

Proyecto Bimestral II: PING-PONG

Viscaino Ochoa Stiven Andrés, stiven.viscaino@epn.edu.ec, Sangucho Coronel Eddy Alejandro, eddy.sangucho@epn.edu.ec.

Escuela Politécnica Nacional, Facultad de Sistemas

Programación I

Ing. Hernan Ordonez, 18 de febrero del 2024

Resumen— Este proyecto presenta la creación de un juego de ping-pong en C++ utilizando Visual Studio y la biblioteca Raylib. El juego incluye controles mediante teclado, la detección al momento que la pelota impacta contra los bordes de la pantalla, puntuación, velocidad variable de la pelota y sus respectivos credits.

I. INTRODUCCIÓN

En el ámbito de la programación, la creación de videojuegos presenta un desafío técnico y creativo que nos permitió poner a prueba lo aprendido durante este curso de programación.

En esta ocasión abordamos el desarrollo de un juego de ping-pong implementado el lenguaje de programación C++ mediante el entorno de desarrollo Visual Studio y haciendo uso de la biblioteca Raylib la cual es una biblioteca utilizada para crear videojuegos.

Donde el videojuego consiste en el clásico juego PING-PONG, tenemos 2 paletas y una pelota, los jugadores se enfrentaran entre si y el jugador que obtenga 10 puntos será el ganador, para obtener los puntos hay que evitar que la pelota choque contra los bordes de nuestra ventana.

II. DESARROLLO

Empezamos incluyendo las librerías raylib.h, conio.h, fstream y la principal iostream

Después declaramos las variables Player_score y player_wins tanto para el jugador 1 como para el jugador 2.

Se utilizo variables color, para usarlos en el menú y el contador

Se imprime el menú, con las opciones Play, scores, credits y exit, estas en ingles debido a las librerías utilizadas.

Se crea una función llamada SaveScores para que esta almacene los puntajes obtenidos en las variables playerwins.

```
1 #include <iostream>
2 #include <raylib.h>
3 #include <conio.h>
4 #include <fstream>
5 using namespace std;
6
7 Color Green = Color{ 38,185,154,255 };
8 Color Dark_Green = Color{ 20,160,133,255 };
9 Color Light_Green = Color{ 129,204,184,255 };
10 Color Yellow = Color{243,213,91,255 };
11
12 int player_score = 0;
13 int player_2_score = 0;
14 bool gameOver = false;
15
16
17 enum MenuOption {
18     PLAY,
19     SCORES,
20     CREDITS,
21     EXIT,
22 };
23
24 //Variables para almacenar las victorias de cada jugador
25 int playerwins = 0;
26 int player_2wins = 0;
27
28 //Funcion para guardar los puntajes
29 void SaveScores() {
30     ofstream scoresFile("scores.txt");
31     if (scoresFile.is_open()) {
32         scoresFile << playerwins << endl;
33         scoresFile << player_2wins << endl;
34     }
35 }
```

```
32 }
33
34 //Funcion para cargar los puntajes desde un archivo de texto
35 void LoadScores() {
36     ifstream scoresFile("scores.txt");
37     if (scoresFile.is_open()) {
38         scoresFile >> playerwins;
39         scoresFile >> player_2wins;
40     }
41     scoresFile.close();
42 }
43
44 //funcion para mostrar los puntajes de los jugadores
45 void ShowScores() {
46     LoadScores();
47     ClearBackground(RAYWHITE);
48     DrawText("Puntajes", 150, 420, 30, RED);
49     //Mostrar los scores que ha ganado cada jugador
50     DrawText(TextFormat("Jugador 1: %d", playerwins), 150, 470, 20, WHITE);
51     DrawText(TextFormat("Jugador 2: %d", player_2wins), 150, 500, 20, WHITE);
52 }
```

Se crea una función llamada LoadScores esta de tipo void, la cual carga los puntajes desde un archivo de texto.

Tambien otra función llamada showscores de tipo void, la cual muestra los puntajes obtenidos por los jugadores, con ayuda de la variable drawtext la cual sirve como un cout dentro de nuestro código.

