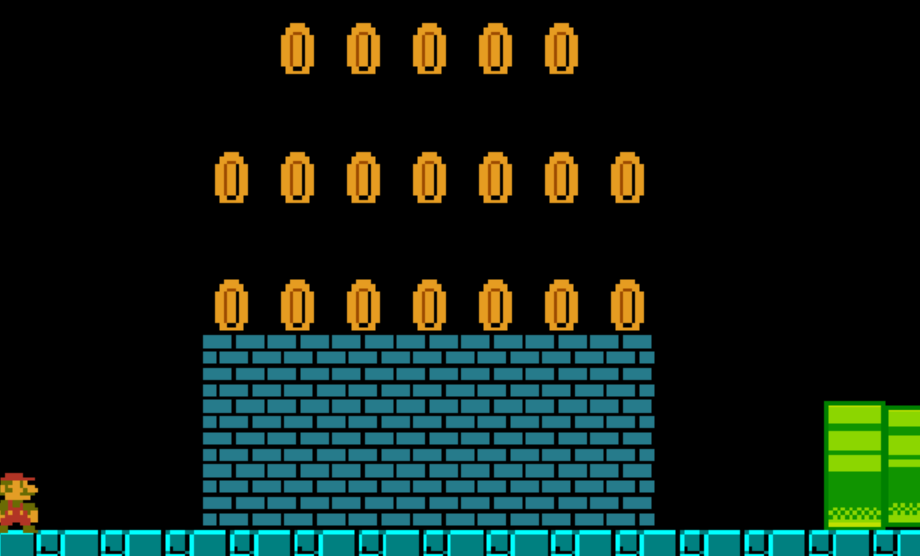
Description automatically generated A picture containing text

Description automatically generated

The game also features various obstacles, such as pits, spikes, and moving platforms. The pits can be crossed by jumping over them, while the spikes can be avoided by careful timing. The moving platforms require the player to jump on them and avoid falling off.

In addition to the enemies and obstacles, the game includes power-ups that Mario can collect to gain additional abilities. The mushroom power-up makes Mario larger and able to withstand an extra hit from enemies. The fire flower power-up allows Mario to shoot fireballs at enemies and defeat them from a distance.

To add a unique twist to the gameplay, I have included hidden blocks throughout the levels that contain coins or power-ups. These blocks can be found by hitting them from underneath, and can sometimes lead to secret areas or shortcuts.

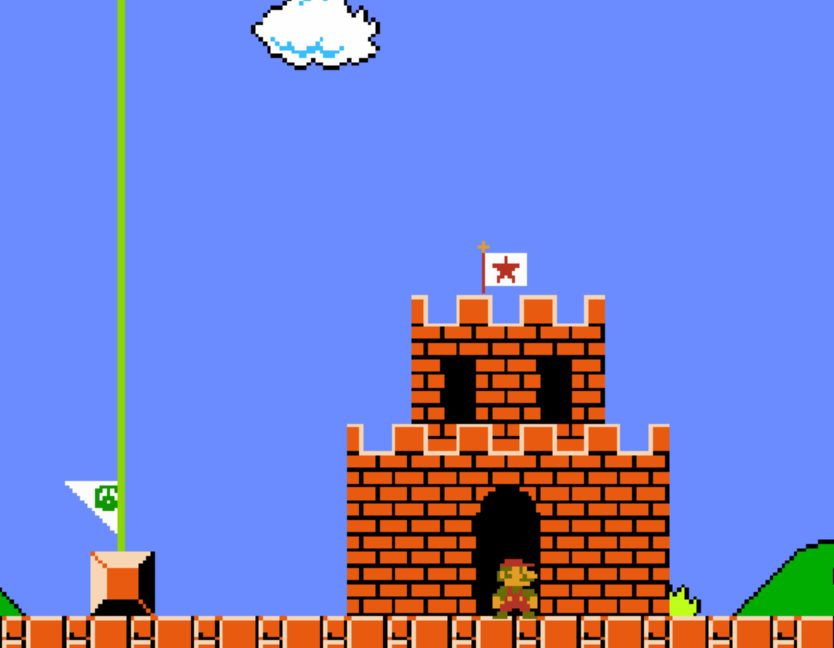


Overall, the game mechanics of the first four levels of my Super Mario Bros. remake are designed to be challenging and engaging, while staying true to the classic gameplay of the original game.

