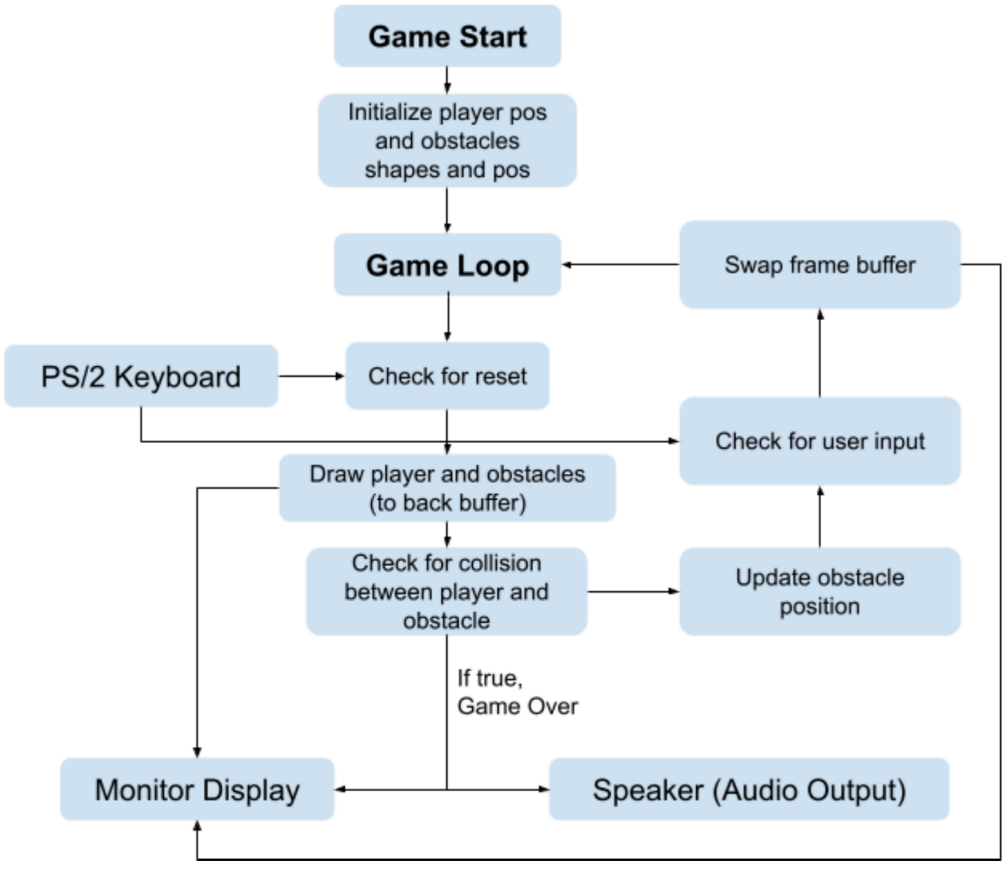
Game Description

User controls a sprite in person perspective using a PS/2 keyboard to dodge the obstacles. There will be multiple lanes with obstacles or power-ups (like a shield) spawning randomly. A point system will be displayed on the screen. The sprite will "run" faster as time goes on (so it will be harder to dodge obstacles) and when the user hits an obstacle it will be game over, an audio clip will be played, and the final score will be displayed.



Milestone 1

What did we do last week

Haozhe

1. Draw 3 objects on the screen that can automatically move downward, and make sure that these objects disappear automatically and generate new objects when they are completely below the screen.
2. Simulate the game logic part in the terminal

Wilbert

1. Implement the PS/2 Keyboard component but failed, implemented key component on DE1-SoC board instead.
2. Modified the game logic part in the terminal

