Universidad Autónoma de Baja California

Facultad de Ingeniería, Arquitectura y Diseño

Materia:

Programacion estructurada

Actividad:

Proyecto final - Juego

Integrantes:

Tony Ozuna Ceseña

Miguel Angel Ramirez Monjaraz

Grupo:

Ingeniero en Computación - 432

Profesor:

Pedro Nunez Yepiz

Ensenada, Baja California a 12 de Diciembre del 2023.

```c
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include <conio.h>
#include <stdio.h>
#include "raylib.h"
#include "raymath.h"
//----------------------------------------------------------------------------------//
//                                  CONSTANTES                                      //
//----------------------------------------------------------------------------------//
#define PLAYER_LIFES 5
#define BRICKS_LINES 5
#define BRICKS_PER_LINE 20
#define BRICKS_POSITION_Y 50
#define GAME_DURATION 30 // Duración del juego en segundos

//----------------------------------------------------------------------------------//
//                                  ESTRUCTURAS                                     //
//----------------------------------------------------------------------------------//
typedef enum GameScreen
{
    LOGO,
    TITLE,
    GAMEPLAY,
    ENDING
} GameScreen;

// Player structure
typedef struct Player
{
    Vector2 position;
    Vector2 speed;
    Vector2 size;
    Rectangle bounds;
    int lifes;
} Player;

// Ball structure
typedef struct Ball
{
    Vector2 position;
    Vector2 speed;
    float radius;
    bool active;
} Ball;

// Bricks structure
typedef struct Brick
{
    Vector2 position;
    Vector2 size;
    Rectangle bounds;
    int resistance;
    bool active;
} Brick;

typedef struct
{
    char pregunta[200];
    bool respuestacorrecta;
} Pregunta;

//----------------------------------------------------------------------------------//
//                                  FUNCIONES                                       //
//----------------------------------------------------------------------------------//

void inicializaPreguntas(Pregunta estructura[31]);
Pregunta generaPregunta(const char *enunciado, bool respuestaCorrecta);
//----------------------------------------------------------------------------------//
//                                  PROGRAMA                                        //
//----------------------------------------------------------------------------------//
int main()
{
    //----------------------------------------------------------------------------------//
    //                                  VARIABLES                                       //
    //----------------------------------------------------------------------------------//
    const int screenWidth = 980;
    const int screenHeight = 520;
    int framesCounter = 0;
    int gameResult = -1;
    double questionElapsedTime = 0.0;
    const double timeBetweenQuestions = 30.0;
    int gameTimer = GAME_DURATION * 30;
    int initialGameTimer;
    bool showMessage = false;
    int centerX = screenWidth / 2;
    int centerY = screenHeight / 2;
    char question[200];
    int correctAnswer;
    int answers[4];
    bool gamePaused = false;    // Pausa
    float questionTime = 30.0f; // Tiempo en segundos para mostrar cada pregunta
    double questionTimer = (double)questionTime;
    char fcounter[20];
    bool showfcounter = false;
    bool enPregunta = false;
    bool respuesta;
    Pregunta quest[31];
    int i = 0;
    bool isFullscreen = true; // Set to true to start in fullscreen mode

    srand(time(NULL)); // Inicializar la semilla del generador de números aleatorios

    //----------------------------------------------------------------------------------//
    //                    INICIALIZAR AUDIO Y VENTANA                                   //
    //----------------------------------------------------------------------------------//
    InitWindow(screenWidth, screenHeight, "PELOTA CON REBOTE");
    SetWindowState(FLAG_VSYNC_HINT | FLAG_MSAA_4X_HINT);
    InitAudioDevice();

    //----------------------------------------------------------------------------------//
