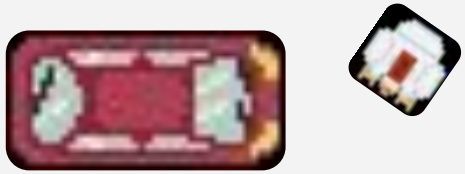
“Journey to the Other Side”

ECE241 Final Project



Why did the chicken cross the road?

By: David Jung & Christopher Lee

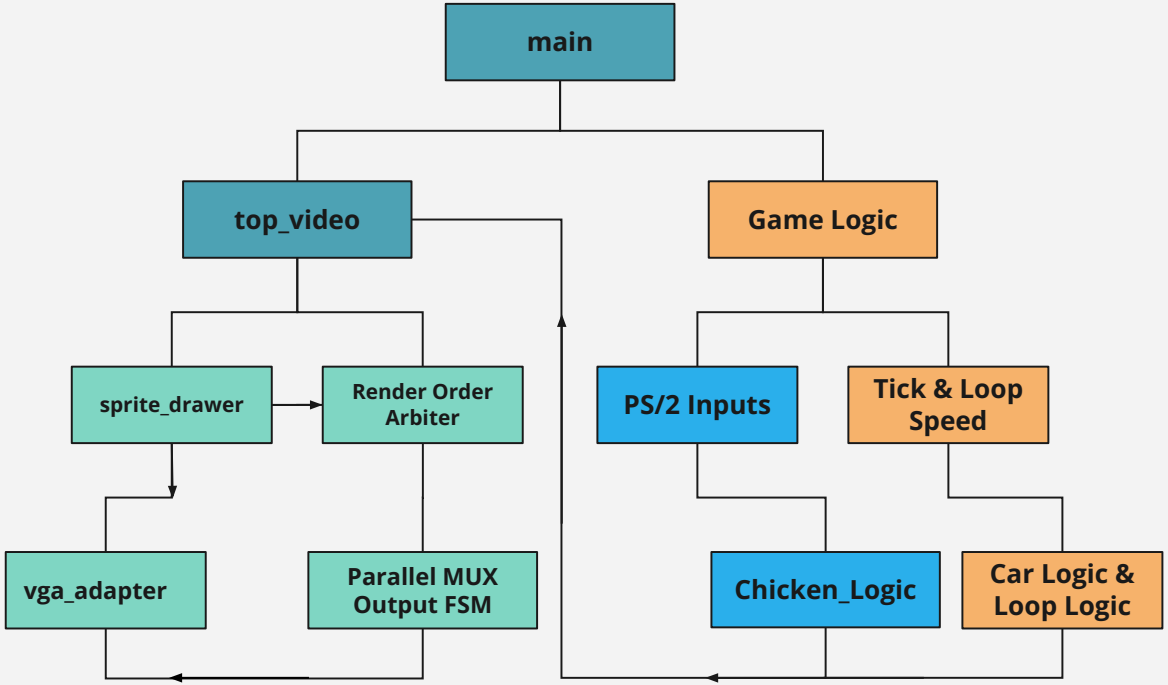
What is our game?

- Inspired by Crossy Road
- Player controls chicken with WASD
- Screen continuously scrolls down trying to catch the chicken
- Chicken needs to cross roads with cars
- Each tile it goes up by is 1 point
- Game ends when chicken is hit by car or if the screen catches them

“🐔+🚗=🛣️☠️”

