
Monkey Madness
By Kyle Edwards, Jake Torres, Madeline Skeel, Sadie Freisthler, William Freedman
slidesmania.com

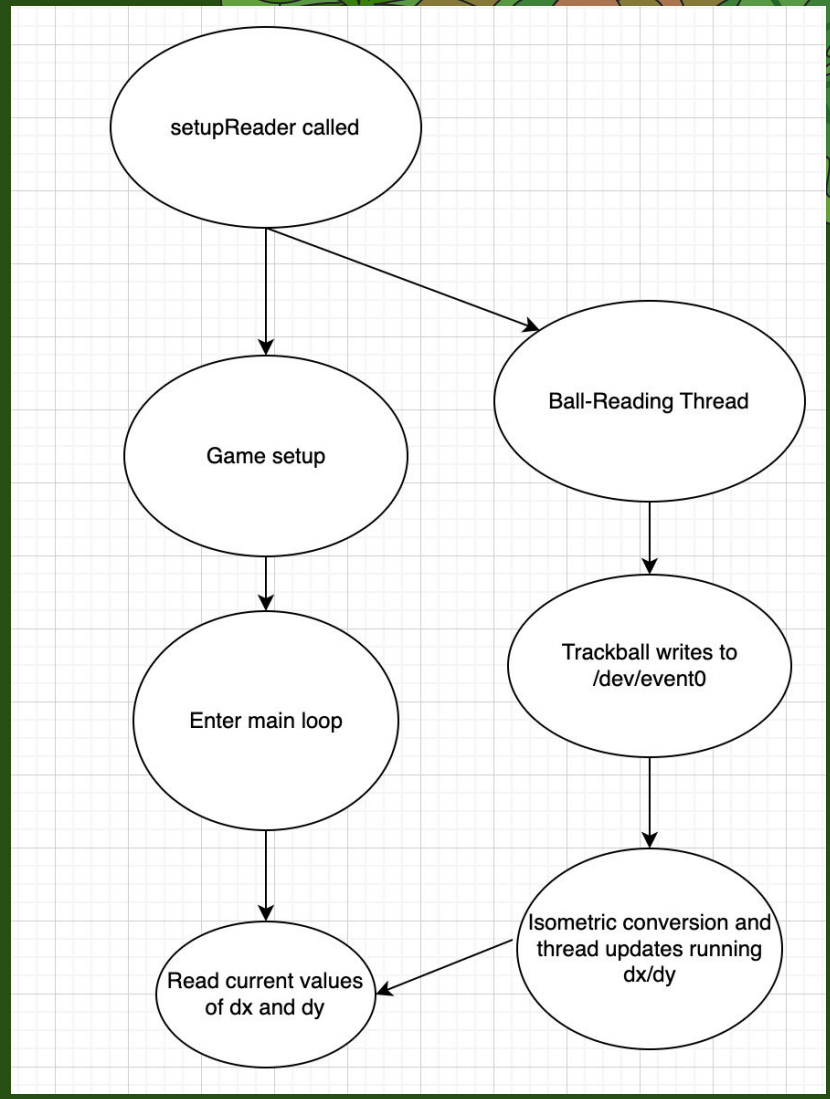
Table of contents.

01 Overview

- Simplified recreation of Marble Madness inspired by Super Monkey Ball
- Isometric projection rendering
- Trackball Input

