

# Monkey Madness

By Kyle Edwards, Jake Torres,  
Madeline Skeel, Sadie Freisthler,  
William Freedman





# Table of contents.

01 | Overview

02 | Trackball

03 | Game  
Logic

04 | Software  
Implementation

05 | Device Driver  
+ Hardware

06 | Demo +  
Conclusion

# 01 Overview

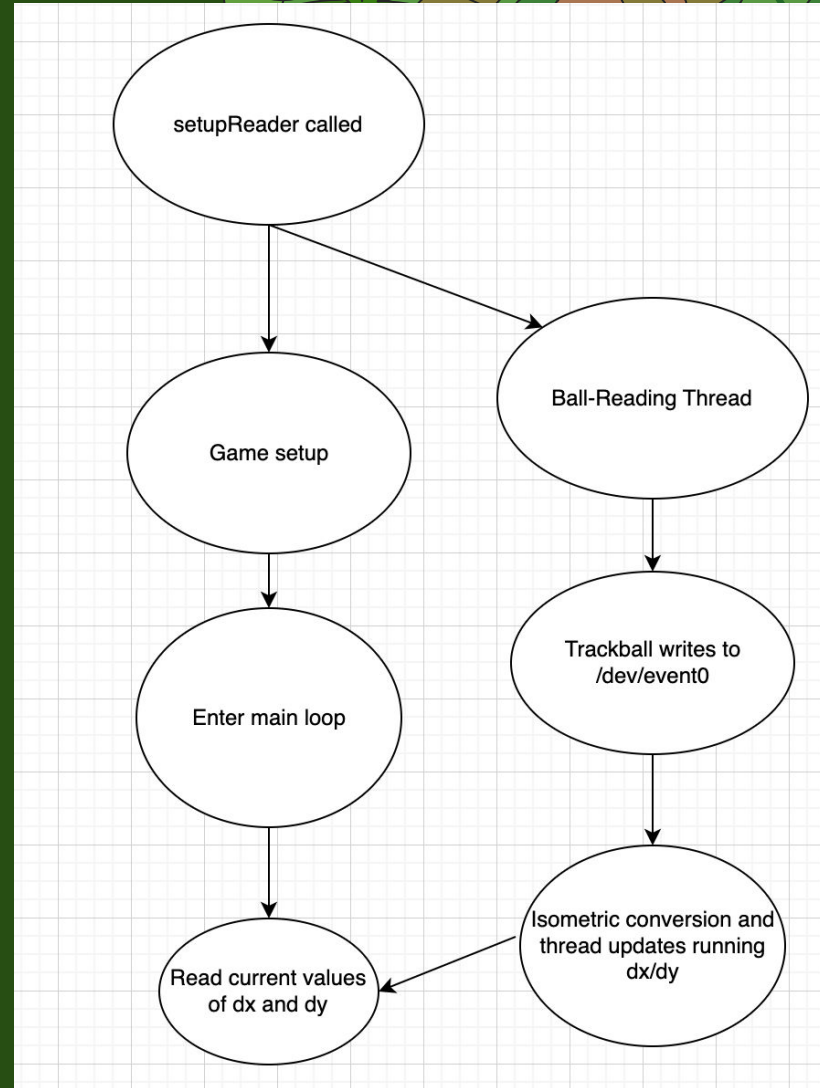
- Simplified recreation of Marble Madness inspired by Super Monkey Ball
- Isometric projection rendering
- Trackball Input