    //                    JUGADOR, BALON, BLOQUES Y GAME SCREEN                         //
    //----------------------------------------------------------------------------------//
    GameScreen screen = LOGO; // Current game screen state
    Player player = {0};
    Ball ball = {0};
    Brick bricks[BRICKS_LINES][BRICKS_PER_LINE] = {0};

    // Jugador
    int anchoplayer = 110;
```

```
player.position = (Vector2){screenWidth / 2, screenHeight * 7 / 8};
player.speed = (Vector2){9.0f, 0.0f};
player.size = (Vector2){anchoplayer, 24};
player.lifes = PLAYER_LIFES;

// Balon
ball.radius = 10.0f;
ball.active = false;
ball.position = (Vector2){player.position.x + player.size.x / 2, player.position.y - ball.radius * 2};
ball.speed = (Vector2){6.0f, 6.0f};
Texture2D brickTextures[10];

brickTextures[0] = LoadTexture("vidas/rojo.png");
brickTextures[1] = LoadTexture("vidas/verde.png");
brickTextures[2] = LoadTexture("vidas/celeste.png");
brickTextures[3] = LoadTexture("vidas/rosa.png");
brickTextures[4] = LoadTexture("vidas/morado.png");
brickTextures[5] = LoadTexture("vidas/rojo.png");
brickTextures[6] = LoadTexture("vidas/verde.png");
brickTextures[7] = LoadTexture("vidas/celeste.png");
brickTextures[8] = LoadTexture("vidas/rosa.png");
brickTextures[9] = LoadTexture("vidas/morado.png");

// Bloques
for (int j = 0; j < BRICKS_LINES; j++)
{
    for (int i = 0; i < BRICKS_PER_LINE; i++)
    {
        bricks[j][i].size = (Vector2){screenWidth / BRICKS_PER_LINE, 20};
        bricks[j][i].position = (Vector2){i * bricks[j][i].size.x, j * bricks[j][i].size.y + BRICKS_POSITION_Y};
        bricks[j][i].bounds = (Rectangle){bricks[j][i].position.x, bricks[j][i].position.y, bricks[j][i].size.x, bricks[j][i].size.y};
        bricks[j][i].active = true;
    }
}

questionElapsedTime = 0.0;

SetTargetFPS(60); // Set desired framerate (frames per second)
//--------------------------------------------------------------------------------------//
//                      CARGAR IMAGENES Y SONIDO PARA EL JUEGO                          //
//--------------------------------------------------------------------------------------//

// Sonido
Sound perder = LoadSound("perdervida.mp3");
Sound hitSound = LoadSound("touch.mp3");
Music logoMusic = LoadMusicStream("gamecube.mp3");
Music titleMusic = LoadMusicStream("street.mp3");
Music gameMusic = LoadMusicStream("Wily.mp3");
Music pauseMusic = LoadMusicStream("fnaf.mp3");
Music gameOver = LoadMusicStream("gameover.mp3");

// Textura
Texture2D texture[5];

// Imageness
Image image1 = LoadImage("vidas/vida1.png");
Image image2 = LoadImage("vidas/vida2.png");
Image image3 = LoadImage("vidas/vida3.png");
Image image4 = LoadImage("vidas/vida4.png");
Image image5 = LoadImage("vidas/vida5.png");
Image imageFondo1 = LoadImage("vidas/fondo1.png");
Image imageFondo2 = LoadImage("vidas/fondo2.png");
Image fondo_ending = LoadImage("ending.png");
Image fondo_main = LoadImage("vidas/trasero.png");
Image icon = LoadImage("icono.ico");

// Textura de cada uno
Texture2D fondo1 = LoadTextureFromImage(imageFondo1);
Texture2D fondo2 = LoadTextureFromImage(imageFondo2);
Texture2D fondo3 = LoadTextureFromImage(fondo_ending);
Texture2D fondo4 = LoadTextureFromImage(fondo_main);

texture[0] = LoadTextureFromImage(image1);
texture[1] = LoadTextureFromImage(image2);
texture[2] = LoadTextureFromImage(image3);
texture[3] = LoadTextureFromImage(image4);
texture[4] = LoadTextureFromImage(image5);

// Establecer el icono de la ventana
SetWindowIcon(icon);

// Descargar la imagen, ya que ya no es necesaria
UnloadImage(icon);
UnloadImage(fondo_main);
UnloadImage(fondo_ending);
UnloadImage(imageFondo1);
UnloadImage(imageFondo2);
UnloadImage(image1);
UnloadImage(image2);
UnloadImage(image3);
UnloadImage(image4);
UnloadImage(image5);

// Reproduce la música del logo al inicio
PlayMusicStream(logoMusic);
SetMusicVolume(logoMusic, 0.5f);

// Juego principal - Bucle
while (!WindowShouldClose()) // Mientras que no cierre la ventana
{
    UpdateMusicStream(logoMusic);

    switch (screen)
    {
    case LOGO:
    {
        framesCounter++;
        if (framesCounter > 400)
        {
