

ESCUELA POLITECNICA NACIONAL

PROYECTO  
BIMESTRAL  
II: PING-PONG EN  
C++

}



# Contenidos

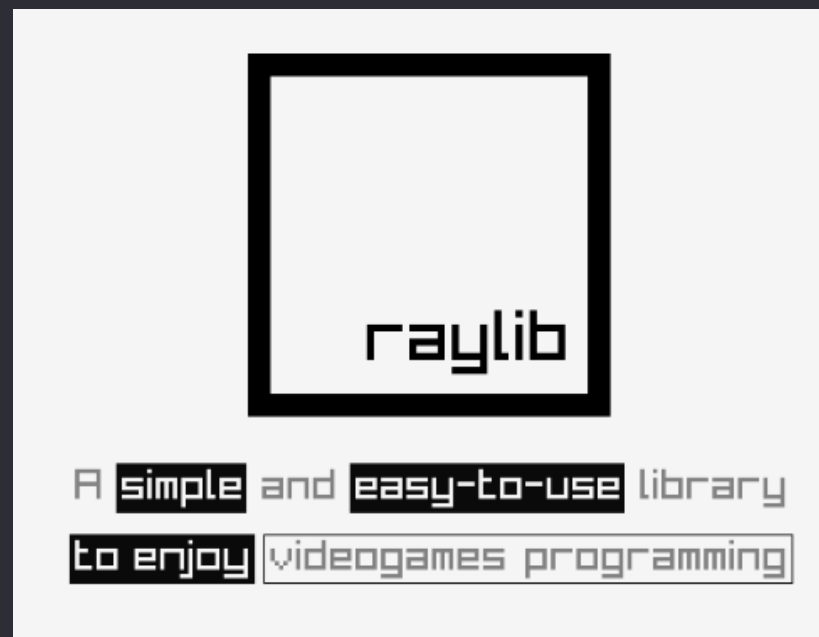
## Resumen:

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 para 2 jugadores, la detección al momento que la pelota impacta contra los bordes de la pantalla, puntuación, velocidad variable de la pelota y sus respectivos credits.

# Desarrollo

El código comienza incluyendo las librerías **raylib.h**, **conio.h**, **fstream**, y la principal **iostream**. Luego, se declaran las variables **Player\_score** y **player\_wins** para los jugadores 1 y 2, junto con variables de color para usar en el menú y el contador.

Posteriormente, se imprime un menú con las opciones jugar, puntajes, credits y salir, utilizando las librerías mencionadas. Además, se crea una función llamada **SaveScores** que tiene como objetivo almacenar los puntajes obtenidos en las variables **player\_wins**.



```
Project
1
2  ~#include <iostream>
3  ~#include <raylib.h>
4  ~#include <conio.h>
5  ~#include <fstream>
6  ~using namespace std;
7
8  ~Color Green = Color{ 38,185,154,255 };
9  ~Color Dark_Green = Color{ 20,160,133,255 };
10 ~Color Light_Green = Color{ 129,204,184,255 };
11 ~Color Yellow = Color{243,213,91,255 };
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 ~//Variables para almacenar las victorias de cada jugador
24 ~int playerwins = 0;
25 ~int player_2wins = 0;
26 ~//Funcion para guardar los puntajes
27 ~void SaveScores() {
28 ~    ~ofstream scoresFile("scores.txt");
29 ~    ~if (scoresFile.is_open()) {
30 ~        ~scoresFile << playerwins << endl;
31 ~        ~scoresFile << player_2wins << endl;
32 ~    }
33 }
```

}

## Desarrollo

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.

```
32     }
33 }
34 //Funcion para cargar los puntajes desde un archivo de texto
35 void LoadScores() {
36     ifstream scoresFile("scores.text");
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 las veces 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 }
```

}

# AHORA EL DESARROLLO DE LA PELOTA

Se define la clase Ball con variables para posición, velocidad y radio. La función Draw() utiliza DrawCircle() para dibujar la pelota en la pantalla. La función Update actualiza la posición de la pelota, incrementando las coordenadas x e y según las velocidades. Se verifica si la pelota alcanza los límites superior o inferior. Si cruza los límites izquierdo o derecho, se incrementan los marcadores y se reinicia la posición con ResetBall().

```
54 ~class Ball {
55     public:
56         float x, y;
57         int speed_x, speed_y;
58         int radius;
59
60     void Draw() {
61         DrawCircle(x, y, radius, YELLOW);
62     }
63
64     void Update() {
65         x += speed_x;
66         y += speed_y;
67
68         if (y + radius >= GetScreenHeight() || y - radius <= 0){
69             speed_y *= -1;
70         }
71         if (x + radius >= GetScreenWidth()) {
72             player_2_score++;
73             ResetBall();
74         }
75         if (x - radius <= 0) {
76             player_score++;
77             ResetBall();
78         }
79     }
80
81     void ResetBall() {
82         x = GetScreenWidth() / 2;
83         y = GetScreenHeight() / 2;
84         int speed_choices[2] = { -1, 1 };
85         speed_x = speed_choices[GetRandomValue(0, 1)];
86         speed_y = speed_choices[GetRandomValue(0, 1)];
87     }
88 }
```

## AHORA EL DESARROLLO DE LAS PALETAS

Se ha creado una clase llamada Paddle en C++, que representa una paleta de pin-pon. La clase tiene variables float x, y para las coordenadas, float width, height para el tamaño, e int speed para la velocidad de movimiento.