Project Demo / Showcase

Project Breakdown



Top Level Modules

~400 lines

Input Controllers

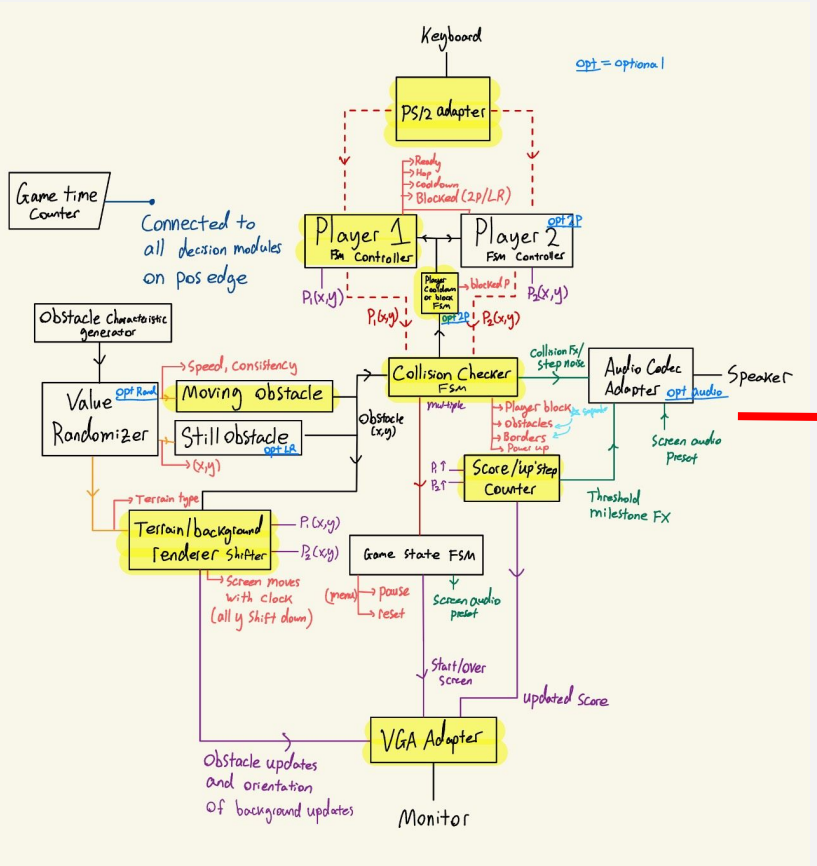
~200 lines

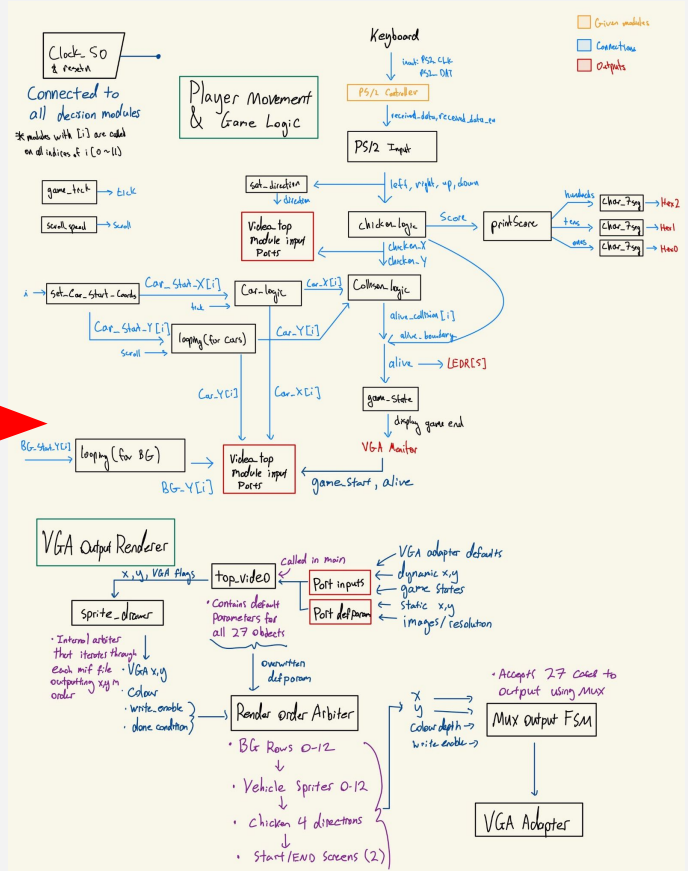
VGA Output

~1800 lines

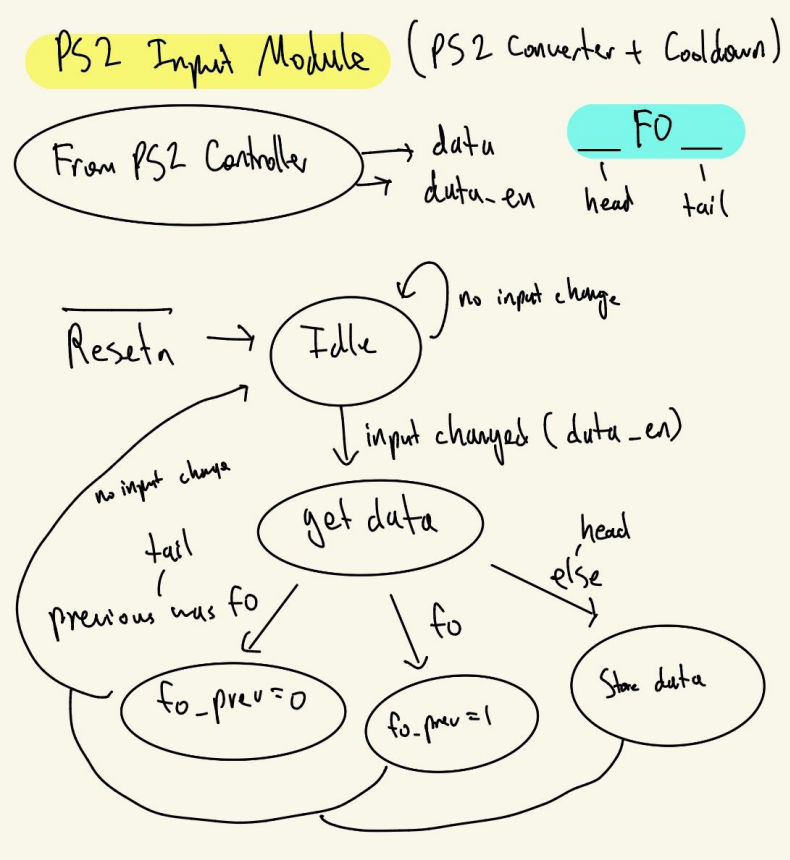
Game Logic

~300 lines



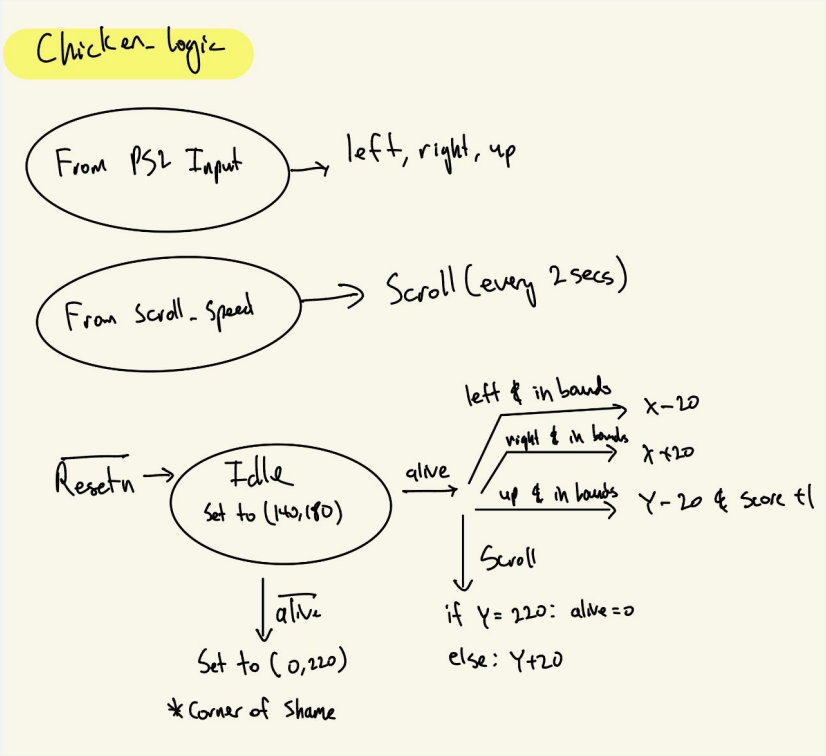


Top Level Modules




```
always @(posedge clock) begin
    if (reset == 0) begin
        left <= 0;
        right <= 0;
        up <= 0;
        down <= 0;
    f0_prev <= 0;
    end else if (received_data_en == 1) begin
     if (f0_prev == 1) begin
         left <= 0;
         right <= 0;
         up <= 0;
         down <= 0;
         f0_prev <= 0;
     end else if (received_data == 8'hF0) begin
         left <= 0;
         right <= 0;
         up <= 0;
         down <= 0;
         f0_prev <= 1;