**Technical Details:**

My Super Mario Bros. remake was developed using Python and the Pygame library. Pygame is a set of Python modules designed for writing video games and multimedia applications.

The game consists of multiple Python scripts, each responsible for a specific aspect of the game. The main script, "main.py", handles the game loop and is responsible for updating the game state and rendering the graphics. Other scripts include "sprites.py", which defines the different game objects such as Mario, enemies, and power-ups, and "levels.py", which defines the layout and properties of each level.



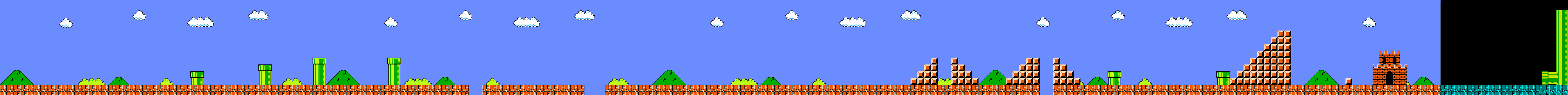
The game uses a tile-based system to construct the levels. Each level is defined using a two-dimensional array of integers, where each integer represents a specific tile type. For example, the number 0 represents an empty space, 1 represents a solid block, and 2 represents a question mark block.

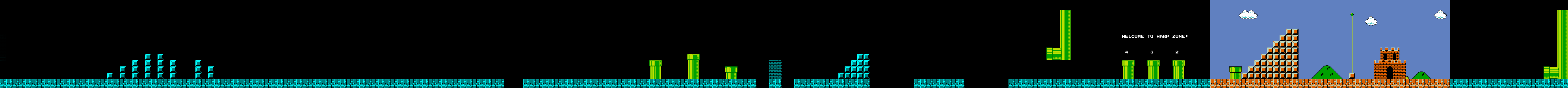
The game also includes collision detection to ensure that Mario and the enemies interact with the game objects correctly. When Mario collides with an enemy or an obstacle, the game checks whether he should be killed or if he should bounce back. Similarly, when an enemy collides with a solid block or a pit, the game determines how it should react.

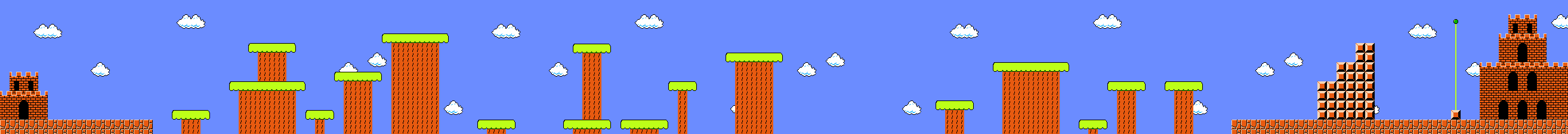
To improve performance and reduce the number of sprites that need to be drawn each frame, the game uses a technique called sprite batching. This involves grouping similar sprites together into a single image, which can then be drawn with a single draw call.

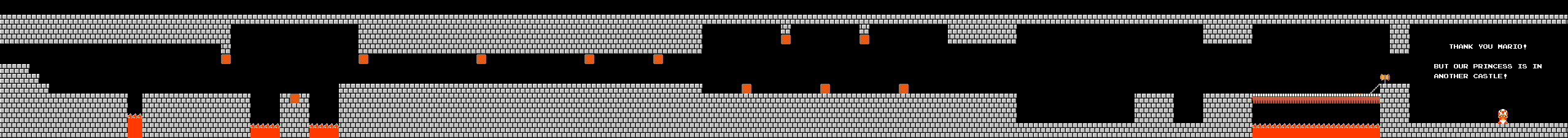
Overall, the technical details of my Super Mario Bros. remake demonstrate the use of various programming techniques and libraries to create an engaging and functional game.