La función Draw() utiliza DrawRectangleRounded() de Raylib para dibujar la paleta con esquinas redondeadas. Se especifica el rectángulo con las coordenadas (x, y), ancho (width), alto (height), y un radio de esquinas redondeadas de 0.8. El color de la paleta es blanco.

La función Update() actualiza la posición de la paleta en cada fotograma, verificando límites para asegurar que no sobrepase los límites superior o inferior de la pantalla. Este proceso se realiza tanto para la paleta uno como para la dos.

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

```
113
114 class Paddle_2 {
115 public:
116     float x, y;
117     float width, height;
118     int speed;
119
120     void Draw() {
121         DrawRectangleRounded(Rectangle{ x,y,width,height }, 0.8, 0, WHITE);
122     }
123
124     void Update() {
125         if (IsKeyDown(KEY_W)) {
126             y = y - speed;
127         }
128         if (IsKeyDown(KEY_S)) {
129             y = y + speed;
130         }
131         if (y <= 0) {
132             y = 0;
133         }
134         if (y + height >= GetScreenHeight()) {
135             y = GetScreenHeight() - height;
136         }
137     }
138 };
139
```

Se crean instancias de las clases Ball, Paddle, y Paddle\_2 llamadas ball, player, y player\_2. Se definen variables para el ancho y alto de la ventana, y la variable ingame indica si el juego está en curso.

La variable selectedOption, de tipo MenuOption, almacena la opción seleccionada del menú. SetTargetFPS(60) establece la meta de 60 fotogramas por segundo.

Se configuran las propiedades iniciales: la pelota (ball) se coloca en el centro con radio 20 y velocidad inicial. Las paletas (player y player\_2) se colocan en los lados con dimensiones y velocidades específicas.

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

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
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202

const char* winnerText = nullptr;
while (WindowShouldClose() == false) {
    //verificar si estamos en el juego o en el menu
    if (!inGame) {
        //dibujar el menu
        BeginDrawing();
        ClearBackground(DARKBLUE);
        DrawText("PING-PONG", screen_width / 2 - MeasureText("PING-PONG", 150) / 2, 100, 150, WHITE);
        DrawText("1. Jugar", screen_width / 2 - MeasureText("1. Jugar", 60) / 2, 330, 60, YELLOW);
        DrawText("2. Puntajes", screen_width / 2 - MeasureText("2. Puntajes", 60) / 2, 420, 60, YELLOW);
        DrawText("3. Creditos", screen_width / 2 - MeasureText("3. Creditos", 60) / 2, 510, 60, YELLOW);
        DrawText("4. Salir", screen_width / 2 - MeasureText("4. Salir", 60) / 2, 600, 60, YELLOW);
        EndDrawing();

        //Verificar si se ha presionado alguna tecla
        if (IsKeyPressed(KEY_ONE)) {
            selectedOption = PLAY;
            inGame = true;
        }
        else if (IsKeyPressed(KEY_TWO)) {
            selectedOption = SCORES;
        }
        else if (IsKeyPressed(KEY_THREE)) {
            selectedOption = CREDITS;
        }
        else if (IsKeyPressed(KEY_FOUR)) {
            selectedOption = EXIT;
            break;
        }
    }
    else {

```



# 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()).

```
203 else {
204     //si estamos en el juego
205     //verificar si se ha presionado la barra espaciadora
206     if (IsKeyPressed(KEY_SPACE)) {
207         inGame = false;
208         selectedOption = PLAY;
209     }
210     //LOGICA DEL JUEGO
211     BeginDrawing();
212     //Colisiones
213     if (CheckCollisionCircles(Vector2{ ball.x , ball.y }, ball.radius, Rectangle{ player.x,player.y,player.width,player.height }) {
214         ball.speed_x *= -1;
215     }
216     if (CheckCollisionCircles(Vector2{ ball.x,ball.y }, ball.radius, Rectangle{ player_2.x,player_2.y,player_2.width,player_2.height }) {
217         ball.speed_x *= -1;
218     }
219     //Updating
220     ball.Update();
221     player.Update();
222     player_2.Update();
223     //Draws
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     else {
236         ClearBackground(Dark_Green);
237         DrawRectangle(screen_width / 2, 0, screen_width / 2, screen_height, Green);
238     }
```

En el juego, el fondo es verde oscuro, con la mitad derecha en verde. Se dibuja un círculo y una línea vertical en el centro. Se muestran la pelota, las paletas y la puntuación. Después de salir, se guardan los puntajes.

Si se elige ver puntajes, se muestra la pantalla correspondiente. En caso de seleccionar créditos, se muestra un mensaje y se cierra la ventana