     end else
         case (received_data)
             8'h1C: left <= 1;
             8'h23: right <= 1;
             8'h1D: up <= 1;

         endcase
 end else if (received_data_en == 0) begin
     left <= 0;
     right <= 0;
     up <= 0;
     down <= 0;
 end
```

Input Controller Modules Pt. 1

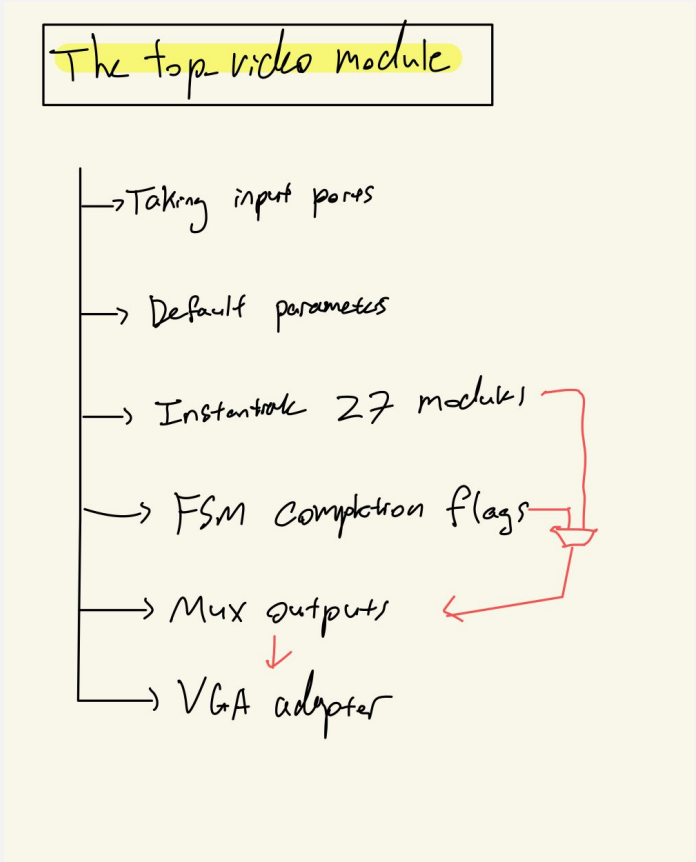



```
always @(posedge clock) begin
    if (resetn == 0) begin
        Chicken_X <= 9'd140;
        Chicken_Y <= 9'd180;
        score <= 0;
        LEDR[8:5] <= 4'b0000;
        alive_boundary <= 1;
        game_start <= 0;
    end else begin
        if (alive == 1) begin
            if (left && Chicken_X >= 60) begin
                Chicken_X <= Chicken_X - 20;
                game_start <= 1;

            end else if (right && Chicken_X <= 9'd280)  begin
                Chicken_X <= Chicken_X + 20;
                game_start <= 1;

            end else if (up && Chicken_Y >= 20) begin
                Chicken_Y <= Chicken_Y - 20;
                game_start <= 1;

                score <= score + 1;
            end else if (down && Chicken_Y <= 9'd219) begin
                Chicken_Y <= Chicken_Y + 20;
                game_start <= 1;

            end

            if (game_start) begin
                if (scroll) begin
                    if (Chicken_Y == 220) alive_boundary <= 0;
                    Chicken_Y <= Chicken_Y + 20;
                end
                if (Chicken_Y == 60) Chicken_Y <= Chicken_Y + 20;
            end
        end else begin
            Chicken_X <= 0;
            Chicken_Y <= 9'd220;
            game_start <= 0;
        end
    end
```