What are we going to do next week

Haozhe

1. Implement all game logic parts to board
2. Create a simple start screen and game over screen

Wilbert

1. Finish the keyboard component
2. Create a game over sound generator function
3.

Code

Here's our code on the board

```
#include <stdlib.h>
#include <stdbool.h>
#include <time.h>
#define HALF_WIDTH_BOX 2
#define PLAYER_SPEED 5
#define player_pos_y 220
#define SCREEN_HEIGHT 240
#define SCREEN_WIDTH 320
#define PLYAER_X_OFFSET 3
#define PLAYER_Y_OFFSET 5

int pixel_buffer_start; // global variable
short int Buffer1[240][512]; // 240 rows, 512 (320 + padding) columns
short int Buffer2[240][512];


int player_pos_x = SCREEN_WIDTH / 2;
int old_player_pos_x;
int obstacles_pos[3][2], obstacle_height[3], obstacle_width[3];
int obstacles_old_pos[3][2], obstacle_old_height[3], obstacle_old_width[3];
int current_speed = 5;
int arrow_input = 0;

void swap(int *a, int *b);
void plot_pixel(int x, int y, short int line_color);
```

```
void draw_ranctangle(int x, int y, int half_witdh, int half_height, short
int line_color);
void clear_screen();
void wait_for_vsync();
void resetObstacle(int pos[2], int *height, int *width, int idx);
void resetObstacle2(int pos[2], int *height, int *width, int idx);
void get_button_input();
void get_keyboard_input();

int main(void)
{
    arrow_input = 0;
    srand(time(NULL));
    volatile int * pixel_ctrl_ptr = (int *)0xFF203020;
    // declare other variables(not shown)
    // initialize location and direction of rectangles(not shown)

    /* set front pixel buffer to Buffer 1 */
    *(pixel_ctrl_ptr + 1) = (int) &Buffer1; // first store the address in
the  back buffer
    /* now, swap the front/back buffers, to set the front buffer location
*/
    wait_for_vsync();
    /* initialize a pointer to the pixel buffer, used by drawing functions
*/
    pixel_buffer_start = *pixel_ctrl_ptr;
    clear_screen(); // pixel_buffer_start points to the pixel buffer

    /* set back pixel buffer to Buffer 2 */
    *(pixel_ctrl_ptr + 1) = (int) &Buffer2;
    pixel_buffer_start = *(pixel_ctrl_ptr + 1); // we draw on the back
buffer
    clear_screen(); // pixel_buffer_start points to the pixel buffer

    // draw player for the first time
    old_player_pos_x = player_pos_x;
    draw_ranctangle(player_pos_x, player_pos_y, PLYAER_X_OFFSET,
PLAYER_Y_OFFSET, 0xFFFF);

    // draw the obstacles for the first time
    for (int i = 0; i < 3; ++i) {
        resetObstacle(obstacles_pos[i], &obstacle_width[i],
&obstacle_height[i], i);
        obstacles_pos[i][1] -= 150;
        obstacles_old_pos[i][0] = obstacles_pos[i][0];
        obstacles_old_pos[i][1] = obstacles_pos[i][1];
        obstacle_old_height[i] = obstacle_height[i];
        obstacle_old_width[i] = obstacle_width[i];
        draw_ranctangle(obstacles_pos[i][0], obstacles_pos[i][1],
obstacle_width[i]/2, obstacle_height[i]/2, 0xFFC0CB);
    }
```

```
    for (;;) {
        // ==================clear screen==================
        // clear player
        draw_ranctangle(old_player_pos_x, player_pos_y, PLYAER_X_OFFSET +
PLAYER_SPEED, PLAYER_Y_OFFSET, 0);

        // clear obstacles
        for (int i = 0; i < 3; ++i) {
            draw_ranctangle(obstacles_old_pos[i][0], obstacles_old_pos[i]
[1], obstacle_old_width[i]/2 + current_speed, obstacle_old_height[i]/2 +
current_speed, 0);
        }
        // ================end if clear screen===============

        // ================draw new elements=================
        // draw new player
        draw_ranctangle(player_pos_x, player_pos_y, PLYAER_X_OFFSET,
PLAYER_Y_OFFSET, 0xFFFF);

        // draw the new obstacles
        for (int i = 0; i < 3; ++i) {
            draw_ranctangle(obstacles_pos[i][0], obstacles_pos[i][1],
obstacle_width[i]/2, obstacle_height[i]/2, 0xFFC0CB);
        }
        // Wait for vertical sync to swap buffers
        // =============end of draw new elements==============

        wait_for_vsync();

        // Update the pixel_buffer_start pointer to the new back buffer
        pixel_buffer_start = *(pixel_ctrl_ptr + 1);