# 02 Trackball

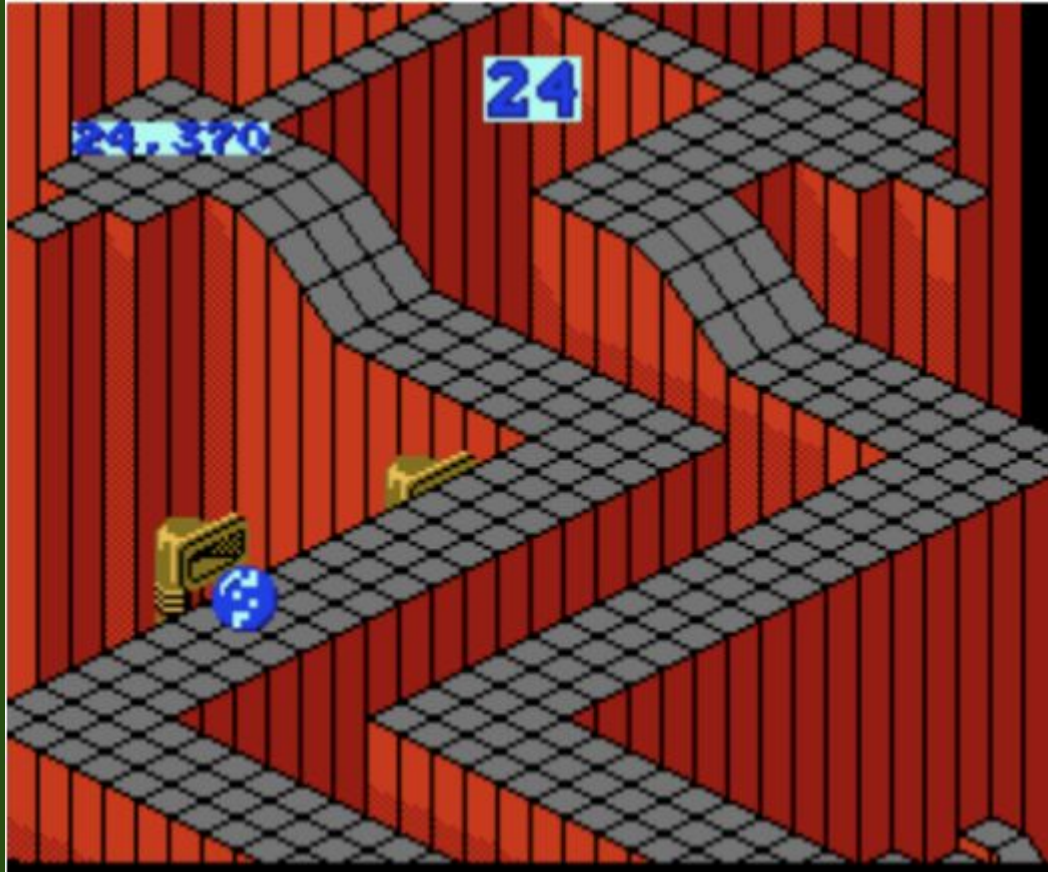


# 02 Trackball



# 03

## Game Logic/Level Design





# 03

## Game Logic/Level Design

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

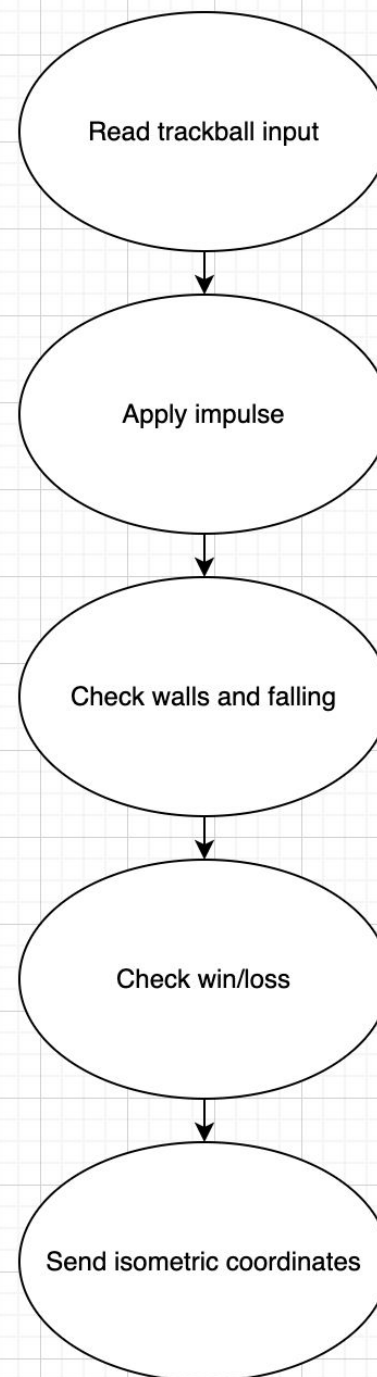


0,2	0,2	0,2	-1,2	0,0	0,0	0,0	0,0	0,0	0,0	0,0
0,2	0,2	0,2	-1,2	-1,2	-1,2	-1,2	-1,2	-1,2	-1,2	0,0
0,2	0,2	0,2	0,2	0,2	0,2	0,2	0,2	0,2	-1,2	0,0
0,2	0,2	0,2	0,2	0,2	0,2	0,2	0,2	0,2	-1,2	0,0
0,2	0,2	0,2	0,2	0,4	0,4	0,2	0,2	0,2	-1,2	0,0
0,2	0,2	0,2	0,3	1,2	1,2	0,7	0,2	0,2	-1,2	0,0
0,2	0,2	0,2	0,3	1,2	1,2	0,7	0,2	0,2	-1,2	0,0
0,2	0,2	0,2	0,2	0,6	0,6	0,2	0,2	0,2	-1,2	0,0
0,2	0,2	0,2	0,2	0,2	0,2	0,2	0,2	0,2	-1,2	0,0
0,2	0,2	0,2	0,2	0,2	0,2	0,2	0,2	0,2	-1,2	0,0
0,2	0,2	0,2	0,2	0,2	0,2	0,2	0,2	0,2	-1,2	0,0
0,0	0,0	0,0	0,0	0,4	0,4	0,0	0,0	0,0	0,0	0,0

```
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  
};
```

