‘C’ Programming Project
ON
‘Ball Game’

Title: ‘Ball Game’

Made by: Marjan Hossain Mahin

Technologies: C, Raylib

Assigned Date: 16/4/23

Github: https://github.com/MarjanHossain01/raylibFlappyBird

Table of Contents

Introduction **2**
Structs **2**
Functions **3**
Flowchart 4
Screenshots: 5

Introduction

The following project is a Ball Game where you fly as a bird, evading oncoming obstacles as a test of patience and persistence.The game was made with C (Created by the visionary genius Dennis Ritchie) and Raylib (a cross-platform open-source graphics library). The game was created in about ***6 days*** upon being assigned. The game consists of ***3 user-defined structs*** *and* ***7 user-defined functions****.* Like any other game, this game relies on a ***game loop*** which is responsible for displaying frames and updating various properties (i.e objects, scores, etc.) In this game, the game loop exists inside the main function.

User interaction is straightforward, with responsive controls allowing players to navigate using keyboard inputs. The game's interface features clear buttons and labels for easy navigation. For the textures and basic graphics, a graphics module maybe required in order to load these properly.

Structs

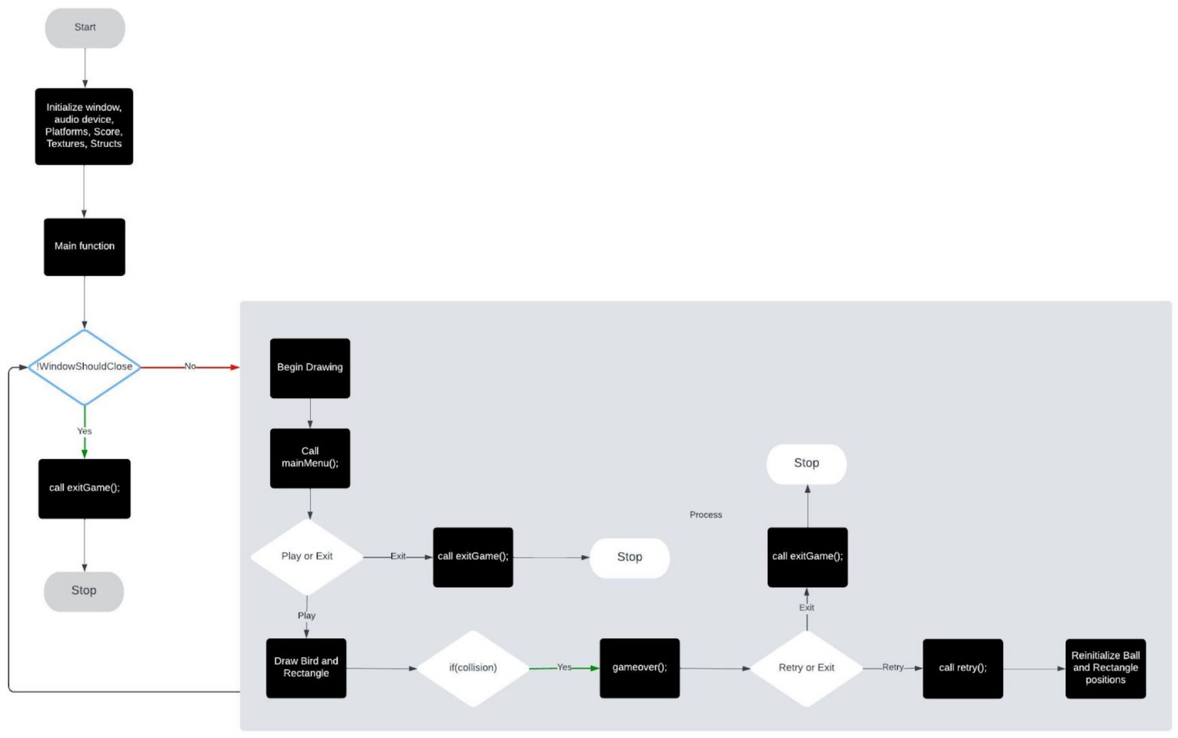
Name	Properties	Description
Bird	**Float** width, **Float** Height **Vector2** Position, **Vector2** Velocity, **Texture2D** Texture	The 'bird' struct contains properties of the bird the user plays as. Float width and Float height control the size of the bird, The Vector2 Position and Vector2 Velocity control the movement of the bird. The Texture2D texture property loads the texture of the bird.
Platform	**Int** x **Int** y **Int** width **Int** height	This struct contains properties for obstacles which the bird has to face. Int x and Int y controls the position, and Int width and Int height controls the size of the obstacles It has 2 members: Top_platforms (obstacles in the top of the screen) and Bottom_platforms (obstacles in the bottom of the screen).
Button	**Rectangle** rect, **Color** color, **Color** textcolor, **char*** text	The 'Button' struct holds properties used for buttons. Buttons in Raylib are created by combining a Rectangle and Text together and detecting mouse input. Here we have a rect property declared with the **Rectangle** type, one of the built-ins of Raylib. Then we have color and textcolor declared with the **Color** type. Another one is text, for the text inside the button. This struct has three members: **Retry**, **Exit** and **Play.**