            StopMusicStream(logoMusic);
            screen = TITLE;
            framesCounter = 0;
            PlayMusicStream(titleMusic);
        }
    }
    break;

    case TITLE:
    {
        UpdateMusicStream(titleMusic);
```

```
    framesCounter++;
    if (IsKeyPressed(KEY_ENTER))
    {
        StopMusicStream(titleMusic);
        screen = GAMEPLAY;
        // Aqui corremos la cancion del juego
        PlayMusicStream(gameMusic);
        framesCounter = 0;
    }
    //
}
break;
case GAMEPLAY:
{
    framesCounter++;
    // Aqui actaulizamos cancion del juego
    UpdateMusicStream(gameMusic);
    UpdateMusicStream(pauseMusic);
    //  Pausa con 'P'
    if (anchoplayer > 10)
    {
        if (framesCounter == 1000)
        {
            enPregunta = true;
            // gamePaused = true;
        }
    }
    if (IsKeyPressed('P'))
    {

        if (gamePaused)
        {
            UpdateMusicStream(pauseMusic);
            // Si el juego estaba pausado, reinicia el temporizador
            gameTimer = initialGameTimer;
            StopMusicStream(pauseMusic);
            PlayMusicStream(gameMusic);
            gamePaused = !gamePaused;
        }
        else
        {
            StopMusicStream(gameMusic);
            PlayMusicStream(pauseMusic);
            // PlayMusicStream(pauseMusic);
            gamePaused = !gamePaused;
        }
        // StopMusicStream(pauseMusic);
        // PlayMusicStream(gameMusic);
    }
    if (!gamePaused) // se mueven o no bolita y jugador
    {
        UpdateMusicStream(gameMusic);
        // Movimiento del jugador
        if (IsKeyDown(KEY_LEFT))
            player.position.x -= player.speed.x;
        if (IsKeyDown(KEY_RIGHT))
            player.position.x += player.speed.x;

        if ((player.position.x) <= 0)
            player.position.x = 0;
        if ((player.position.x + player.size.x) >= screenWidth)
            player.position.x = screenWidth - player.size.x;

        player.bounds = (Rectangle){player.position.x, player.position.y, player.size.x, player.size.y};

        // En el caso GAMEPLAY, ajusta la lógica de pausa para las preguntas
        if (IsKeyPressed(KEY_SPACE))
        {
            gamePaused = true;
            showMessage = true;
            questionTimer = questionTime;
            gamePaused = false;

            // Añade esta línea para reiniciar el temporizador de preguntas
            questionElapsedTime = 0.0;
        }

        if (ball.active)
        {
            // Balon movimiento
            ball.position.x += ball.speed.x;
            ball.position.y += ball.speed.y;

            // Colision logico: Balon vs pantalla-límites
            if (((ball.position.x + ball.radius) >= screenWidth) || ((ball.position.x - ball.radius) <= 0))
                ball.speed.x *= -1;
            if ((ball.position.y - ball.radius) <= 0)
                ball.speed.y *= -1;

            // Colision logico: Balon vs Jugador
            if (CheckCollisionCircleRec(ball.position, ball.radius, player.bounds))
            {
                ball.speed.y *= -1;
                ball.speed.x = (ball.position.x - (player.position.x + player.size.x / 2)) / player.size.x * 5.0f;
            }
            // Colision logico: Balon vs Bloques
            for (int j = 0; j < BRICKS_LINES; j++)
            {
                for (int i = 0; i < BRICKS_PER_LINE; i++)
                {
                    if (bricks[j][i].active && (CheckCollisionCircleRec(ball.position, ball.radius, bricks[j][i].bounds)))
                    {
                        bricks[j][i].active = false;
                        ball.speed.y *= -1;
                        PlaySound(hitSound);

                        break;
                    }
                }
            }

            // Game ending logic
            if ((ball.position.y + ball.radius) >= screenHeight)
            {
                ball.position.x = player.position.x + player.size.x / 2;
                ball.position.y = player.position.y - ball.radius - 1.0f;
                ball.speed = (Vector2){0, 0};
                ball.active = false;

                player.lifes--;
```

