Mario Bros
Classic

Isabelle Chong
Jose Guajardo
Nancy Hidalgo
What is Mario Bros Classic?
System Block Diagram

- **IMU Controller Block**
  - Converts IMU acceleration data into a left, right and jump movement signal to control player movement

- **Game Mechanics Block**
  - Mario & Goomba Movement
  - Collision Detection
  - Game State FSM

- **Video and Display Block**
  - CoE and ROM for Sprites
  - Background/Map ROM
  - Side-scrolling Module
  - XVGA

- **Clock**
- **Reset**
- **Calibrate**

Output: VGA Out
Our Implementation - Terminology
Our Implementation - Terminology
Our Implementation - Terminology

Left edge

Right edge = Left edge + 1024
Our Implementation - Terminology

For each object (cloud, platform, hole), Map ROM will store:

\{abs\_loc\_x, abs\_loc\_y, width, height, sprite\_id\}
Different ROMs

- Location ROM
- Sprite ROM
- Background ROMs
Game Mechanics

**Button Inputs**
- Pause
- Reset

**IMU Inputs**
- Up
- Right
- Left

Game State FSM
- Based on user inputs, Mario's position, etc., outputs a state (Game play, pause, game over, reset).
- Keeps track of metrics

Collision Detection
- Based on ALL Sprites position determines collisions, and updates the game FSM and the movement modules
- Mario and Goomba position
- Collision type

Movement
- Updates Mario's relative position based on user inputs and collision with other sprites

You died
- To Sprite Pipeline

You won
- To Sprite Pipeline
Video and Graphics Block

**Sprite Pipeline:**
1. Updates a counter that keeps track of which object is being rendered (updated as soon as `sprite_complete` goes high)
2. Contains logic to determine addresses for Location ROM
3. Calculates relative position of objects depending on absolute positions and camera location

**Object Generation FSM Module**
- Generates Sprite address depending on sprite ID, rel. location, width, height, and h/v count. Once the maximum Sprite address is reached for that item, send out a `sprite_complete` flag.

**Background**
- Updates background depending on Mario's velocity

**Enable Object**
- Updates an enable array with objects that have absolute position in camera

**Location ROM**
- Stores information about each object in the map/level.
- Inputs: `obj_enable_array`, `read_address` (counter), `abs. location width, height`, `sprite ID`, `valid`

**Mario**
- Includes 8-bit Mario generation
- Inputs: `Mario position & velocity`, `h/v count`, `pixel`

**Sprite ROM**
- Stores pixel values for objects in the map. Sprite address depends on the sprite ID, which describes the type of object (cloud, platform, hole)

**Background ROM**
- Stores pixel values for one column of the background (including ground + sky)
Enable Object and Location ROM Modules

Sprite Pipeline:
1. Updates a counter that keeps track of which object is being rendered (updated as soon as sprite_complete goes high)
2. Contains logic to determine addresses for Location ROM
3. Calculates relative position of objects depending on absolute positions and camera location

Object Generation FSM Module
Generates Sprite address depending on sprite ID, rel. location, width, height, and h/v count. Once the maximum Sprite address is reached for that item, send out a sprite_complete flag.

Background
Updates background depending on Mario’s velocity

Sprite ROM
Stores pixel values for objects in the map. Sprite address depends on the sprite ID, which describes the type of object (cloud, platform, hole)

Background ROM
Stores pixel values for one column of the background (including ground + sky)
Which objects should be displayed, and where?

Left edge

Right edge = Left edge + 1024

**Relative loc.**

**Absolute loc.**

Enable Object

- Updates an enable array with objects that have absolute position in camera

Location ROM

- Stores information about each object in the map/level.
  - abs. location width, height
  - sprite ID
  - valid

Camera position

Obj enable array

Read enable address (counter)
What if an object is moving due to side-scrolling?
Object and Sprite Generation Modules

**Object and Sprite Generation Modules**

**Object Generation FSM Module**
- Generates Sprite address depending on sprite ID, rel. location, width, height, and h/v count. Once the maximum Sprite address is reached for that item, send out a sprite_complete flag.

**Sprite ROM**
- Stores pixel values for objects in the map. Sprite address depends on the sprite ID, which describes the type of object (cloud, platform, hole).