Level 1:



Level 2:  


Level 3:

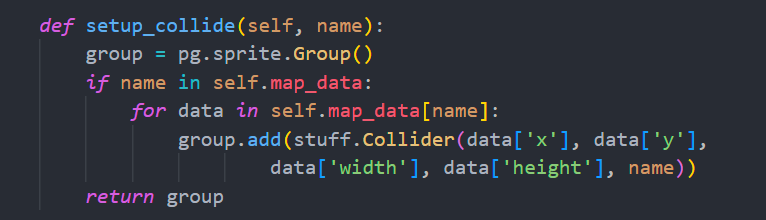
Level 4:  


**Testing and Debugging:**

During the development of my Super Mario Bros. remake, I conducted extensive testing and debugging to ensure that the game was functional and free of bugs.

I started by testing each component of the game individually, such as the movement of Mario, the behavior of enemies, and the collision detection system. I also tested each level to ensure that they were playable and challenging, but not too difficult or frustrating.

To assist with testing and debugging, I used various debugging tools and techniques, such as printing debug statements to the console, using breakpoints to step through the code, and using the Pygame built-in event logger to track user input.

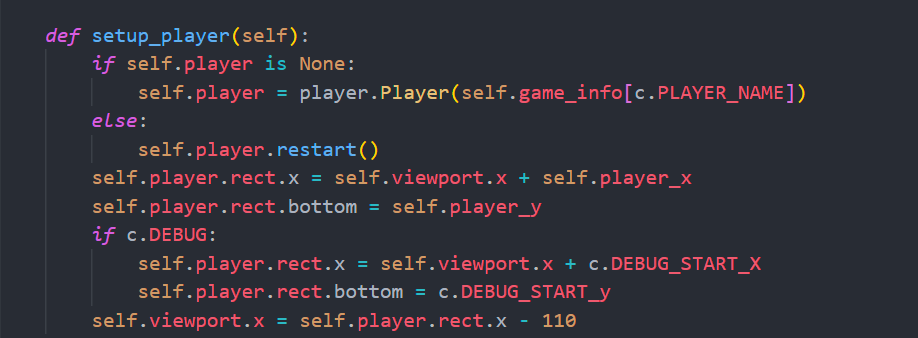
 A screenshot of a computer

Description automatically generated with medium confidence

A screenshot of a computer

Description automatically generated with medium confidence Text

Description automatically generated

 Text

Description automatically generated

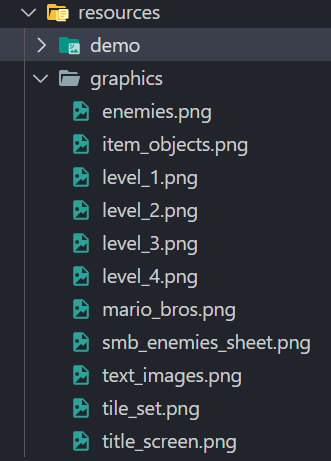
Throughout the development process, I encountered several bugs and issues, such as incorrect sprite positioning, inaccurate collision detection, and unexpected behaviors of enemies. To fix these issues, I used a combination of trial and error, logic, and online resources such as forums and documentation.

Once I had fixed all known bugs and issues, I conducted a final round of testing to ensure that the game was stable and free of crashes. I also tested the game on different systems and screen resolutions to ensure compatibility.

Overall, the testing and debugging process was crucial to ensuring that my Super Mario Bros. remake was a functional and enjoyable game. By testing each component individually and conducting multiple rounds of testing, I was able to identify and fix bugs and issues that would have otherwise affected the gameplay experience.

Text

Description automatically generatedText

Description automatically generated with medium confidence

Text

Description automatically generated with medium confidence

**Future Improvements:**

While my Super Mario Bros. remake is currently functional and enjoyable, there are several areas that could be improved to enhance the gameplay experience:

More Levels: Currently, the game only includes the first four levels of the first world. Adding more levels would increase the game's replayability and provide more challenges for the player.

More Power-Ups: In addition to the mushroom and fire flower power-ups, adding more power-ups such as the star or the cape feather would add more variety to the gameplay and allow for new strategies.

Improved AI: The behavior of enemies in the game is currently quite simple. Improving the AI of the enemies to make them more challenging and dynamic would increase the game's difficulty and provide a greater sense of satisfaction when defeating them.