AHORA EL DESARROLLO DE LA PELOTA

Declaramos una clase de tipo Ball, donde definimos x y y como una float .

Despues definimos speed_x y y , acompañados de un radio de la pelota de tipo enteros.

La función Draw() se encarga de dibujar la pelota en la pantalla utilizando la función DrawCircle() de la biblioteca Raylib. La posición y el radio de la pelota se toman de las variables de la clase.

La función Updtae actualiza la posición de la pelota en cada fotograma del juego, gracias a la biblioteca raylib y también se incrementan las coordenadas x e y según las velocidades speed_x y speed_y.

Se comprueba si la pelota ha alcanzado los límites superior o inferior de la pantalla

La condición if (x - radius <= 0) verifica si la pelota ha cruzado el límite izquierdo de la pantalla, y la condición if (x + radius >= GetScreenWidth()) verifica si ha cruzado el límite derecho. Si alguna de estas condiciones es verdadera, se incrementan los marcadores de los jugadores (player_score o player_2_score respectivamente) y se llama a la función ResetBall() para reiniciar la posición de la pelota.

```
84 class Ball {
85 public:
86     float x, y;
87     int speed_x, speed_y;
88     int radius;
89
90     void Draw() {
91         DrawCircle(x, y, radius, YELLOW);
92     }
93
94     void Update() {
95         x += speed_x;
96         y += speed_y;
97
98         if (y + radius >= GetScreenHeight() || y - radius <= 0) {
99             speed_y *= -1;
100         }
101
102         if (x + radius >= GetScreenWidth()) {
103             player_2_score++;
104             ResetBall();
105         }
106
107         if (x - radius <= 0) {
108             player_score++;
109             ResetBall();
110         }
111
112     void ResetBall() {
113         x = GetScreenWidth() / 2;
114         y = GetScreenHeight() / 2;
115         int speed_choices[2] = { -1, 1 };
116         speed_x = speed_choices[GetRandomValue(0, 1)];
117         speed_y = speed_choices[GetRandomValue(0, 1)];
118     }
119 }
```

AHORA EL DESARROLLO DE LAS PALETAS

Creamos una clase llamada Paddle que representara a una paleta de pin pon en c++

Donde:

float x, y: Representan las coordenadas x e y de la paleta.

float width, height: Definen el ancho y alto de la paleta.

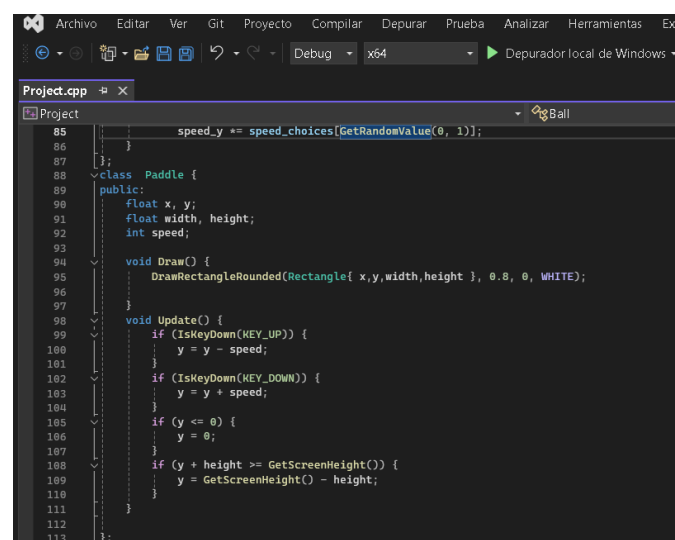
int speed: Indica la velocidad de movimiento de la paleta.

