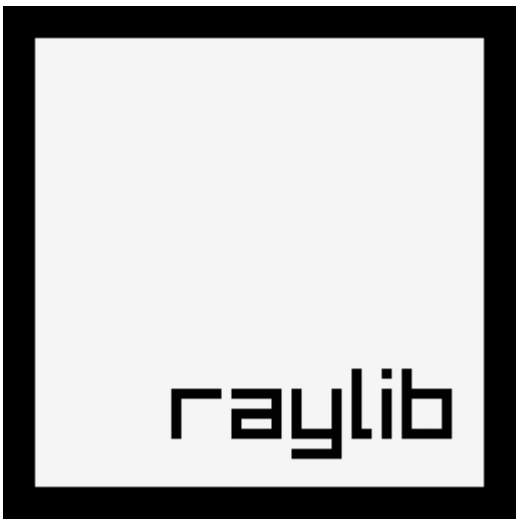
Pesquisa sobre a Biblioteca Raylib

Vinícius Silva de Sampaio

Introdução

O Raylib é uma biblioteca de desenvolvimento de jogos em C, super simples de se usar. Foi criada pensando justamente em ajudar quem está começando a entender os conceitos de programação gráfica e desenvolvimento de jogos. O Raylib é muito útil tanto para quem está iniciando quanto para quem já tem mais experiência. Com exemplos bem práticos e uma documentação objetiva, ela facilita bastante na hora de desenvolver jogos 2D, 3D e até outras aplicações multimídia.



Onde é utilizado e suas características:

Desde seu lançamento, o Raylib tem ficado bem popular, principalmente entre quem desenvolve jogos indie e educadores que ensinam programação de jogos. A ideia principal do Raylib é tornar o aprendizado acessível para qualquer pessoa, independente do nível de experiência. Ela segue o estilo "faça você mesmo", sem depender de ferramentas gráficas sofisticadas ou interfaces complexas, te incentivando a codar de verdade

.

Características Principais

Simplicidade: A API do Raylib é feita pra ser bem fácil de entender. Isso permite que a gente foque na parte divertida (a lógica do jogo) em vez de se perder com detalhes técnicos mais chatos.

Ferramentas simples e código sustentável: O Raylib prioriza a simplicidade. O objetivo é que o desenvolvimento seja fluido, com um código limpo que vai funcionar a longo prazo.

Documentação minimalista: Ao invés de ter uma documentação cheia de tutoriais longos, o Raylib tem uma folha de dicas com as funcionalidades mais importantes e vários exemplos práticos. Assim, a ideia é que você aprenda diretamente mexendo no código e entendendo como ele funciona na prática.

Multiplataforma: É possível utilizar o Raylib em vários sistemas, como Windows, macOS, Linux, FreeBSD, Raspberry Pi, Android e até HTML5. Ou seja, É possível desenvolver em quase qualquer ambiente.

Modularidade: Ela é bem modular, o que significa que você pode usar alguns dos seus módulos de forma isolada, sem precisar da biblioteca completa.

Desempenho: Mesmo sendo fácil de usar, o Raylib ainda tem um bom desempenho, especialmente pra jogos 2D e 3D, oferecendo gráficos de qualidade.