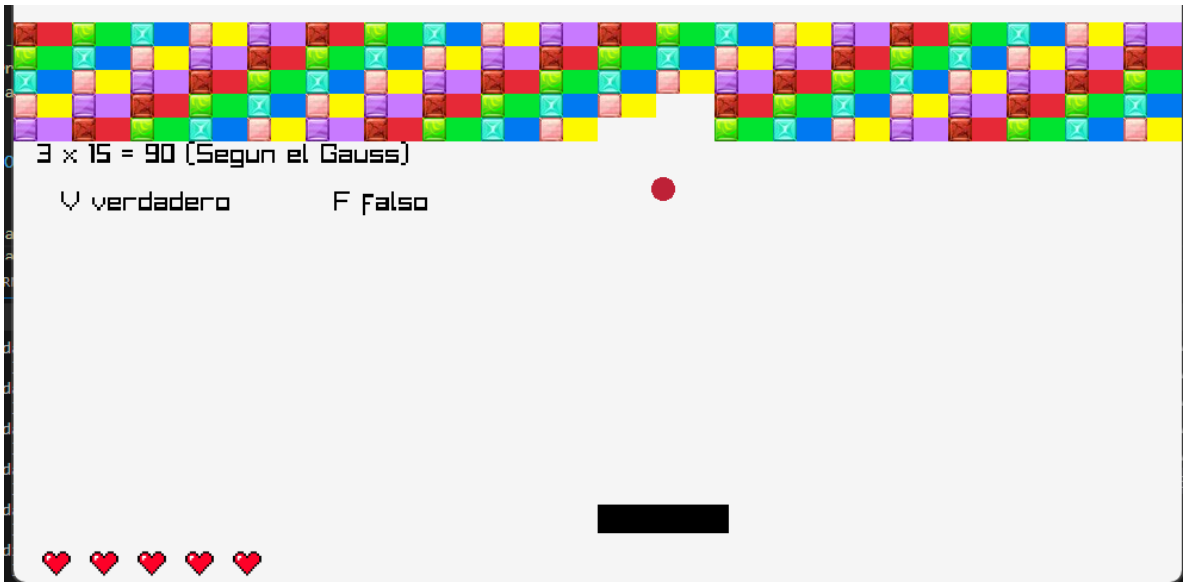
```
                PlaySound(perder);
            }
            if (player.lifes < 0)
            {
                screen = ENDING;
                player.lifes = 5;
                framesCounter = 0;
                PlayMusicStream(gameOver);
            }
        }
        else
        {
            // Reset ball position
            ball.position.x = player.position.x + player.size.x / 2;

            // LESSON 03: Inputs management (keyboard, mouse)
            if (IsKeyPressed(KEY_SPACE))
            {
                // Activate ball logic
                ball.active = true;
                ball.speed = (Vector2){0, -5.0f};
            }
        }
        if (enPregunta)
        {
            // DrawText("Rusia fue el pais con mas medallas en los juegos olimpicos de 2012", screenWidth / 2 - MeasureText("Rusia fue el pais con mas medallas en los juegos ol
            // obtengo la estructura
            // estructura[pregunta].pregunta = texto
            // estructura[pregunta].respuesta = bool
            question[i] = '\0';
            inicializaPreguntas(quest);
            DrawText(quest[i].pregunta, 20, 150, 22, BLACK);
            DrawText("V verdadero          F falso", 40, 190, 22, BLACK);
            // gamePaused = true;
            if (framesCounter == 1720)
            {
                anchoplayer -= 20;
                player.size = (Vector2){anchoplayer, 24};
                enPregunta = false;
                // gamePaused = false;
                framesCounter = 0;
                i++;
            }
            if (IsKeyPressed('V'))
            {
                // en base a estructura[pregunta].respuesta  se decide si es correcto o no, si no es, se le quitan 30 px
                respuesta = true;
                if (respuesta != quest[i].respuestacorrecta)
                {
                    anchoplayer -= 20;
                    player.size = (Vector2){anchoplayer, 24};
                    enPregunta = false;
                    // gamePaused = false;
                    framesCounter = 0;
                    i++;
                }
                else
                {
                    enPregunta = false;
                    // gamePaused = false;
                    framesCounter = 0;
                    i++;
                }
            }
            if (IsKeyPressed('F'))
            {
                // en base a estructura[pregunta].respuesta  se decide si es correcto o no, si no es, se le quitan 30 px
                respuesta = false;
                if (respuesta != quest[i].respuestacorrecta)
                {
                    anchoplayer -= 20;
                    player.size = (Vector2){anchoplayer, 24};
                    enPregunta = false;
                    // gamePaused = false;
                    framesCounter = 0;
                    i++;
                }
                else
                {
                    enPregunta = false;
                    // gamePaused = false;
                    framesCounter = 0;
                    i++;
                }
            }

            // if key press = V
            // decido si es correcto o no.
            // generar lista de preguntas 5,2,7,4,6   int listapreg[5]
            // estructura[listapreg[preguntacual]].
            // int preguntacual = 0;

            // Aparece pregunta (nueva si la anterior se contesto)  estructura[listapreg[preguntaactual]]

            // checa si presionaron una tecla
            // si la tecla es a o b, se revisa respuesta
            // preguntaactual++
            // framecounter =0
        }
    }
}
break;
case ENDING:
{
    // Update END screen data here!
    UpdateMusicStream(gameOver);
    framesCounter++;
    if (IsKeyPressed(KEY_ENTER))
    {
        // Replay / Exit game logic
        screen = TITLE;
    }
}
break;
default:
    break;
}
//----------------------------------------------------------------------------------//