Se creo una función llamada Draw, la cual dibuja la paleta en la pantalla

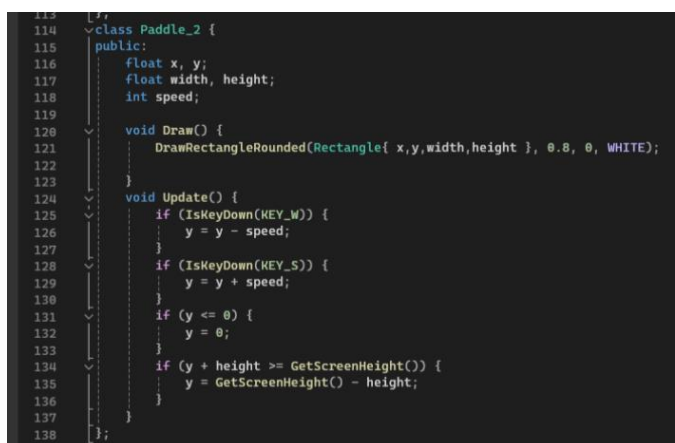
La función Draw() se encarga de dibujar la paleta en la pantalla utilizando la función DrawRectangleRounded() de la biblioteca Raylib. Se utiliza un rectángulo redondeado definido por las coordenadas (x, y), el ancho (width), el alto (height), y se establece un radio de esquinas redondeadas de 0.8. El color de la paleta es blanco (WHITE).

La función Update() se encarga de actualizar la posición de la paleta en cada fotograma del juego.

Se verifican límites para asegurar que la paleta no se salga de los límites superior o inferior de la pantalla, todo esto tanto para la paleta uno como para la dos.



```
85         speed_y == speed_choices[GetRandomValue(0, 1)];
86     }
87 }
88
89 class Paddle {
90 public:
91     float x, y;
92     float width, height;
93     int speed;
94
95     void Draw() {
96         DrawRectangleRounded(Rectangle{ x,y,width,height }, 0.8, 0, WHITE);
97     }
98
99     void Update() {
100         if (IsKeyDown(KEY_UP)) {
101             y = y - speed;
102         }
103         if (IsKeyDown(KEY_DOWN)) {
104             y = y + speed;
105         }
106         if (y <= 0) {
107             y = 0;
108         }
109         if (y + height >= GetScreenHeight()) {
110             y = GetScreenHeight() - height;
111         }
112     }
113 }
```



```
114
115 class Paddle_2 {
116 public:
117     float x, y;
118     float width, height;
119     int speed;
120
121     void Draw() {
122         DrawRectangleRounded(Rectangle{ x,y,width,height }, 0.8, 0, WHITE);
123     }
124
125     void Update() {
126         if (IsKeyDown(KEY_W)) {
127             y = y - speed;
128         }
129         if (IsKeyDown(KEY_S)) {
130             y = y + speed;
131         }
132         if (y <= 0) {
133             y = 0;
134         }
135         if (y + height >= GetScreenHeight()) {
136             y = GetScreenHeight() - height;
137         }
138     }
139 }
```

Se crean instancias de las clases Ball, Paddle, y Paddle_2 llamadas ball, player, y player_2

Definimos variables para el ancho y alto de la ventana y el ingame es el cual nos indicara si el juego esta en curso.

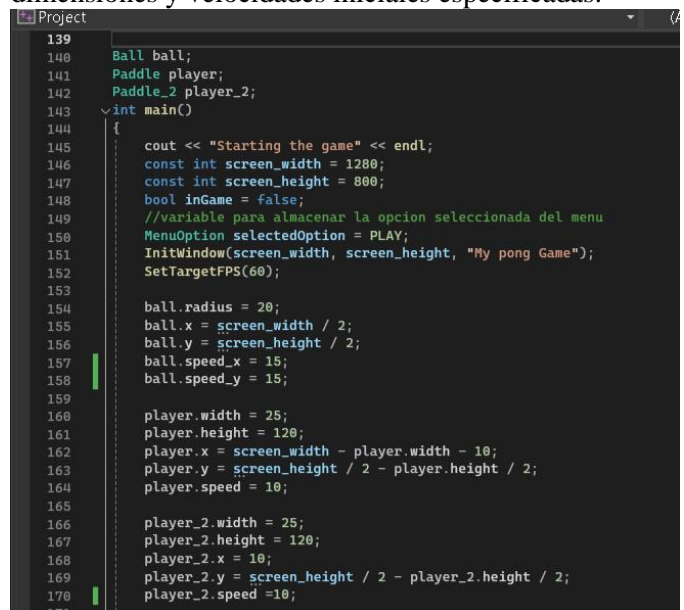
Mientras que selectedOption es una variable de tipo MenuOption que almacena la opción seleccionada del menú.