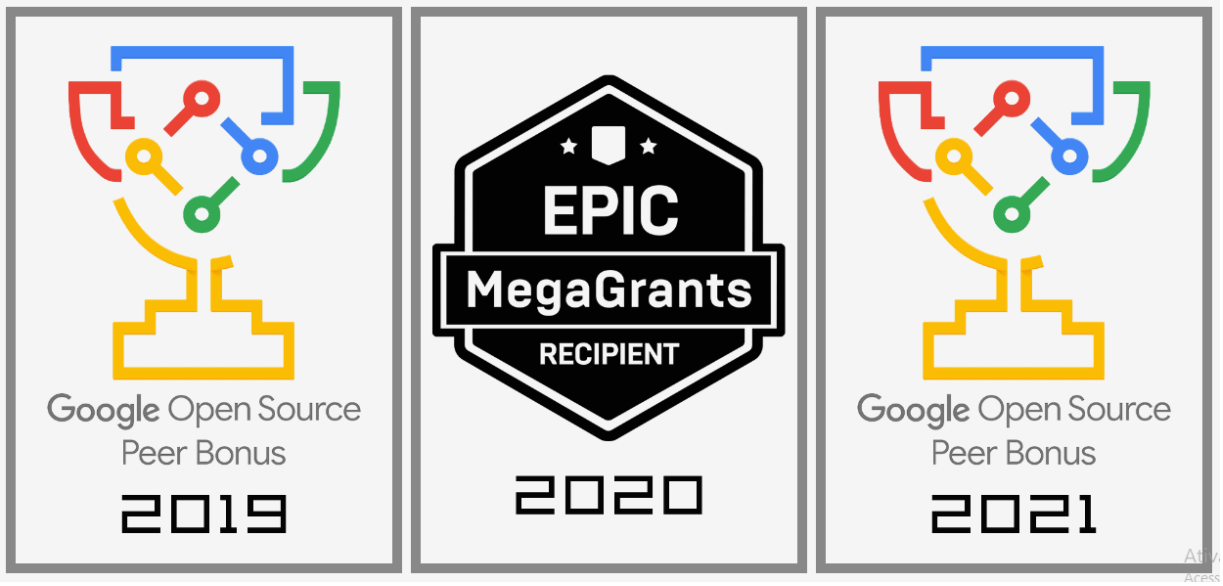
Prêmios e Reconhecimentos

O Raylib já ganhou alguns prêmios bem legais por ser uma ferramenta open-source e por ajudar no ensino de programação de jogos:

Google Open Source Peer Bonus – 2019

Epic MegaGrant Recipient – Outono de 2020

Google Open Source Peer Bonus – 2021



Quem desenvolveu

Ramon Santamaría, um desenvolvedor de software da Espanha que se destacou na comunidade de código aberto com o projeto Raylib. Ele começou sua jornada no mundo dos videogames em 2006 e, alguns anos depois, começou a ensinar desenvolvimento de jogos para jovens com perfis artísticos. O desafio principal era ensinar programação para pessoas que nunca haviam escrito uma linha de código.

Vendo as limitações das ferramentas disponíveis na época, Ramon decidiu criar sua própria biblioteca. Ele queria algo que fosse acelerado por hardware, com nomes de funções claros, bem organizada e estruturada, e que usasse codificação simples. A ideia principal era que essa biblioteca ajudasse os alunos a aprenderem programação de jogos de forma acessível.

O Raylib começou como um projeto de fim de semana, mas, após três meses de trabalho intenso, a versão 1.0 foi lançada em novembro de 2013. A última atualização do Raylib foi há um ano, mas o Ramon ainda tem vários planos para o futuro. Um deles é o Raylib Project Creator, uma ferramenta que criará automaticamente a estrutura completa de projetos, com os sistemas de build já configurados. Tudo o que será preciso é de um arquivo de código e alguns parâmetros. O Raylib continua sendo bem utilizada e segue como uma ótima opção pra quem quer desenvolver jogos de forma simples e prática.

Exemplo de Códigos:

Os exemplos de raylib são classificados por complexidade com estrelas. Uma estrela para os exemplos básicos e quatro estrelas para os mais complexos.

Exemplo 1 (uma estrela): Controla do cubo pra cima e pra baixo com o scroll do mouse:

```
/*******************************************************************************************
**
*
*   raylib [core] examples - Mouse wheel input
*
*   Example originally created with raylib 1.1, last time updated with raylib 1.3
*
*   Example licensed under an unmodified zlib/libpng license, which is an OSI-certified,
*   BSD-like license that allows static linking with closed source software
*
*   Copyright (c) 2014-2024 Ramon Santamaria (@raysan5)
*
********************************************************************************************
**/

#include "raylib.h"

//------------------------------------------------------------------------------------
// Program main entry point
//------------------------------------------------------------------------------------
int main(void)
{
    // Initialization
    //--------------------------------------------------------------------------------------
--
    const int screenWidth = 800;
    const int screenHeight = 450;

    InitWindow(screenWidth, screenHeight, "raylib [core] example - input mouse wheel");

    int boxPositionY = screenHeight/2 - 40;
    int scrollSpeed = 4;            // Scrolling speed in pixels

    SetTargetFPS(60);               // Set our game to run at 60 frames-per-second
    //--------------------------------------------------------------------------------------
--

    // Main game loop
    while (!WindowShouldClose())    // Detect window close button or ESC key
    {
        // Update
        //----------------------------------------------------------------------------------
--
        boxPositionY -= (int)(GetMouseWheelMove()*scrollSpeed);
        //----------------------------------------------------------------------------------
--

        // Draw
        //----------------------------------------------------------------------------------
--
        BeginDrawing();

            ClearBackground(RAYWHITE);

            DrawRectangle(screenWidth/2 - 40, boxPositionY, 80, 80, MAROON);
```

```
            DrawText("Use mouse wheel to move the cube up and down!", 10, 10, 20, GRAY);
            DrawText(TextFormat("Box position Y: %03i", boxPositionY), 10, 40, 20,
LIGHTGRAY);

        EndDrawing();
        //----------------------------------------------------------------------------------
--
    }

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

    return 0;
}
```