//                                      DIBUJO                                      //
//----------------------------------------------------------------------------------//
```

```
BeginDrawing();
ClearBackground(RAYWHITE);
switch (screen)
{
case LOGO:
{
    // Draw LOGO screen here!
    DrawTexture(fondo1, 0, 0, WHITE);
    DrawText("JUEGA", 20, 100, 40, LIGHTGRAY);
    DrawText("Y", 20, 150, 40, LIGHTGRAY);
    DrawText("APRENDE!", 20, 200, 40, LIGHTGRAY);

    fcounter[0] = '\0';
    sprintf(fcounter, "L Fcounter = %d", framesCounter);
    DrawText(fcounter, 5, 5, 20, GRAY);
}
break;
case TITLE:
{
    // Draw TITLE screen here!

    DrawTexture(fondo2, 0, 0, WHITE);
    DrawText("PELOTA REBOTA ", 310, 150, 38, PURPLE);
    DrawText("CON PREGUNTAS", 310, 190, 38, PURPLE);
    DrawText("GAME", 350, 240, 40, PURPLE);
    if ((framesCounter / 30) % 2 == 0)
        DrawText("PRESS [ENTER] to START", GetScreenWidth() / 2 - MeasureText("PRESS [ENTER] to START", 20) / 2, GetScreenHeight() / 2 + 60, 20, PURPLE);
    fcounter[0] = '\0';
    sprintf(fcounter, "T Fcounter = %d", framesCounter);
    DrawText(fcounter, 5, 5, 20, GRAY);
}
break;
case GAMEPLAY:
{
    // Dibujar GAMEPLAy pantalla
    // DrawTexture(fondo4, 0, 0, WHITE);
    DrawRectangle(player.position.x, player.position.y, player.size.x, player.size.y, BLACK); // Dibujar barra del jugador
    DrawCircleV(ball.position, ball.radius, MAROON);                                          // Dibujar balon

    for (int j = 0; j < BRICKS_LINES; j++)
    {
        for (int i = 0; i < BRICKS_PER_LINE; i++)
        {
            if (bricks[j][i].active)
            {
                Color brickColor;
                switch ((i + j) % 5)
                {
                case 0:
                    brickColor = RED;
                    break;
                case 1:
                    brickColor = GREEN;
                    break;
                case 2:
                    brickColor = BLUE;
                    break;
                case 3:
                    brickColor = YELLOW;
                    break;
                case 4:
                    brickColor = PURPLE;
                    break;
                default:
                    brickColor = WHITE; // Color por defecto si necesitas más variedad
                    break;
                }

                DrawRectangle(bricks[j][i].position.x, bricks[j][i].position.y, bricks[j][i].size.x, bricks[j][i].size.y, brickColor);
                for (int j = 0; j < BRICKS_LINES; j++)
                {
                    for (int i = 0; i < BRICKS_PER_LINE; i++)
                    {
                        if (bricks[j][i].active)
                        {
                            // Use the appropriate texture based on the modulo result
                            DrawTexture(brickTextures[(i + j) % 5], (int)bricks[j][i].position.x, (int)bricks[j][i].position.y, WHITE);
                        }
                    }
                }
            }
        }
    }
    // Dibujar vidas del jugador
    for (int i = 0; i < player.lifes; i++)
    {
        // Calcular la posición de dibujo para cada vida
        int posX = 20 + 40 * i;
        int posY = screenHeight - 30;

        // Dibujar la textura en la posición calculada
        DrawTexture(texture[i], posX, posY, WHITE);
    }
    if (!enPregunta)
    {
        framesCounter++;
    }
    // Draw pause message when required
    if (gamePaused)
    {
        DrawText("GAME PAUSED", screenWidth / 2 - MeasureText("GAME PAUSED", 40) / 2, screenHeight / 2 + 60, 40, GRAY);
    }

    // aqui va el counter
    fcounter[0] = '\0';
    sprintf(fcounter, "Fcounter = %d", framesCounter);
    DrawText(fcounter, 5, 5, 20, GRAY);
}
break;
    PlayMusicStream(gameOver);
    UpdateMusicStream(gameOver);
case ENDING:
{
    PlayMusicStream(gameOver);
    UpdateMusicStream(gameOver);
    // Draw END screen here!
    DrawTexture(fondo3, 0, 0, WHITE);
    DrawText("GAME OVER", 400, 250, 32, PURPLE);

    if ((framesCounter / 30) % 2 == 0)
```