SetTargetFPS(60) configura la cantidad objetivo de fotogramas por segundo, esto de acuerdo a lo que nosotros queramos, como lo más estable actualmente elegimos 60 fps

Se configuran las propiedades iniciales de la pelota y las paletas.

La pelota (ball) se coloca en el centro de la pantalla, con un radio de 20 píxeles y velocidad inicial tanto en la dirección x como en la dirección y de 12 píxeles por fotograma.

Las paletas (player y player_2) se colocan en los lados derecho e izquierdo de la pantalla respectivamente, con dimensiones y velocidades iniciales especificadas.



```
139
140 Ball ball;
141 Paddle player;
142 Paddle_2 player_2;
143
144 int main()
145 {
146     cout << "Starting the game" << endl;
147     const int screen_width = 1280;
148     const int screen_height = 800;
149     bool inGame = false;
150     //Variable para almacenar la opcion seleccionada del menu
151     MenuOption selectedOption = PLAY;
152     InitWindow(screen_width, screen_height, "My pong Game");
153     SetTargetFPS(60);
154
155     ball.radius = 20;
156     ball.x = screen_width / 2;
157     ball.y = screen_height / 2;
158     ball.speed_x = 15;
159     ball.speed_y = 15;
160
161     player.width = 25;
162     player.height = 120;
163     player.x = screen_width - player.width - 10;
164     player.y = screen_height / 2 - player.height / 2;
165     player.speed = 10;
166
167     player_2.width = 25;
168     player_2.height = 120;
169     player_2.x = 10;
170     player_2.y = screen_height / 2 - player_2.height / 2;
171     player_2.speed = 10;
172 }
```

Se declara un **puntero** a una cadena de caracteres llamado winnerText y se inicializa con un valor nulo (nullptr).

Se inicia un **bucle** principal que continúa mientras la ventana no se cierre (WindowShouldClose() == false).

Si no estamos en el juego (!inGame), se dibuja un menú en la ventana con opciones para jugar, ver puntajes, créditos y salir.

Se verifica si se presiona alguna tecla correspondiente a las opciones (1, 2, 3, 4).

Dependiendo de la tecla presionada, se actualiza la opción seleccionada (selectedOption) y se cambia el estado del juego (inGame).

```

171 const char* winnerText = nullptr;
172 while (WinShouldClose() == false) {
173     //Verificar si estamos en el juego o en el menu
174     if (!inGame) {
175         //Inicio el menu
176         BeginDrawing();
177         ClearBackground(DARKBLUE);
178         DrawText("PING-PONG", screen_width / 2 - MeasureText("PING-PONG", 150) / 2, 100, 150, WHITE);
179         DrawText("1. Jugar", screen_width / 2 - MeasureText("1. Jugar", 60) / 2, 330, 60, YELLOW);
180         DrawText("2. Puntajes", screen_width / 2 - MeasureText("2. Puntajes", 60) / 2, 420, 60, YELLOW);
181         DrawText("3. Credits", screen_width / 2 - MeasureText("3. Credits", 60) / 2, 510, 60, YELLOW);
182         DrawText("4. Salir", screen_width / 2 - MeasureText("4. Salir", 60) / 2, 600, 60, YELLOW);
183         EndDrawing();
184     }
185     //Verificar si se ha presionado alguna tecla
186     if (IsKeyPressed(KEY_ONE)) {
187         selectedOption = PLAY;
188         inGame = true;
189     }
190     else if (IsKeyPressed(KEY_TWO)) {
191         selectedOption = SCORES;
192     }
193     else if (IsKeyPressed(KEY_THREE)) {
194         selectedOption = CREDITS;
195     }
196     else if (IsKeyPressed(KEY_FOUR)) {
197         selectedOption = EXIT;
198         break;
199     }
200     else {
201         //Si no se presiona ninguna tecla
202         continue;
203     }
204 }

```

Se verifica si la tecla de barra espaciadora (KEY_SPACE) se ha presionado.

Si es así, se sale del juego (inGame = false) y se establece la opción seleccionada como jugar (selectedOption = PLAY).