02 Trackball

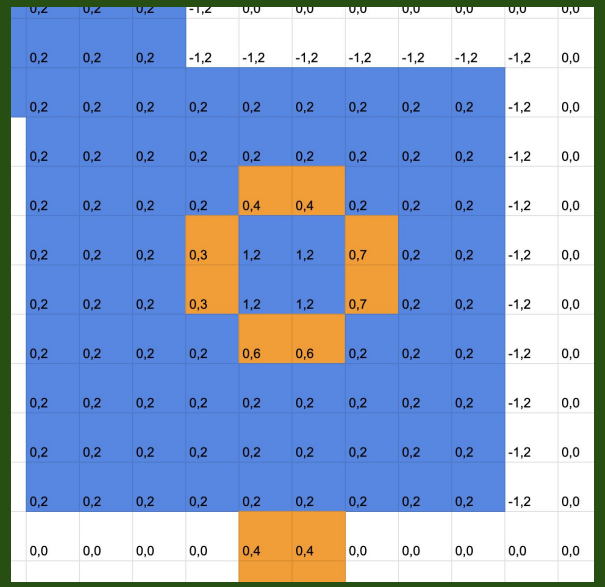
02 Trackball



03 Game Logic/Level Design

03 Game Logic/Level Design

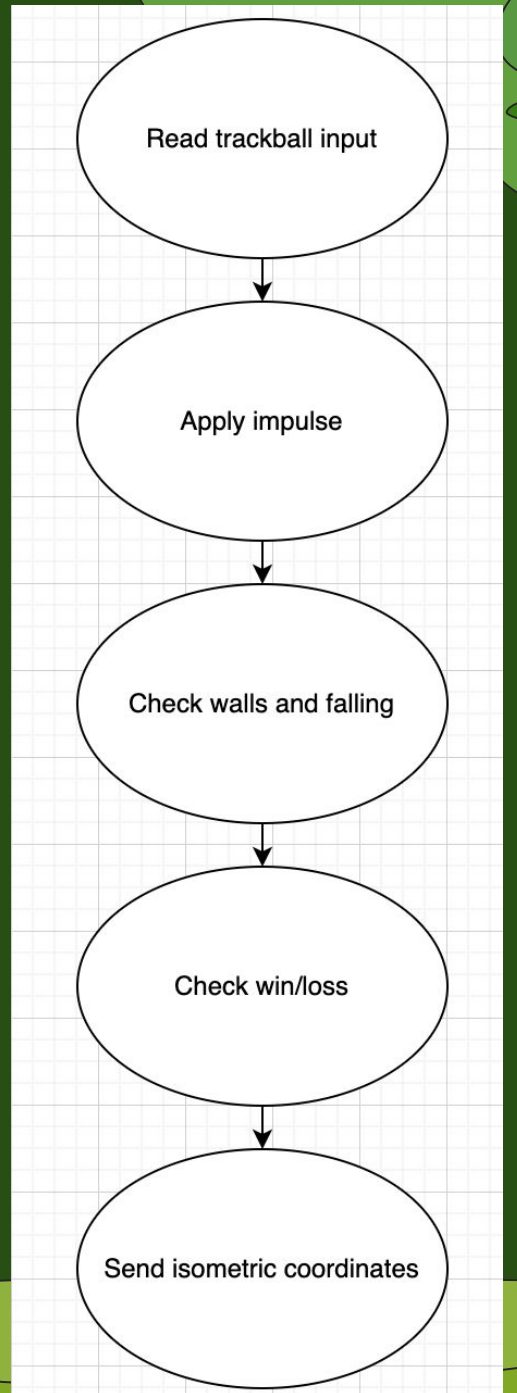
- Excel Sheet -> csv -> read_level() -> Tile array in memory, set ball's starting position.




```
typedef struct {
    int x_idx;
    int y_idx;
    int z_idx;
    int type;
} Tile;
```

