PHOTO: DARCY PETERKA

4840 Final Project: Forest Fire and Ice' Game

Yonghao Lin (yl5763), Zhenqi Li (zl3508), Yang Cao (yc4535)
Yifan Mao (ym3064), Weijie Wang (ww2739)

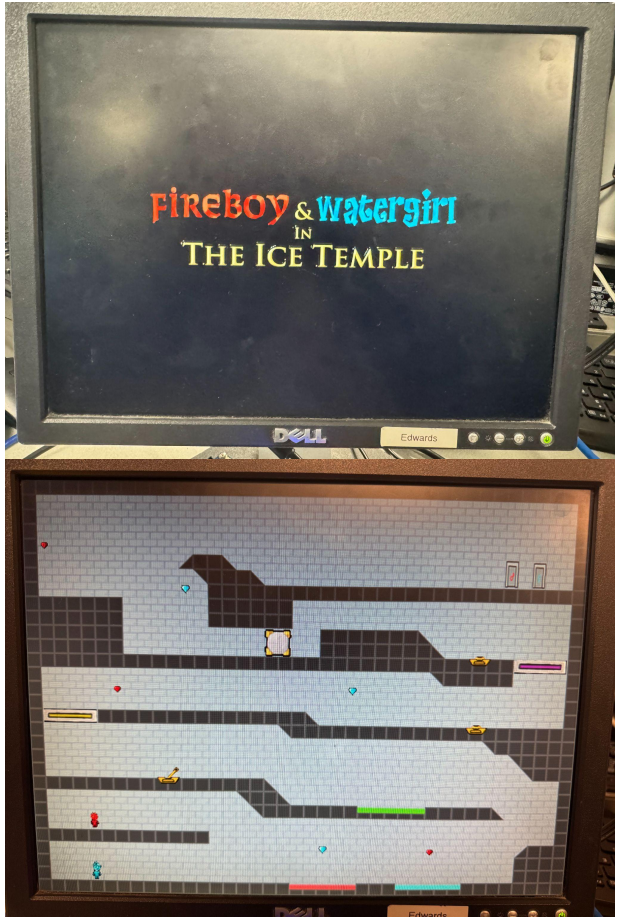
EMBEDDING SYSTEM

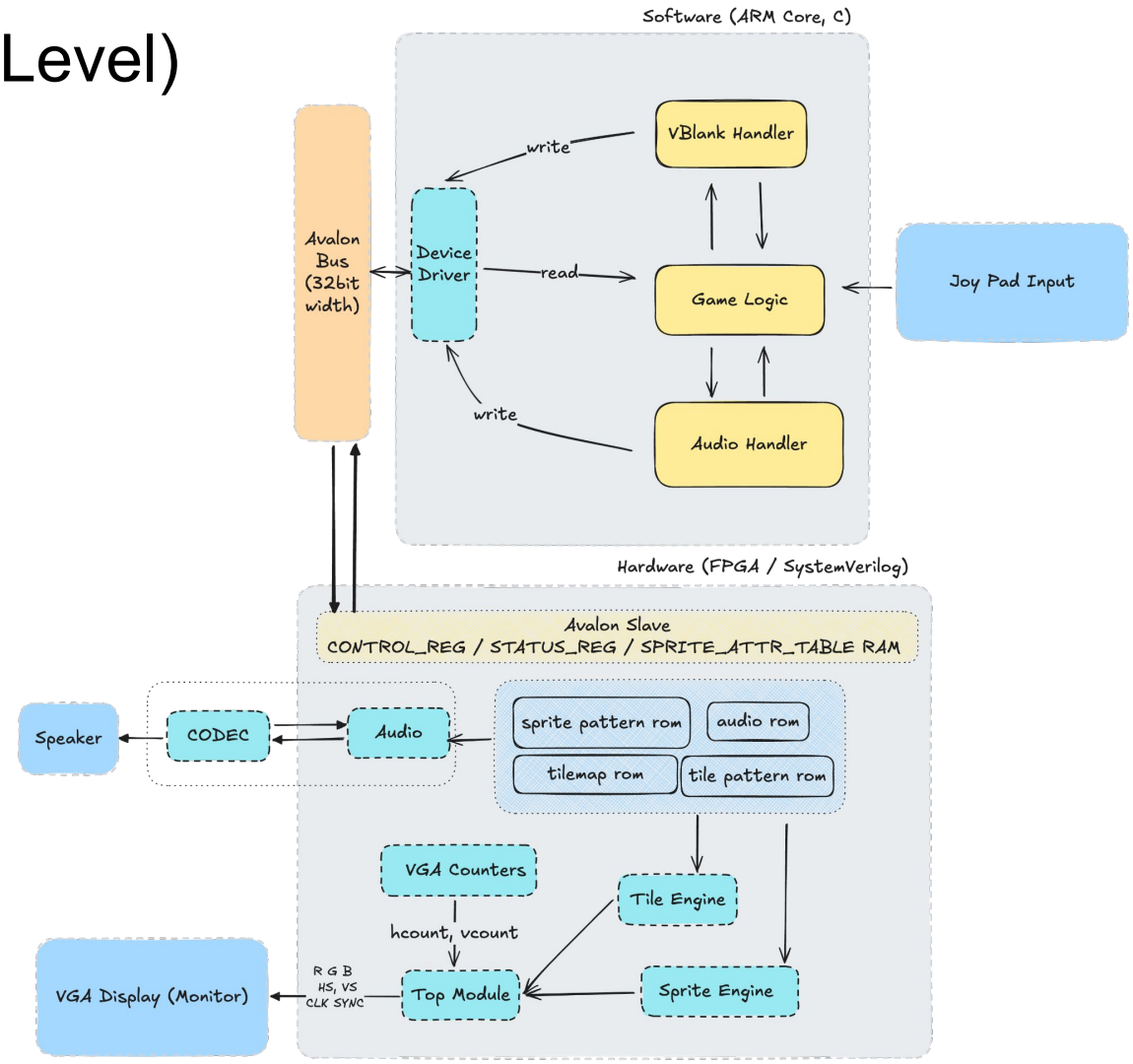
COLUMBIA | ENGINEERING
The Fu Foundation School of Engineering and Applied Science

Content

1. Overview Diagram
2. Hardware
3. Hardware and Software Interface
4. Resource Budget
5. Software
6. Demo

Overview Diagram (High Level)

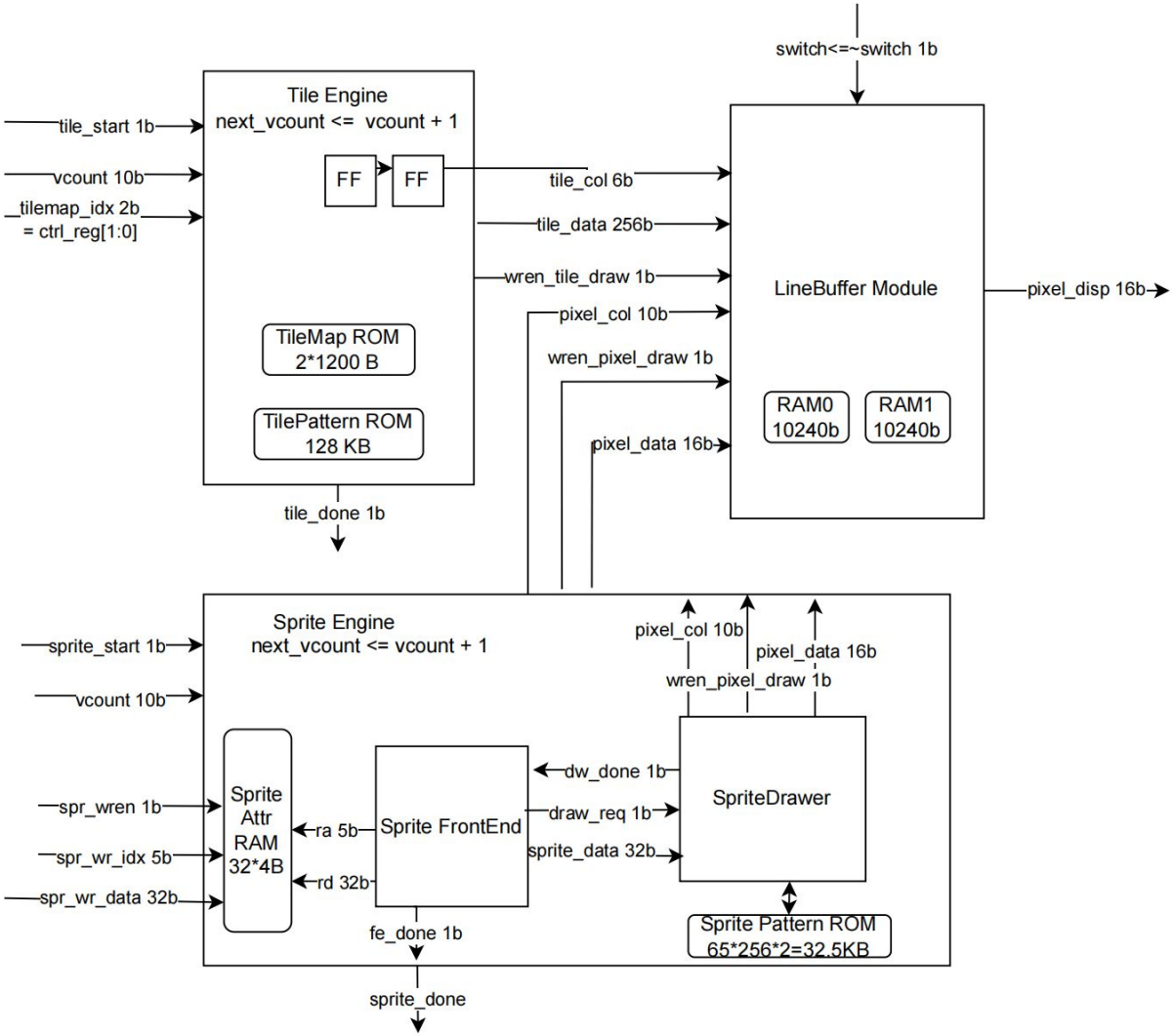




HW–VGA Display

VGA Display Overview

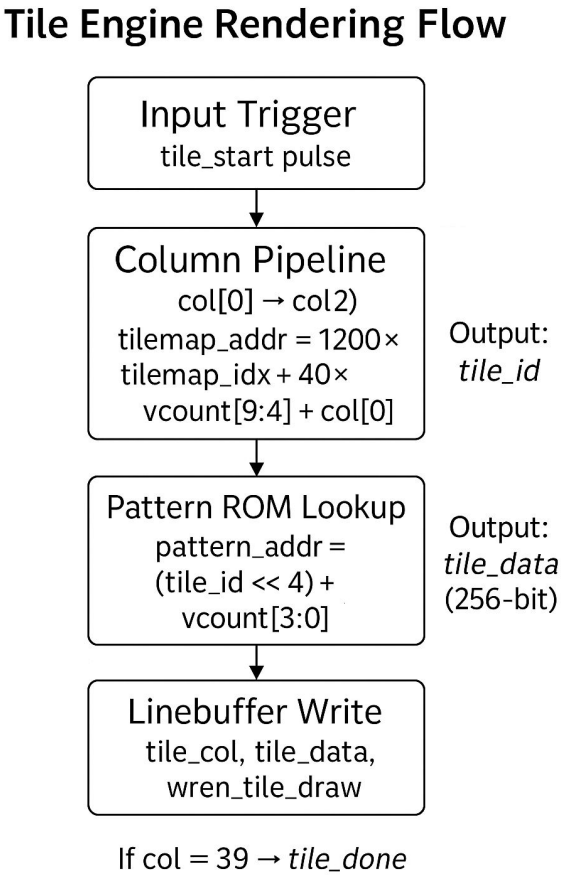
- `vga_top` is the main module for VGA output at 640×480 @ 60Hz.
- Uses a 50 MHz clock and Avalon bus for control.
- Internally connects:
  - `tile_engine` to draw background tiles (40 per line)
  - `sprite_engine` to overlay sprites with per-pixel transparency
- Pixel format is **T**RGB555 with the **T**/MSB as a transparency bit.
- A **dual-bank line buffer** allows rendering and displaying to happen in **parallel**.
- Each 5-bit RGB value is left-shifted to generate 8-bit VGA output



HW–Tile Engine

Tile Engine

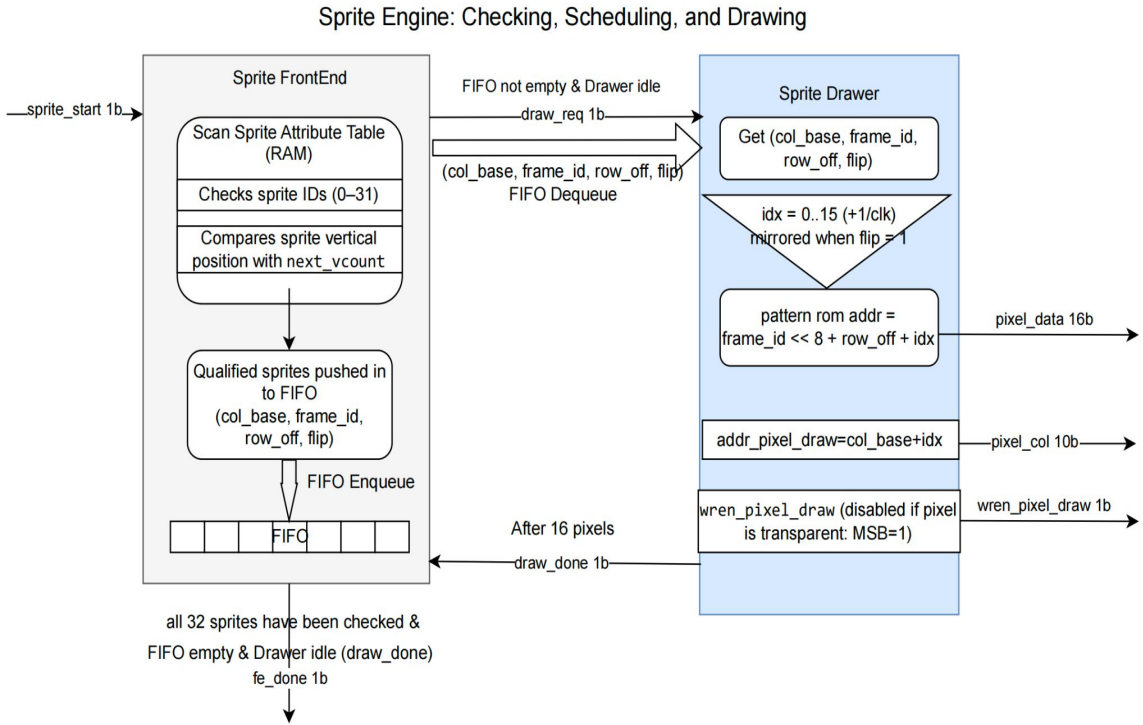
- Generates background layer: 40 tiles per line, 1 line of tile (16 pixels) per clk.
- Uses `tilemap_idx` and `next_vcount` to fetch tile tile IDs and tile pattern data.
- Outputs:
  - `tile_col` (0–39): tile col index
  - `tile_data` (256 bits): 16 **T**RGB555 pixels
  - `wren_tile_draw`: write enable for line buffer
- When `tile_col` 39 is done, `tile_done` tells sprite rendering can begin.
- All tile pixels are non-transparent (**T**/MSB = 0), so sprite frame pixel can transparently overlay them.



HW–Sprite Engine

Sprite Engine

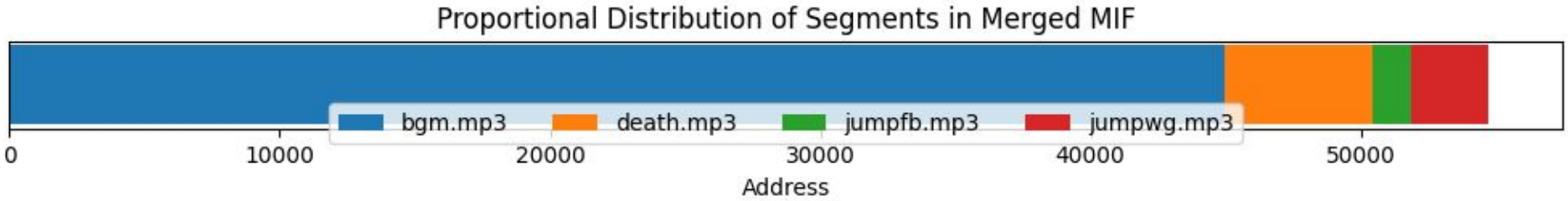
- `sprite_engine` draws the sprite layer over background tiles.
- Internally has:
  - `sprite_frontend`: checks 32 sprite entries each line, queues visible ones
  - `sprite_drawer`: draws 16-pixel-wide sprite rows from Pattern ROM
- Sprite attributes (32bit per sprite): `enable`, `x`, `y`, `frame_id`, `flip`
- Visible sprites in current drawing line are pushed to a FIFO; drawn in order (later = higher priority)
- Only non-transparent pixels (MSB = 0) are written to the linebuffer
- Exposes simple interface to `vga_top`: starts on `sprite_start`, signals done via `sprite_done`



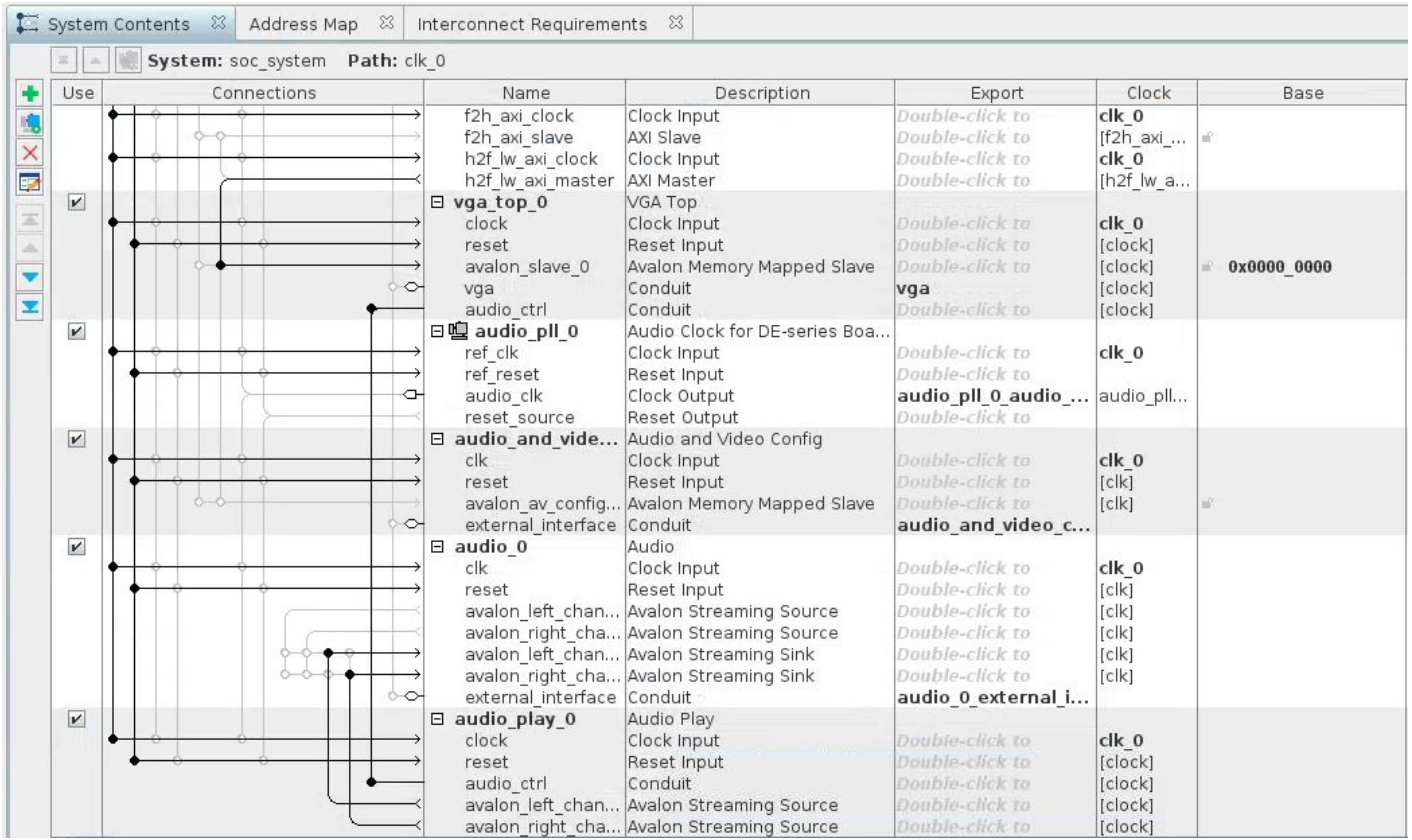
HW–Audio Play

Including Background Music & Sound Effects:

- Merged into a combined .mif file.
- Left Channel always plays BGM, Right Channel for Sound Effects (Plays BGM when it’s vacant).
- Transmit through CODEC.



HW–Audio Play



HW/SW Interface

Offset	Register	Description	Bits	Value Range	R/W
0x00	`CTRL_REG`	Control register (tilemap index, audio ctrl)	[31:0]	See bit field below	W
0x04	`STATUS_REG`	Current pixel column and row	[19:0]	[19:10]: col (0–639) [9:0]: row (0–479)	R
0x08–0x7F	Reserved	Reserved for future use	–	–	–
0x80–0xFF	`SPRITE_ATTR_TABLE[n]`	Sprite attribute table (32 entries, 4 bytes each)	[31:0]	See format below	W

Table 1: Register Map

HW/SW Interface–CTRL_REG Bit Field

CTRL_REG Bit Field Description

The CTRL_REG Bit field description is shown in Table 2.

Bits	Name	Description
[1:0]	`tilemap_idx`	Tilemap index (2 bits): selects one of 4 tilemaps (0–3)
[28:2]	Reserved	Unused, reserved for future use
[30:29]	`sfx_sel`	Sound effect selector: 00 = None, 01/10/11 = 3 sound effects
[31]	`bgm_en`	Background music enable: 1 = On, 0 = Off

Table 2: CTRL_REG Bit Field Description

HW/SW Interface–SPRITE_ATTR_TABLE RAM

SPRITE_ATTR_TABLE Format (Each Entry = 4 Bytes)

The SPRITE_attr_table format is shown in Table 3. Each entry at offset: `0x80 + (n × 4)`, where $n \in [0, 31]$:

Bits	Field	Description
[31]	`enable`	1 = visible, 0 = hidden
[30]	`flip`	1 = horizontally flipped
[29:27]	Reserved	Unused
[26:18]	`sprite_y`	Vertical position (0–479)
[17:8]	`sprite_x`	Horizontal position (0–639)
[7:0]	`frame_id`	Sprite frame index (0–255)

Table 3: SPRITE_ATTR_TABLE Entry Format

On-Chip Memory (BRAM) Usage

Component	Memory Type	Size Estimate	Description
Tile Maps (x2)	BRAM (ROM)	2.4 KB	2 static tilemaps, each 40×30 tiles (1 byte per tile)
Tile Pattern	BRAM (ROM)	69 KB	138 tiles, each 16×16 pixels, 16-bit color
Sprite Pattern	BRAM (ROM)	33 KB	Multiple sprite animation frames, 16-bit color per pixel (Details see Table 5)
Sprite Attribute Table	BRAM (RAM)	128 B	32 sprite attribute entries, each 4 bytes
Audio Sample ROM	BRAM (ROM)	54 KB	$\leq$7 seconds of mono 8kHz 8-bit audio
MMIO Control Registers	Register	negligible	CONTROL_REG, STATUS_REG, etc.
Total		**~158 KB**	Well within ~495 KB BRAM budget of the Cyclone V FPGA

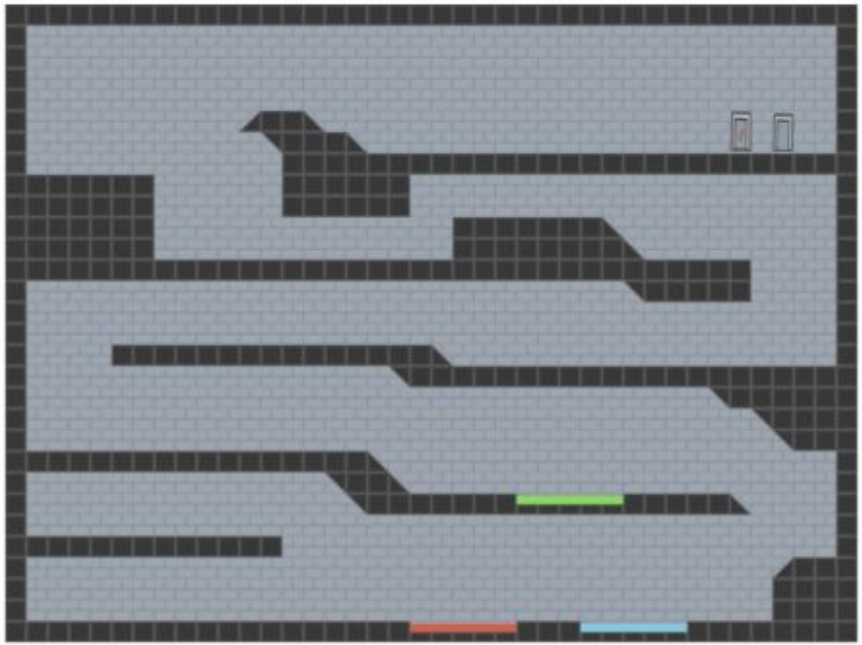
Tilemap background 1

Background 1

```
WIDTH=256;
DEPTH=2000;
ADDRESS_RADIX=HEX;
DATA_RADIX=HEX;
CONTENT BEGIN
```

125 tiles

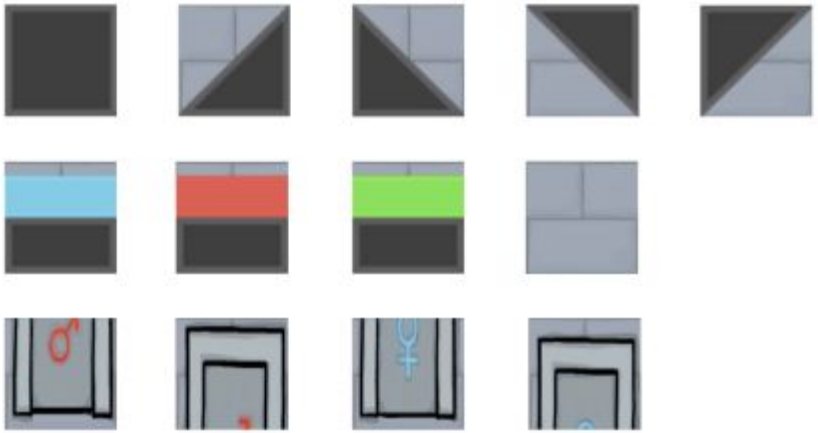
Tilemap background 2



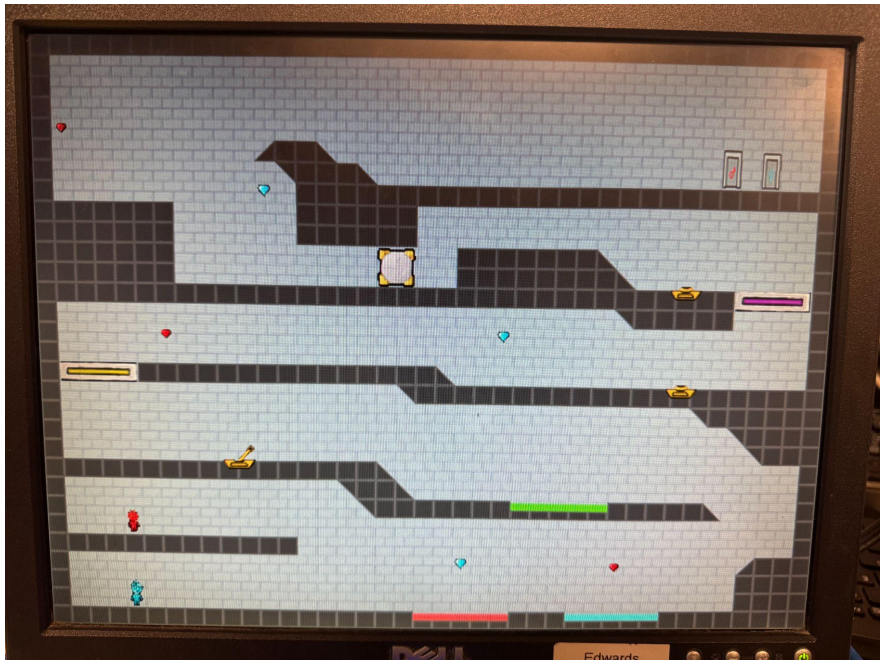
Background 2

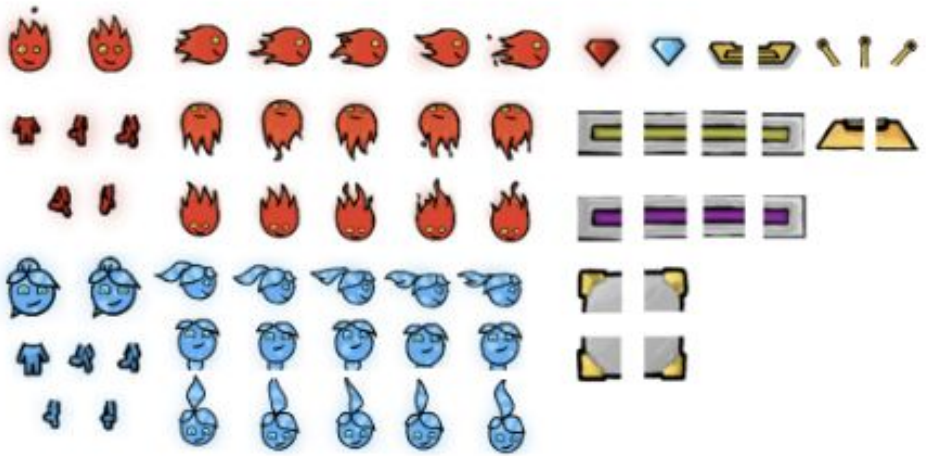
```
WIDTH=256;
DEPTH=208;
ADDRESS_RADIX=HEX;
DATA_RADIX=HEX;
CONTENT BEGIN
```

13 tiles



Frame patterns





Sprite Pattern Resource

Sprite Pattern	Frame Size(Bytes)	Frame Adress	Frames	Total Memory (Bytes)	Description
Fireboy head	16x16x2	0x0000–0x10FF	17	8704	stand(2), walk(5), drop(5), down(5)
Fireboy feet	16x16x2	0x1100–0x15FF	5	2560	stand(1), walk(3), drop & down(1)
Watergirl head	16x16x2	0x1600–0x26FF	17	8704	stand(2), walk(5), drop(5), down(5)
Watergirl feet	16x16x2	0x2700–0x2BFF	5	2560	stand(1), walk(3), drop & down(1)
Red Gem (x3)	16x16x2	0x2C00–0x2CFF	1	512	Collection, disappears after collision
Blue Gem (x3)	16x16x2	0x2D00–0x2DFF	1	512	Collection, disappears after collision
Yellow Button (x6)	16x16x2	0x2E00–0x2FFF	2	1024	Pressed/unpressed animation states
Yellow Lever (x2)	16x16x2	0x3000–0x32FF	3	1536	Toggleable lever: left/mid/right state
Yellow Elevator (x4)	16x16x2	0x3300–0x36FF	4	2048	Moving platform (up/down) animation
Purple Button (x2)	16x16x2	0x3700–0x38FF	2	1024	Pressed/unpressed animation states
Purple Elevator (x4)	16x16x2	0x3900–0x3CFF	4	2048	Moving platform (up/down) animation
Box (x4)	16x16x2	0x3D00–0x40FF	4	2048	Movable block used in puzzles
Total: 32	**16x16x2**	**0x0000–0x40FF**	**65**	**33280**	Total number

128B

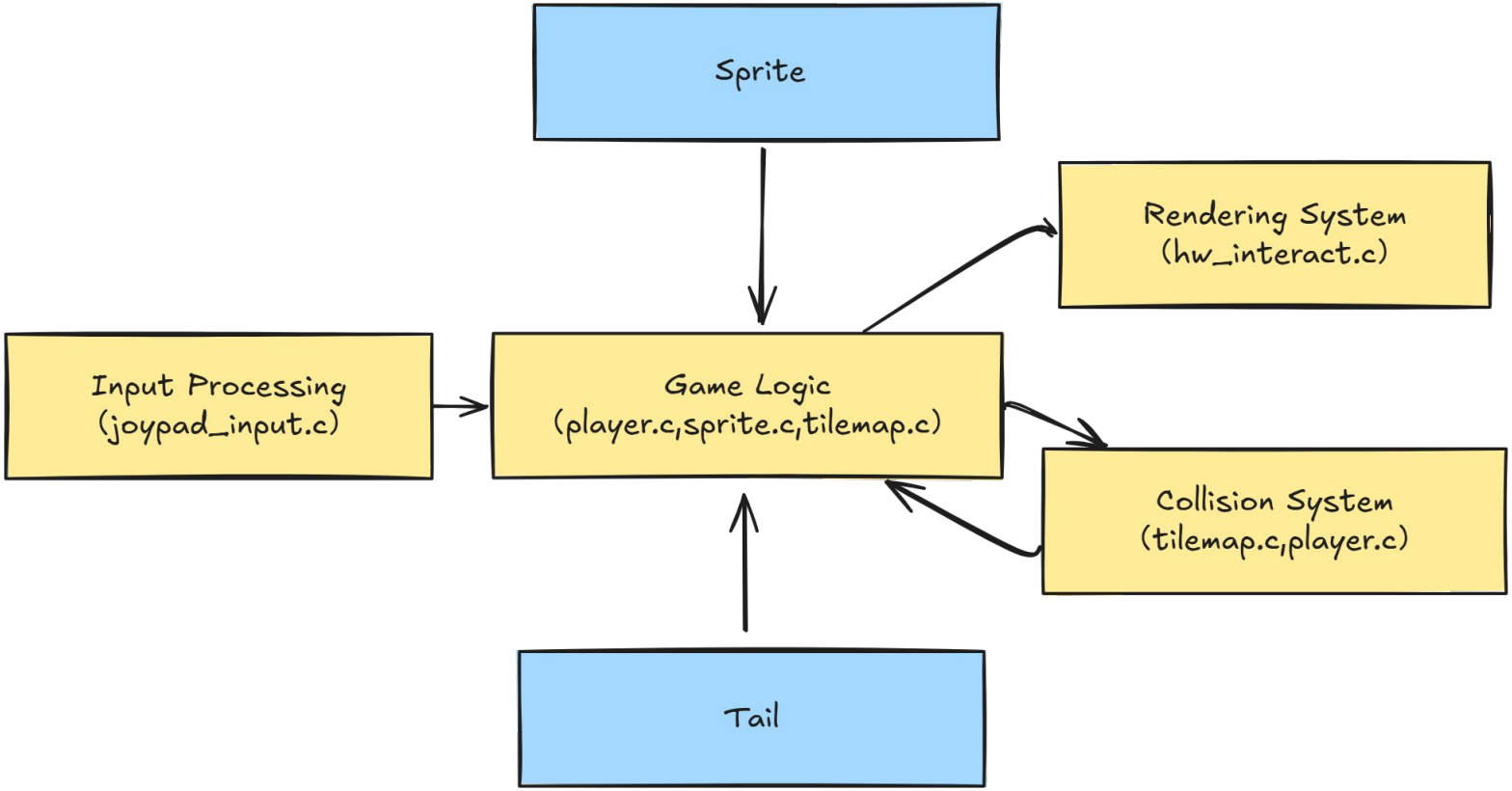
32.5KB

Software tile_map

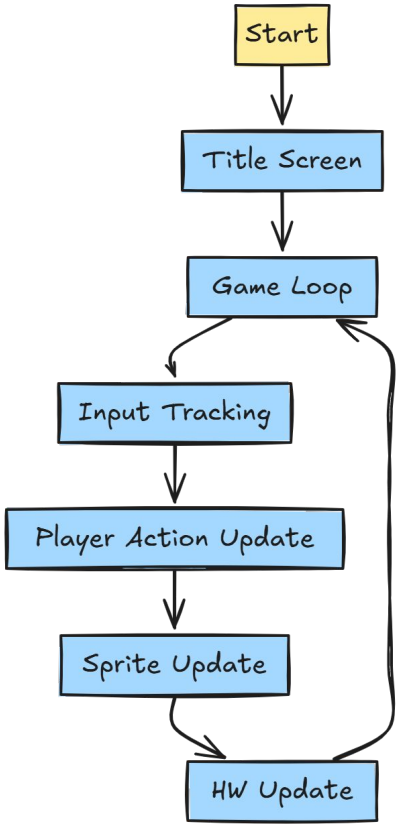
Tile Code	Description
0	TILE_EMPTY: Empty space, walkable
1	TILE_WALL: Solid wall, blocks movement
2	TILE_FIRE: Fire pool, only Fireboy survives
3	TILE_WATER: Water pool, only Watergirl survives
4	TILE_Green: Green toxic pool, fatal to both characters
5	TILE_SLOPE_L_UP: Left-high to right-low slope (\)
6	TILE_SLOPE_R_UP: Right-high to left-low slope (/)
7	TILE_CEIL_R: Ceiling slope, left-high to right-low (\)
8	TILE_CEIL_L: Ceiling slope, left-low to right-high (/)
9	TILE_GOAL: Level goal tile, for WaterGirl
10	TILE_GOAL2: Level goal tile, for FireBpyz

```
//             x              1              5              2              5              3              5              4
{1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1},
{1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1},
{1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1},
{1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1},
{1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1}, //
{1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 6, 1, 1, 5, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 9, 0, 10, 0, 0, 1},
{1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 7, 1, 1, 1, 5, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 9, 0, 10, 0, 0, 1},
{1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1},
{1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1},
{1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1}, // 1
{1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 5, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1},
{1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 5, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1},
{1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 1},
{1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 7, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1},
{1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1}, //
{1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1},
{1, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 5, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1},
{1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 7, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1},
{1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 7, 1, 1, 1, 1, 1, 1},
{1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 7, 1, 1, 1, 1}, // 2
{1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 7, 1, 1, 1},
{1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 5, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1},
{1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 7, 1, 1, 5, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1},
{1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 7, 1, 1, 1, 1, 1, 1, 1, 4, 4, 4, 4, 4, 1, 1, 1, 1, 1, 5, 0, 0, 0, 0, 1},
{1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1}, //
{1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1},
{1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 6, 1, 1, 1},
{1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1},
{1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1},
{1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 1, 1, 1, 3, 3, 3, 3, 3, 1, 1, 1, 1, 1, 1, 1, 1}};
```

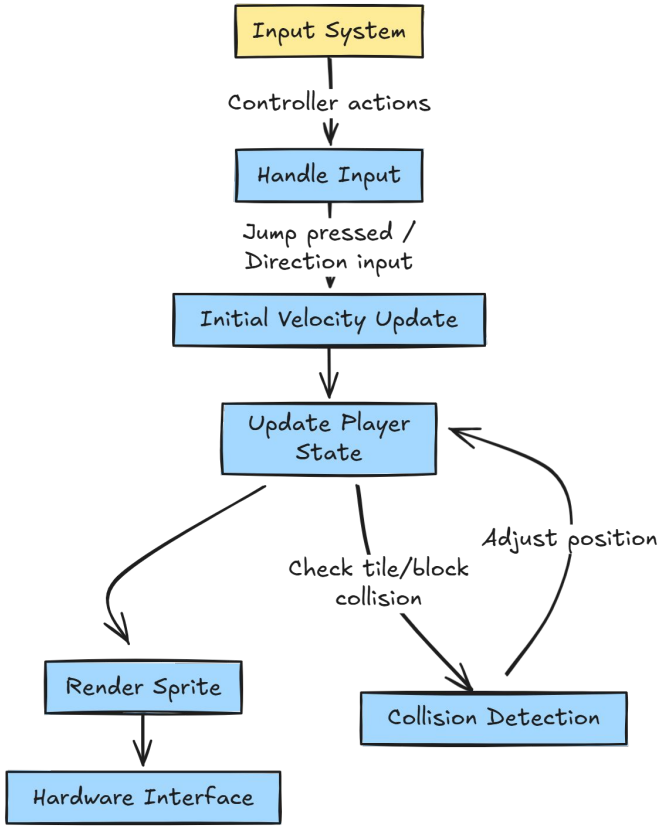
System Overview



Main Game Logic

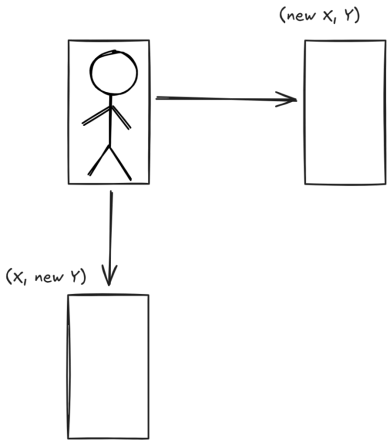


Player Action System

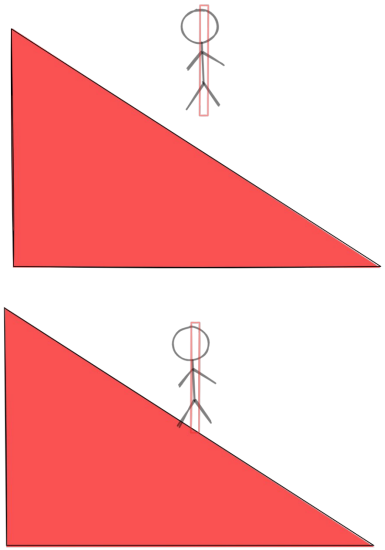


Collision Detection

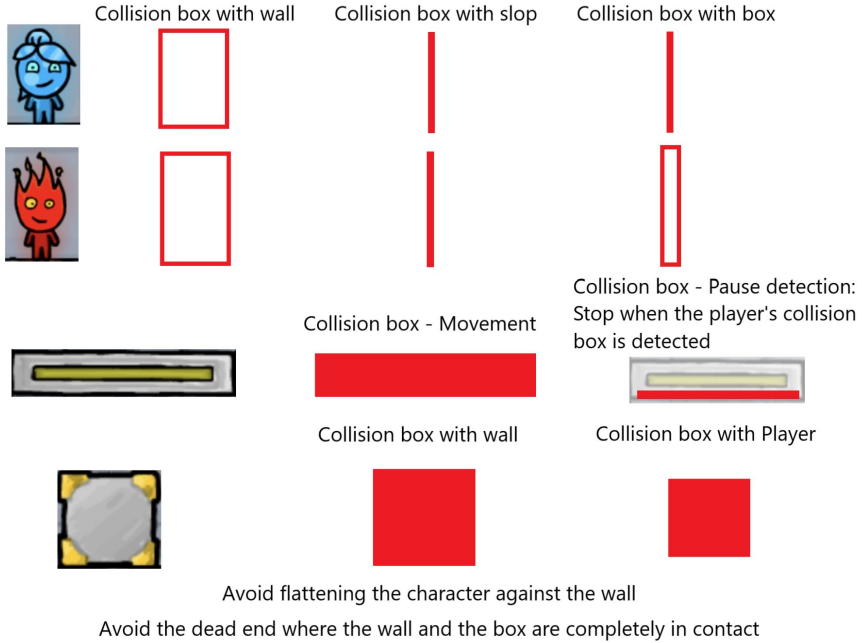
Prediction



Correction



Collision Box



Avoid flattening the character against the wall

Avoid the dead end where the wall and the box are completely in contact