Logica del juego:

Se inicia el dibujo en la ventana (BeginDrawing()).

Se verifican colisiones entre la pelota y las paletas (player y player_2). Si hay colisión, se invierte la dirección horizontal de la pelota.

Se actualizan las posiciones de la pelota, el jugador y el jugador 2 llamando a sus respectivos métodos Update().

Si el marcador de jugador 2 o el marcador del jugador 1 alcanzan o superan 10 puntos, se muestra un mensaje indicando al ganador y se reinician los marcadores.

Se finaliza el dibujo en la ventana (EndDrawing()).

```

205 //Si estamos en el juego
206 //Verificar si se ha presionado la barra espaciadora
207 if (IsKeyPressed(KEY_SPACE)) {
208     inGame = false;
209     selectedOption = PLAY;
210 }
211 //Inicio del juego
212 BeginDrawing();
213 if (CheckCollisionCircleVec(Vector2{ball.x, ball.y}, ball.radius, Rectangle{player.x, player.y, player.width, player.height})) {
214     ball.speed.x = -ball.speed.x;
215 }
216 if (CheckCollisionCircleVec(Vector2{ball.x, ball.y}, ball.radius, Rectangle{player_2.x, player_2.y, player_2.width, player_2.height})) {
217     ball.speed.x = -ball.speed.x;
218 }
219 //Actualizar
220 ball.Update();
221 player.Update();
222 player_2.Update();
223 //Verificar si se ha alcanzado el límite de puntos
224 if (player_2.score > 10 || player.score > 10) {
225     ClearBackground(DARK_GREEN);
226     if (player.score > 10) {
227         playerWins++;
228         DrawText("Jugador 1 es el ganador!", 240, 340, 60, RED);
229     }
230     else if (player_2.score > 10) {
231         player_2Wins++;
232         DrawText("Jugador 2 es el ganador!", 240, 340, 60, RED);
233     }
234 }
235 //Si no se alcanza el límite de puntos
236 ClearBackground(DARK_GREEN);
237 DrawRectangle(screen_width / 2, 0, screen_width / 2, screen_height, Green);
238 DrawCircle(screen_width / 2, screen_height / 2, 150, LightGreen);
239 DrawLine(screen_width / 2, 0, screen_width / 2, screen_height, WHITE);
240 ball.Draw();
241 player.Draw();
242 player_2.Draw();
243 EndDrawing();
244 //Guardar los puntajes antes de salir del juego
245 SaveScores();
246 if (selectedOption == SCORES) {
247     ShowScores();
248 }
249 if (selectedOption == CREDITS) {
250     DrawText("Hecho por [ALEJANDRO SANGUCHO]", 10, 10, 30, PURPLE);
251     DrawText("Hecho por [STIVEN VISCAINOS]", 10, 50, 30, PURPLE);
252 }
253 CloseWindow();
254 return 0;
255 }

```

Si no estamos en el menú y no hemos salido del juego, entonces estamos en el juego.

Se configura el fondo de la pantalla como verde oscuro (ClearBackground(Dark_Green)).

Se dibuja la mitad derecha de la pantalla de color verde (DrawRectangle(screen_width / 2, 0, screen_width / 2, screen_height, Green)).

Se dibuja un círculo en el centro de la pantalla (DrawCircle(screen_width / 2, screen_height / 2, 150, Light_Green)).

Se dibuja una línea vertical en el centro de la pantalla (DrawLine(screen_width / 2, 0, screen_width / 2, screen_height, WHITE)).

Se dibujan la pelota y las paletas llamando a sus respectivos métodos Draw().

Se muestra la puntuación de los jugadores en la parte superior de la pantalla.

Se finaliza el dibujo en la ventana (EndDrawing()).

Después de salir del bucle, se guardan los puntajes antes de cerrar el juego llamando a la función SaveScores().

Si la opción seleccionada es ver los puntajes (SCORES), se muestra la pantalla de puntajes llamando a la función ShowScores().

Si la opción seleccionada es ver los créditos (CREDITS), se muestra un mensaje de créditos con nuestros nombres.

Se cierra la ventana y se finaliza el programa llamando a CloseWindow().