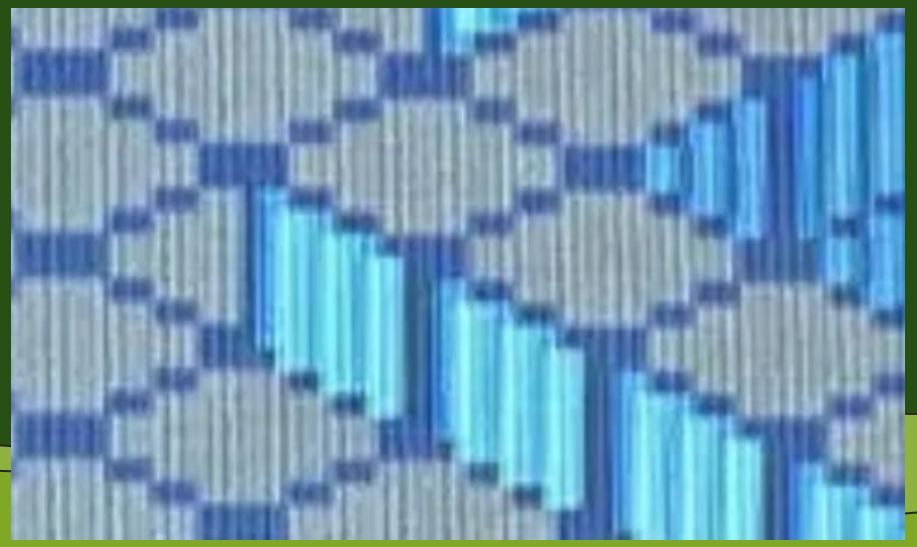
```
enum TileType {
    NO_TILE = 0,
    START_TILE = 1,
    FLAT = 2,
    UP_Y_RAMP = 3,
    UP_X_RAMP = 4,
    DOWN_X_RAMP = 6,
    DOWN_Y_RAMP = 7,
    WIN_TILE = 8
};
```

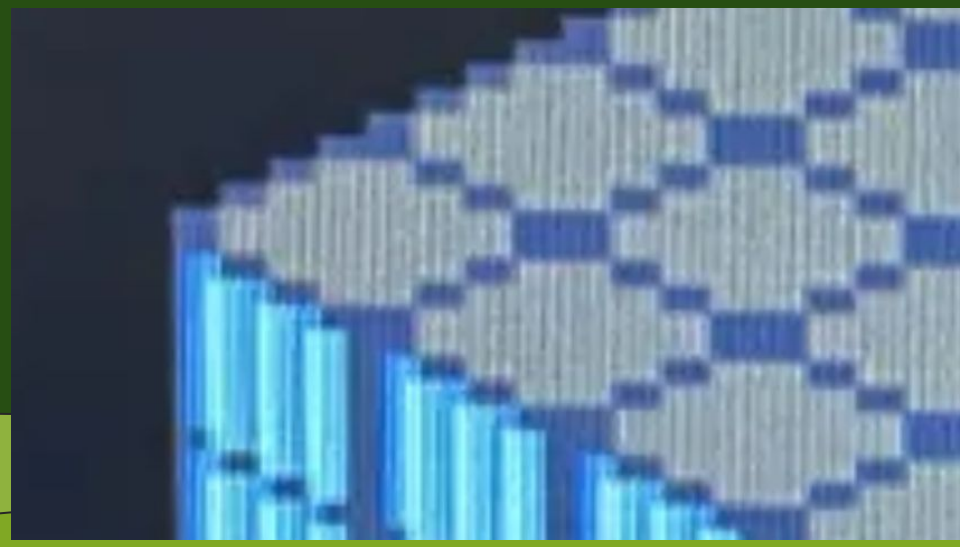
03.1 Game Loop



03.1 Game Loop

- Two boundaries to check and handle
  - Walls
    - If the ball is approaching the wall, invert its velocity in the direction of the normal
  - Fall
    - Apply an impulse in the positive z direction





03.2 Isometric ball projection

- The ball's screen position is calculated independently from the rest of the tiles.
- Small additions to the normal isometric formula
  - We multiply x and y by 2 and 4 respectively
    - Projected tiles are twice as wide as they are tall
  - We add an offset to account for our origin not being in the corner of the screen

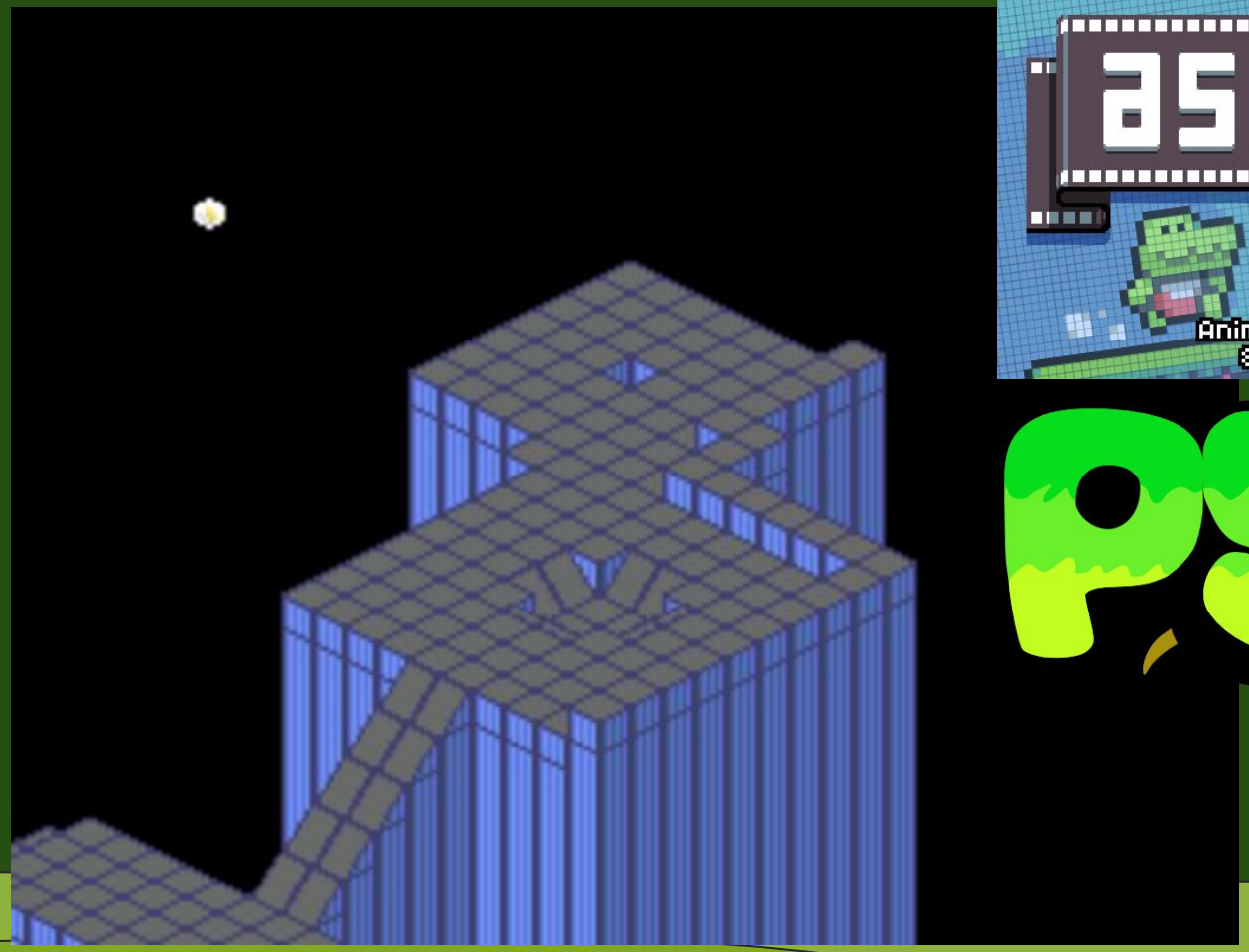
```
Vec2 project_3D_to_2D(Vec3 pos3D) {
    Vec2 pos2D;

    pos2D.x = (2 * pos3D.y + 2 * pos3D.x) * 2;
    pos2D.y = (pos3D.x + pos3D.y + 2 * pos3D.z) * 4;

    pos2D.x += 120.0;
    pos2D.y += 50.0;

    return pos2D;
}
```

04 Software Implementation

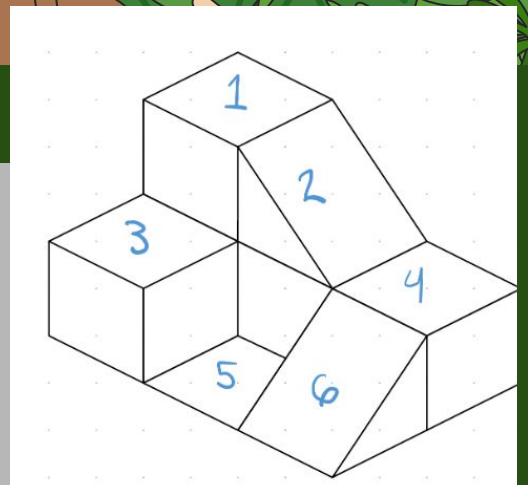


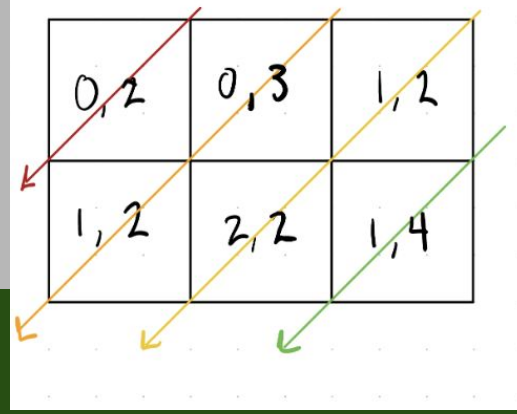
pygame

04 Software Implementation

Name	Graphic	Size (bits)	csv value
Banana Ball		16x16	N/A
Floor Tile		16x16	2
Ramp Left Tile		16x16	4
Ramp Right Tile		16x16	3
Back Ramp Right Tile		16x16	6
Back Ramp Left Tile		16x16	7

```
for diagonal in rows+cols:
    for i in reversed(rows):
        j = diagonal - i
        // skip if oob
        x = (2 * j - 2 * i) *
                (tile_width // 4)
              + x_offset
        y = (i + j + 2 * z) *
             (tile_height // 2)
            + y_offset
```





04 Software

- Render -> 4 binary files -> Device Driver
- Headers of the binary files
  - Number of palettes
  - Height + Width of Map
  - Number of Texture
- Read & Write Texture, Palette + Level Data -> Read & Write Sprite Data -> Game Loop