Exemplo 2 (duas estrelas): Esse código faz com que dois olhos sigam o ponteiro do mouse, com as pupilas se movendo dentro da parte branca do olho (esclera):

```
/*******************************************************************************************
**
*
*   raylib [shapes] example - following eyes
*
*   Example originally created with raylib 2.5, last time updated with raylib 2.5
*
*   Example licensed under an unmodified zlib/libpng license, which is an OSI-certified,
*   BSD-like license that allows static linking with closed source software
*
*   Copyright (c) 2013-2024 Ramon Santamaria (@raysan5)
*
********************************************************************************************
**/

#include "raylib.h"

#include <math.h>       // Required for: atan2f()

//----------------------------------------------------------------------------------
// Program main entry point
//----------------------------------------------------------------------------------
int main(void)
{
    // Initialization
    //--------------------------------------------------------------------------------------
--
    const int screenWidth = 800;
    const int screenHeight = 450;

    InitWindow(screenWidth, screenHeight, "raylib [shapes] example - following eyes");

    Vector2 scleraLeftPosition = { GetScreenWidth()/2.0f - 100.0f, GetScreenHeight()/2.0f
};
    Vector2 scleraRightPosition = { GetScreenWidth()/2.0f + 100.0f, GetScreenHeight()/2.0f
};
    float scleraRadius = 80;

    Vector2 irisLeftPosition = { GetScreenWidth()/2.0f - 100.0f, GetScreenHeight()/2.0f };
    Vector2 irisRightPosition = { GetScreenWidth()/2.0f + 100.0f, GetScreenHeight()/2.0f
};
    float irisRadius = 24;
```

```
    float angle = 0.0f;
    float dx = 0.0f, dy = 0.0f, dxx = 0.0f, dyy = 0.0f;

    SetTargetFPS(60);               // Set our game to run at 60 frames-per-second
    //--------------------------------------------------------------------------------------
--

    // Main game loop
    while (!WindowShouldClose())    // Detect window close button or ESC key
    {
        // Update
        //----------------------------------------------------------------------------------
--
        irisLeftPosition = GetMousePosition();
        irisRightPosition = GetMousePosition();

        // Check not inside the left eye sclera
        if (!CheckCollisionPointCircle(irisLeftPosition, scleraLeftPosition, scleraRadius
- irisRadius))
        {
            dx = irisLeftPosition.x - scleraLeftPosition.x;
            dy = irisLeftPosition.y - scleraLeftPosition.y;

            angle = atan2f(dy, dx);

            dxx = (scleraRadius - irisRadius)*cosf(angle);
            dyy = (scleraRadius - irisRadius)*sinf(angle);

            irisLeftPosition.x = scleraLeftPosition.x + dxx;
            irisLeftPosition.y = scleraLeftPosition.y + dyy;
        }

        // Check not inside the right eye sclera
        if (!CheckCollisionPointCircle(irisRightPosition, scleraRightPosition,
scleraRadius - irisRadius))
        {
            dx = irisRightPosition.x - scleraRightPosition.x;
            dy = irisRightPosition.y - scleraRightPosition.y;

            angle = atan2f(dy, dx);

            dxx = (scleraRadius - irisRadius)*cosf(angle);
            dyy = (scleraRadius - irisRadius)*sinf(angle);

            irisRightPosition.x = scleraRightPosition.x + dxx;
            irisRightPosition.y = scleraRightPosition.y + dyy;
        }
        //----------------------------------------------------------------------------------
--

        // Draw
        //----------------------------------------------------------------------------------
--
        BeginDrawing();

            ClearBackground(RAYWHITE);

            DrawCircleV(scleraLeftPosition, scleraRadius, LIGHTGRAY);
            DrawCircleV(irisLeftPosition, irisRadius, BROWN);
            DrawCircleV(irisLeftPosition, 10, BLACK);

            DrawCircleV(scleraRightPosition, scleraRadius, LIGHTGRAY);
            DrawCircleV(irisRightPosition, irisRadius, DARKGREEN);
            DrawCircleV(irisRightPosition, 10, BLACK);

            DrawFPS(10, 10);

        EndDrawing();
        //----------------------------------------------------------------------------------
--
```

```
    }

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

    return 0;
}
```