```
Project (Ámbito 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, Light_Green);
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 }
258 CloseWindow();
259 return 0;
260
261 }
```

Funcionamiento del juego:

MENU:

# PING-PONG

1. Jugar
2. Puntajes
3. Creditos
4. Salir

# Puntajes

## PING-PONG

Puntajes

Jugador 1: 0

Jugador 2: 0

1. Jugar

2. Puntajes

3. Creditos

4. Salir

# Creditos:

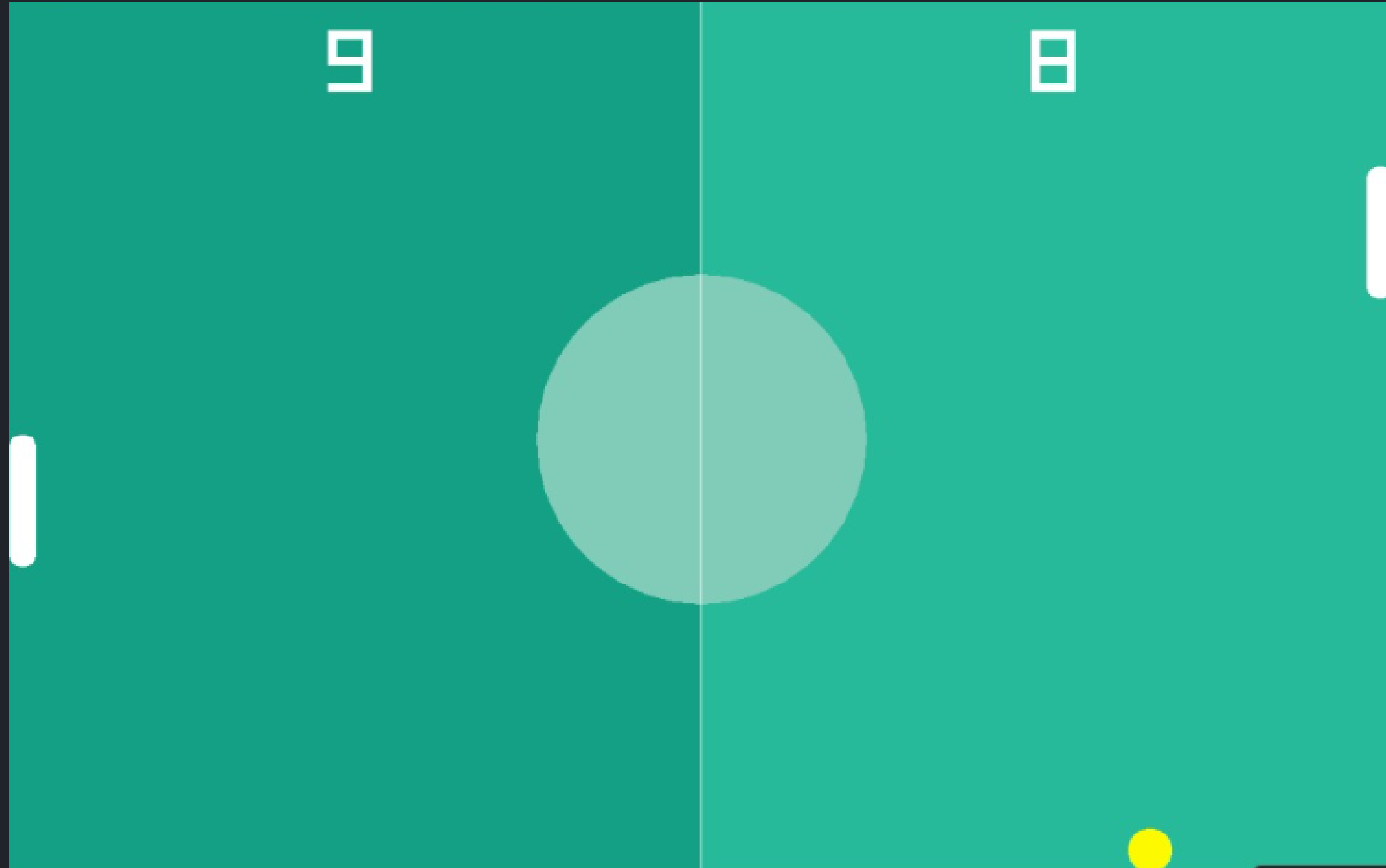
Hecho por [ALEJANDRO SANGUCHO]

Hecho por [STIVEN VISCAINOS]

# PING-PONG

1. Jugar
2. Puntajes
3. Creditos
4. Salir

Juego:



Juego:

!Jugador 1 es el ganador!

Gracias Por su  
atencion {

}