# PING PONG


Gonzalo Espinoza, Diego Osuna.
Universidad Autónoma de Baja California
Facultad de Ingeniería Arquitectura y Diseño
Programación Estructurada

Pedro Nuñes Yepiz

**12 de Diciembre de 2023**

# Código:

```c
#include <SDL2/SDL_error.h>
#include <SDL2/SDL_keycode.h>
#include <SDL2/SDL_render.h>
#include <SDL2/SDL_surface.h>
#include <SDL2/SDL_timer.h>
#include <SDL2/SDL_video.h>
#include <SDL2/SDL_ttf.h>
#ifdef _WIN32
#define _UNICODE
#endif
#include <GL/glew.h>
#include <SDL2/SDL.h>
#include <SDL2/SDL_opengl.h>
#include <SDL2/SDL_mixer.h>
#include <SDL2/SDL_image.h>
#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>
#include <time.h>

#include "shaders.h"

typedef struct _paddle
{
float x;
float y;
float width;
float height;
int score;
int up;
int down;
SDL_Rect sprite;
SDL_Texture *tex;
int noted;
} tpaddle;

typedef struct _ball
{
float x;
float y;
float spd_x;
```

```
                float spd_y;
                  float r;
               SDL_Rect sprite;
              SDL_Texture *tex;
                int vert_dir;
                int hort_dir;
                } tball;

            typedef tpaddle *paddleObj;
              typedef tball *ballObj;

            SDL_Surface *surface = NULL;
             SDL_Window *window = NULL;
             SDL_Surface *image = NULL;

  void handleKeyboardEvents(SDL_Event * event, tpaddle *paddle_p1,
              tpaddle *paddle_p2, int *value);
          void menuKeyEvents(SDL_Event * event);
            void movePaddle(tpaddle *paddle);
void reset_game(tpaddle *paddle_p1, tpaddle *paddle_p2, tball *ball);

               SDL_Rect introPosition;

                    #ifdef _WIN32
            int wmain(int argc, char *argv[])
                          {
                        #endif
                    #ifdef __linux__

             int main(int argc, char *argv[])
                          {
                        #endif
        paddleObj paddle_p1 = malloc(sizeof(tpaddle));
        paddleObj paddle_p2 = malloc(sizeof(tpaddle));
            ballObj ball = malloc(sizeof(tball));
        char *p1_text_score = malloc(sizeof(char)*32);
        char *p2_text_score = malloc(sizeof(char)*32);
        char *scoreboards = malloc(sizeof(char)*1500);
                Mix_Chunk *bounce1;
              Mix_Chunk *russiaWins;
             Mix_Chunk *americaWins;
               int game_start = 0;
                 int scored = 0;
```

```
float mult_vert = 2.5f;
float mult_hor = 2.5f;
FILE *archivo;
const char *score_file = "scores.txt";
const char *story_file = "guion.txt";
time_t current_time;
struct tm *info_time;
int menu_opt = 0;

float y_text_pos = 0;

Mix_MasterVolume(75);
paddle_p1->x = 25.0f;
paddle_p1->y = 150.0f;
paddle_p1->width = 20.0f;
paddle_p1->height = 100.0f;
paddle_p1->score = 0;
paddle_p1->sprite.x = (int)paddle_p1->x;
paddle_p1->sprite.y = (int)paddle_p1->y;
paddle_p1->sprite.w = (int)paddle_p1->width;
paddle_p1->sprite.h = (int)paddle_p1->height;
paddle_p1->up = 0;
paddle_p1->down = 0;
paddle_p1->noted = 0;

paddle_p2->x = 750.0f;
paddle_p2->y = 150.0f;
paddle_p2->width = 20.0f;
paddle_p2->height = 100.0f;
paddle_p2->score = 0;
paddle_p2->sprite.x = paddle_p2->x;
paddle_p2->sprite.y = paddle_p2->y;
paddle_p2->sprite.w = paddle_p2->width;
paddle_p2->sprite.h = paddle_p2->height;
paddle_p2->up = 0;
paddle_p2->down = 0;
paddle_p2->noted = 0;

ball->x = 390.0f;
ball->y = 190.0f;
ball->spd_x = 2.5f;
ball->spd_y = 2.0f;
ball->r = 20;
```

```
                    ball->sprite.x = ball->x;
                    ball->sprite.y = ball->y;
                    ball->sprite.w = ball->r;
                    ball->sprite.h = ball->r;
                       ball->vert_dir = 1;
                       ball->hort_dir = 1;

        SDL_Surface* iconSurface = IMG_Load("iconopp.png");
                       if (!iconSurface) {
                  printf("NO SE CARGO EL ICONO");
                                }

                 if (SDL_Init(SDL_INIT_VIDEO) < 0)
                                {
            printf("SDL no pudo inicializarse. Error: %s\n",
                          SDL_GetError());
                                return -1;
                                }
     if (Mix_OpenAudio(44100, MIX_DEFAULT_FORMAT, 2, 2048) < 0)
                                {
    printf("SDL no pudo inicializar el mixer %s\n", Mix_GetError);
                               SDL_Quit();
                                return -1;
