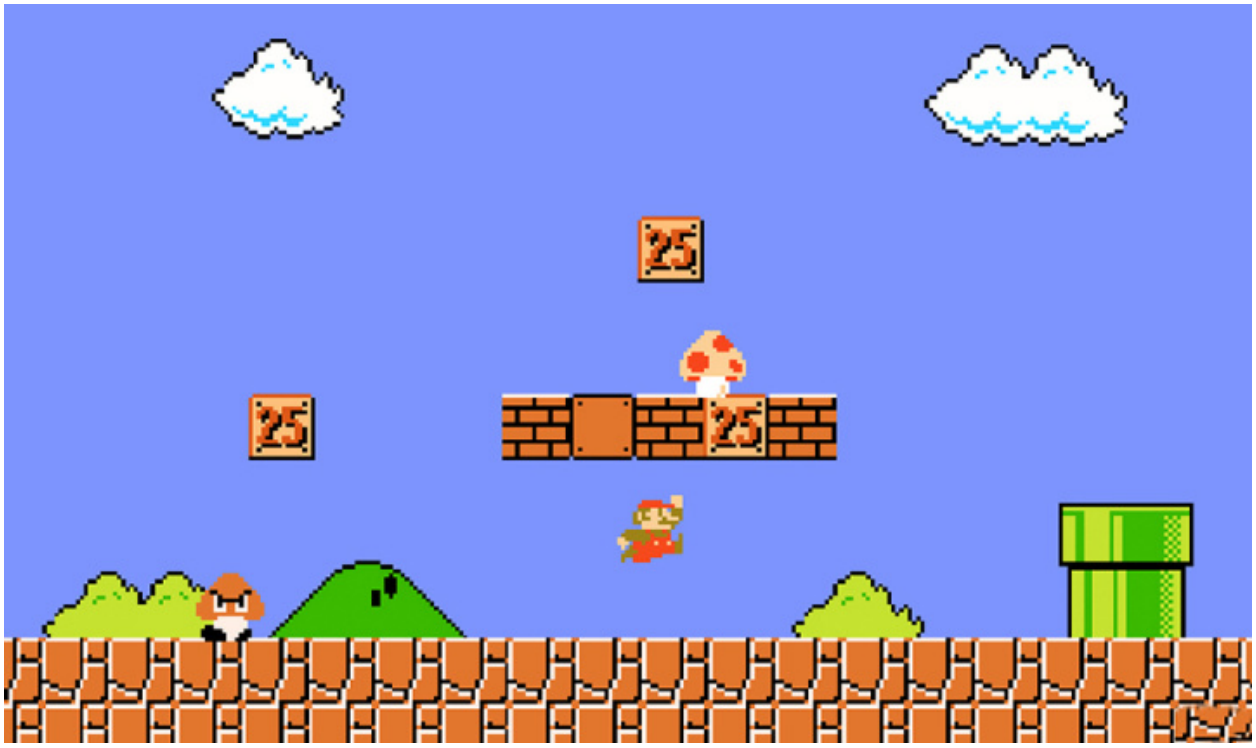
Super Mario Design Document

Groupe members: Hangpu Cao (hc3346), Zeqi Li (zl3202), Shen Gao (sg4140), Han Yang (hy2759), Zhiyuan Liu (zl3208)



Overview

Our project goal is to construct a Mario-like 2D side-scroller game. We aim to attempt the reconstruction of core game features including 2D map scrolling, item acquisitions, and enemy interactions with the means of a collision detection algorithm.

Design Outline

- 2D Side-scrolling
    - The character should stay in the center of the screen while scrolling
- Hit Box Detection
    - The character should be blocked by the ground & walls. Enemies and coins should have the proper interaction when hitting the character.

- Audio Output
    - The game should have BGM and audio effects (jump, enemy killed, coins)
- Animation
    - Add score animation, death animation, kill animation, and impact animation. Complete the character's interaction with terrain and enemies.

