Pesquisa Raylib – Vyttor Ramos

Raylib é uma biblioteca de código aberto voltada ao desenvolvimento de software, focada principalmente na criação de jogos e aplicações gráficas. Criada em 2013 por Ramon Santamaria, a Raylib visa facilitar o aprendizado de programação de jogos, com um foco em simplicidade e praticidade. Sua interface intuitiva torna o desenvolvimento acessível tanto para programadores iniciantes quanto para desenvolvedores mais experientes. Escrita em C, ela utiliza OpenGL para renderização gráfica, e é projetada para criar aplicações em 2D e 3D.

A Raylib é amplamente utilizada em diversas áreas, sendo um recurso valioso para o desenvolvimento de jogos independentes, aplicações gráficas, e até ferramentas de ensino. Ela é compatível com várias plataformas, incluindo sistemas desktop , dispositivos móveis, e sistemas embarcados. Além disso, com suporte a HTML5, é possível compilar projetos Raylib para rodar diretamente em navegadores. Seu uso se estende para áreas educacionais, onde é uma ferramenta eficiente no ensino de programação gráfica e jogos. Graças à sua documentação enxuta e exemplos práticos, é ideal para quem quer aprender de forma prática, com exemplos reais de código.

Ramon Santamaria, desenvolvedor espanhol, começou a desenvolver a Raylib em 2013 como parte de um curso de programação de jogos. Seu objetivo era fornecer uma ferramenta simples para aqueles que estavam começando na área, permitindo que os alunos se concentrassem em criar jogos, sem a complexidade excessiva de outras bibliotecas gráficas.

Exemplo de código:

```
/*******************************************************************************************
*
*   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

//----------------------------------------------------------------------------------
// 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
//----------------------------------------------------------------------------------
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();
}
```

O exemplo anterior se refere ao clássico jogo snake, que tem como objetivo atingir uma cobra de maior dimensão possível. O jogo chega ao fim quando a cabeça da cobra encostar em qualquer parte de seu corpo ou se o jogador decidir encerrá-lo.

Existem alguns outros jogos feitos utilizando raylib, como: Drift, jogo de corrida em que os carros derrapam nas curvas, Floppy que é umarecriação do “Flappy Bird”, onde os jogadores controlam um pássaro e precisam desviar de obstáculos, entre outros.

Referencias bibliográficas

https://www.raylib.com/

https://en.wikipedia.org/wiki/Raylib

https://github.com/raysan5/raylib-games/blob/master/classics/src/snake.c