**Background**
- Updates background depending on Mario’s velocity.

**Background ROM**
- Stores pixel values for one column of the background (including ground + sky).

**Object and Sprite Generation Modules**

**Enable Object**
- Updates an enable array with objects that have absolute position in camera.

**Location ROM**
- Stores information about each object in the map/level.

**Mario**
- Includes 8-bit Mario generation.

**Sprite Pipeline**
- 1. Updates a counter that keeps track of which object is being rendered (sprite_complete goes high)
- 2. Contains logic to determine addresses for Location ROM
- 3. Calculates relative position of objects depending on absolute positions and camera location

**Background**
- Updates background depending on Mario’s velocity.

**Background ROM**
- Stores pixel values for one column of the background (including ground + sky).
Accessing and Generating Sprites

- **Idle**: Waiting for valid signal to go high
- **Generating Sprite**: Calculates Sprite address. Sends address to Sprite ROM
- **Sprite Complete**: Communicates with Sprite Pipeline to render next object’s sprite
Saving Sprites as Logic and in Memory

- 8-bit sprite drawing modules
  - Mario
  - Goomba
  - Floor gaps

- COE files
  - Platforms
  - Pipes
  - Clouds and bushes
  - Gameboy background interface
Background Generation Modules

Enable Object
- Updates an enable array with objects that have absolute position in camera

Location ROM
- Stores information about each object in the map/level.

Mario
- Includes 8-bit Mario generation

Sprite Pipeline:
1. Updates a counter that keeps track of which object is being rendered (updated as soon as sprite_complete goes high)
2. Contains logic to determine addresses for Location ROM
3. Calculates relative position of objects depending on absolute position and camera location

Object Generation FSM Module
- Generates Sprite address depending on sprite ID, rel. location, width, height, and h/v count. Once the maximum Sprite address is reached for that item, send out a sprite_complete flag.

Sprite ROM
- Stores pixel values for objects in the map. Sprite address depends on the sprite ID, which describes the type of object (cloud, platform, hole)

Background
- Updates background depending on Mario's velocity

Background ROM
- Stores pixel values for one column of the background (including ground + sky)
Background Generation

- Display objects that are always on screen
  - Sky and floor in game environment background
  - Gameboy interface from COE
    - Saves memory by making gameplay area smaller
    - Enhances user experience- buttons on interface will be overlaid with blob sprites that change color to appear “pressed”
**IMU Controller**

- Similar to implementation used in lab 5b - interface with Teensy
- Tilting in the x direction will determine if Mario moves forward or backward
- Tilting in the y direction determines if Mario jumps
## Goals

<table>
<thead>
<tr>
<th>Commitment</th>
<th>Baseline</th>
<th>Stretch</th>
</tr>
</thead>
<tbody>
<tr>
<td>Working game with button inputs</td>
<td>Working game with IMU controller</td>
<td>Pick out of the following:</td>
</tr>
<tr>
<td>One direction of movement and side scrolling and jumping</td>
<td>Two-directional movement</td>
<td>IMU Speed</td>
</tr>
<tr>
<td>End game flag movement</td>
<td>Add a Star Coin</td>
<td>Player Avatars</td>
</tr>
<tr>
<td></td>
<td>SD Card Audio</td>
<td>Two-Player Mode</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Wireless IMU Controller</td>
</tr>
<tr>
<td></td>
<td>Week 1</td>
<td>Week 2</td>
</tr>
<tr>
<td>-------</td>
<td>-------------------------------------------------</td>
<td>---------------------------------------------</td>
</tr>
<tr>
<td>Nancy</td>
<td>Create Game FSM and collision detection modules.</td>
<td>Create movement module</td>
</tr>
<tr>
<td>Jose</td>
<td>Create Enable Object Modules, Location ROM and Sprite Pipeline</td>
<td>Create Object Generation Module and integrate with Sprite Pipeline and Sprite ROM</td>
</tr>
<tr>
<td>Izzy</td>
<td>Generate COE files and bit art for game sprites and environment background</td>
<td>Implement IMU controller and Gameboy User Interface (GUI ;)</td>
</tr>
</tbody>
</table>
Questions