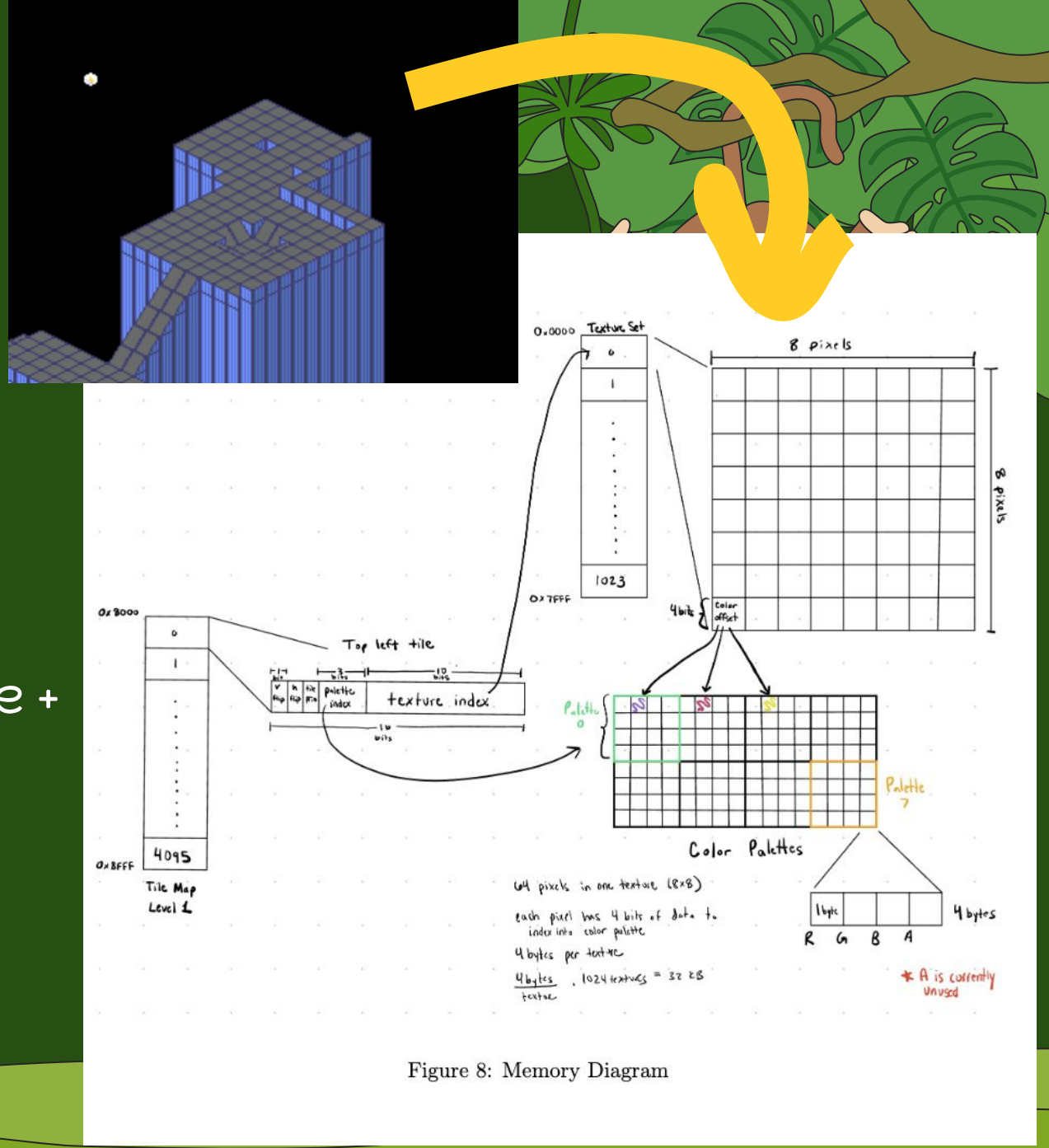