Functions

Type	Name	Arguments	Description
int	main	**None**	The main function in this game contains the **gameloop** and initializes the window and struct members such as platforms. It is also responsible for loading textures, sound, and drawing the graphics for the game.
Int	mainMenu	**None**	The mainMenu function displays the logo of the game, and gives the user 2 options: **Play** and **Exit**
void	Update_platforms	**Platform** Top_platforms[], **Platform** Bottom_platforms[]	This function takes two arguments: **Platform Top_platforms[]** and **Platform Bottom_platforms[]**. It is called from the **main** function to create obstacles and update their positions respectively. Obstacles that appear at the top and bottom of the screen are divided into **2** arrays: **Top_platforms** and **Bottom_platforms.** This function draws **4** rectangles on top and **4** at the bottom. Once these rectangles cross the left side of the screen (0), their positions get reset to the right side of the screen (820).
int	collision	**Platform** Top_platforms[], **Platform** Bottom_platforms[]	This function checks if the ball has touched any of the rectangles, the bottom or top of the screen. If the ball collides with any of the mentioned surfaces, it returns **1** to the main function and the main function calls the **gameOver** function.
int	scoreUpdate	**Platform** Top_platforms[] **int*** score	This function increases the score whenever the ball goes through the rectangles. The **crossed** static boolean variable checks if the ball is still on the platform or not. If the ball is still on the platform, the score will not go up. This was done because the score was initially being incremented by 4 for all the platforms.