                                }
                 bounce1 = Mix_LoadWAV("bounce.wav");
              americaWins = Mix_LoadWAV("americaWins.wav");
               russiaWins = Mix_LoadWAV("russiaWin.wav");
                            if (!bounce1)
                                {
       printf("No se pudo cargar el sonido %s\n", Mix_GetError);
                            Mix_CloseAudio();
                               SDL_Quit();
                                return -1;
                                }
                       if (TTF_Init() == -1)
                                {
      printf("Error al iniciar las fuentes %s\n", TTF_GetError);
                               SDL_Quit();
                                return -1;
                                }

    window = SDL_CreateWindow("Ping-Pong", SDL_WINDOWPOS_CENTERED,
      SDL_WINDOWPOS_CENTERED, 1000, 400, SDL_WINDOW_SHOWN);
```

```
                        if (!window)
                            {
    printf("No se pudo crear la ventana SDL. Error: %s\n",
                    SDL_GetError());
                        SDL_Quit();
                        return -1;
                            }

      SDL_SetWindowIcon(window, iconSurface);//<----

    SDL_Renderer* renderer = SDL_CreateRenderer(window, -1,
                SDL_RENDERER_ACCELERATED);
                        if (!renderer) {
    printf("Error al crear el renderer: %s\n", SDL_GetError());
                    SDL_DestroyWindow(window);
                        IMG_Quit();
                        SDL_Quit();
                        return -1;
                            }

                Mix_Music *backgroundMusic;

                // Superficie para dibujar
            surface = SDL_GetWindowSurface(window);
                        if (!surface)
                            {
printf("No se pudo obtener la superficie de la ventana. Error:
                %s\n", SDL_GetError());
                    SDL_DestroyWindow(window);
                        SDL_Quit();
                        return -1;
                            }
    SDL_Surface *img_surf = IMG_Load("background.png");
                        if (!img_surf)
                            {
    printf("Error al cargar la imagen %s\n", IMG_GetError);
                    SDL_DestroyWindow(window);
                SDL_DestroyRenderer(renderer);
                        IMG_Quit();
                        SDL_Quit();
                        return -1;
                            }
```

```
    SDL_Texture *img_tex = SDL_CreateTextureFromSurface(renderer,
                                img_surf);
                    img_surf = IMG_Load("menu_bkg.png");
     SDL_Texture *menu_bkg = SDL_CreateTextureFromSurface(renderer,
                                img_surf);
                img_surf = IMG_Load("PaddleSovietico.png");
paddle_p1->tex = SDL_CreateTextureFromSurface(renderer, img_surf);
                img_surf = IMG_Load("PaddleAmericano.png");
paddle_p2->tex = SDL_CreateTextureFromSurface(renderer, img_surf);
                    img_surf = IMG_Load("ball.png");
     ball->tex = SDL_CreateTextureFromSurface(renderer, img_surf);
                        SDL_FreeSurface(img_surf);
                    img_surf = IMG_Load("guion_bkg.png");
    SDL_Texture *guion_bkg = SDL_CreateTextureFromSurface(renderer,
                                img_surf);
                  img_surf = IMG_Load("scores_bkg.png");
   SDL_Texture *scores_bkg = SDL_CreateTextureFromSurface(renderer,
                                img_surf);

                                if (!img_tex)
                                        {
        printf("Error al crear la textura %s\n", SDL_GetError());
                                SDL_DestroyWindow(window);
                               SDL_DestroyRenderer(renderer);
                                        IMG_Quit();
                                        SDL_Quit();
                                        return -1;
                                        }

         TTF_Font *font = TTF_OpenFont("Strengthen.ttf", 50);
                               if (font == NULL)
                                        {
      printf("Error al cargar nuestra fuente %s\n", TTF_GetError);
                                SDL_DestroyWindow(window);
                               SDL_DestroyRenderer(renderer);
                                        IMG_Quit();