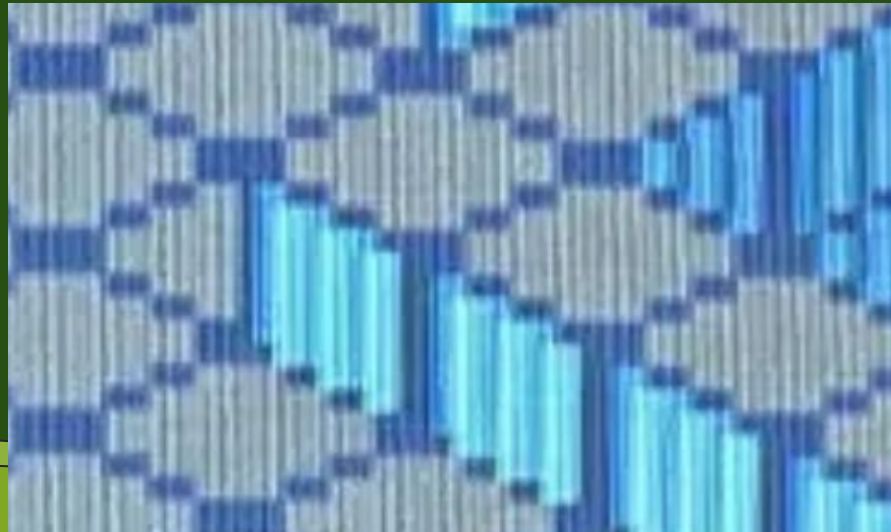
# 03.1 Game Loop





# 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

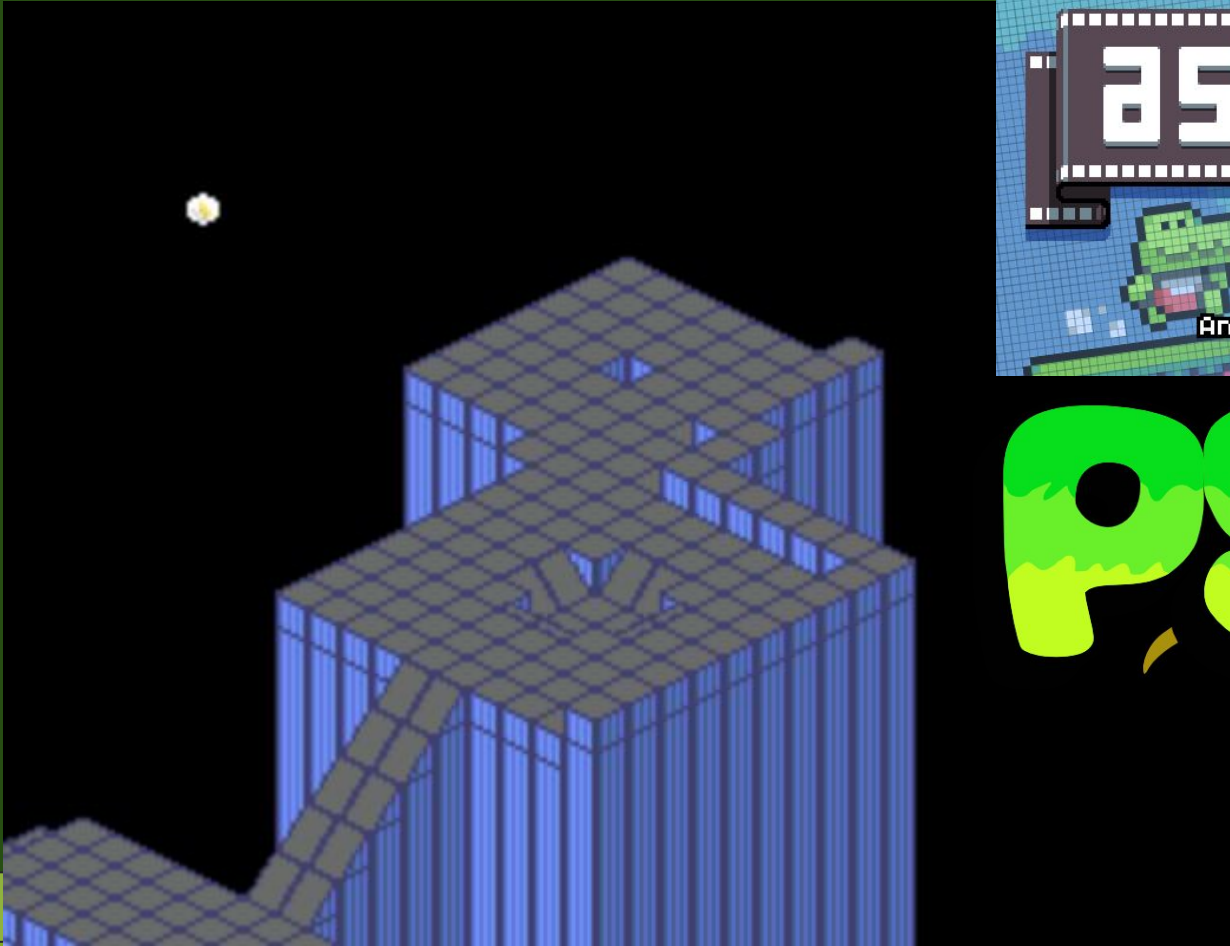
- The ball's screen position is calculated independently from the rest of the tiles.
- Small additions to the normal isometric formula
  - We multiply x and y by 2 and 4 respectively
    - Projected tiles are twice as wide as they are tall
  - We add an offset to account for our origin not being in the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