Exemplo 3 (três estrelas): Esse código permite ajustar a densidade da névoa em um ambiente com modelos 3D usando as setas para cima e para baixo; para cima aumenta a névoa, e para baixo diminui:

```
/*******************************************************************************************
**
*
*   raylib [shaders] example - fog
*
*   NOTE: This example requires raylib OpenGL 3.3 or ES2 versions for shaders support,
*         OpenGL 1.1 does not support shaders, recompile raylib to OpenGL 3.3 version.
*
*   NOTE: Shaders used in this example are #version 330 (OpenGL 3.3).
*
*   Example originally created with raylib 2.5, last time updated with raylib 3.7
*
*   Example contributed by Chris Camacho (@chriscamacho) and reviewed by Ramon Santamaria
(@raysan5)
*
*   Example licensed under an unmodified zlib/libpng license, which is an OSI-certified,
*   BSD-like license that allows static linking with closed source software
*
*   Copyright (c) 2019-2024 Chris Camacho (@chriscamacho) and Ramon Santamaria (@raysan5)
*
********************************************************************************************
**/

#include "raylib.h"

#include "raymath.h"

#define RLIGHTS_IMPLEMENTATION
#include "rlights.h"

#if defined(PLATFORM_DESKTOP)
    #define GLSL_VERSION            330
#else   // PLATFORM_ANDROID, PLATFORM_WEB
    #define GLSL_VERSION            100
#endif

//------------------------------------------------------------------------------------
// Program main entry point
//------------------------------------------------------------------------------------
int main(void)
{
    // Initialization
    //--------------------------------------------------------------------------------------
    const int screenWidth = 800;
    const int screenHeight = 450;

    SetConfigFlags(FLAG_MSAA_4X_HINT);  // Enable Multi Sampling Anti Aliasing 4x (if
available)
```

```
    InitWindow(screenWidth, screenHeight, "raylib [shaders] example - fog");

    // Define the camera to look into our 3d world
    Camera camera = { 0 };
    camera.position = (Vector3){ 2.0f, 2.0f, 6.0f };    // Camera position
    camera.target = (Vector3){ 0.0f, 0.5f, 0.0f };      // Camera looking at point
    camera.up = (Vector3){ 0.0f, 1.0f, 0.0f };          // Camera up vector (rotation
towards target)
    camera.fovy = 45.0f;                                // Camera field-of-view Y
    camera.projection = CAMERA_PERSPECTIVE;             // Camera projection type

    // Load models and texture
    Model modelA = LoadModelFromMesh(GenMeshTorus(0.4f, 1.0f, 16, 32));
    Model modelB = LoadModelFromMesh(GenMeshCube(1.0f, 1.0f, 1.0f));
    Model modelC = LoadModelFromMesh(GenMeshSphere(0.5f, 32, 32));
    Texture texture = LoadTexture("resources/texel_checker.png");

    // Assign texture to default model material
    modelA.materials[0].maps[MATERIAL_MAP_DIFFUSE].texture = texture;
    modelB.materials[0].maps[MATERIAL_MAP_DIFFUSE].texture = texture;
    modelC.materials[0].maps[MATERIAL_MAP_DIFFUSE].texture = texture;

    // Load shader and set up some uniforms
    Shader shader = LoadShader(TextFormat("resources/shaders/glsl%i/lighting.vs",
GLSL_VERSION),
                               TextFormat("resources/shaders/glsl%i/fog.fs",
GLSL_VERSION));
    shader.locs[SHADER_LOC_MATRIX_MODEL] = GetShaderLocation(shader, "matModel");
    shader.locs[SHADER_LOC_VECTOR_VIEW] = GetShaderLocation(shader, "viewPos");

    // Ambient light level
