Assembly Project: Dr Mario

Ahnaf Keenan Ardhito
Ibrahim Bilal

March 24, 2025

1 Instruction and Summary

1. Which milestones were implemented?
   (a) Milestone 1
   (b) Milestone 2
   (c) Milestone 3
   (d) Milestone 4 and 5 :
      i. Easy:
         A. Implement gravity, so that each second that passes will automatically move the capsule down one row.
         B. Assuming that gravity has been implemented, have the speed of gravity increase gradually over time, or after the player completes a certain number of rows
         C. When the player has reached the "game over" condition, display a Game Over screen in pixels on the screen. Restart the game if a "retry" option is chosen by the player. Retry should start a brand new game (no state is retained from previous attempts).
         D. Add sound effects for different conditions like rotating and dropping capsules, removing a row of squares, for beating a level and the game over condition.
         E. If the player presses the keyboard key p, display a "Paused" message on screen until they press p a second time, at which point the original game will resume.
         F. Add levels to the game that trigger after the player eliminates all of the viruses in the current level. The next level should be more difficult than the previous one (i.e. speed and number of viruses). Make sure that the increased difficulty is different from the Easy/Medium/Hard difficulty in some way (the same level can't count for two features).
         G. Show an outline where the capsule will end up if you drop it.
         H. Have a panel on the side that displays a preview of the next capsule that will appear.
         I. Draw Dr. Mario and the viruses on the side panels, as in Figure 2.1
         J. Assuming you've drawn the viruses from the previous feature, have each virus image disappear as the viruses of that colour are eliminated from the playing field (play the game if you're unclear on what this means).
      ii. Hard:
         A. Play the Dr. Mario's theme music in the background while playing the game.
2. How to view the game:
   (a) Unit width in pixels: 2
   (b) Unit height in pixels: 2

(c) Display width in pixels: 64

(d) Display height in pixels: 64

(e) Base Address for Display: 0x10008000

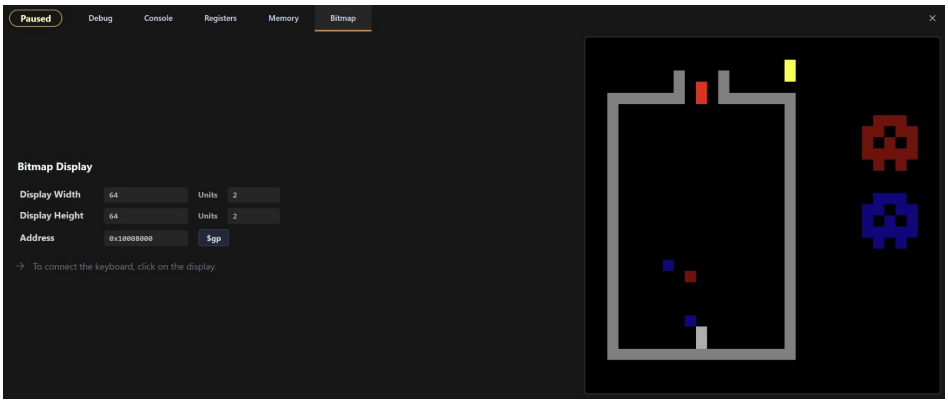

Figure 1: How the game looks running on Saturn

3. Game Summary:

- This game is our attempt on recreating the famous classic Dr. Mario game, implemented entirely in MIPS assembly.
- The player controls falling capsules to align matching colors and eliminate viruses.
- Controls are as follows:
  - **W** - Rotate capsule
  - **A** - Move capsule left
  - **D** - Move capsule right
  - **S** - Instantly drop capsule
  - **P** - Pause the game
  - **Q** - Quit the game
  - **R** - Reset the game
- We successfully implemented the original "Fever" background music from Dr. Mario to enhance the gameplay experience.
- The game features infinite level progression — each new level adds one virus and increases the capsule fall speed (gravity).
- The game continues until the player loses, which occurs when capsules block the neck of the medicine bottle (top of the play area).

2 Attribution Table

Ahnaf Keenan A. 1010236219	**Ibrahim Bilal, 1010437270**
Implemented drawing the medicine bottle	Generated capsules with randomized colors
Implemented left-right movement and gravity	Enhanced movement and added rotation (including instant drop)
Implemented match checking (4+ same-color in a row)	Applied gravity only to unsupported capsules after clearing
Implemented virus drawing with randomized rows and columns spawn points	Improved virus rendering using color variation + normalization for match detection
Implemented win detection (progresses to next level when viruses are cleared)	Implemented Pause and Game Over screens
Added background theme song	Implemented virus display in the sidebar
Added sound effects to user movements (rotate, place down, win)	Implemented capsule preview before it drops

Theme Song Integration

One of the most challenging and time-consuming parts of the extra features of our project was the creation and integration of the Dr. Mario theme song and sound effects. Rather than using a pre-existing MIDI file or music engine, we chose to manually transcribe the song into MIPS Assembly, note by note, bar by bar.

This required:

- Listening closely to the original theme and identifying each note's pitch and rhythm.
- Translating each note into a four-field representation: `<Pitch, Duration, Instrument, Volume>`.
- Matching the tempo and structure to fit the pace of the game while maintaining musical accuracy.
- Adding bass lines and harmony using a second instrument channel, further enhancing the audio quality.
- Ensuring that the music playback was synchronized with the game loop using carefully tuned delays and counters.

We implemented a tick-based playback system using a dedicated timer that triggers a note read and playback syscall every fixed number of frames. The song data is stored as a series of `.word` instructions, and a note index is incremented each time a new note is played. Volume, instrument, and pitch changes are all dynamically handled. Nonetheless, this feature gives the game a polished and nostalgic touch, adding to its charm and authenticity. It's one of the components we're most proud of in this project.