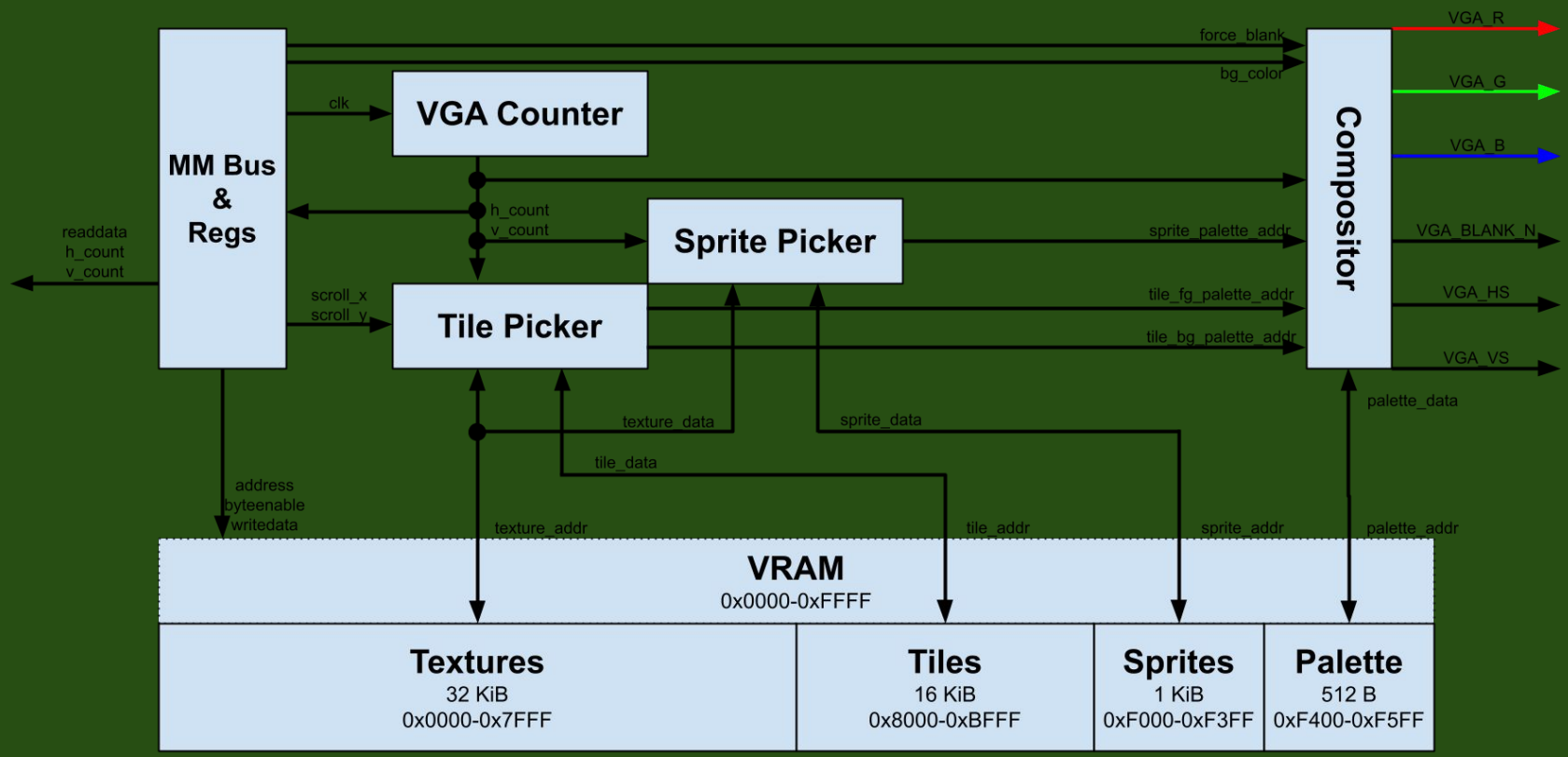


