PESQUISA RAYLIB

Aluno: André Luiz Lima Rodrigues

Professor: Paulo Giovani de Faria Zeferino

Disciplina: Programação Orientada a Objetos

1. Introdução

O Raylib é uma biblioteca de código aberto e multiplataforma, projetado para facilitar o desenvolvimento de jogos 2D e 3D. Ela apresenta um conjunto de funcionalidades abrangentes por meio de uma API simples, tornando o Raylib uma excelente opção tanto para iniciantes quanto para desenvolvedores experientes. A biblioteca tem grande destaque pela sua performance, flexibilidade e por ser leve, o que a torna ideal para a criação de jogos de diversos gêneros e complexidades.

Seu principal uso é na desenvolvida e criação de jogos, sendo amplamente adotada por desenvolvedores amadores que desejam criar jogos mais simplificados para ganhar experiencias. No entanto, seu uso não se restringe somente aos iniciantes, muitos veteranos também utilizam o raylib devido à sua facilidade de suso, que permite a prototipagem rápida, experimentação de mecânicas e ideias de jogo.

Além disso, muitos desenvolvedores independentes escolhem o Raylib por sua curva de aprendizagem suave e pela possibilidade de portar jogos para diversas plataformas. A biblioteca é usada com frequência graças a cursos e tutoriais disponibilizados pela comunidade na internet disseminando ainda mais o Raylib.

2. Desenvolvedores

A biblioteca Raylib foi criada por Ramon Santamaria , um desenvolvedor independente e alguns contribuintes. A primeira versão do raylib foi disponibilizada para download em 18 de novembro de 2013 escrita em linguagem C e adaptada para várias outras linguagens de programação.

Sua inspiração vem da Borland BGI Graphics e do XNA framework, sendo especialmente indicada para prototipagem, ferramentas, aplicações gráficas e educação. A biblioteca é bem avaliada tanto pelo público quanto por profissionais, e seu suporte a diversas linguagens e sistemas operacionais resultou em prêmios de qualidade, como o Google Open Source Peer Bonus em 2019 e 2021, além de ser reconhecida pela Epic Games com o Epic MegaGrants Recipient em 2020.

3. Exemplos de Jogos e Códigos

A Raylib possui uma vasta coleção de exemplos que demonstram como utilizar as diversas funcionalidades da biblioteca. Esses exemplos cobram desde conceitos básicos, como criação de janelas e desenho de formas, até tópicos avançados, como física, animações e carregamento de modelos 3D.

Segue um código de jogo da cobrinha feiro por Ian Eito, Albert Marcos e Ramon Santamaria, obtido da página do GitHub do Raylib.

```
/*******************************************************************************
*
*   raylib - classic game: snake
*
*   Sample game developed by Ian Eito, Albert Martos and Ramon Santamaria
*
*   This game has been created using raylib v1.3 (www.raylib.com)
*   raylib is licensed under an unmodified zlib/libpng license (View raylib.h
for details)
*
*   Copyright (c) 2015 Ramon Santamaria (@raysan5)
*
********************************************************************************/

#include "raylib.h"

#if defined(PLATFORM_WEB)
    #include <emscripten/emscripten.h>
#endif

//----------------------------------------------------------------------------------
// Some Defines
//----------------------------------------------------------------------------------
#define SNAKE_LENGTH   256
#define SQUARE_SIZE     31
```

```c
//----------------------------------------------------------------------------------
// Types and Structures Definition
//----------------------------------------------------------------------------------
typedef struct Snake {
    Vector2 position;
    Vector2 size;
    Vector2 speed;
    Color color;
} Snake;

typedef struct Food {
    Vector2 position;
    Vector2 size;
    bool active;
    Color color;
} Food;

//----------------------------------------------------------------------------------
// Global Variables Declaration
//----------------------------------------------------------------------------------
static const int screenWidth = 800;
static const int screenHeight = 450;

static int framesCounter = 0;
static bool gameOver = false;
static bool pause = false;

static Food fruit = { 0 };
static Snake snake[SNAKE_LENGTH] = { 0 };
static Vector2 snakePosition[SNAKE_LENGTH] = { 0 };
static bool allowMove = false;
static Vector2 offset = { 0 };
static int counterTail = 0;

//----------------------------------------------------------------------------------
// Module Functions Declaration (local)
//----------------------------------------------------------------------------------
static void InitGame(void);         // Initialize game
static void UpdateGame(void);       // Update game (one frame)
static void DrawGame(void);         // Draw game (one frame)
static void UnloadGame(void);       // Unload game
static void UpdateDrawFrame(void);  // Update and Draw (one frame)

//----------------------------------------------------------------------------------
// Program main entry point
```

```c
//------------------------------------------------------------------------------------
int main(void)
{
    // Initialization (Note windowTitle is unused on Android)
    //---------------------------------------------------------
    InitWindow(screenWidth, screenHeight, "classic game: snake");

    InitGame();

#if defined(PLATFORM_WEB)
    emscripten_set_main_loop(UpdateDrawFrame, 60, 1);
#else
    SetTargetFPS(60);
    //--------------------------------------------------------------------------------

    // Main game loop
    while (!WindowShouldClose())    // Detect window close button or ESC key
    {
        // Update and Draw
        //----------------------------------------------------------------------------
        UpdateDrawFrame();
        //----------------------------------------------------------------------------
    }
#endif
    // De-Initialization
    //--------------------------------------------------------------------------------
    UnloadGame();         // Unload loaded data (textures, sounds, models...)

    CloseWindow();        // Close window and OpenGL context
    //--------------------------------------------------------------------------------

    return 0;
}

//------------------------------------------------------------------------------------
// Module Functions Definitions (local)
//------------------------------------------------------------------------------------

// Initialize game variables
void InitGame(void)
{
    framesCounter = 0;
    gameOver = false;
    pause = false;

    counterTail = 1;
```

```
    allowMove = false;

    offset.x = screenWidth%SQUARE_SIZE;
    offset.y = screenHeight%SQUARE_SIZE;

    for (int i = 0; i < SNAKE_LENGTH; i++)
    {
        snake[i].position = (Vector2){ offset.x/2, offset.y/2 };
        snake[i].size = (Vector2){ SQUARE_SIZE, SQUARE_SIZE };
        snake[i].speed = (Vector2){ SQUARE_SIZE, 0 };

        if (i == 0) snake[i].color = DARKBLUE;
        else snake[i].color = BLUE;
    }

    for (int i = 0; i < SNAKE_LENGTH; i++)
    {
        snakePosition[i] = (Vector2){ 0.0f, 0.0f };
    }

    fruit.size = (Vector2){ SQUARE_SIZE, SQUARE_SIZE };
    fruit.color = SKYBLUE;
    fruit.active = false;
}

// Update game (one frame)
void UpdateGame(void)
{
    if (!gameOver)
    {
        if (IsKeyPressed('P')) pause = !pause;

        if (!pause)
        {
            // Player control
            if (IsKeyPressed(KEY_RIGHT) && (snake[0].speed.x == 0) &&
allowMove)
            {
                snake[0].speed = (Vector2){ SQUARE_SIZE, 0 };
                allowMove = false;
            }
            if (IsKeyPressed(KEY_LEFT) && (snake[0].speed.x == 0) &&
allowMove)
            {
                snake[0].speed = (Vector2){ -SQUARE_SIZE, 0 };
                allowMove = false;
```

```
            }
            if (IsKeyPressed(KEY_UP) && (snake[0].speed.y == 0) && allowMove)
            {
                snake[0].speed = (Vector2){ 0, -SQUARE_SIZE };
                allowMove = false;
            }
            if (IsKeyPressed(KEY_DOWN) && (snake[0].speed.y == 0) &&
allowMove)
            {
                snake[0].speed = (Vector2){ 0, SQUARE_SIZE };
                allowMove = false;
            }

            // Snake movement
            for (int i = 0; i < counterTail; i++) snakePosition[i] =
snake[i].position;

            if ((framesCounter%5) == 0)
            {
                for (int i = 0; i < counterTail; i++)
                {
                    if (i == 0)
                    {
                        snake[0].position.x += snake[0].speed.x;
                        snake[0].position.y += snake[0].speed.y;
                        allowMove = true;
                    }
                    else snake[i].position = snakePosition[i-1];
                }
            }

            // Wall behaviour
            if (((snake[0].position.x) > (screenWidth - offset.x)) ||
                ((snake[0].position.y) > (screenHeight - offset.y)) ||
                (snake[0].position.x < 0) || (snake[0].position.y < 0))
            {
                gameOver = true;
            }

            // Collision with yourself
            for (int i = 1; i < counterTail; i++)
            {
                if ((snake[0].position.x == snake[i].position.x) &&
(snake[0].position.y == snake[i].position.y)) gameOver = true;
            }

            // Fruit position calculation
```

```
            if (!fruit.active)
            {
                fruit.active = true;
                fruit.position = (Vector2){ GetRandomValue(0,
(screenWidth/SQUARE_SIZE) - 1)*SQUARE_SIZE + offset.x/2, GetRandomValue(0,
(screenHeight/SQUARE_SIZE) - 1)*SQUARE_SIZE + offset.y/2 };

                for (int i = 0; i < counterTail; i++)
                {
                    while ((fruit.position.x == snake[i].position.x) &&
(fruit.position.y == snake[i].position.y))
                    {
                        fruit.position = (Vector2){ GetRandomValue(0,
(screenWidth/SQUARE_SIZE) - 1)*SQUARE_SIZE + offset.x/2, GetRandomValue(0,
(screenHeight/SQUARE_SIZE) - 1)*SQUARE_SIZE + offset.y/2 };
                        i = 0;
                    }
                }
            }

            // Collision
            if ((snake[0].position.x < (fruit.position.x + fruit.size.x) &&
(snake[0].position.x + snake[0].size.x) > fruit.position.x) &&
                (snake[0].position.y < (fruit.position.y + fruit.size.y) &&
(snake[0].position.y + snake[0].size.y) > fruit.position.y))
            {
                snake[counterTail].position = snakePosition[counterTail - 1];
                counterTail += 1;
                fruit.active = false;
            }

            framesCounter++;
        }
    }
    else
    {
        if (IsKeyPressed(KEY_ENTER))
        {
            InitGame();
            gameOver = false;
        }
    }
}

// Draw game (one frame)
void DrawGame(void)
{
```

```
    BeginDrawing();

        ClearBackground(RAYWHITE);

        if (!gameOver)
        {
            // Draw grid lines
            for (int i = 0; i < screenWidth/SQUARE_SIZE + 1; i++)
            {
                DrawLineV((Vector2){SQUARE_SIZE*i + offset.x/2, offset.y/2},
(Vector2){SQUARE_SIZE*i + offset.x/2, screenHeight - offset.y/2}, LIGHTGRAY);
            }

            for (int i = 0; i < screenHeight/SQUARE_SIZE + 1; i++)
            {
                DrawLineV((Vector2){offset.x/2, SQUARE_SIZE*i + offset.y/2},
(Vector2){screenWidth - offset.x/2, SQUARE_SIZE*i + offset.y/2}, LIGHTGRAY);
            }

            // Draw snake
            for (int i = 0; i < counterTail; i++)
DrawRectangleV(snake[i].position, snake[i].size, snake[i].color);

            // Draw fruit to pick
            DrawRectangleV(fruit.position, fruit.size, fruit.color);

            if (pause) DrawText("GAME PAUSED", screenWidth/2 -
MeasureText("GAME PAUSED", 40)/2, screenHeight/2 - 40, 40, GRAY);
        }
        else DrawText("PRESS [ENTER] TO PLAY AGAIN", GetScreenWidth()/2 -
MeasureText("PRESS [ENTER] TO PLAY AGAIN", 20)/2, GetScreenHeight()/2 - 50,
20, GRAY);

    EndDrawing();
}

// Unload game variables
void UnloadGame(void)
{
    // TODO: Unload all dynamic loaded data (textures, sounds, models...)
}

// Update and Draw (one frame)
void UpdateDrawFrame(void)
{
    UpdateGame();
```

```
    DrawGame();
}
```

Algoritmo 1 – Jogo da Cobrinha

A seguir, apresentamos uma imagem do jogo, obtida diretamente da página oficial do Raylib, que ilustra a interface e a dinâmica do jogo desenvolvido.

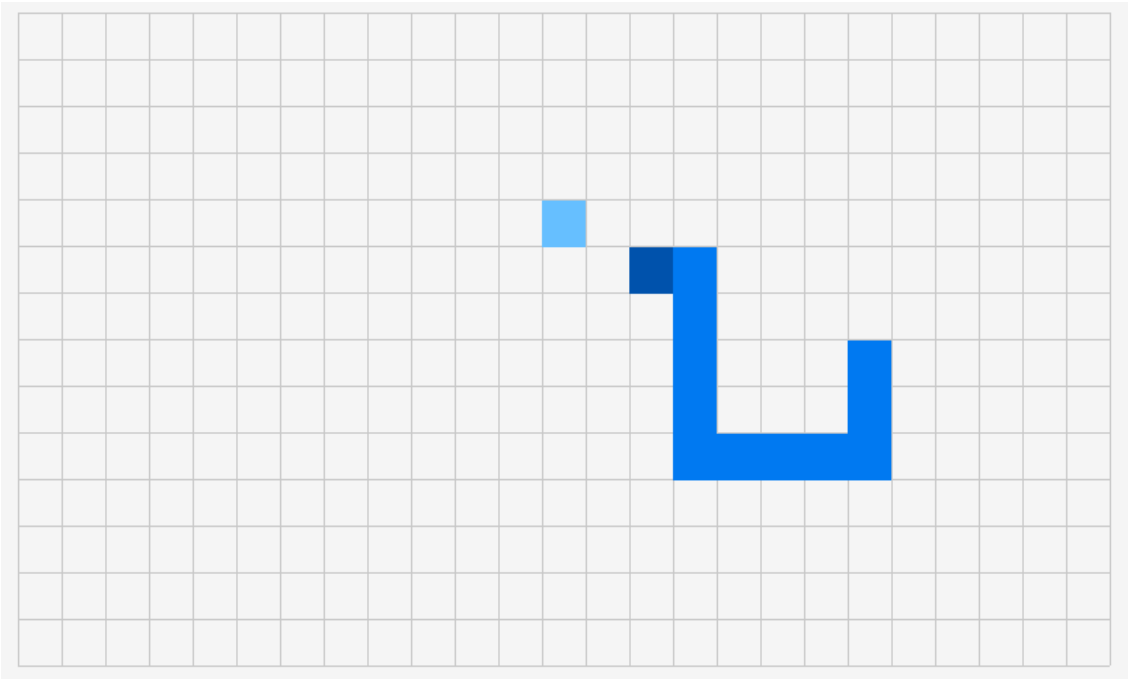

Figura 1 – Imagem do jogo da cobrinha.

O jogo da cobrinha, desenvolvido por Ian Eito, Albert Marcos e Ramon Santamaria, exemplifica bem as capacidades da biblioteca. O código apresentado ilustra como, através de uma estrutura clara e organizada, é possível criar um jogo envolvente e divertido. A capacidade de manipular gráficos, detectar colisões e gerenciar estados do jogo são apenas algumas das funcionalidades que tornam o Raylib uma ferramenta atraente para quem deseja se aventurar no desenvolvimento de jogos.

4. Conclusão

A utilização da biblioteca Raylib se mostrou uma excelente escolha para o desenvolvimento de jogos, tanto para iniciantes quanto para desenvolvedores experientes. Sua simplicidade na API, combinada com um conjunto robusto de funcionalidades, permite uma rápida prototipação e experimentação de mecânicas de jogo. Além disso, o fato de ser multiplataforma e suportar várias

linguagens de programação amplia suas possibilidades de uso e adoção por parte da comunidade de desenvolvedores.

5. Referências Bibliográficas

- RAYLIB. Raylib. Disponível em: https://www.raylib.com/. Acesso em: 06 out. 2024.
- RAYAN5. Raylib games: Snake. Disponível em: https://github.com/raysan5/raylib-games/blob/master/classics/src/snake.c. Acesso em: 06 out. 2024.
- GAMES. Raylib: Snake game tutorial. Disponível em: https://youtu.be/TgCwzMemb_M?si=bKoJKT4xx-6kmvm3. Acesso em: 06 out. 2024.
- RAYLIB. Raylib games. Disponível em: https://www.raylib.com/games.html. Acesso em: 06 out. 2024.