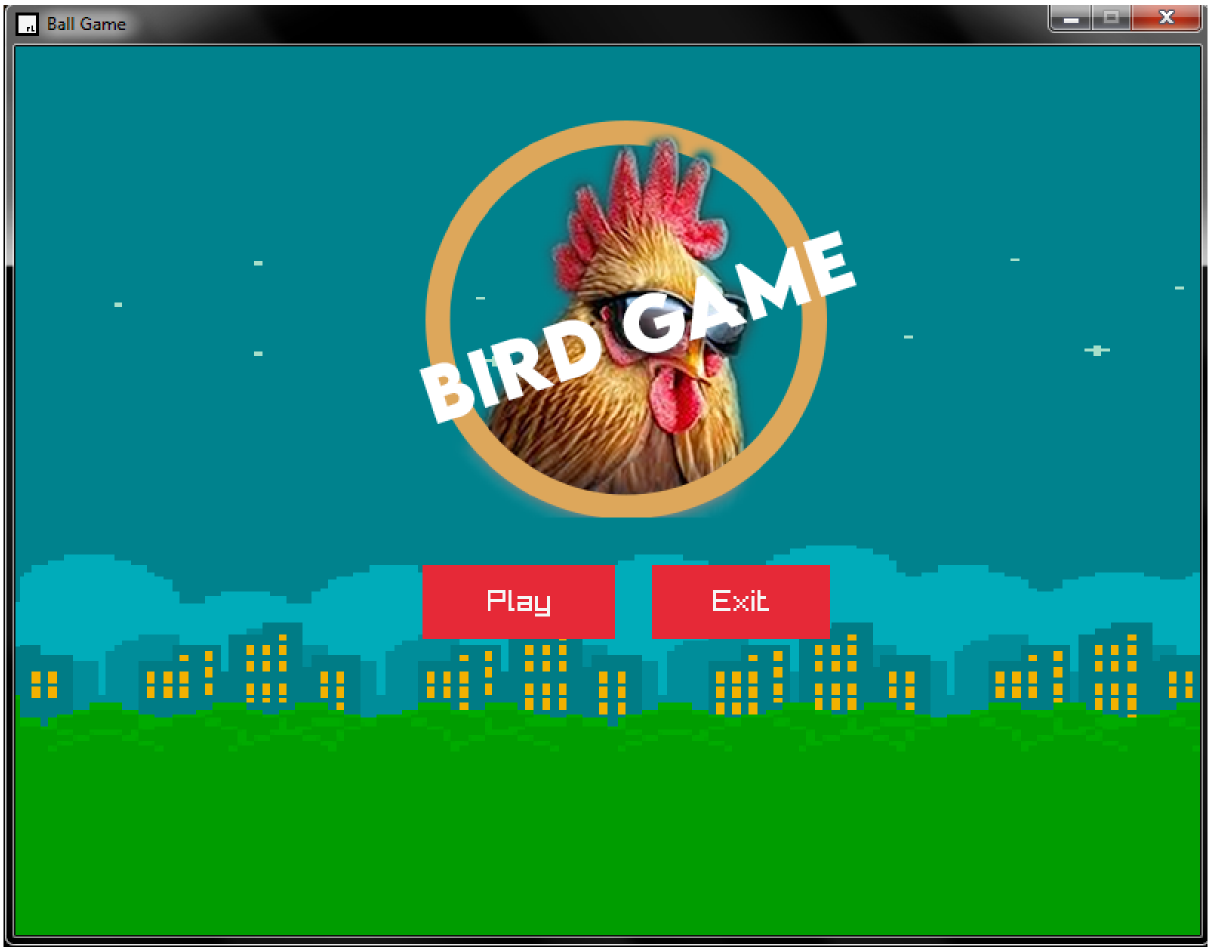
int	gameOver	**Platform** Top_platforms[] **Platform** Bottom_platforms[] **int*** score	This function stops the ball and the obstacles from moving upon the ball colliding with any object. It then presents the user with 2 buttons: **Retry** and **Exit**. The user can select the Retry button with which the retry function will be called, or can select exit with which the exitgame function is called.
void	retry	**Platform** Top_platforms[] **Platform** Bottom_platforms[]	This function resets the ball and rectangle position and resets the score to 0.
void	exitgame	**None**	This function exits the game.