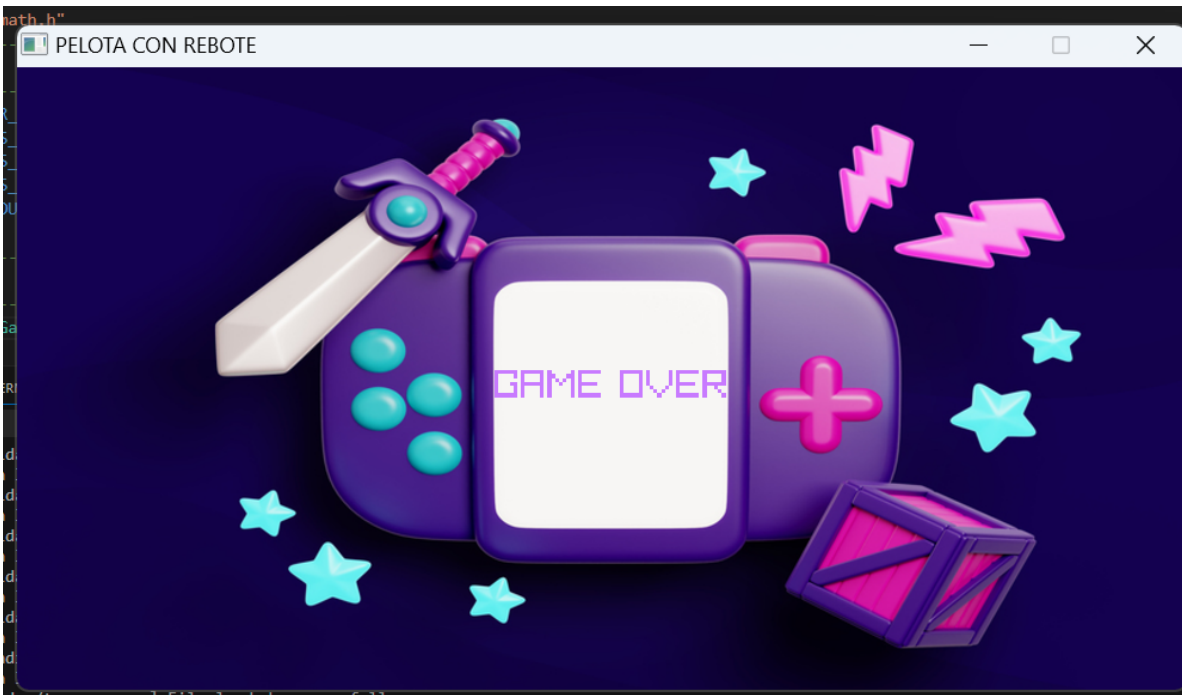
```
                DrawText("PRESS [ENTER] TO PLAY AGAIN", 400, 280, 12, PURPLE);
            PauseMusicStream(gameOver);
        }
        break;
        default:
            break;
        }

        EndDrawing();
        //----------------------------------------------------------------------------------
    }
    // Descargar musica
    for (int i = 0; i < 5; i++)
    {
        UnloadTexture(brickTextures[i]);
    }

    UnloadMusicStream(gameOver);
    UnloadMusicStream(logoMusic);
    UnloadSound(hitSound);
    UnloadMusicStream(gameMusic);
    UnloadMusicStream(titleMusic);
    UnloadMusicStream(pauseMusic);
    // Cerrar dispositivo de audio y la ventana
    CloseAudioDevice();
    CloseWindow();
    return 0;
}

Pregunta generaPregunta(const char *enunciado, bool respuestaCorrecta)
{
    Pregunta nuevaPregunta;
    strcpy(nuevaPregunta.pregunta, enunciado);
    nuevaPregunta.respuestacorrecta = respuestaCorrecta;
    return nuevaPregunta;
}

void inicializaPreguntas(Pregunta estructura[31])
{
    // Utilice el 1 para cuando la respuesta1 es la correcta y el 2 para cuando la respuesta2 es la correcta

    estructura[0] = generaPregunta("3 x 15 = 90 (Segun el Gauss)", true);
    estructura[1] = generaPregunta("Peru se encuentra en el continente Africano", false);
    estructura[2] = generaPregunta("La raiz cuadrada de 625 es 25", true);
    estructura[3] = generaPregunta("El balon del Mundial de futbol 1978 se llamaba Adidas Tango", true);
    estructura[4] = generaPregunta("Rusia fue el pais con mas medallas en los juegos olimpicos de 2012", false);
    estructura[5] = generaPregunta("Los partidos de rugby duran 80 minutos", true);
    estructura[6] = generaPregunta("Magneto es un villano de Batman", false);
    estructura[7] = generaPregunta("Sheldon Cooper es un personaje de The Big Bang Theory", true);
    estructura[8] = generaPregunta("El Kpop es pop Japones", false);
    estructura[9] = generaPregunta("La sal es el compuesto quimica NaCl", true);
    estructura[10] = generaPregunta("Edison fue el inventor de la bombilla", true);
    estructura[11] = generaPregunta("La columna vertebral humana tiene 34 vertebras", false);
    estructura[12] = generaPregunta("En matematicas el numero PI es igual a 3.1415....", true);
    estructura[13] = generaPregunta("El flamenco es tipico de Brasil", false);
    estructura[14] = generaPregunta("Alemania es el pais con mas podios en la Copa Mundial de futbol", true);
    estructura[15] = generaPregunta("Rusia es el pais mas grande del mundo", true);
    estructura[16] = generaPregunta("Las pulgas tienen alas", false);
    estructura[17] = generaPregunta("Groenlandia se encuentra en el emisferio Norte", true);
    estructura[18] = generaPregunta("Las Palomas asadas son un platillo tipico de Perú", false);
    estructura[19] = generaPregunta("El chino es el idiomas mas hablado en el mundo", true);
    estructura[20] = generaPregunta("La segunda guerra mundial acabo en 1945", true);
    estructura[21] = generaPregunta("Windsor es el apellido de la Reina Isable 11", true);
    estructura[22] = generaPregunta("Freddie Mercury murio en 1992", false);
    estructura[23] = generaPregunta("La revolucion mexicana se celebra el 20 de Noviembre", true);
    estructura[24] = generaPregunta("La primera Guerra Mundial comenzo en 1915", false);
    estructura[25] = generaPregunta("5 + 31 - 41 + 8 - 9 + 10 = 4", true);
    estructura[26] = generaPregunta("La revolucion francesa se produjo en 1789", true);
    estructura[27] = generaPregunta("Cristobal Colon llego a America en 1492", true);
    estructura[28] = generaPregunta("El muro de Berlin cayo en el año 1990", false);
    estructura[29] = generaPregunta("Austria se encuentra en Asia", false);
    estructura[30] = generaPregunta("La bandera de China cuenta con 7 estrellas", false);
}
```

Capturas del programa:

INICIO



PARTIDA



FIN DEL JUEGO

Conclusion sobre el curso y el juego.

El haber entendido cada parte del curso, aportaba notoriamente tu saber en este lenguaje, el cual fue fundamental, ademas de mejorar la logica, el entendimiento del curso nos ayudo en saber como desarrollar nuestro juego. En este proyecto, se ha desarrollado un juego básico utilizando la biblioteca Raylib en el lenguaje de programación C. A través de este proceso, se han abordado varios conceptos y prácticas de programación que proporcionan valiosas lecciones de aprendizaje:

En resumen, el curso de programación estructurada ha sido fundamental para mi comprensión de este lenguaje, mejorando significativamente mi lógica de programación. La estructuración del código en funciones y la implementación de estructuras específicas para el juego, como jugador, bola y bloques, han demostrado ser esenciales. Esta práctica de organización ha facilitado no solo la comprensión del flujo de juego, sino también la capacidad de realizar cambios y expansiones de manera más efectiva. Los conceptos clave de la lógica de juego, incluido el manejo de colisiones, el control de entrada del jugador y la integración de un temporizador para preguntas educativas, han proporcionado valiosas lecciones que seguramente aplicaré en futuros proyectos. En definitiva, el curso no solo ha ampliado mi conocimiento técnico, sino que ha cambiado mi enfoque hacia la resolución de problemas de programación. Estoy emocionado de aplicar estas habilidades en proyectos futuros y continuar mi crecimiento.