Figure 8: Memory Diagram

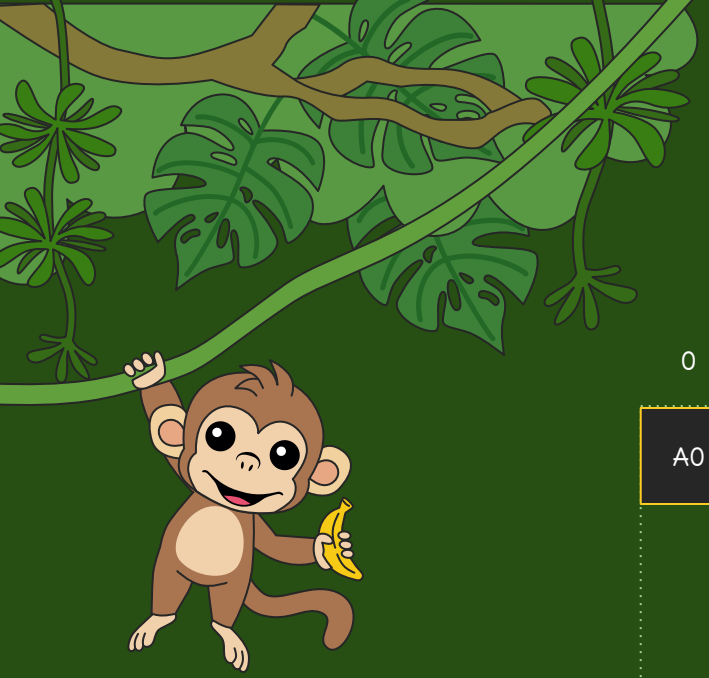
05 Hardware

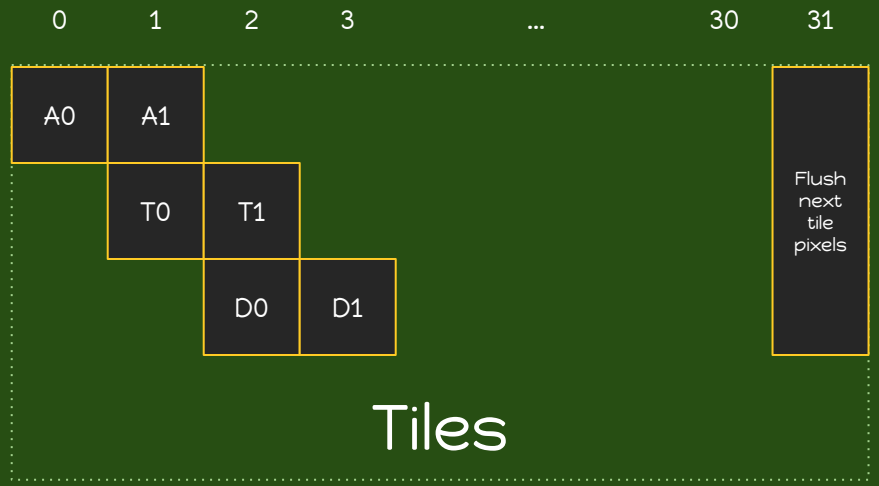


Memory Map

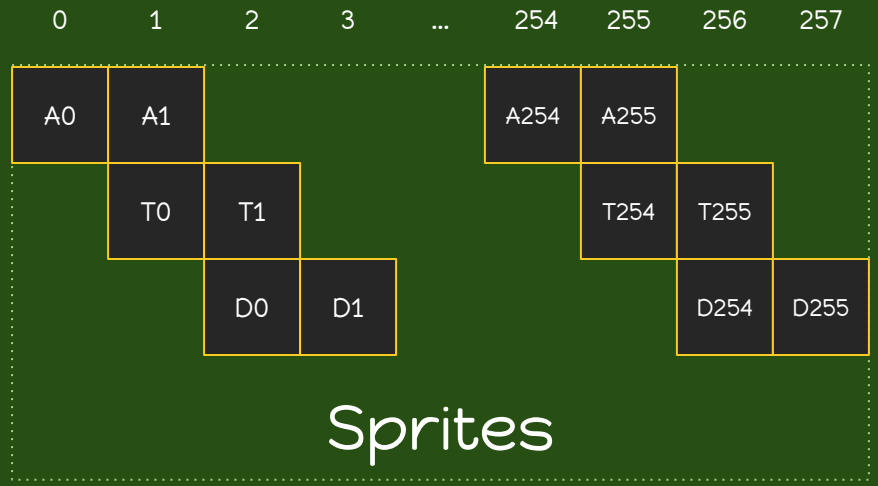
	0x0000
Texture Data	0x8_7_6_5_4_3_2_1
	0x8000
Tiles	0bV_H_0_PPP_TTTTTTTTTT
	0xBFFF
	0xF000
Sprites	0bV_H_PPP_TTTTTTTTTT_YYYYYYYY_XXXXXXXXX
	0xF400
Palette	0xRR_GG_BB_AA
	0xF5FF
	0xFF00
Registers	Depends
	0xFFFF

Rendering Pipelines





Tiles



Sprites

06 Takeaways + Demo

Division of Labor

Kyle

Hardware + Driver

Jake

Driver + Python ->
Software + Artwork

Madeline

Physics + Trackball +
Software

Sadie

Physics + Trackball +
Software

William

Physics + Trackball +
Software


Thank you!!
slidesmania.com