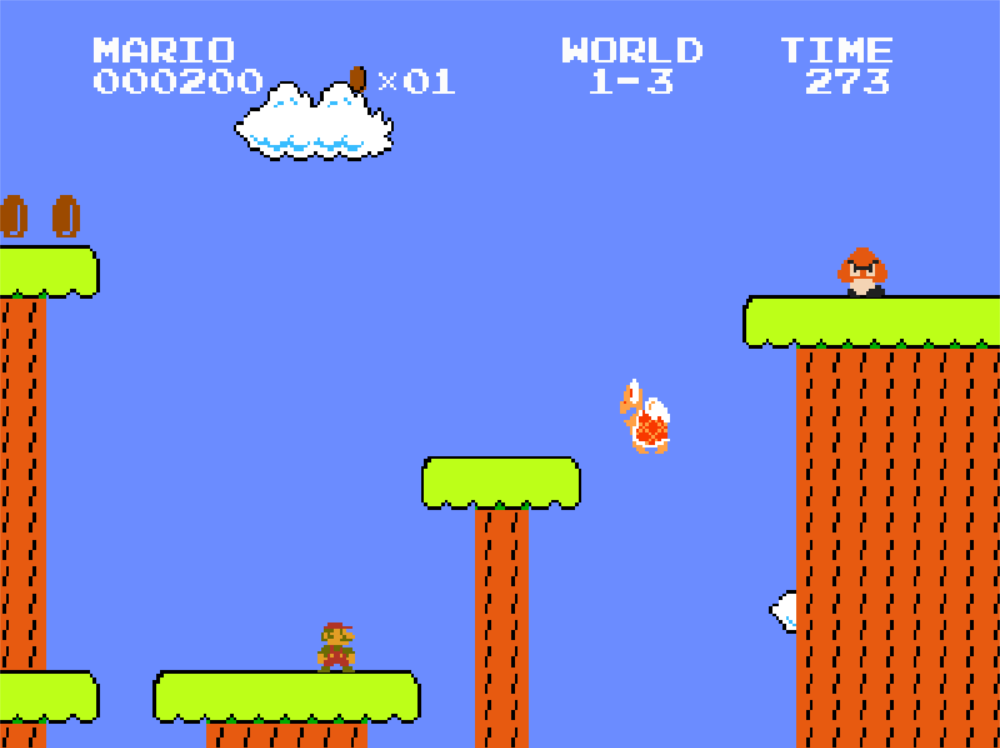
Multiplayer Mode: Adding a multiplayer mode would allow for players to compete against each other or work together to complete levels. This would add a new dimension to the gameplay and increase the game's social appeal.

Sound Effects and Music: Currently, the game has limited sound effects and no music. Adding appropriate sound effects and music would greatly enhance the immersion of the player and add to the overall atmosphere of the game.

Improved Graphics: While the current graphics of the game are functional, improving the graphics and visual effects would make the game more appealing and engaging to players.

Level Editor: Adding a level editor would allow players to create and share their own levels, increasing the game's replayability and fostering a sense of community among players.

Overall, there are many areas where my Super Mario Bros. remake could be improved. By adding new features, improving existing ones, and listening to feedback from players, the game could continue to evolve and remain relevant and enjoyable for years to come.

 A screenshot of a video game

Description automatically generated with medium confidence

**Conclusion:**

In conclusion, the Super Mario Bros. remake that I created using Python and Pygame is a functional and enjoyable game that stays true to the original game while adding some new features and mechanics. The game mechanics include moving Mario, jumping, collecting power-ups, and defeating enemies.

Throughout the development process, I encountered several challenges and opportunities for learning, such as improving my knowledge of object-oriented programming and implementing collision detection algorithms.

I conducted extensive testing and debugging to ensure that the game was free of bugs and playable. Additionally, I have identified several areas where the game could be improved, such as adding more levels, power-ups, and a multiplayer mode.

Overall, the experience of creating this game was rewarding and allowed me to develop my programming skills and understanding of game development. I hope that players enjoy playing the game and that my report provides some insight into the development process and potential future improvements.