I/O Device

- Video Output: VGA
- Audio Output: 3.5mm audio jack
- Controller Input: Keyboard

Milestones

- Implement of the use of keyboard control Mario’s movement, including jumping and moving forward and backward.
- Map design & 2D map scrolling.
- Animation & Audio; Test and debug the game.

Workflow

Project Setup

1.1. Set up development environment: Install necessary software tools, libraries, and FPGA development tools for the DE1-SOC platform.

1.2. Familiarize with FPGA DE1-SOC documentation and tools: Understand the FPGA DE1-SOC's capabilities, constraints, and recommended practices.

1.3. Establish team roles and responsibilities: Assign specific tasks to team members based on their expertise and interests.

Hardware Configuration

2.1. Configure VGA video output: Set up the FPGA to generate the correct signals for VGA display and implement the necessary hardware interfaces.

2.2. Configure 3.5mm audio jack output: Set up the audio output using the DE1-SOC's audio codec and implement the necessary hardware interfaces.

2.3. Configure keyboard controller input: Implement the required hardware interfaces to read keyboard input from a PS/2 or USB port.

Game Design

3.1. Develop game concept and objectives: Define the game's overall theme, gameplay mechanics, and goals for the player.

3.2. Design game levels and maps: Create the layout for each level, including terrain, obstacles, and enemy placements.

3.3. Define game characters, enemies, and items: Design the visual appearance and behavior of the game's characters, enemies, and power-ups.

3.4. Plan game scoring and progression system: Determine how players earn points, advance through levels, and achieve victory.

Software Development

4.1. Implement core game engine:

> 4.1.1. Implement 2D map scrolling: The logic that moves the map in response to character movement, updating the display accordingly.
>
> 4.1.2. Implement item acquisitions and power-ups: The logic for item pickups, their effects on the player character, and their durations.
>
> 4.1.3. Implement enemy interactions: Implement enemy movement, AI behavior, and consequences of player-enemy collisions.

4.2. Implement collision detection algorithm: Create an efficient algorithm to detect collisions between game objects, such as the player, enemies, and items.

4.3. Create character animations and graphics: Design and implement sprites for the player character, enemies, and items, along with their respective animations.

4.4. Develop audio assets and sound effects: Create background music, sound effects for character actions, and other audio elements to enhance the game experience.

4.5. Integrate keyboard input for character control:

> 4.5.1. Implement character movement: Program the player character's movement (left and right) based on keyboard input.
>
> 4.5.2. Implement character jumping: Program the player character's jumping mechanics, including jump height and duration, based on keyboard input.

4.6. Implement game scoring and progression system: Program the logic for calculating player scores, saving progress, and moving between levels.

4.7. Create game menu and user interface: Design and implement the game's menu system,

including options for starting a new game, pausing, and adjusting settings.

Testing and Debugging

5.1. Test individual components and features: Verify that each component, such as map scrolling, collision detection, and item pickups, works as intended.

5.2. Perform integration testing: Test the game as a whole, ensuring that all components work together seamlessly.

5.3. Test game performance and optimize code: Monitor the game's performance on the FPGA DE

(Alternative)

5.1. Test individual components and features: Verify the functionality of each component in isolation.

5.2. Perform integration testing: Test the combined functionality of all components to ensure proper interaction and compatibility.

5.3. Test game performance and optimize code: Analyze the game's performance, identify bottlenecks, and optimize the code to improve performance.

5.4. Debug any issues discovered during testing: Identify and fix any problems found during testing to ensure a smooth gameplay experience.

Finalization

6.1. Perform final playtesting and gather feedback: Conduct thorough playtesting with team members and external testers to gather feedback on the gameplay experience, identify any remaining issues, and collect suggestions for improvement.

6.2. Make necessary adjustments based on feedback: Implement changes to the game based on the feedback received during playtesting to improve the overall quality and user experience.

6.3. Complete documentation: Write comprehensive documentation for the project, including a user manual, technical documentation, and any other necessary materials.

6.4. Package and release the game: Prepare the game for distribution by creating a final build, packaging all necessary files, and releasing the game to the intended audience.

Project Schematic: