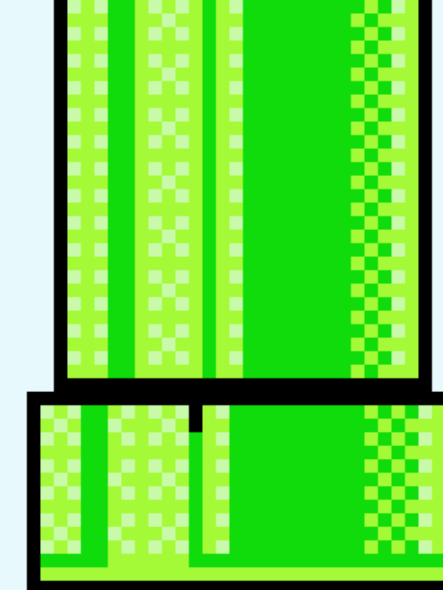
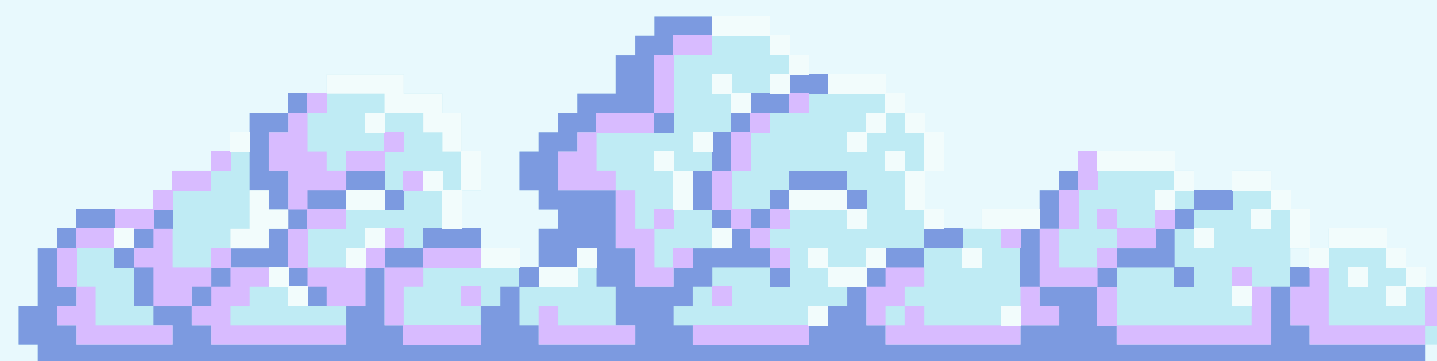


FLAPPY BIRD



START



CREATED BY
SAPTARSHI TALUKDAR AND
ROHAN DAGA

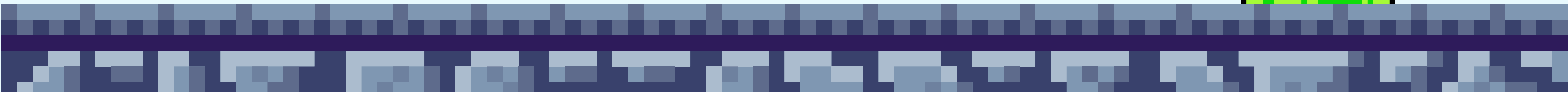
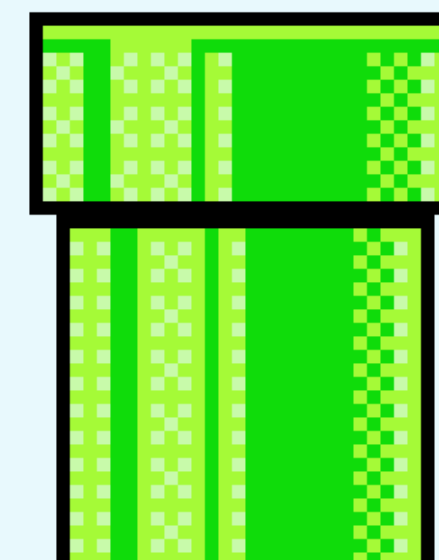






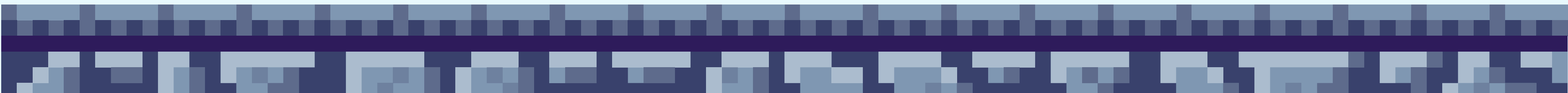
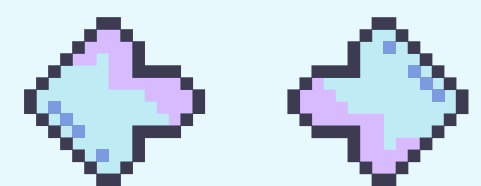
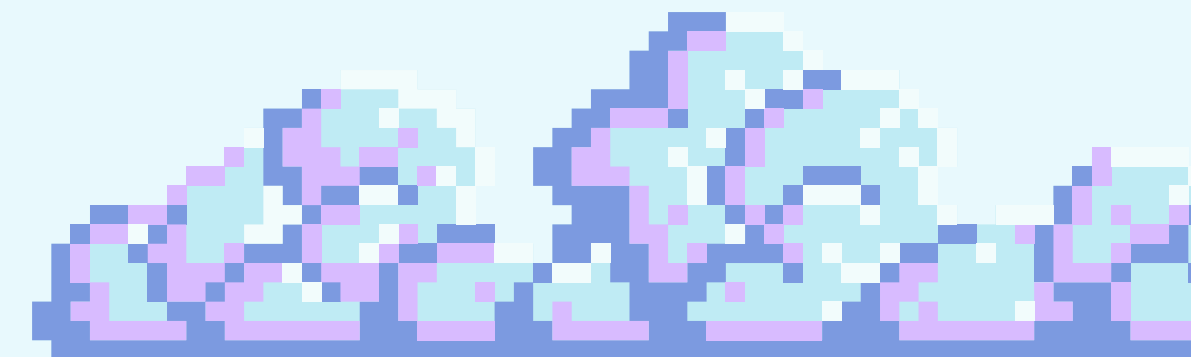




TABLE OF CONTENTS

-  Introduction
-  High Level Block Diagram
-  Game Logic
-  Draw Frame Module
-  Future Improvements
-  Division of Work



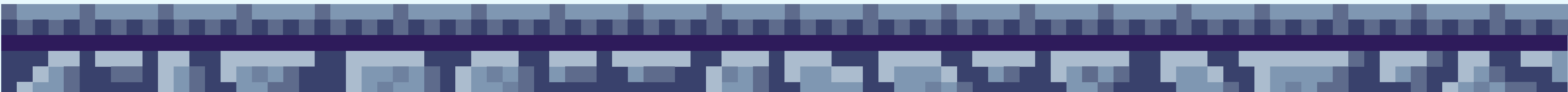
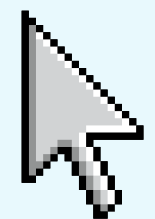
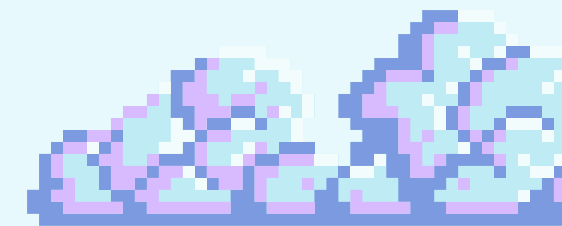
INTRODUCTION

Our Project is an Flappy Bird-inspired game developed on the FPGA DE1-SoC board. The game showcases a seamless integration of visuals, game logic and user interaction all implemented directly on the platform.

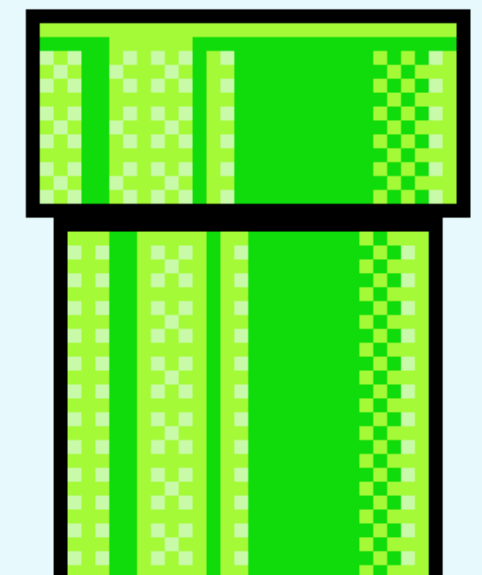
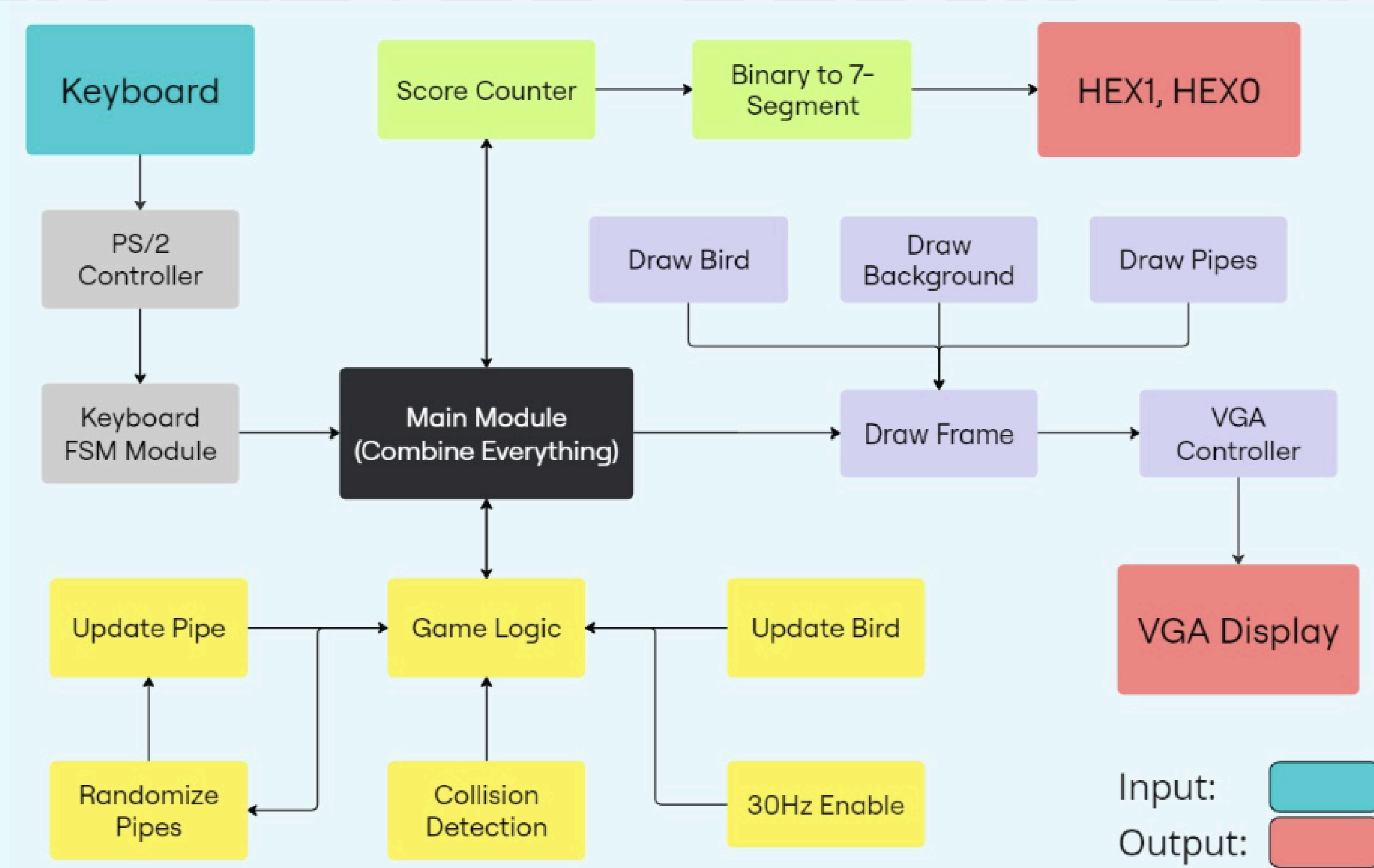
The visuals bring the game to life with elements like the background, dynamic pipes, the bird and an introductory screen.

The game logic is the core of the project, handling collision detection updating the positions of the bird and pipes, maintaining a 30Hz refresh rate and score tracking.

The keyboard implementation allows the player to interact with the game, using key inputs to control the bird's movement.



HIGH LEVEL BLOCK DIAGRAM





KEYBOARD DRIVER

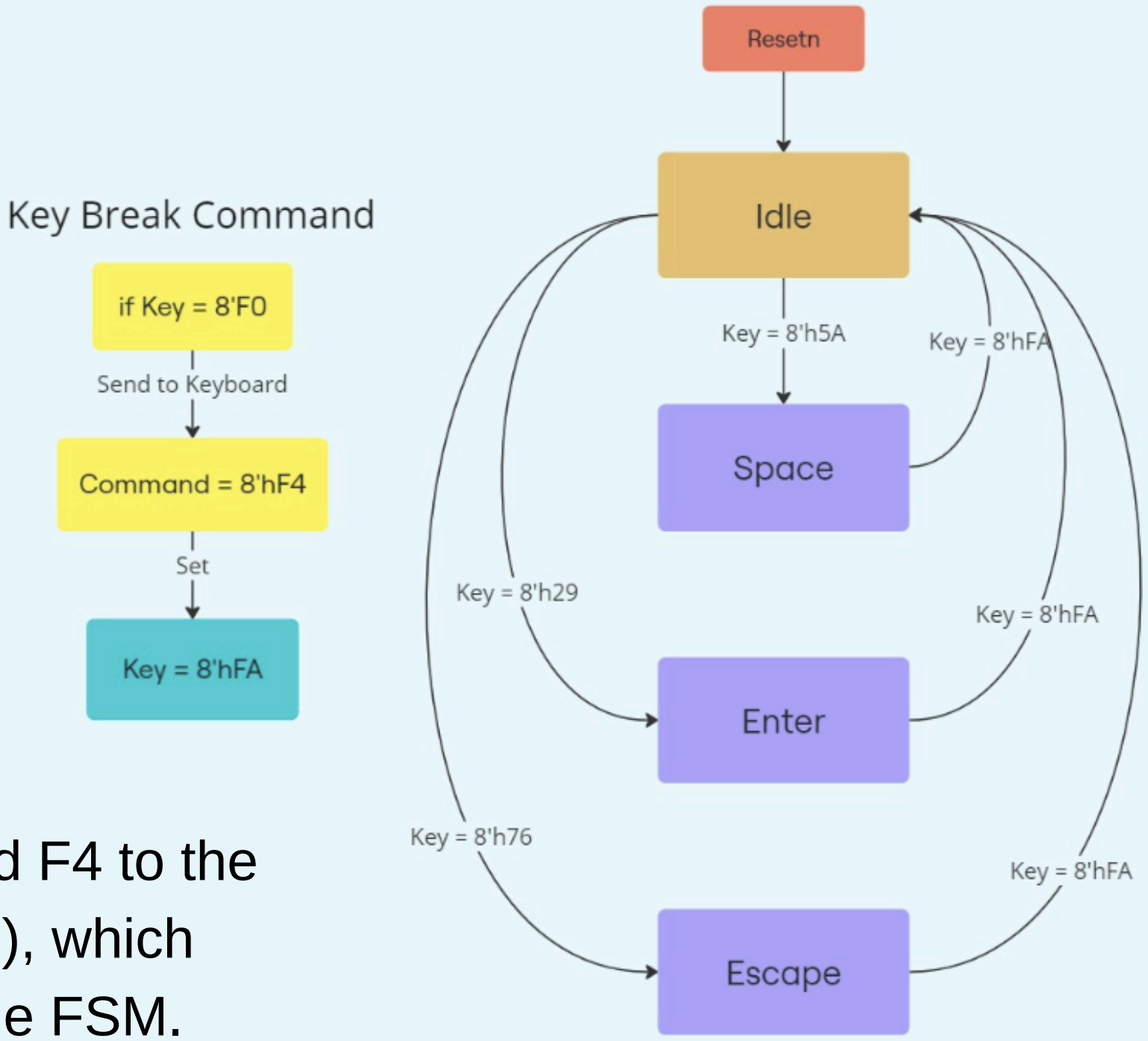
We receive input from the PS/2 Keyboard using the provided PS/2 Controller.

The keyboard outputs “Space”, “Enter”, and “Escape” are controlled using an FSM.

“Space” is for jump, “Enter” is for restarting the game, and “Escape” is currently unused.

Bug: The Keyboard would get stuck in one state until another key was pressed.

Fix: Added break code functionality that sends the command F4 to the keyboard whenever a key is released (detected by Key = F0), which changes Key to FA. This was added as a state variable to the FSM.





GAME LOGIC (30HZ ENABLE)

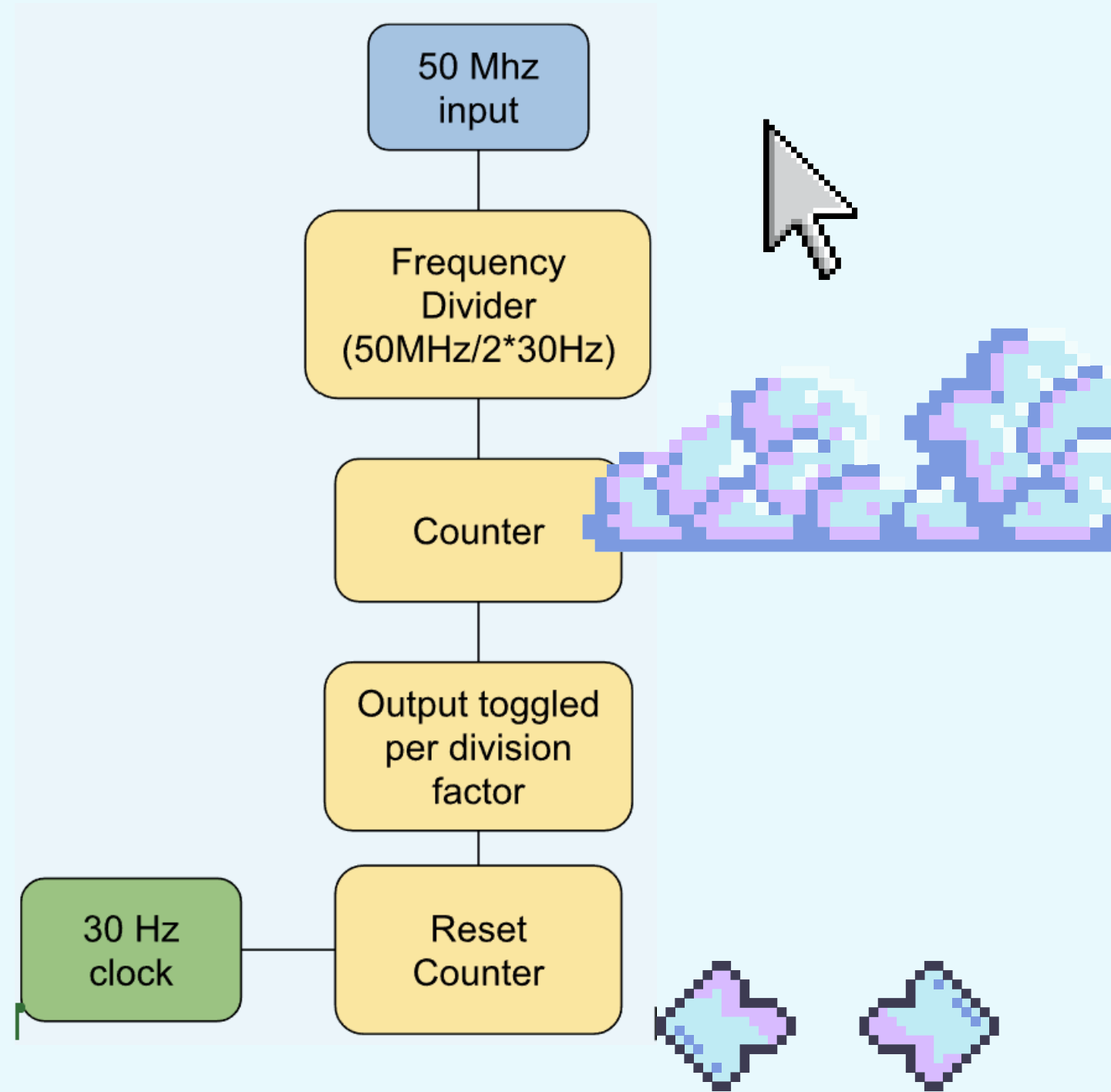
We use a 50Mhz clock as an input and calculate the division factor using the frequency divider (50MHz/60Hz).

We implement a counter to count up to the division factor.

The clock output is toggled when the counter reaches the division factor.

The counter is reset and the process is repeated.

The 30 Hz clock is then connected to the main module to implement a 30Hz refresh rate. Each frame is redrawn every 33.33 ms.

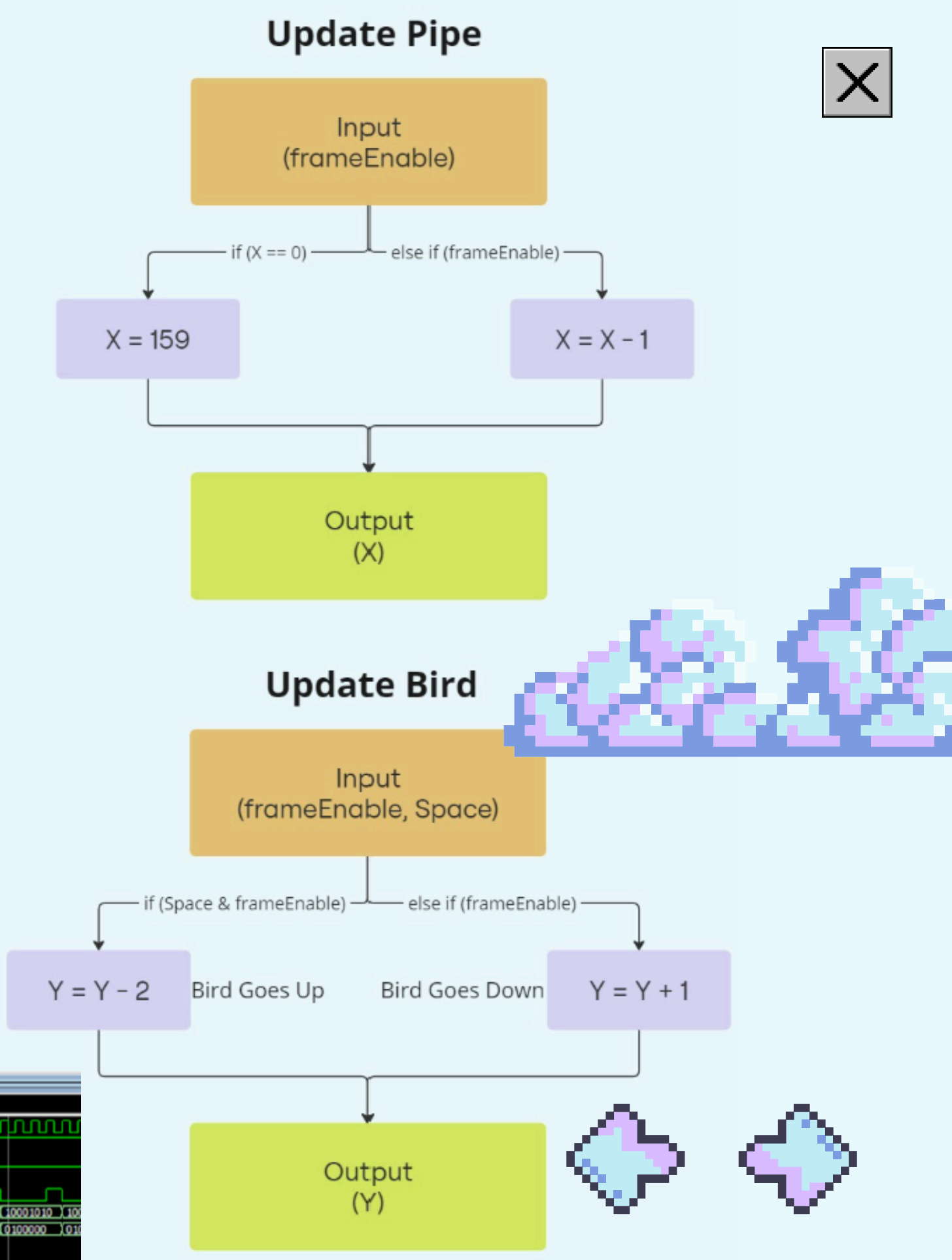
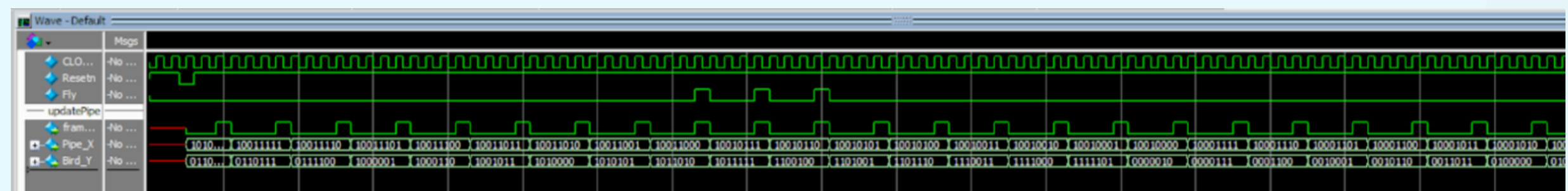


GAME LOGIC: UPDATE PIPE & BIRD

The update pipe module decrements the x-coordinate of the pipe every frame (frameEnable signal).

Every frame, the Update Bird module decrements the y-coordinate of the bird every frame by 2 if the user presses the spacebar, and increments by 1 if spacebar is not pressed.

Note: Decrementing the y-coordinate makes the bird go up, and incrementing makes the bird go down.





GAME LOGIC: COLLISION DETECTION

The Collision Detection module checks for overlap between the bird and the pipes using the if statement:

```
if (bird_y < pipe_y || (bird_y + bird_height) > (pipe_y + hole_height));
```

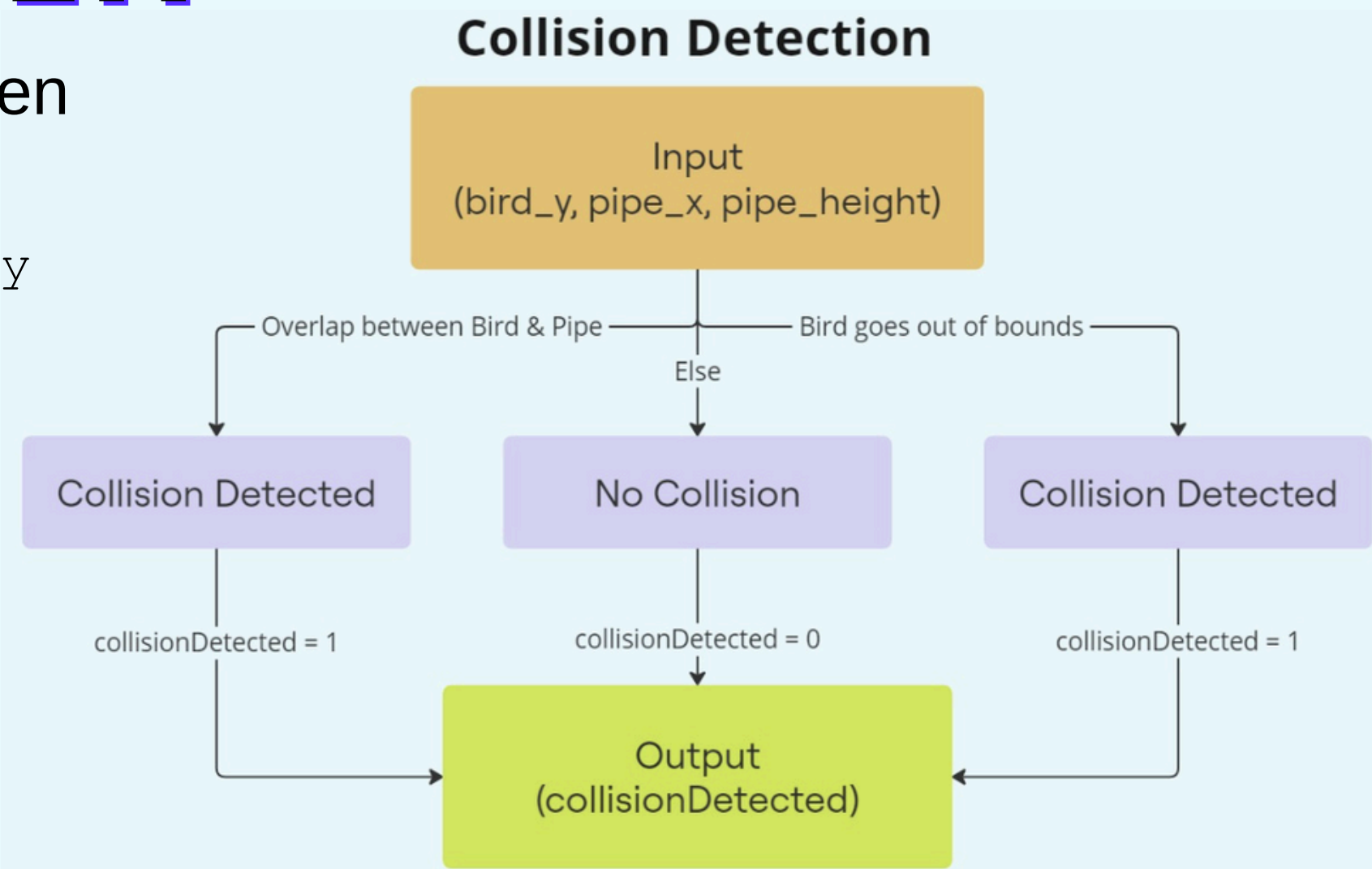
And checks if the bird goes out of bounds using the if statement:

```
if (bird_y + bird_height > 120 || bird_y <= 0);
```

If a collision is detected, collisionDetected is set to 1.

Bug: The hitbox of the bird was a rectangle due to which the collision was detected even when the corner of the bird was out of the frame.

Fix: We reduced the height and width of the bird by 1-2 pixels to bring the hitbox closer to the body. This fix mimics the hitbox being ellipse without adding too much complexity.



DRAW FRAME MODULE: DRAW BIRD & PIPES

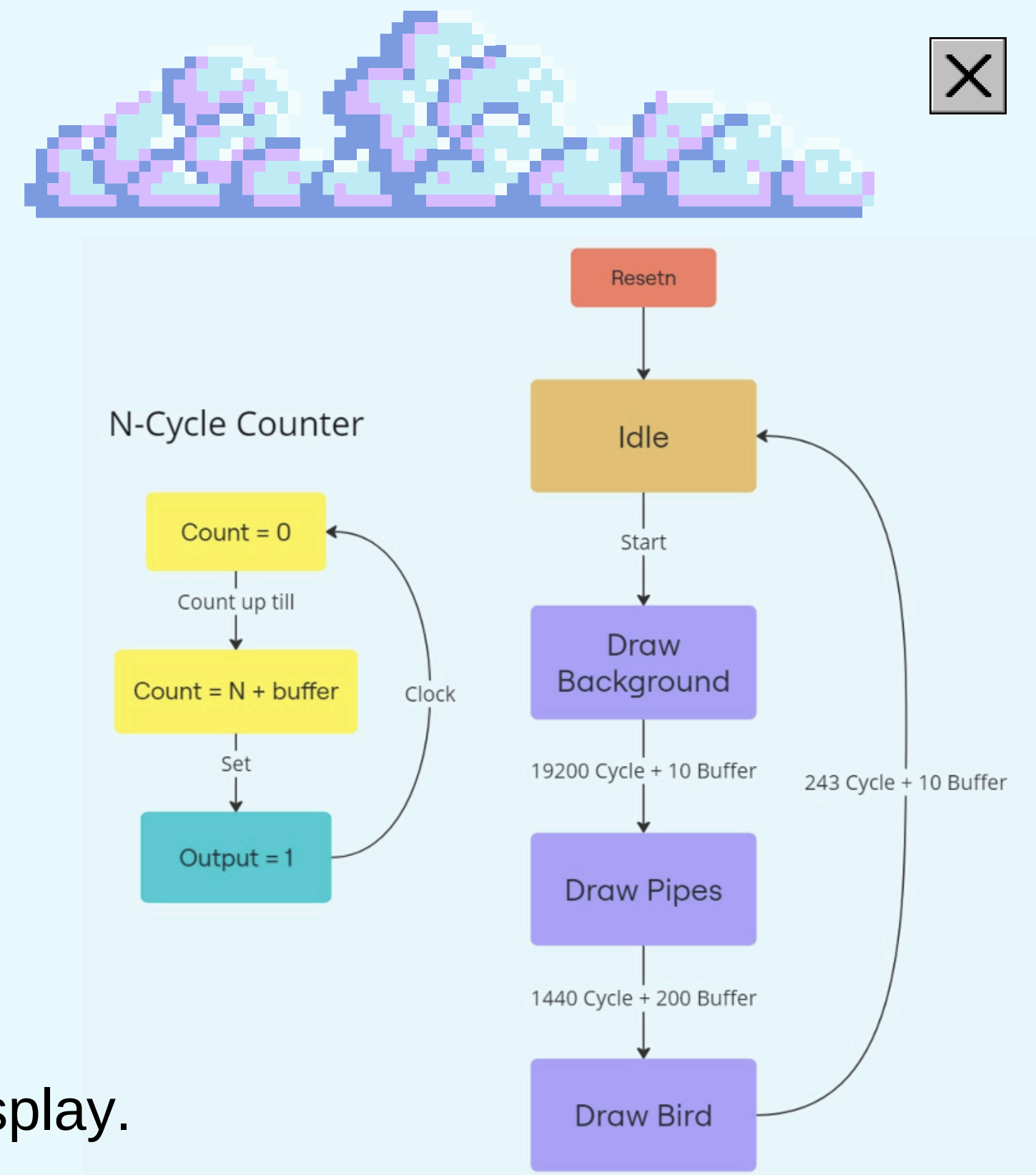
The Draw Frame module implements a timed FSM that draws the background, the pipes and the bird in order. The module draws each element after a set delay.

The N-Cycle counter counts up till N (number of pixels) plus a buffer to ensure the previous element is fully drawn before moving onto the next element.

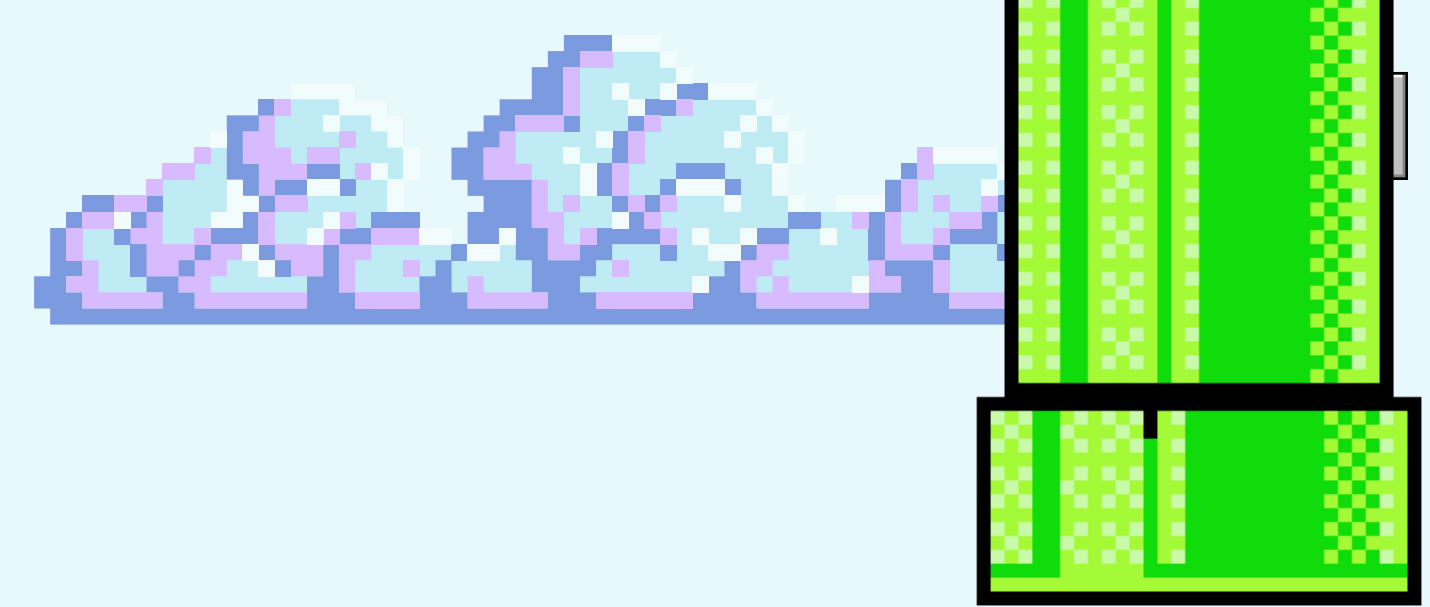
The Draw Bird module takes the y-coordinate of the bird as the input and the Draw Pipes module take the pipe height and x-coordinate as the inputs and draws the elements.

Bug: There was a lot of screen tearing while using the VGA display.

Fix: We changed the timer to display a frame every 1.68 mil clock cycles (29.76 FPS). This localized the screen tearing to only the top of the display and overall made the game look cleaner.



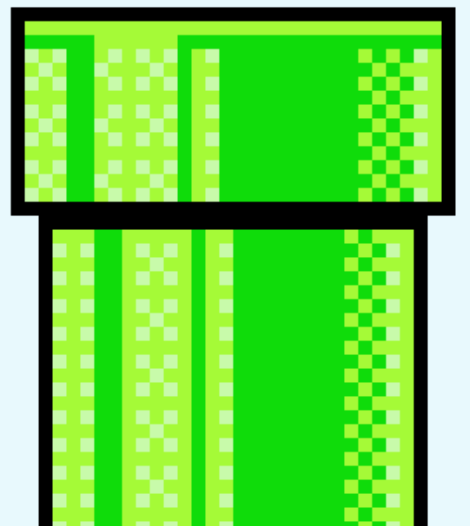
DRAW FRAME MODULE: SCREEN_FILLER MODULE



The screen_filler module outputs pixel data to fill a screen with a background image stored in memory. It reads pixel colors from a preloaded memory file (e.g., .mif) and maps them to (x, y) screen coordinates, iterating through each pixel row-by-row. The module ensures synchronized output with a clock signal and provides valid data only when the drawStart signal is active.

Bug: The Background was being overwritten every frame by the bird and pipes.

Fix: The screen_filler module was incorporated in the draw frame module to print the background every frame before any object.

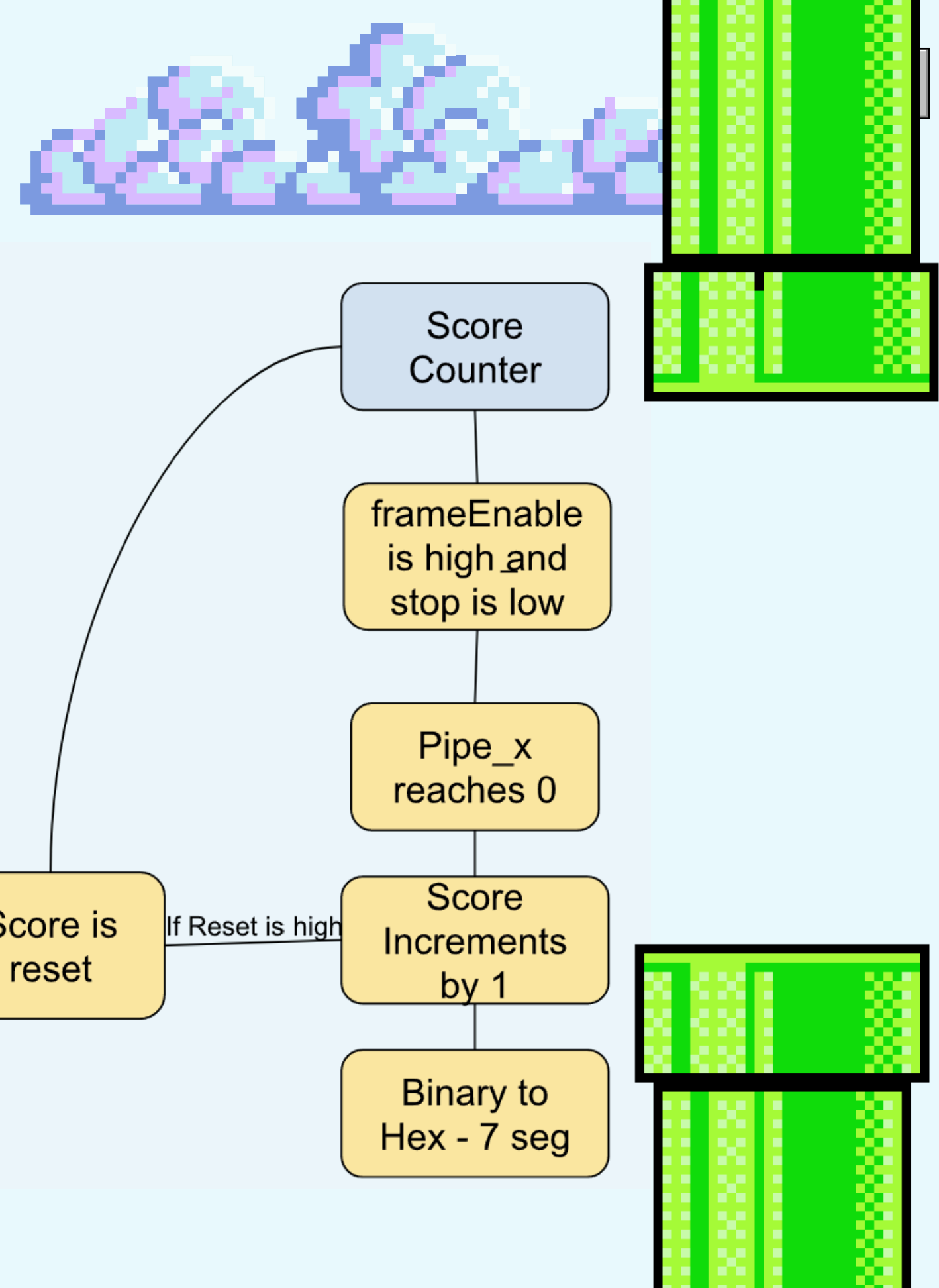
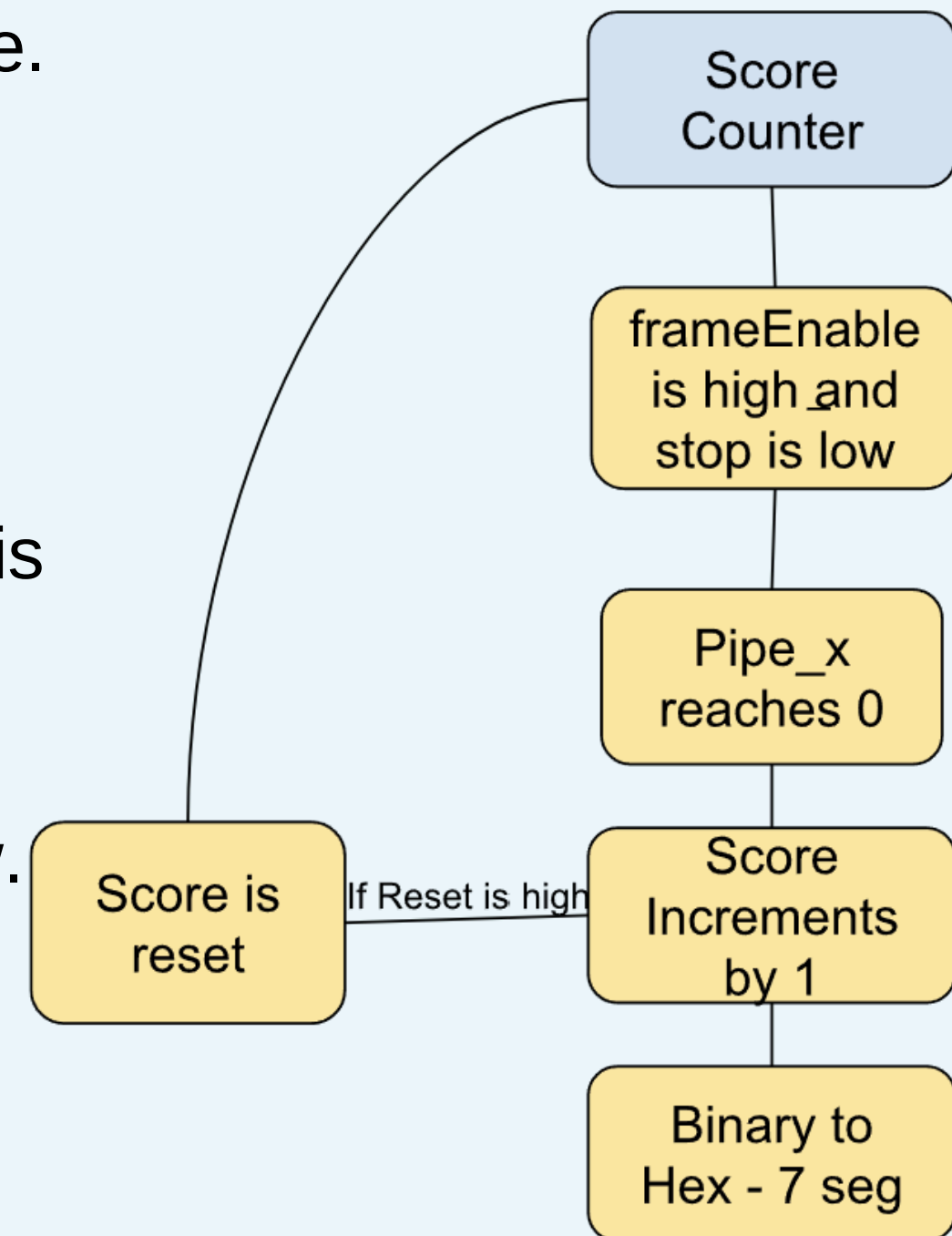


SCORE COUNTER

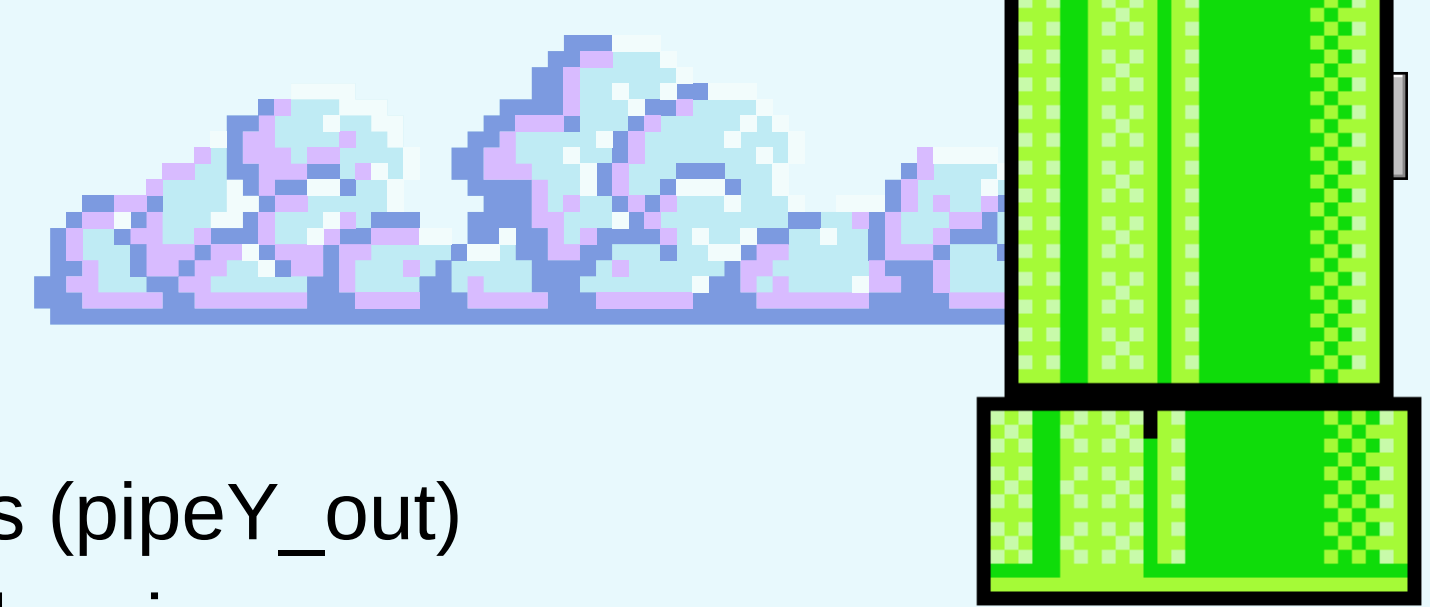
The scoreCounter module tracks the player's score in a game. It increments the score by 1 whenever the pipe_x signal (representing the position of a pipe) reaches 0, indicating the bird has successfully passed the pipe. The counter only increments if the frameEnable signal is high (indicating an active frame) and the stop signal is low (indicating the game is ongoing).

The score resets to 0 when the Resetn signal is asserted low. The module operates synchronously with the CLOCK_50 clock signal, ensuring updates occur at consistent intervals.

The score is displayed on the 7-segment in HEX.



RANDOMIZED PIPES

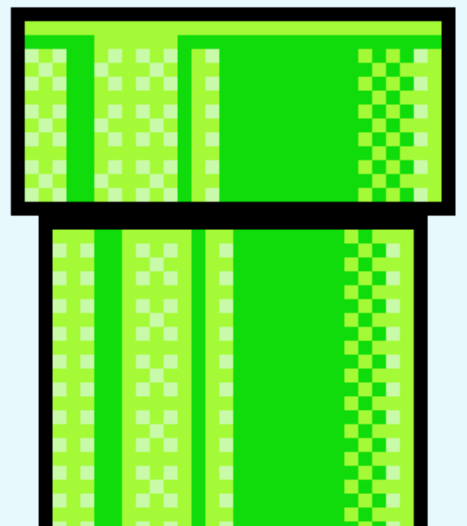


The randomizePipes module generates random vertical positions (pipeY_out) for pipes in a game. It increments a 4-bit counter (Q) each time the pipe reaches x=0 and the frameEnable signal is active, cycling through 16 predefined positions.

The mux_16 submodule uses the value of Q to select one of 16 preset heights as the output pipeY_out.

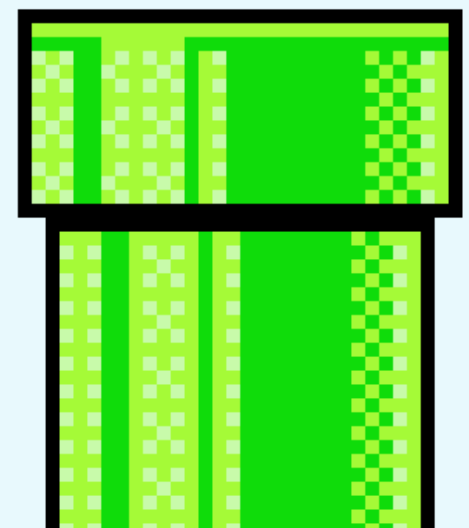
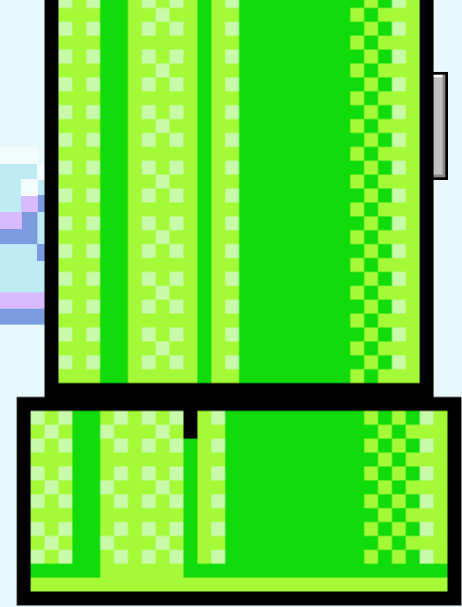
This ensures that pipes appear at varying heights, adding randomness to the game.

The counter resets when Resetn is asserted low, starting the sequence over.



FUTURE IMPROVEMENTS

- Display the score on the VGA display instead of using the 7 Segment Displays.
- Add a navigable main menu and pause screen to the game.
- Add more features to the pipes and make them truly randomized.
- Increase the resolution and colour depth of the game.
- Implement double buffering to the VGA for smoother gameplay.
- Make the hitbox of the bird the actual body of the bird and not just a rectangle around the bird.
- Implement more features in the game like speeding up after crossing a certain number of pipes and maintaining a High Score.



DISTRIBUTION OF WORK

SAPTARSHI

- Implemented the Keyboard Controller
- Implemented the updateBird/Pipe modules.
- Made a collision detection module that stops the game when the bird collides with the screen's top or bottom or the pipes.
- Implemented the drawframe - drawbird and drawpipe modules.

ROHAN

- Implemented the score counter module which increments whenever pipe_x reaches 0.
- Created a 30Hz clock to refresh and print a new frame every 33.33ms.
- Implemented the screen_filler module in draw frame to print the background every frame
- Implemented a randomize pipe module.