                                        SDL_Quit();
                                        return -1;
                                        }
                  SDL_Color color = {255, 255, 255, 255};
       img_surf = TTF_RenderText_Solid(font, "hello", color);
    SDL_Texture *score_tex = SDL_CreateTextureFromSurface(renderer,
                                img_surf);
```

```
                    SDL_UpdateWindowSurface(window);
                            SDL_Event event;
                              int menu = 1;

   introPosition.x = 0; // Posición x de la esquina superior izquierda
   introPosition.y = 0; // Posición y de la esquina superior izquierda
      introPosition.w = 1000; // Ancho de tu imagen de introducción
                          introPosition.h = 400; //
         // Duración total de la introducción en milisegundos

                  SDL_Texture *introTexture = NULL;
      SDL_Surface *introSurface = IMG_Load("fondo_intro.png");
                          if (!introSurface) {
                 printf("NO SE CARGO LA INTRO IMAGEN");
                                  }

introTexture = SDL_CreateTextureFromSurface(renderer, introSurface);
                    SDL_FreeSurface(introSurface);

            const Uint32 TIEMPO_TOTAL_INTRODUCCION = 5000;
                Uint32 introStartTime = SDL_GetTicks();

          Uint32 introTime = 0; // Inicializar introTime aquí

             while (introTime <= TIEMPO_TOTAL_INTRODUCCION) {
               Uint32 introTime = SDL_GetTicks() - introStartTime;

          // Procesar la introducción según el tiempo transcurrido
                          if (introTime <= 1000) {
          // Ajustar gradualmente la transparencia para aparecer en 1
                                  segundo
            Uint8 alphaValue = (Uint8)((introTime / 1000.0f) * 255);
               SDL_SetTextureAlphaMod(introTexture, alphaValue);
             } else if (introTime > 1000 && introTime <= 4000) {
          // Mantener la imagen visible durante 5 segundos (5000 ms)
                   SDL_SetTextureAlphaMod(introTexture, 255);
                } else if (introTime > 4000 && introTime <=
                      TIEMPO_TOTAL_INTRODUCCION) {
                  // Desvanecer la imagen durante 1 segundo
            Uint8 alphaValue = (Uint8)(((TIEMPO_TOTAL_INTRODUCCION -
                      introTime) / 1000.0f) * 255);
               SDL_SetTextureAlphaMod(introTexture, alphaValue);
```

```
                } else {
            // La animación de introducción ha finalizado
        break; // Salir del bucle después de que la introducción haya
                finalizado
                }

    // Limpiar el renderer y copiar la textura en cada iteración
                SDL_RenderClear(renderer);
    SDL_RenderCopy(renderer, introTexture, NULL, &introPosition);
                SDL_RenderPresent(renderer);

    // Pequeña pausa para controlar la velocidad de la introducción
                SDL_Delay(10);
                }



                while (menu)
                {
            while (SDL_PollEvent(&event))
                    {
                if (event.type == SDL_QUIT)
                    {
                SDL_DestroyWindow(window);
                    SDL_Quit();
                    return 0;
                    }
                if (event.type == SDL_KEYDOWN)
                    {
            if (event.key.keysym.sym == SDLK_ESCAPE)
                    {
                SDL_DestroyWindow(window);
                    SDL_Quit();
                    return 0;
                    }
            if (event.key.keysym.sym == SDLK_RETURN)
                    {
                    menu = 0;
                    }
            if (event.key.keysym.sym == SDLK_BACKSPACE)
                    {
                    menu_opt = 0;
                    archivo = NULL;
                    }
```

```
                    if (event.key.keysym.sym == SDLK_s)
                                {
                            menu_opt = 1;
                           y_text_pos = 0;
                                }
                    if (event.key.keysym.sym == SDLK_g)
                                {
                            menu_opt = 2;
                                }
                              }
                            }

                      if (menu_opt == 0) {
                     SDL_RenderClear(renderer);

               SDL_RenderCopy(renderer, menu_bkg, NULL, NULL);

                     SDL_RenderPresent(renderer);
                               }
                      if (menu_opt == 1) {
                     SDL_RenderClear(renderer);
               SDL_RenderCopy(renderer, scores_bkg, NULL, NULL);

                  FILE *archivo = fopen(score_file, "r");
                          if (archivo == NULL) {
                   printf("No se pudo abrir el archivo\n");
                                }

                           float y_pos = 400;

                           char buffer[256];

          while (fgets(buffer, sizeof(buffer), archivo) != NULL) {
                     char *newline = strchr(buffer, '\n');
                            if (newline != NULL) {
                                 *newline = '\0';
                                }

                     SDL_Color color = {45, 35, 35, 255};
               SDL_Rect dstRect = {250, y_pos+y_text_pos, 500, 25};
                SDL_Surface *img_surf = TTF_RenderText_Solid(font,
                          buffer, color);
```

```
                    SDL_Texture *score_tex =
SDL_CreateTextureFromSurface(renderer, img_surf);
        SDL_RenderCopy(renderer, score_tex, NULL, &dstRect);

                            y_pos += 25;

                        SDL_FreeSurface(img_surf);
                    SDL_DestroyTexture(score_tex);
                                }
                        y_text_pos -= 0.5;

                        fclose(archivo);

                    SDL_RenderPresent(renderer);
                                }
                    if (menu_opt == 2) {
                    SDL_RenderClear(renderer);
        SDL_RenderCopy(renderer, guion_bkg, NULL, NULL);

                FILE *archivo = fopen(story_file, "r");
                        if (archivo == NULL) {
                printf("No se pudo abrir el archivo\n");
                            return 1;
                              }

                        float y_pos = 350;

                        char buffer[256];

    while (fgets(buffer, sizeof(buffer), archivo) != NULL) {
                char *newline = strchr(buffer, '\n');
                        if (newline != NULL) {
                             *newline = '\0';
                                  }

         SDL_Rect dstRect = {25, y_pos+y_text_pos, 950, 25};
          SDL_Surface *img_surf = TTF_RenderText_Solid(font,
                    buffer, color);
                        SDL_Texture *score_tex =
SDL_CreateTextureFromSurface(renderer, img_surf);
        SDL_RenderCopy(renderer, score_tex, NULL, &dstRect);

                            y_pos += 30;
```

```
                SDL_FreeSurface(img_surf);
                SDL_DestroyTexture(score_tex);
                }
                y_text_pos -= 10;

                fclose(archivo);

                SDL_RenderPresent(renderer);
                }
                SDL_Delay(10);
                }
        backgroundMusic = Mix_LoadMUS("musica_fondo.wav");
                if (!backgroundMusic) {
   printf("Error al cargar la música de fondo: %s\n", Mix_GetError());
     // Manejar el error (puede ser que el archivo no exista o no se
                        pueda cargar)
                }
                while (true)
                {
        const int windowHeight = 400; // Altura de la ventana
       const int paddleHeight = 100; // Altura de la barra (paddle)

                while (SDL_PollEvent(&event))
                {
                if (event.type == SDL_QUIT)
                {
                SDL_DestroyWindow(window);
                SDL_Quit();
                return 0;
                }
            handleKeyboardEvents(&event, paddle_p1, paddle_p2,
&game_start); // Llamar a la función para manejar eventos del teclado
                }
                if (Mix_PlayingMusic() == 0) {
            Mix_PlayMusic(backgroundMusic, -1); // -1 para reproducir
                        en bucle
                }
                // Lógica del juego
                if (game_start)
                {
                ball->x += ball->spd_x*ball->hort_dir;
                ball->y += ball->spd_y*ball->vert_dir;
```

```
                                    }
                        ball->sprite.x = ball->x;
                        ball->sprite.y = ball->y;

                    if (ball->y >= 380 || ball->y <= 0)
                                    {
                        ball->vert_dir = -ball->vert_dir;
                        Mix_PlayChannel(-1, bounce1, 0);
                                    }


    if (ball->x <= 10 && (ball->y <= paddle_p1->y || ball->y >=
            paddle_p1->y + paddle_p1->height))
                                    {
                    reset_game(paddle_p1, paddle_p2, ball);
                            paddle_p2->score++;
                           paddle_p1->noted = 1;
                     Mix_PlayChannel(-1, americaWins, 0);
        printf("¡Jugador 2 anotó un punto! Puntuación: Jugador 1:
%d, Jugador 2: %d\n", paddle_p1->score, paddle_p2->score);

        // Incrementar la velocidad de la pelota después de anotar
                            un punto
        printf("Velocidad X: %f, Velocidad Y: %f\n", ball->spd_x,
                        ball->spd_y);
                             game_start = 0;
                                    }

   // Si la pelota cruza la línea 730 y no está cerca del paddle
                        derecho, se reinicia
    if (ball->x >= 780 && (ball->y <= paddle_p2->y || ball->y >=
            paddle_p2->y + paddle_p2->height))
                                    {
                      Mix_PlayChannel(-1, russiaWins, 0);
                    reset_game(paddle_p1, paddle_p2, ball);
                             game_start = 0;
                           paddle_p1->score++;
                           paddle_p2->noted = 1;
        printf("¡Jugador 1 anotó un punto! Puntuación: Jugador 1:
%d, Jugador 2: %d\n", paddle_p1->score, paddle_p2->score);

        // Incrementar la velocidad de la pelota después de anotar
                            un punto
```

```
            printf("Velocidad X: %f, Velocidad Y: %f\n", ball->spd_x,
                                ball->spd_y);

                                    }

                if (ball->x >= paddle_p1->x && ball->x <
  paddle_p1->width+ball->r && ball->y + ball->r >= paddle_p1->y &&
            ball->y <= paddle_p1->y + paddle_p1->height)
                                    {
                              //ball->spd_x = -ball->spd_x;
                           ball->hort_dir = -ball->hort_dir;
                           Mix_PlayChannel(-1, bounce1, 0);
             if (!paddle_p1->noted && !paddle_p2->noted && scored)
                                        {
                             ball->spd_x /= mult_hor;
                            ball->spd_y /= mult_vert;
                                   scored = 0;
                                ball->x += 8.0f;
                                        }
                             if (paddle_p1->noted)
                                        {
                             ball->spd_x *= mult_hor;
                            ball->spd_y *= mult_vert;
                              paddle_p1->noted = 0;
                                   scored = 1;
                                        }
                                    }

             if(paddle_p1->score >= 5 || paddle_p2->score >= 5)
                                    {
                          archivo = fopen(score_file, "r");
                                if (archivo == NULL)
                                        {
                            printf("Creando archivo\n");
                           archivo = fopen(score_file, "w");
                                 if (archivo == NULL)
                                          {
                             printf("Algo falló\n");
                                    return 1;
                                        }
                                    }
                                  else
                                        {
```

```
                                fclose(archivo);
                            archivo = fopen(score_file, "a");
                                        }
                                    if (archivo == NULL)
                                        {
                                printf("No se pudo abrir el archivo\n");
                                        }
                                    time(&current_time);
                                info_time = localtime(&current_time);
                                fprintf(archivo, "%02d/%02d/%04d\n",
                                        info_time->tm_mday,
                                        info_time->tm_mon,
                                        info_time->tm_year);
                                if(paddle_p1->score > paddle_p2->score)
                                        {
                                        printf("jugador 1 gano\n");
                                fprintf(archivo, "Ganador: Union Sovietica\n");
                                        }
                                if(paddle_p2->score > paddle_p1->score)
                                        {
                                        printf("jugador 2 gano\n");
                                fprintf(archivo, "Ganador: Estados Unidos\n");
                                        }
                                fprintf(archivo, "USA[%d] vs URS[%d]\n\n",
                            paddle_p1->score, paddle_p2->score);
                                        fclose(archivo);
                                    printf("Datos guardados\n");
                                        paddle_p1->score = 0;
                                        paddle_p2->score = 0;

                                        }

            if (ball->x >= paddle_p2->x - ball->r && ball->x <=
    paddle_p2->x && ball->y + ball->r >= paddle_p2->y && ball->y <=
                    paddle_p2->y + paddle_p2->height)
                                        {
                                ball->hort_dir = -ball->hort_dir;
                                Mix_PlayChannel(-1, bounce1, 0);
                if (!paddle_p1->noted && !paddle_p2->noted && scored)
                                        {
                                        ball->spd_x /= mult_hor;
                                        ball->spd_y /= mult_vert;
                                                scored = 0;
```

```
                    ball->x -= 8.0f;
                }
            if (paddle_p2->noted)
                {
            ball->spd_x *= mult_vert;
             ball->spd_y *= mult_hor;
              paddle_p2->noted = 0;
                   scored = 1;
                }
             }


        if (paddle_p1->y > windowHeight)
                {
    paddle_p1->y = -paddleHeight; // Si la barra sale por
            abajo, aparece por arriba
                }
       else if (paddle_p1->y < -paddleHeight)
                {
     paddle_p1->y = windowHeight; // Si la barra sale por
            arriba, aparece por abajo
                }

        if (paddle_p2->y > windowHeight)
                {
  paddle_p2->y = -paddleHeight; // Lo mismo para la segunda
                barra
                }
       else if (paddle_p2->y < -paddleHeight)
                {
    paddle_p2->y = windowHeight; // Si la barra sale por
            arriba, aparece por abajo
                }
           // En el bucle principal:
            movePaddle(paddle_p1);
            movePaddle(paddle_p2);
          SDL_RenderClear(renderer);

      SDL_RenderCopy(renderer, img_tex, NULL, NULL);

    SDL_RenderFillRect(renderer, &(paddle_p2->sprite));

SDL_RenderCopy(renderer, ball->tex, NULL, &ball->sprite);
```

```
                SDL_RenderCopy(renderer, paddle_p1->tex, NULL,
                        &paddle_p1->sprite);
                SDL_RenderCopy(renderer, paddle_p2->tex, NULL,
                        &paddle_p2->sprite);

                    SDL_Rect dstRect = {830, 50, 30, 50};
               sprintf(p1_text_score, "%d", paddle_p1->score);
          img_surf = TTF_RenderText_Solid(font, p1_text_score, color);
         score_tex = SDL_CreateTextureFromSurface(renderer, img_surf);
              SDL_RenderCopy(renderer, score_tex, NULL, &dstRect);
                              dstRect.x = 940;
               sprintf(p2_text_score, "%d", paddle_p2->score);
          img_surf = TTF_RenderText_Solid(font, p2_text_score, color);
         score_tex = SDL_CreateTextureFromSurface(renderer, img_surf);
              SDL_RenderCopy(renderer, score_tex, NULL, &dstRect);


                          SDL_RenderPresent(renderer);

                               SDL_Delay(10);
                                    }
                        Mix_FreeMusic(backgroundMusic);
                            SDL_FreeSurface(image);
                         SDL_FreeSurface(iconSurface);
                        SDL_DestroyWindow(window);
                               SDL_Quit();
                                return 0;
                                  }

    void handleKeyboardEvents(SDL_Event * event, tpaddle *paddle_p1,
                    tpaddle *paddle_p2, int *game_start)
                                    {
                         if (event->type == SDL_KEYDOWN)
                                        {
                                *game_start = 1;
                        if (event->key.keysym.sym == SDLK_w)
                                         {
                                 paddle_p1->up = 1;
                                         }
                        if (event->key.keysym.sym == SDLK_s)
                                         {
                                paddle_p1->down = 1;
                                         }
```

```
                if (event->key.keysym.sym == SDLK_UP)
                                {
                            paddle_p2->up = 1;
                                }
                if (event->key.keysym.sym == SDLK_DOWN)
                                {
                           paddle_p2->down = 1;
                                }

            if (event->key.keysym.sym == SDLK_ESCAPE)
                                {
                     SDL_DestroyWindow(window);
                             SDL_Quit();
                                }
                            }

              if (event->type == SDL_KEYUP)
                            {
              if (event->key.keysym.sym == SDLK_w)
                                {
                            paddle_p1->up = 0;
                                }
              if (event->key.keysym.sym == SDLK_s)
                                {
                           paddle_p1->down = 0;
                                }
               if (event->key.keysym.sym == SDLK_UP)
                                {
                            paddle_p2->up = 0;
                                }
                if (event->key.keysym.sym == SDLK_DOWN)
                                {
                           paddle_p2->down = 0;
                                }
                            }


                        }

            void movePaddle(tpaddle *paddle)
                            {

                       if (paddle->up)
                                {
```

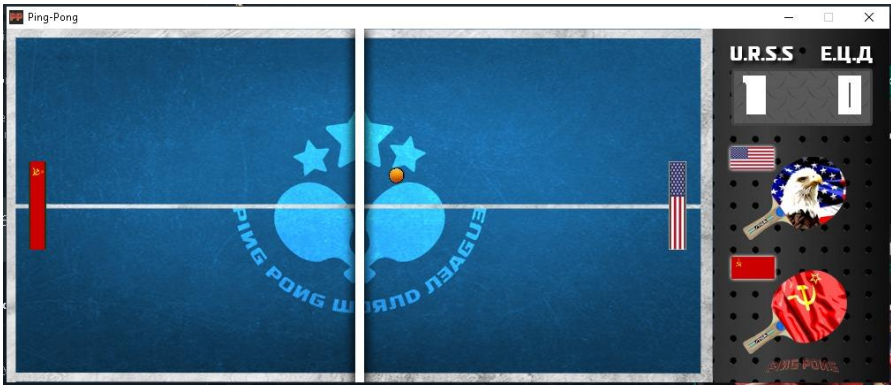
```
                    // Mover la raqueta izquierda hacia arriba
                                paddle->y -= 4.0f;
                            paddle->sprite.y = paddle->y;
                                        }
                                    if (paddle->down)
                                        {
                    // Mover la raqueta izquierda hacia abajo
                                paddle->y += 4.0f;
                            paddle->sprite.y = paddle->y;
                                        }
                                    }

void reset_game(tpaddle *paddle_p1, tpaddle *paddle_p2, tball *ball)
                                    {
                                ball->x = 390;
                                ball->y = 190;
                            paddle_p1->y = 150.0f;
                            paddle_p2->y = 150.0f;
                        paddle_p1->sprite.y = paddle_p1->y;
                        paddle_p2->sprite.y = paddle_p2->y;
                                ball->spd_x = 2.5f;
                                ball->spd_y = 2.0f;
                                    }
```

## Pantallas:


Pantalla de carga


Pantalla del menú




Pantalla de juego

## Conclusión:

Para este proyecto utilizamos el lenguaje C para la creación de un juego donde utilizamos la biblioteca SDL. El juego se basa en el juego de Ping Pong, implica dos raquetas controladas por los  jugadores que intentan golpear la pelota para anotar puntos. A lo largo del código, se emplean diversas funciones de la biblioteca SDL para gestionar eventos de teclado como manipular gráficos, reproducir música de fondo y efectos de sonido.

A la hora de implementar la lógica del juego como las colisiones, el seguimiento de puntuación y la dinámica de la pelota y las raquetas, nos obligó a pensar en soluciones y utilizar lo visto la clase de programación estructurada lo que nos permite ver cómo es que se aplican los conocimientos vistos en clase, así también nos permitió trabajar con nuevas herramientas como los son las bibliotecas gráficas permitiendo ver las utilidades que estas nos brindan .