Input Controller Modules Pt. 2

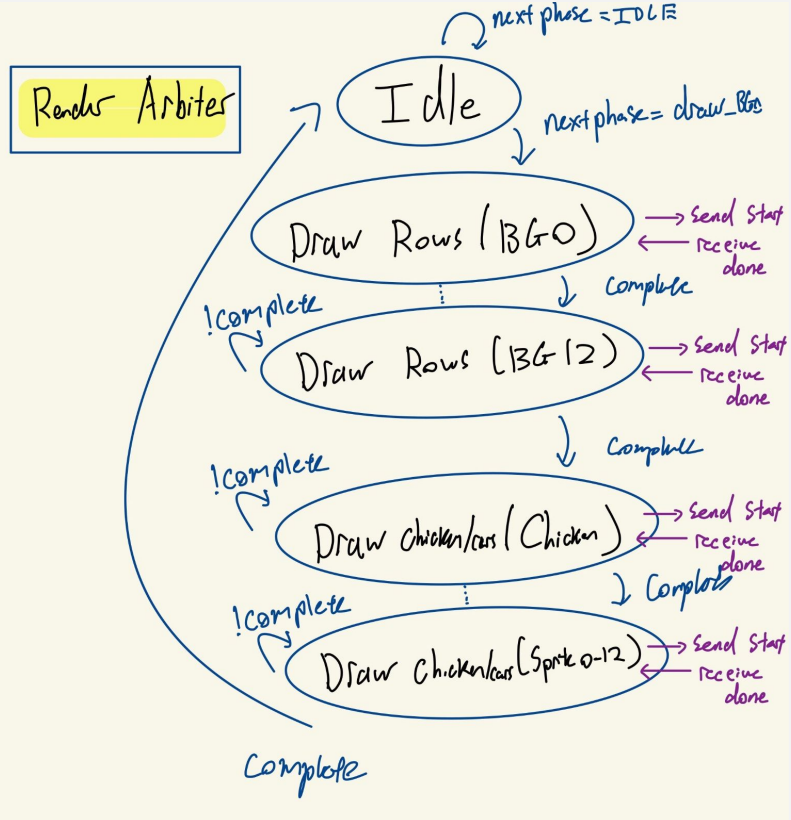
```
top_video_display (
    .CLOCK_50(CLOCK_50),
    .KEY(KEY),
    .VGA_R(VGA_R), //colour bass
    .VGA_G(VGA_G),
    .VGA_B(VGA_B),
    .VGA_HS(VGA_HS), ///horizontal sync
    .VGA_VS(VGA_VS), //vertical sync
    .VGA_BLANK_N(VGA_BLANK_N),
    .VGA_SYNC_N(VGA_SYNC_N),
    .VGA_CLK(VGA_CLK),

    // frames logic
    .tick(tick), //pass in the game
    .resetn(resetn),
    .game_start(game_start),
    .alive(alive),

    //background ports
    // Each BG*_Y is a 320x20 strip scrolled vertically by BS_Y[]
    .BG0_Y(BG_Y[0]),
    .BG1_Y(BG_Y[1]),
    .BG2_Y(BG_Y[2]),
    .BG3_Y(BG_Y[3]),
    .BG4_Y(BG_Y[4]),
    .BG5_Y(BG_Y[5]),
    .BG6_Y(BG_Y[6]),
    .BG7_Y(BG_Y[7]),
    .BG8_Y(BG_Y[8]),
    .BG9_Y(BG_Y[9]),
    .BG10_Y(BG_Y[10]),
    .BG11_Y(BG_Y[11]),

    // chicken sprite ports
    .CHICKEN_DIRECTION(direction),
    .CHICKEN_SPRITE_X(Chicken_X),
    .CHICKEN_SPRITE_Y(Chicken_Y),

    // car sprite portsys
    // Map each lane's car X/Y into a foreground sprite slot
    .SPRITE0_X(Car_X[0]),
    .SPRITE0_Y(Car_Y[0]),
    .SPRITE1_X(Car_X[1]),
    .SPRITE1_Y(Car_Y[1]),
    .SPRITE2_X(Car_X[2]),
    .SPRITE2_Y(Car_Y[2]),
    .SPRITE3_X(Car_X[3]),
    .SPRITE3_Y(Car_Y[3]),
    .SPRITE4_X(Car_X[4]),
    .SPRITE4_Y(Car_Y[4]),
    .SPRITE5_X(Car_X[5]),
    .SPRITE5_Y(Car_Y[5]),
    .SPRITE6_X(Car_X[6]),
    .SPRITE6_Y(Car_Y[6]),
    .SPRITE7_X(Car_X[7]),
    .SPRITE7_Y(Car_Y[7]),
    .SPRITE8_X(Car_X[8]),
    .SPRITE8_Y(Car_Y[8]),
    .SPRITE9_X(Car_X[9]),
    .SPRITE9_Y(Car_Y[9]),
    .SPRITE10_X(Car_X[10]),
    .SPRITE10_Y(Car_Y[10]),
    .SPRITE11_X(Car_X[11]),
    .SPRITE11_Y(Car_Y[11])
);
```

```
//Row 0 Tile
defparam display.BG0_WIDTH = 320;
defparam display.BG0_HEIGHT = 20;
defparam display.BG0_X = 0;
defparam display.BG0_IMAGE = "./MIF/dirtRoad2_320_9.mif";

//Row 1 Tile
defparam display.BG1_WIDTH = 320;
defparam display.BG1_HEIGHT = 20;
defparam display.BG1_X = 0;
defparam display.BG1_IMAGE = "./MIF/grassStrip2_320_9.mif";

//Row 2 Tile
defparam display.BG2_WIDTH = 320;
defparam display.BG2_HEIGHT = 20;
defparam display.BG2_X = 0;
defparam display.BG2_IMAGE  = "./MIF/roadStrip2_320_9.mif";

//Row 3 Tile
defparam display.BG3_WIDTH = 320;
defparam display.BG3_HEIGHT = 20;
defparam display.BG3_X = 0;
defparam display.BG3_IMAGE  = "./MIF/grassStrip2_320_9.mif";

//Row 4 Tile
defparam display.BG4_WIDTH = 320;
defparam display.BG4_HEIGHT = 20;
defparam display.BG4_X = 0;
defparam display.BG4_IMAGE  = "./MIF/roadStrip2_320_9.mif";

//Row 5 Tile
defparam display.BG5_WIDTH = 320;
defparam display.BG5_HEIGHT = 20;
defparam display.BG5_X = 0;
defparam display.BG5_IMAGE  = "./MIF/grassStrip2_320_9.mif";

//Row 6 Tile
defparam display.BG6_WIDTH = 320;
defparam display.BG6_HEIGHT = 20;
defparam display.BG6_X = 0;
defparam display.BG6_IMAGE = "./MIF/roadStrip2_320_9.mif";

//Row 7 Tile
defparam display.BG7_WIDTH = 320;
defparam display.BG7_HEIGHT = 20;
defparam display.BG7_X = 0;
defparam display.BG7_IMAGE = "./MIF/grassStrip2_320_9.mif";

//Row 8 Tile
defparam display.BG8_WIDTH = 320;
defparam display.BG8_HEIGHT = 20;
defparam display.BG8_X = 0;
defparam display.BG8_IMAGE = "./MIF/dirtRoad2_320_9.mif";

//Row 9 Tile
defparam display.BG9_WIDTH = 320;
defparam display.BG9_HEIGHT = 20;
defparam display.BG9_X = 0;
defparam display.BG9_IMAGE = "./MIF/grassStrip2_320_9.mif";

//Row 10 Tile
defparam display.BG10_WIDTH = 320;
defparam display.BG10_HEIGHT = 20;
defparam display.BG10_X = 0;
defparam display.BG10_IMAGE = "./MIF/roadStrip2_320_9.mif";
```



VGA Output Modules Pt. 1

```
//stores arbiter progress
reg [4:0] current_video_phase;
reg [4:0] next_video_phase;

// fsm states
localparam [4:0] IDLE = 5'd0;

// 14 background sprite states
localparam [4:0] DRAW_BG0 = 5'd1;
localparam [4:0] DRAW_BG1 = 5'd2;
localparam [4:0] DRAW_BG2 = 5'd3;
localparam [4:0] DRAW_BG3 = 5'd4;
localparam [4:0] DRAW_BG4 = 5'd5;
localparam [4:0] DRAW_BG5 = 5'd6;
localparam [4:0] DRAW_BG6 = 5'd7;
localparam [4:0] DRAW_BG7 = 5'd8;
localparam [4:0] DRAW_BG8 = 5'd9;
localparam [4:0] DRAW_BG9 = 5'd10;
localparam [4:0] DRAW_BG10 = 5'd11;
localparam [4:0] DRAW_BG11 = 5'd12;
// localparam [4:0] DRAW_BG12 = 5'd13;
// localparam [4:0] DRAW_BG13 = 5'd14;
// NOTE: currently only DRAW_BG0..DRAW_BG10 are used
// DRAW_BG11 is left defined but not entered by the F

// 14 sprite states (chicken is SPRITE0)
localparam [4:0] DRAW_CHICKEN = 5'd15;
localparam [4:0] DRAW_SPRITE0 = 5'd16;
localparam [4:0] DRAW_SPRITE1 = 5'd17;
localparam [4:0] DRAW_SPRITE2 = 5'd18;
localparam [4:0] DRAW_SPRITE3 = 5'd19;
localparam [4:0] DRAW_SPRITE4 = 5'd20;
localparam [4:0] DRAW_SPRITE5 = 5'd21;
localparam [4:0] DRAW_SPRITE6 = 5'd22;
localparam [4:0] DRAW_SPRITE7 = 5'd23;
localparam [4:0] DRAW_SPRITE8 = 5'd24;
localparam [4:0] DRAW_SPRITE9 = 5'd25;
localparam [4:0] DRAW_SPRITE10 = 5'd26;
localparam [4:0] DRAW_SPRITE11 = 5'd27;
// localparam [4:0] DRAW_SPRITE13 = 5'd28;
localparam [4:0] DRAW_START_SCREEN = 5'd29;
localparam [4:0] DRAW_GAMEOVER_SCREEN = 5'd30;
```




```
//=============================================================
// Priority DRAWER Arbiter FSM based on priority STATE arbiter FSM
// STEP 1: IDLE: do not draw anything
// STEP 2: DRAW_BACKGROUND: draw full background once
// STEP 3: DRAW_SPRITE: draw sprite(s) once on top
//
// after sprite done, return to IDLE and wait for next game tick input from port.
//=============================================================

// currently selected pixel info to send to VGA adapter (unqie so fsm is needed))
reg [8:0] mux_selected_pixel_x_9bit;
reg [7:0] mux_selected_pixel_y_8bit;
reg [8:0] mux_selected_pixel_color_9bit;
reg mux_selected_pixel_write_enable;


always @(*) begin
    // Default: nothing drawn
    mux_selected_pixel_x_9bit = 9'd0;
    mux_selected_pixel_y_8bit = 8'd0;
    mux_selected_pixel_color_9bit = 9'd0;
    mux_selected_pixel_write_enable = 1'b0;

    case (current_video_phase)
        // Use background module output
        DRAW_BG0: begin
            mux_selected_pixel_x_9bit = bg0_pixel_x;
            mux_selected_pixel_y_8bit = bg0_pixel_y;
            mux_selected_pixel_color_9bit = bg0_pixel_color_9bit;
            mux_selected_pixel_write_enable = bg0_pixel_write_enable;
        end

        //BACKGROUND sprites
        DRAW_BG1: begin
```

Game Logic Modules Pt. 1

Encountered Bugs Pt. 1

PS/2 Controller

PS/2 keyboard vs controller

VGA DESIM VS FPGA OH MY GOD

VGA Demos

I**ssue:** Professor brown's demo's didnt have any examples for calling multiple instantiations of .mif files + didn't account for overwriting

Solution: Adapt the vga_adapter into a sprite_drawer that could process the images sequentially

***Initially we had two drawer modules for a static and dynamic objects, but realized we needed only dynamic.

Array Complexity

Problem: Initially tried to input arrays directly into each module is a TERRIBLE IDEA

Solution: Called each module 12 times w/ inputs/outputs being the elements itself

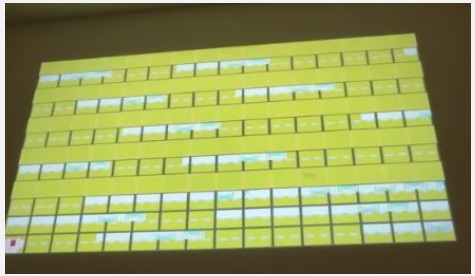
Encountered Bugs Pt 2.

DeSim VGA Adapter

Issue: We built our project on DeSim at home and realized after milestone 2 that the adapter were different.

Solution: We needed to add an RGB port and horizopntal/vertical sync ports

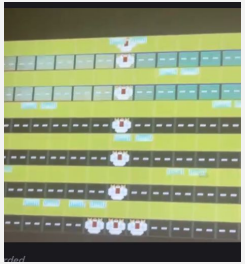
Integration Debugging



Lots of bugs were discovered once we put the theoretical logic onto a visible screen.

We worked through each of the issues together.

Shadow Clones



The shadow cloning was caused by a .mif file logic change.

Since we removed the static background and shifted everything over to our other helper, all the images that were black became transparent.

Game Improvements

Add 2 Players

We drew a second duck sprite that could have been implemented, though it would have convoluted our current code.

Add Audio

We only used 40% memory, could have had more memory intensive additions

Add Static Obstacles

More collision logic to look into, between player(s)

Double Buffering

Removes the flickering of full renders per drawn frame

Attribution Table

	David	Chris
Module Contributions	All the game logic + main: 1. Set-up modules: set_direction, PS2_Input, game_tick, scroll_speed, printScore 2. Chicken: chicken_logic 3. Car: set_Car_Start_Coords, car_logic, looping (for car) 4. Background: looping (for BG) 5. Alive condition: collision_logic	All the VGA output logic + main 1.Main top_video calls 2. Sprite_drawer helper module 3. All visual .mif files hand drawn/converted 4. Order arbiter FSM 5. Final MUX to vga_adapter call

Thanks!

Do you have any questions?

youremail@freepik.com
+34 654 321 432
yourwebsite.com