Flowchart

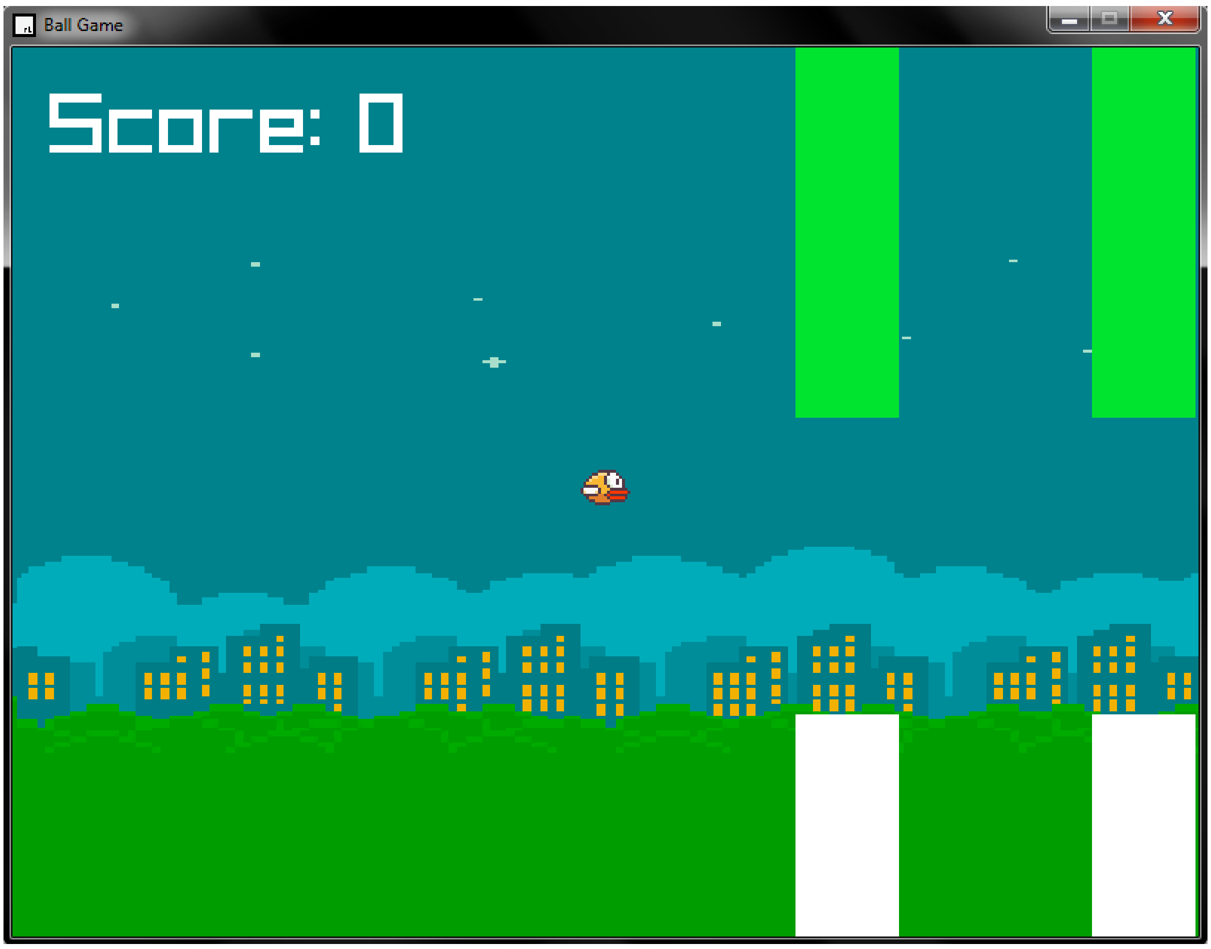

Start
Initialize window, audio device, Platforms, Score, Textures, Structs
Main function
!WindowShouldClose
No
Yes
call exitGame();
Stop
Begin Drawing
Call mainMenu();
Play or Exit
Exit
call exitGame();
Stop
Play
Draw Bird and Rectangle
if(collision)
Yes
gameover();
Retry or Exit
Exit
call exitGame();
Stop
Retry
call retry();
Reinitialize Ball and Rectangle positions
Process

Screenshots:

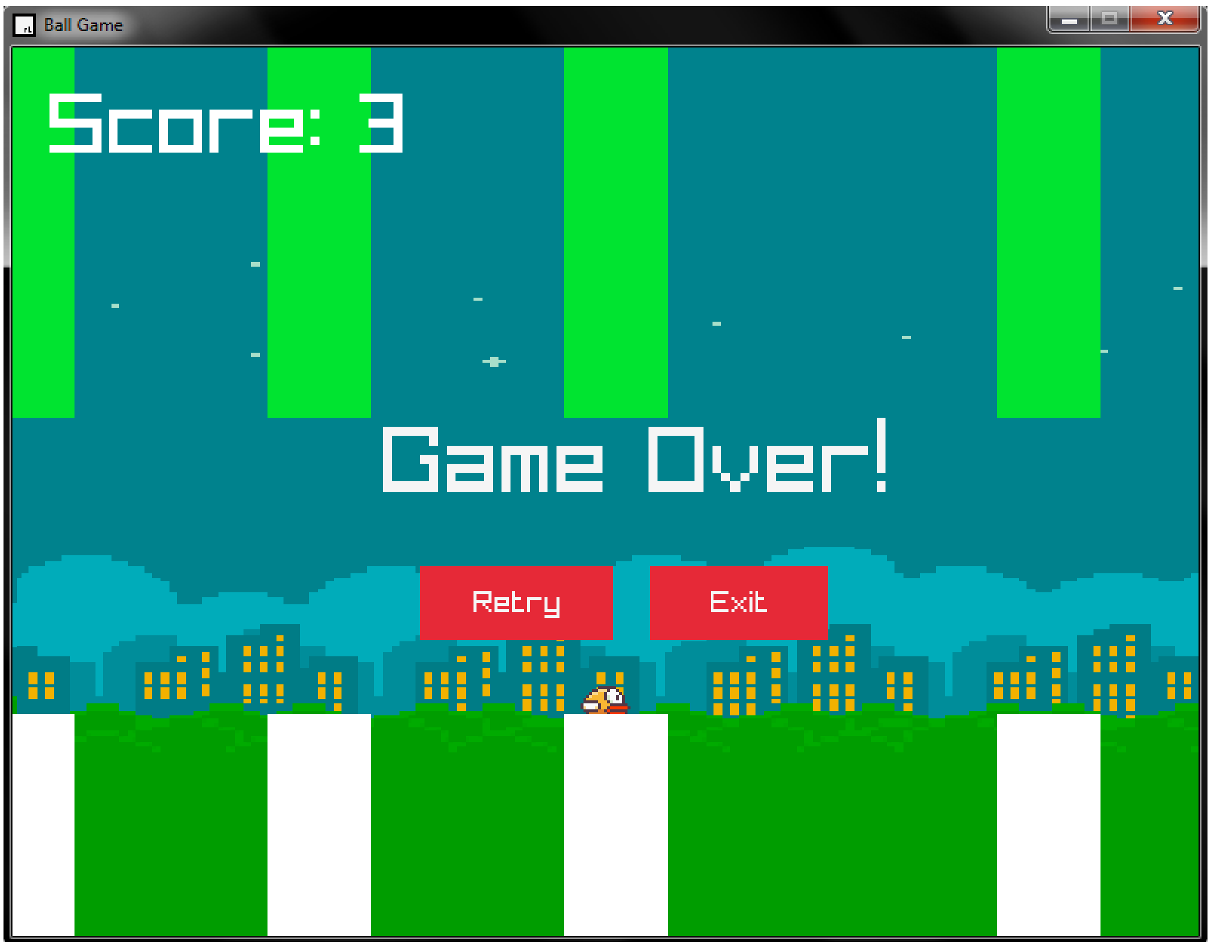
Menu:



Game Start:



Game End:



<u>Code</u>

```
#include "raylib.h"
#include <stdio.h>
#include <time.h>

const int width = 800;
const int height = 600;

bool death = false;

// Bird Properties:
typedef struct Bird {
    float width;
    float height;
    Vector2 position;
    Vector2 velocity;
    Texture2D Texture;
} Bird;

Bird bird = {
    .width = 34,
    .height = 24,
    .position = {(float)width/2, (float)height/2},
    .velocity = {0, 0}, // Initialize velocity to {0, 0}
    .Texture = {0} // Initialize texture
};

// Platform Properties
typedef struct Platform {
    int x;
    int y;
    int width;
    int height;
} Platform;

typedef struct Button {
    Rectangle rect;
    Color color;
    Color textcolor;
    char* text;
} Button;

struct Button Retry = {
    .rect = {width/2-125, height/2+50, 130, 50},
    .color = RED,
    .textcolor = RAYWHITE,
    .text = "Retry"
};

struct Button Exit = {
    .rect = {width/2+30, height/2+50, 120, 50},
    .color = RED,
    .textcolor = RAYWHITE,
    .text = "Exit"
};

struct Button Play = {
    .rect = {width/2-125, height/2+50, 130, 50},
    .color = RED,
    .textcolor = RAYWHITE,
    .text = "Play"
};

//Function Declarations
int mainMenu();
void Update_platforms(Platform Top_platforms[], Platform Bottom_platforms[]); //Obstacle
generation
int collision(Platform Top_platforms[], Platform Bottom_platforms[]); //Collision detection
int scoreUpdate(Platform Top_platforms[], int* score); //Score Function
int gameOver(Platform Top_platforms[], Platform Bottom_platforms[], int* score); // Game
over stage
void retry(Platform Top_platforms[], Platform Bottom_platforms[]); //Retry
void exitGame(); //Exit

int mainMenu()
{
    Texture2D logo = LoadTexture("F:\\Code\\Project\\RaylibBallGame\\assets\\logo.png");
    DrawTexture(logo, 270, 50, WHITE);

    DrawRectangleRec(Play.rect, Play.color);
        DrawText(Play.text, Play.rect.x + Play.rect.width / 2 - MeasureText(Play.text, 20) / 2,
        Play.rect.y + Play.rect.height / 2 - 10, 20, Play.textcolor);

   DrawRectangleRec(Exit.rect, Exit.color);
        DrawText(Exit.text, Exit.rect.x + Exit.rect.width / 2 - MeasureText(Exit.text, 20) / 2,
        Exit.rect.y + Exit.rect.height / 2 - 10, 20, Exit.textcolor);

   if(CheckCollisionPointRec(GetMousePosition(), Play.rect) &&
IsMouseButtonPressed(MOUSE_BUTTON_LEFT))
   {
      return 1;
   }
```

```
  if(CheckCollisionPointRec(GetMousePosition(), Exit.rect) &&
IsMouseButtonPressed(MOUSE_BUTTON_LEFT))
  {
     exitGame();
  }
  return 0;
}

void Update_platforms(Platform Top_platforms[], Platform Bottom_platforms[]) {
   int i;
   for (i = 0; i < 4; i++) {
       DrawRectangle(Top_platforms[i].x, Top_platforms[i].y, Top_platforms[i].width,
Top_platforms[i].height, GREEN);
       if (Top_platforms[i].x + Top_platforms[i].width < 0) { // if platforms go outside of the
screen they will be teleported to the right side
           Top_platforms[i].x = 820;
       }
       DrawRectangle(Bottom_platforms[i].x, 450, Bottom_platforms[i].width, 150, WHITE);
       if (Bottom_platforms[i].x + Bottom_platforms[i].width < 0) {
           Bottom_platforms[i].x = 820;
       }
       if(death != true)
       {
           Top_platforms[i].x -= 4;
           Bottom_platforms[i].x -= 4;
       }
   }
}

int collision(Platform Top_platforms[], Platform Bottom_platforms[]) {
   if((bird.position.y + bird.height/2) >= height){ // if bird touches the bottom of the screen
       return 1;
   }
   if((bird.position.y - bird.height/2) <= 0){ // if bird touches the top of the screen
       return 1;
   }

   for (int i = 0; i < 4; i++) { // If bird touches any rectangles
       if (CheckCollisionCircleRec(bird.position, bird.height/2, (Rectangle){Top_platforms[i].x,
           Top_platforms[i].y, Top_platforms[i].width, Top_platforms[i].height})) {
           return 1;
       }
       if (CheckCollisionCircleRec(bird.position, bird.height/2,
(Rectangle){Bottom_platforms[i].x,
           Bottom_platforms[i].y, Bottom_platforms[i].width, Bottom_platforms[i].height})) {
           return 1;
       }
```

```
    }
    return 0; // No collision detected
}

int scoreUpdate(Platform Top_platforms[], int* score)
{
    static bool crossed[4] = {false}; // check if the bird has crossed each platform
    for(int i = 0; i < 4; i++) {
        if(bird.position.x > Top_platforms[i].x + Top_platforms[i].width &&
           bird.position.x - bird.width/2 <= Top_platforms[i].x + Top_platforms[i].width &&
           !crossed[i]) //if the ball passes the rectangle and it already hasnt been crossed, score
will go up
        {
           (*score)++;
           crossed[i] = true; // Mark the platform as crossed
        }
        else if (bird.position.x <= Top_platforms[i].x + Top_platforms[i].width) {
            crossed[i] = false; //unmark the check if the bird is still on the platform
        }
    }

    char scoreText[20]; //Display score
    snprintf(scoreText, 20, "Score: %d", *score);
    DrawText(scoreText, 25, 25, 58, WHITE);

    return *score;
}

int gameOver(Platform Top_platforms[], Platform Bottom_platforms[], int* score) {
    bird.velocity = (Vector2){0, 0}; // Stop bird from moving
    death = true;

    DrawText("Game Over!", 250, 250, 62, RAYWHITE);
    {
        DrawRectangleRec(Retry.rect, Retry.color);
        DrawText(Retry.text, Retry.rect.x + Retry.rect.width / 2 - MeasureText(Retry.text, 20) / 2,
//Text
        Retry.rect.y + Retry.rect.height / 2 - 10, 20, Retry.textcolor);

        DrawRectangleRec(Exit.rect, Exit.color);
        DrawText(Exit.text, Exit.rect.x + Exit.rect.width / 2 - MeasureText(Exit.text, 20) / 2,
//Text
        Exit.rect.y + Exit.rect.height / 2 - 10, 20, Exit.textcolor);
    }

    if (CheckCollisionPointRec(GetMousePosition(), Retry.rect) &&
IsMouseButtonPressed(MOUSE_BUTTON_LEFT)){
        *score = 0;
        retry(Top_platforms, Bottom_platforms);
    }
    if (CheckCollisionPointRec(GetMousePosition(), Exit.rect) &&
IsMouseButtonPressed(MOUSE_BUTTON_LEFT)){
        exitGame();
    }
}

void retry(Platform Top_platforms[], Platform Bottom_platforms[]) {
    bird.position.x = width / 2;
    bird.position.y = height / 2;
    death = false;

    for (int i = 0; i < 4; i++) {
        Top_platforms[i].x = 800 + i * 200;
        Bottom_platforms[i].x = 800 + i * 200;
    }
}

void exitGame(){
    EndDrawing();
    CloseWindow();
}

int main() {
    InitWindow(width, height, "Ball Game");
    InitAudioDevice();

    int score = 0;

    //Initializing Platforms
    Platform Top_platforms[4];
    Platform Bottom_platforms[4];
    for (int i = 0; i < 4; i++) {
        Top_platforms[i].x = 800 + i * 200;
        Top_platforms[i].y = 0;
        Top_platforms[i].width = 70;
        Top_platforms[i].height = 250;

        Bottom_platforms[i].x = 800 + i * 200;
        Bottom_platforms[i].y = 450;
        Bottom_platforms[i].width = 70;
        Bottom_platforms[i].height = 150;
    }

    //Assets
    Texture2D background =
LoadTexture("F:\\Code\\Project\\RaylibBallGame\\assets\\background.png");
```

```
    Sound jump = LoadSound("F:\\Code\\Project\\RaylibBallGame\\assets\\boom.mp3");
    bird.Texture = LoadTexture("F:\\Code\\Project\\RaylibBallGame\\assets\\bird.png");

    // Draw the main menu once to display buttons
    int startGame = mainMenu();

    SetTargetFPS(60);
    while (!WindowShouldClose()) {
        BeginDrawing();
        ClearBackground(BLUE);
        DrawTexture(background, 0, 0, RAYWHITE);

        if (startGame == 0) {
            startGame = mainMenu(); // Update startGame based on menu input
        }
        else {
            DrawTexture(bird.Texture, bird.position.x - bird.width/2, bird.position.y - bird.height/2,
WHITE);
```

Total Lines: ***270 (after refactoring)***