        // ==========Update the previous position===========
        old_player_pos_x = player_pos_x;
        for (int i = 0; i < 3; ++i) {
            obstacles_old_pos[i][0] = obstacles_pos[i][0];
            obstacles_old_pos[i][1] = obstacles_pos[i][1];
            obstacle_old_height[i] = obstacle_height[i];
            obstacle_old_width[i] = obstacle_width[i];
        }
        // ======end of Update the previous position=======




        // ===========update elements position=============
        // update player position
        get_button_input();
```

```
        switch (arrow_input)
        {
        case 1:
            if (player_pos_x <= SCREEN_WIDTH - 6)
                player_pos_x += PLAYER_SPEED;
            arrow_input = 0;
            break;
        case 2:
            if (player_pos_x >= 6)
                player_pos_x -= PLAYER_SPEED;
            arrow_input = 0;
            break;
        }
        // update obstacle postion
        for (int i = 0; i < 3; ++i) {
            if (obstacles_pos[i][1] - obstacle_height[i] - 1 >=
SCREEN_HEIGHT) {
                resetObstacle(obstacles_pos[i], &obstacle_height[i],
&obstacle_width[i], i);
            } else {
                obstacles_pos[i][1] += current_speed;
            }
        }
        // ========end ofupdate elements position==========
    }

}

void plot_pixel(int x, int y, short int line_color)
{
    volatile short int *one_pixel_address;
    one_pixel_address = pixel_buffer_start + (y << 10) + (x << 1);
    *one_pixel_address = line_color;
}

void swap(int *a, int *b) {
    int temp = *a;
    *a = *b;
    *b = temp;
}

void draw_ranctangle(int x, int y, int half_witdh, int half_height, short
int line_color) {
    for (int i = ((x - half_witdh >= 0) ? (x - half_witdh) : 0); i < ((x +
half_witdh) <= SCREEN_WIDTH ? (x + half_witdh) : SCREEN_WIDTH); ++i) {
        for (int j = ((y - half_height) >= 0 ? (y - half_height) : 0); j <
((y + half_height) <= SCREEN_HEIGHT ? (y + half_height) : SCREEN_HEIGHT);
++j) {
            plot_pixel(i, j, line_color);
        }
    }
}
```

```
void wait_for_vsync() {
    volatile int * pixel_ctrl_ptr = (int *)0xFF203020;
    * pixel_ctrl_ptr = 1;
    while(*(pixel_ctrl_ptr + 3) & 0b1);
}

void clear_screen() {
    for (int i = 0; i < SCREEN_WIDTH; ++i) {
        for (int j = 0; j < SCREEN_HEIGHT; ++j) {
            plot_pixel(i, j, 0);
        }
    }
}

void resetObstacle(int pos[2], int *height, int *width, int idx) {
    *height = rand() % 80 + 50; // Make sure it's not zero
    *width = 70 + rand() % 20;
    pos[0] = *width / 2 + (idx * 80); // 320 / 3 = 80
    pos[1] = -(int)((100 + rand() % 41) * idx);
}

void get_button_input() {
    volatile int* buttons_ptr = (int *)0xFF200050;
    int button_data = *(buttons_ptr);

    if (button_data & 0b0001) { // key 0
        arrow_input = 1;
    } else if (button_data & 0b0010) { // key 1 pressed
        arrow_input = 2;
    }
}

void get_keyboard_input() {
    volatile int * keyboard_ptr = (int * )0xFF200100;
    int input[3] = {0};
    int inputIndex = 2;

    if (*(keyboard_ptr) & 0xF) {
        for (int i = 0; i < (*(keyboard_ptr) & 0xFFFF0000); i++) {
            input[inputIndex] = *(keyboard_ptr) & 0x7; // save the input
            inputIndex--;
            if (inputIndex == -1) { // reset index
                inputIndex = 2;
            }
            if (input[0] == 0xE0 && input[1] == 0xF0 && input[2] == 0x6B) {
// left arrow key release
                arrow_input = 1;
                break;
            } else if (input[0] == 0xE0 && input[1] == 0xF0 && input[2] ==
0x74) { // right arrow key release
                arrow_input = 2;
```

```
                break;
            } else if (input[0] == 0xE0 && input[1] == 0xF0 && input[2] ==
0x75) { // up arrow key release
                arrow_input = 3;
                break;
            } else if (input[0] == 0xE0 && input[1] == 0xF0 && input[2] ==
0x72) { // down arrow key release
                arrow_input = 4;
                break;
            }
        }
    }
}
```