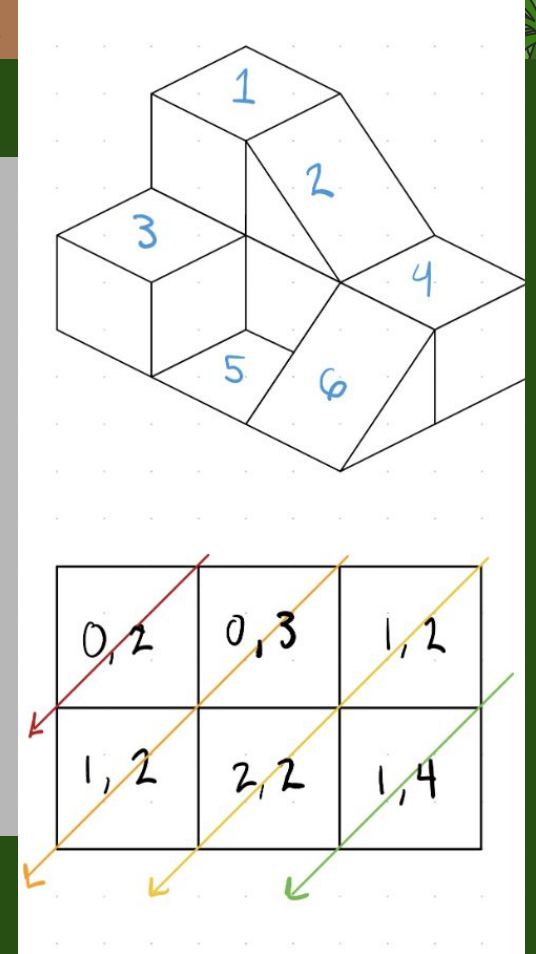
## Software Implementation



# 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
```





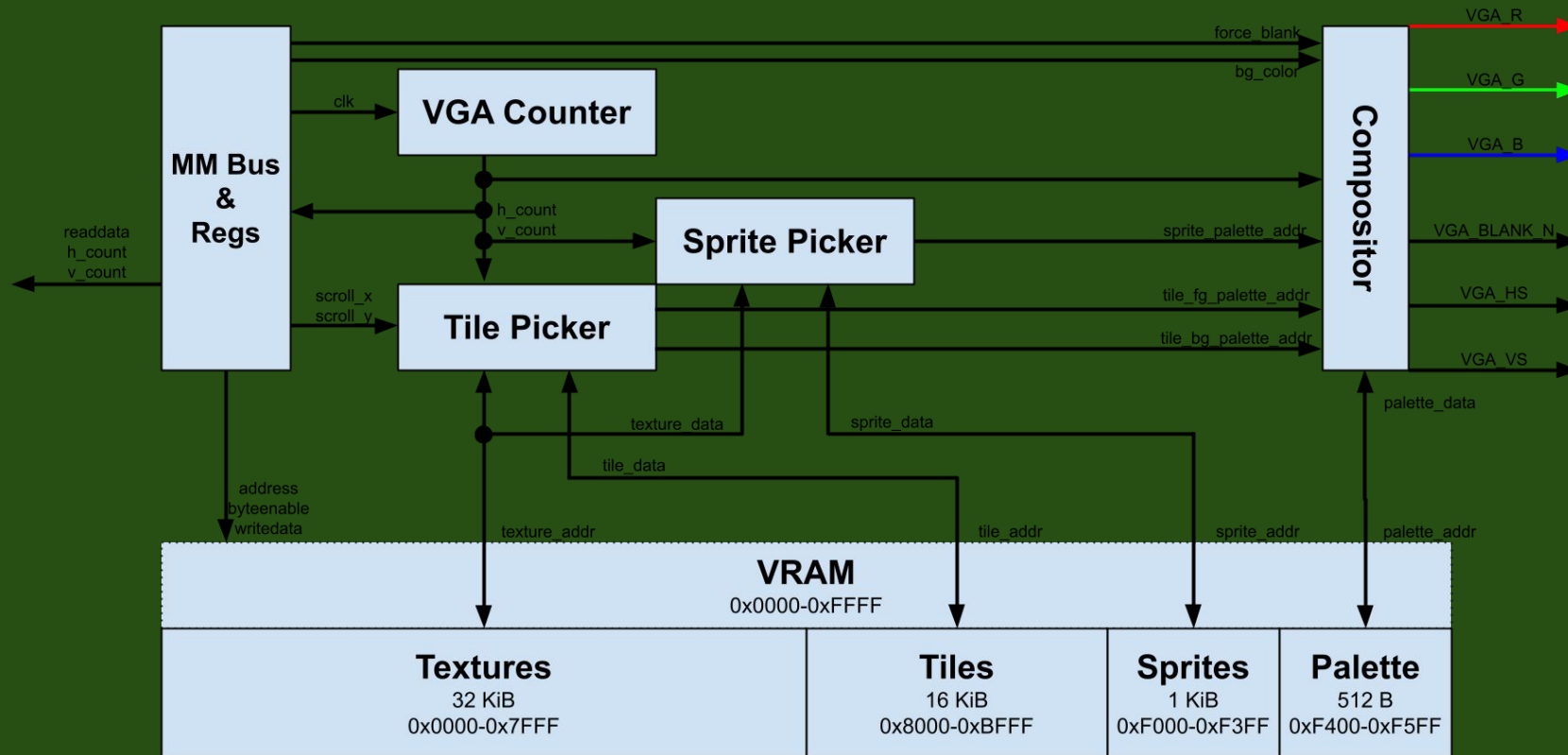
The image is a composite. On the left, a 3D wireframe of a staircase is shown against a black background. A large yellow arrow points from the staircase towards a diagram on the right. The diagram shows a 'Texture Set' with a value of '0.0000' and a '0' in a box, with an arrow pointing to a horizontal bar labeled '8 pixels'. The background of the right side is a green jungle scene with a tree branch and leaves.

- 
- Figure 8: Memory Diagram
- The diagram illustrates the memory layout for a texture map, showing the relationship between the Tile Map Level 1, the texture index, and the color palettes.
- Tile Map Level 1:** A vertical array of 4096 entries (0 to 4095). The top left tile (index 0) is highlighted.
- Top left tile structure:** A 16-bit structure divided into two 8-bit fields:
- texture index (8 bits):** Points to the texture data.
  - palette index (8 bits):** Points to the color palette.
- Texture Data:** A 16x16 grid of pixels. Each pixel is 4 bits in size. The texture index points to the top-left pixel of this grid.
- Color Palettes:** A 4x4 grid of color palettes. Each palette is 4 bits in size. The palette index points to the top-left palette of this grid.
- Color offset:** A 1023-bit field, likely representing a color offset or a pointer to a color table.
- Color Palettes Detail:** A 4x4 grid of color palettes. Each palette is 4 bits in size. The palettes are labeled R, G, B, and A (Alpha). The A channel is currently unused.
- Handwritten Calculations:**
- 64 pixels in one texture (16x4)
  - each pixel has 4 bits of data to index into color palette
  - 4 bytes per texture
  - $\frac{4 \text{ bytes}}{\text{texture}} \cdot 1024 \text{ textures} = 32 \text{ KB}$
- Handwritten Note:** \* A is currently unused

13

# 05

## Hardware







# Memory Map



0x0000

Texture Data

0x8\_7\_6\_5\_4\_3\_2\_1

0x8000

Tiles

0bV\_H\_O\_PPP\_TTTTTTTTTT

0xBFFF



0xF000

Sprites

0bV\_H\_PPP\_TTTTTTTTTT\_YYYYYYYY\_XXXXXXXXXX

0xF400

Palette

0xRR\_GG\_BB\_AA

0xF5FF



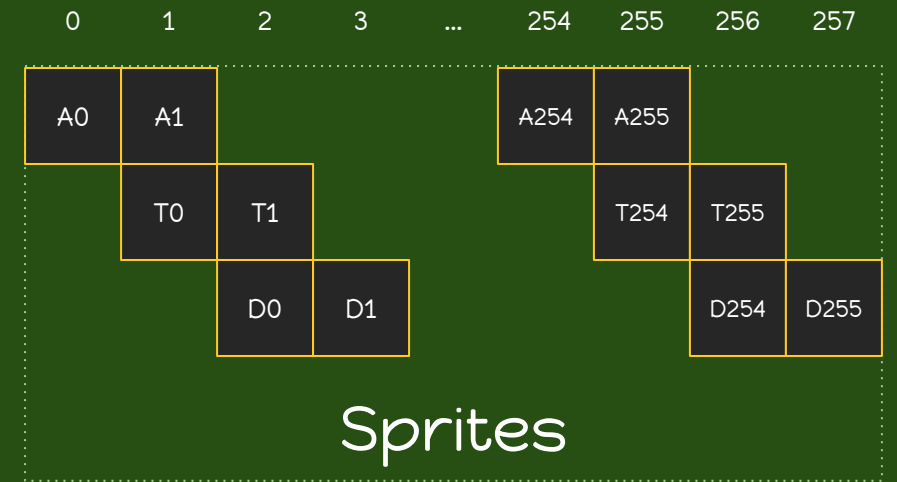
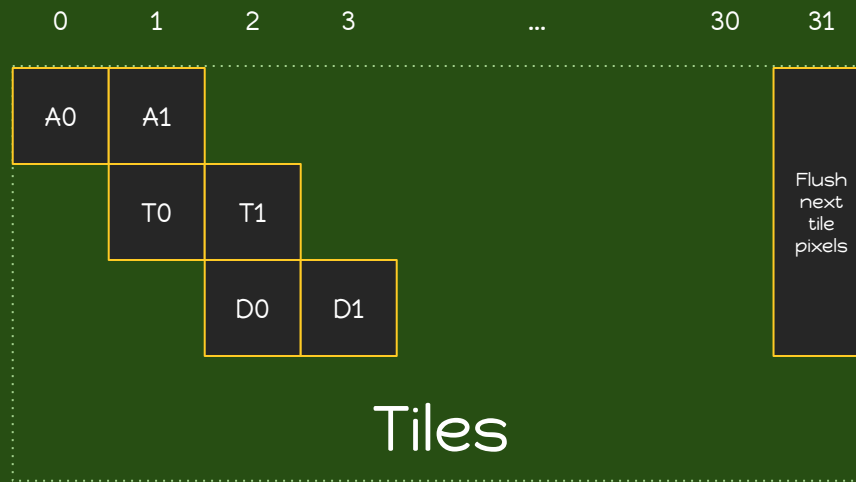
0xFF00

Registers

Depends

0xFFFF

# Rendering Pipelines



# 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!!**