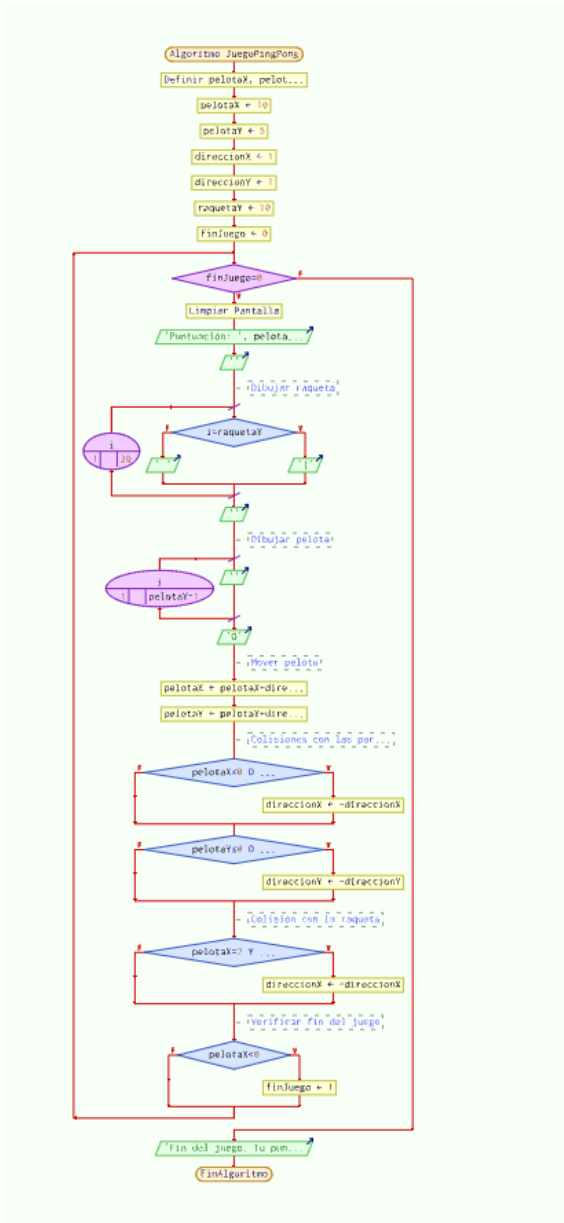
```

Project (Ambito global)
226 playerWins++;
227 DrawText("Jugador 1 es el ganador!", 240, 340, 60, RED);
228
229 } else if (player_2.score > 10) {
230     player_2Wins++;
231     DrawText("Jugador 2 es el ganador!", 240, 340, 60, RED);
232 }
233
234 } else {
235     ClearBackground(Dark_Green);
236     DrawRectangle(screen_width / 2, 0, screen_width / 2, screen_height, Green);
237     DrawCircle(screen_width / 2, screen_height / 2, 150, LightGreen);
238     DrawLine(screen_width / 2, 0, screen_width / 2, screen_height, WHITE);
239     ball.Draw();
240     player_2.Draw();
241     player.Draw();
242     DrawText(TextFormat("%i", player_2.score), screen_width / 4 - 20, 20, 80, WHITE);
243     DrawText(TextFormat("%i", player.score), 3 + screen_width / 4 - 20, 20, 80, WHITE);
244 }
245 EndDrawing();
246
247 //Guardar los puntajes antes de salir del juego
248 SaveScores();
249 if (selectedOption == SCORES) {
250     ShowScores();
251 }
252 if (selectedOption == CREDITS) {
253     DrawText("Hecho por [ALEJANDRO SANGUCHO]", 10, 10, 30, PURPLE);
254     DrawText("Hecho por [STIVEN VISCAINOS]", 10, 50, 30, PURPLE);
255 }
256
257 CloseWindow();
258 return 0;
259 }

```

También se creó un código en Pseint, para simular y obtener el diagrama de flujo de nuestro juego.

Recordemos que Pseint no puede ejecutar códigos muy complejos, ni juegos en sí.



Ejecución del Juego:

Menú Principal:



Puntajes:



Créditos:



Juego:

