

Data Structures and Algorithms-1 Lab

Project Report

Course Title : Data Structures and Algorithms-1 Lab

Course Code : CSE0613212

Title : DSA-1 Lab final project about “2D Game” using Raylib and C++

Submitted to:

|  |  |
| --- | --- |
| **Teacher’s Name** | **Designation** |
| Sadia Afrin Urmi | Lecturer |
| Saima Siddique Tashfia | Lecturer |

Submitted by:

Student Name : Gaus Saraf Murady

Student ID : 0432410005101088

Semester : Autumn, 2024

Batch : 55

Department : CSE

Date of Submission: 27.05.25

Objective:

The objective of this DSA-1 Lab Project is to build a game using C++. We were tasked with making any type of project as long as it is done using C++. Therefore, I decided to make a 2D game using Raylib.

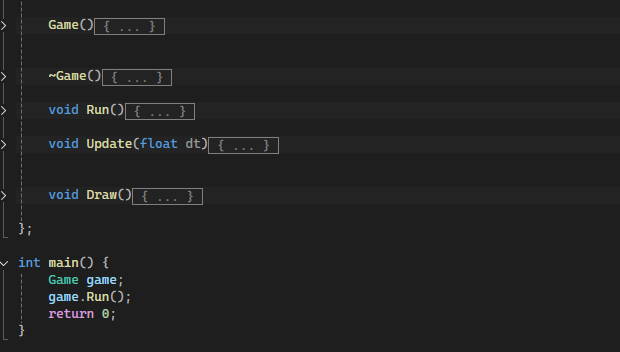
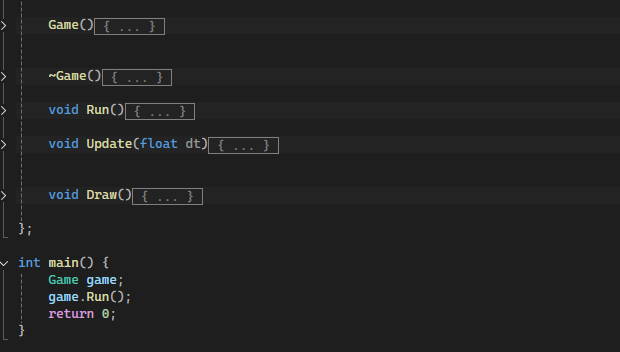
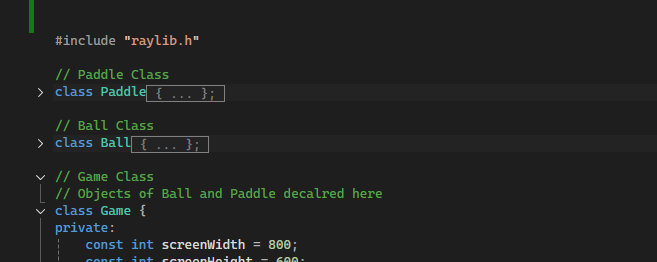
Initially I wanted to make a game using Godot. But only later I realized that it would be more work just to learn in my given deadline. Because Godot Engine is not meant to be used with C++ natively. But I discovered a few ways like using GDExtension and JENOVA framework to work around that hurdle.

Tools:

* Visual Studio 2022
* [raylib | A simple and easy-to-use library to enjoy videogames programming](https://www.raylib.com/)
* [microsoft/vcpkg: C++ Library Manager for Windows, Linux, and MacOS](https://github.com/microsoft/vcpkg)
* GitHub to Store the Game files
* ChatGPT for debugging and documentations

Methodology:

For my final project, I used Raylib; a simple C/C++ library for making 2D games. I planned to make a juggling game where I need to keep bouncing a 2D ball as long as I can. Each hit with the paddle gives a point.



These are the main components of my project:

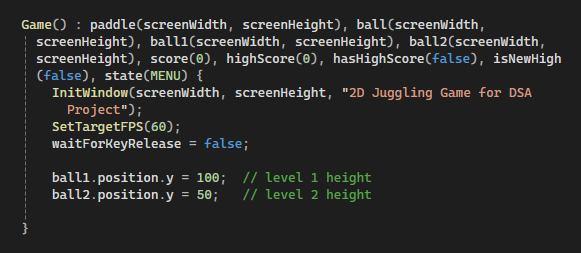
* A Main Function
* A Paddle class (Player controlled)
* A Ball class
* A Game class (Inside which everything else is run)

A screen shot of a computer

AI-generated content may be incorrect.

This part defines the Game Loop, that generates frames for the game. Here the framerate is set to be 60 per second, using SetTargetFPS(60).

A screenshot of a computer program

AI-generated content may be incorrect.

The InItWindow defines the Windows size for the whole game, which is located inside the Game constructor under the Game class.

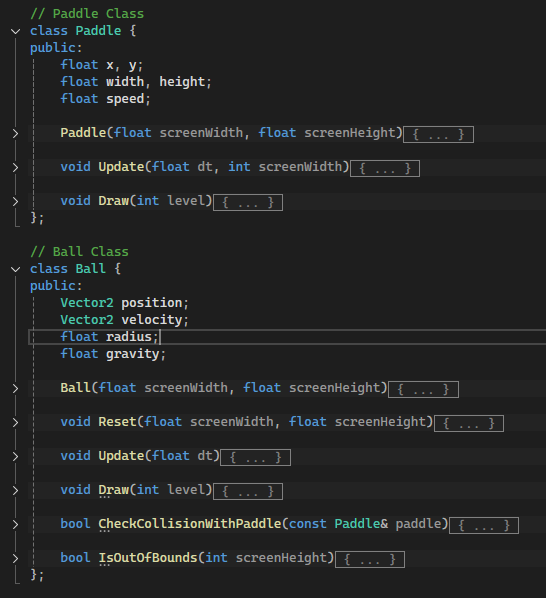
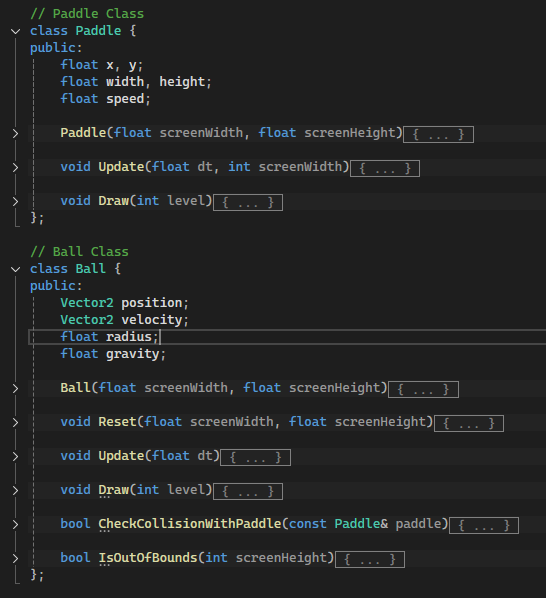
For each frame, the Game Loop calls the Update() and Draw() Methods under the Game class that each call the Update() and Draw() Methods under the paddle and Ball classes. Therefore, generating Paddle and Ball animation for the game.

A screenshot of a computer program

AI-generated content may be incorrect.A computer screen shot of a program code

AI-generated content may be incorrect.

These are the Paddle and Ball classes:



My game has 3 total levels, each denoting greater difficulty than the previous one. I added their themes based on the level we are currently playing. (Player and Ball respectively)

A screen shot of a computer code

AI-generated content may be incorrect.A computer screen shot of a program code

AI-generated content may be incorrect.

A computer screen shot of a program code

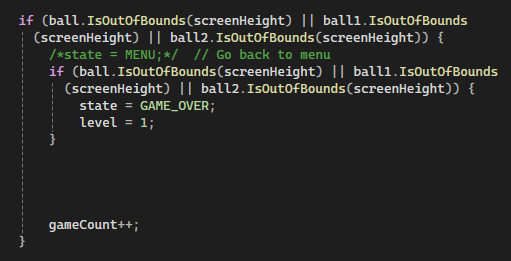
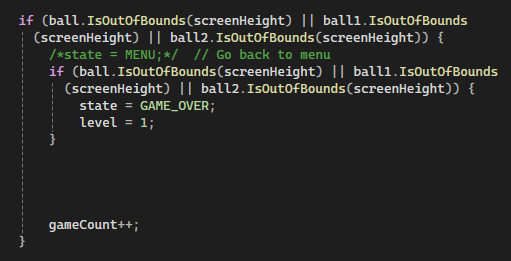
AI-generated content may be incorrect.

This is the scoring logic. Each 5 points, there will be a prompt asking the player if they want to level up, or add difficulty by adding another ball to juggle.

A screen shot of a computer code

AI-generated content may be incorrect.A screen shot of a computer program

AI-generated content may be incorrect.High scores are counted separately. So the game remembers previous highest and highlights when a new Highest Score is achieved. There is not multiple user support, unfortunately.



This is the logic for Game Over, set to show the End Game Screen whenever a ball does not hit the paddle or goes “out of bounds”.

A screen shot of a computer program

AI-generated content may be incorrect.

This portion of the code resets the variables to their default values to start as a New Game. But the High Scores remains as they need to be recorded.

A screen shot of a computer program

AI-generated content may be incorrect.

This the is graphics generating code for End Game screen.

Result:

A screenshot of a video game

AI-generated content may be incorrect. I have made 3 Screens for this Game Project:

* Main Menu / Title Screen
* The main gameplay screen: (each level changes the theme a bit)

A screen shot of a computer

AI-generated content may be incorrect.A screenshot of a video game

AI-generated content may be incorrect.Level-1: Level-2:

A screenshot of a video game

AI-generated content may be incorrect.level-3:

* A screen shot of a computer

  AI-generated content may be incorrect.And the Game Over screen:

Note:

If you want to run the game files, first install Raylib on your computer.

Steps:

* Download the zipped <https://github.com/microsoft/vcpkg>
* Unzip vcpkg in System Root (C drive), rename folder (remove -master) then run in powershell (type “powershell” in the explorer address bar):  
          ./bootstrap-vcpkg.bat  
          ./vcpkg integrate install  
          ./vcpkg install raylib:x64-windows

Conclusion:

I have completed mostly what I wanted to do, but since I had wasted so much time trying to use Godot, I faced a bit of a time shortage and therefore, had to rush to finish the project. But I had nothing planned previously, so everything I did was a full completion.

I want to add multiple user support in the future. I also wanted to add a “Zen Mode” or easy mode where the Game would keep going on as long as any one of the balls are present on the screen or is not “out of bounds”, but that part was hard to achieve with my current expertise of the Raylib framework.

References:

* Tutorials:
  + [New to raylib? Start here by Programming With Nick](https://www.youtube.com/playlist?list=PLwR6ZGPvjVOTIMqUXnqyWaIfQg0xdHNZn)
* Documentations:
  + [raylib - cheatsheet](https://www.raylib.com/cheatsheet/cheatsheet.html)
  + [raylib [shapes] example - colors palette](https://www.raylib.com/examples/shapes/loader.html?name=shapes_colors_palette)
* Reference Projects on GitHub:
  + [Super-Mango/Super Mango/Super Mango at main · Azazo8/Super-Mango](https://github.com/Azazo8/Super-Mango/tree/main/Super%20Mango/Super%20Mango)
  + <https://github.com/educ8s/Cpp-Pong-Game-Raylib>
  + [educ8s/Raylib-CPP-Starter-Template-for-VSCODE-V2: Raylib C++ Starter Template for VSCODE V2](https://github.com/educ8s/Raylib-CPP-Starter-Template-for-VSCODE-V2)
  + [educ8s/Intro-To-raylib-Tutorial: You first raylib project. A white ball that can be controlled from the keyboard. This project is accompanied by a video tutorial that explains everything in detail.](https://github.com/educ8s/Intro-To-raylib-Tutorial)

GitHub Repository:

<https://github.com/b1tranger/DSA-project>

Project Code:

(Added last for keeping the content clean)

#include "raylib.h"

// Paddle Class

class Paddle {

public:

float x, y;

float width, height;

float speed;

Paddle(float screenWidth, float screenHeight) {

width = 100;

height = 20;

x = screenWidth / 2 - width / 2;

y = screenHeight - 50;

speed = 500;

}

void Update(float dt, int screenWidth) {

if (IsKeyDown(KEY\_LEFT)) x -= speed \* dt;

if (IsKeyDown(KEY\_RIGHT)) x += speed \* dt;

// Clamp

if (x < 0) x = 0;

if (x > screenWidth - width) x = screenWidth - width;

}

void Draw(int level) {

if (level == 1) {

DrawRectangleRounded(Rectangle{ x, y, width, height }, 0.8, 0, DARKBLUE);

//DrawRectangle((int)x, (int)y, (int)width, (int)height, SKYBLUE);

}

else if (level == 2) {

DrawRectangleRounded(Rectangle{ x, y, width, height }, 0.8, 0, DARKGRAY);

}

else if (level == 3) {

DrawRectangleRounded(Rectangle{ x, y, width, height }, 0.8, 0, YELLOW);

}

}

};

// Ball Class

class Ball {

public:

Vector2 position;

Vector2 velocity;

float radius;

float gravity;

Ball(float screenWidth, float screenHeight) {

radius = 10;

gravity = 400;

Reset(screenWidth, screenHeight);

}

void Reset(float screenWidth, float screenHeight) {

position = { screenWidth / 2.0f, screenHeight / 2.0f };

velocity = { 200, 0 };

}

void Update(float dt) {

velocity.y += gravity \* dt;

position.x += velocity.x \* dt;

position.y += velocity.y \* dt;

if (position.x - radius <= 0) { //left overflow

velocity.x \*= -1;

position.x = radius;

}

if (position.x + radius >= GetScreenWidth()) { // right overflow

velocity.x \*= -1;

position.x = GetScreenWidth() - radius;

}

}

void Draw(int level) {

if (level == 1) {

DrawCircleV(position, radius, BLACK);

}

else if (level == 2) {

DrawCircleV(position, radius, BLUE);

}

else if (level == 3) {

DrawCircleV(position, radius, RED);

}

}

bool CheckCollisionWithPaddle(const Paddle& paddle) {

return position.y + radius >= paddle.y &&

position.x >= paddle.x &&

position.x <= paddle.x + paddle.width &&

velocity.y > 0;

}

bool IsOutOfBounds(int screenHeight) {

return position.y - radius > screenHeight;

}

};

// Game Class

// Objects of Ball and Paddle decalred here

class Game {

private:

const int screenWidth = 800;

const int screenHeight = 600;

Paddle paddle;

Ball ball;

Ball ball1;

Ball ball2;

int score;

//int highScore;

//GameState state;

int highScore;

bool isNewHigh;

bool hasHighScore; // to avoid showinf on first time

int gameCount = 0;

bool waitForKeyRelease;

// checking for new levels

bool ball1Active = false;

bool ball2Active = false;

//int level = 1;

bool levelUpAvailable = false;

public:

int level = 1; // public for theming

enum GameState {

MENU,

PLAYING,

GAME\_OVER

};

GameState state;

Game() : paddle(screenWidth, screenHeight), ball(screenWidth, screenHeight), ball1(screenWidth, screenHeight), ball2(screenWidth, screenHeight), score(0), highScore(0), hasHighScore(false), isNewHigh(false), state(MENU) {

InitWindow(screenWidth, screenHeight, "2D Juggling Game for DSA Project");

SetTargetFPS(60);

waitForKeyRelease = false;

ball1.position.y = 100; // level 1 height

ball2.position.y = 50; // level 2 height

}

~Game() {

CloseWindow();

}

void Run() {

while (!WindowShouldClose()) {

float dt = GetFrameTime();

Update(dt);

Draw();

}

}

void Update(float dt) {

if (waitForKeyRelease) {

if (!IsKeyDown(KEY\_ENTER)) {

waitForKeyRelease = false; // key released

}

return; // skip rest of update

}

if (state == MENU) {

if (IsKeyPressed(KEY\_ENTER)) {

state = PLAYING;

//state = MENU;

score = 0;

ball.Reset(screenWidth, screenHeight);

ball.Reset(screenWidth, screenHeight);

ball1.Reset(screenWidth, screenHeight);

ball2.Reset(screenWidth, screenHeight);

ball1.position.y = 100;

ball2.position.y = 50;

ball1Active = false;

ball2Active = false;

level = 1;

}

}

else if (state == PLAYING) {

paddle.Update(dt, screenWidth);

ball.Update(dt);

if (!ball.IsOutOfBounds(screenHeight) && ball.CheckCollisionWithPaddle(paddle)) {

ball.velocity.y \*= -1;

score++;

if (score > highScore) {

highScore = score;

isNewHigh = true;

hasHighScore = true;

}

}

//new level balls

if (ball1Active) {

ball1.Update(dt);

if (!ball1.IsOutOfBounds(screenHeight) && ball1.CheckCollisionWithPaddle(paddle)) {

ball1.velocity.y \*= -1;

score++;

if (score > highScore) {

highScore = score;

isNewHigh = true;

hasHighScore = true;

}

}

}

if (ball2Active) {

ball2.Update(dt);

if (!ball2.IsOutOfBounds(screenHeight) && ball2.CheckCollisionWithPaddle(paddle)) {

ball2.velocity.y \*= -1;

score++;

if (score > highScore) {

highScore = score;

isNewHigh = true;

hasHighScore = true;

}

}

}

if (ball.IsOutOfBounds(screenHeight) || ball1.IsOutOfBounds(screenHeight) || ball2.IsOutOfBounds(screenHeight)) {

/\*state = MENU;\*/ // Go back to menu

if (ball.IsOutOfBounds(screenHeight) || ball1.IsOutOfBounds(screenHeight) || ball2.IsOutOfBounds(screenHeight)) {

state = GAME\_OVER;

level = 1;

}

gameCount++;

}

if (score % 5 == 0 && score > 0) {

if (score >= 5 && !ball1Active) {

levelUpAvailable = true;

}

else if (score >= 10 && !ball2Active) {

levelUpAvailable = true;

}

}

if (levelUpAvailable && IsKeyPressed(KEY\_SPACE)) {

if (!ball1Active) {

ball1Active = true;

level++;

levelUpAvailable = false;

}

else if (!ball2Active) {

ball2Active = true;

level++;

levelUpAvailable = false;

}

}

}

else if (state == GAME\_OVER && IsKeyPressed(KEY\_ENTER)) {

score = 0;

isNewHigh = false;

/\*HighShown = false;\*/ // reset highlight

ball.Reset(screenWidth, screenHeight);

state = MENU;

}

}

void Draw() {

BeginDrawing();

// background color

if (level == 1) {

ClearBackground(RAYWHITE);

}

else if (level == 2) {

ClearBackground(RAYWHITE);

}

else if (level == 3) {

ClearBackground(DARKGRAY);

}

if (state == MENU) {

DrawText("JUGGLING GAME", screenWidth / 2 - 240, screenHeight / 2 - 120, 60, DARKGRAY);

DrawText("Press [ENTER] to Start", screenWidth / 2 - 120, screenHeight / 2 + 30, 20, GRAY);

DrawText("UITS ID: 0432410005101088", screenWidth - 780, screenHeight - 100, 20, BLACK);

DrawText("Prototype for DSA Project", screenWidth - 780, screenHeight - 70, 20, BLACK);

DrawText("made by @b1tranger using Raylib", screenWidth - 780, screenHeight - 40, 20, GRAY);

if (gameCount > 0) {

DrawText(TextFormat("Current Record: %d", highScore), screenWidth / 2 - 130, screenHeight / 2 - 20, 30, GRAY);

}

}

else if (state == PLAYING) {

paddle.Draw(level);

ball.Draw(level);

if (ball1Active) ball1.Draw(level); // new balls

if (ball2Active) ball2.Draw(level);

// Draw current score

DrawText(TextFormat("Score: %d", score), 10, 10, 20, BLACK);

// level show

DrawText(TextFormat("Level: %d", level), 10, 70, 20, BLACK);

// level up prompt

if (levelUpAvailable) {

DrawText("Level Up Available! Press [SPACE]", screenWidth / 2 - 160, 20, 20, ORANGE);

}

if (hasHighScore && gameCount > 0) {

int highScoreFontSize = isNewHigh ? 30 : 20;

int PositionX = isNewHigh ? 300 : 10;

Color highScoreColor = isNewHigh ? RED : DARKGRAY;

DrawText(TextFormat("Highest Score: %d", highScore), PositionX, 40, highScoreFontSize, highScoreColor);

}

}

else if (state == GAME\_OVER) {

DrawText("GAME OVER", screenWidth / 2 - 100, screenHeight / 2 - 60, 40, RED);

DrawText(TextFormat("Score: %d", score), screenWidth / 2 - 60, screenHeight / 2, 30, BLACK);

if (isNewHigh) {

DrawText("New High Score!", screenWidth / 2 - 80, screenHeight / 2 + 40, 20, RED);

}

DrawText("Press [ENTER] to return to Menu", screenWidth / 2 - 160, screenHeight / 2 + 80, 20, DARKGRAY);

}

EndDrawing();

}

};

int main() {

Game game;

game.Run();

return 0;}