    int ambientLoc = GetShaderLocation(shader, "ambient");
    SetShaderValue(shader, ambientLoc, (float[4]){ 0.2f, 0.2f, 0.2f, 1.0f },
SHADER_UNIFORM_VEC4);

    float fogDensity = 0.15f;
    int fogDensityLoc = GetShaderLocation(shader, "fogDensity");
    SetShaderValue(shader, fogDensityLoc, &fogDensity, SHADER_UNIFORM_FLOAT);

    // NOTE: All models share the same shader
    modelA.materials[0].shader = shader;
    modelB.materials[0].shader = shader;
    modelC.materials[0].shader = shader;

    // Using just 1 point lights
    CreateLight(LIGHT_POINT, (Vector3){ 0, 2, 6 }, Vector3Zero(), WHITE, shader);

    SetTargetFPS(60);                   // Set our game to run at 60 frames-per-second
    //--------------------------------------------------------------------------------------

    // Main game loop
    while (!WindowShouldClose())        // Detect window close button or ESC key
    {
        // Update
        //----------------------------------------------------------------------------------
        UpdateCamera(&camera, CAMERA_ORBITAL);

        if (IsKeyDown(KEY_UP))
        {
            fogDensity += 0.001f;
            if (fogDensity > 1.0f) fogDensity = 1.0f;
        }

        if (IsKeyDown(KEY_DOWN))
        {
            fogDensity -= 0.001f;
            if (fogDensity < 0.0f) fogDensity = 0.0f;
        }
```

```
        SetShaderValue(shader, fogDensityLoc, &fogDensity, SHADER_UNIFORM_FLOAT);

        // Rotate the torus
        modelA.transform = MatrixMultiply(modelA.transform, MatrixRotateX(-0.025f));
        modelA.transform = MatrixMultiply(modelA.transform, MatrixRotateZ(0.012f));

        // Update the light shader with the camera view position
        SetShaderValue(shader, shader.locs[SHADER_LOC_VECTOR_VIEW], &camera.position.x,
SHADER_UNIFORM_VEC3);
        //----------------------------------------------------------------------------------
--

        // Draw
        //----------------------------------------------------------------------------------
--
        BeginDrawing();

            ClearBackground(GRAY);

            BeginMode3D(camera);

                // Draw the three models
                DrawModel(modelA, Vector3Zero(), 1.0f, WHITE);
                DrawModel(modelB, (Vector3){ -2.6f, 0, 0 }, 1.0f, WHITE);
                DrawModel(modelC, (Vector3){ 2.6f, 0, 0 }, 1.0f, WHITE);

                for (int i = -20; i < 20; i += 2) DrawModel(modelA,(Vector3){ (float)i, 0,
2 }, 1.0f, WHITE);

            EndMode3D();

            DrawText(TextFormat("Use KEY_UP/KEY_DOWN to change fog density [%.2f]",
fogDensity), 10, 10, 20, RAYWHITE);

        EndDrawing();
        //----------------------------------------------------------------------------------
--
    }

    // De-Initialization
    //--------------------------------------------------------------------------------------
--
    UnloadModel(modelA);        // Unload the model A
    UnloadModel(modelB);        // Unload the model B
    UnloadModel(modelC);        // Unload the model C
    UnloadTexture(texture);     // Unload the texture
    UnloadShader(shader);       // Unload shader

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

    return 0;
}
```

Exemplo 4 (quatro estrelas): O código divide a tela entre 2 jogadores em um plano 2D, onde o da direita é controlado pelas teclas W, A, S, D e o da esquerda pelas setas do teclado.

```
/*******************************************************************************************
**
*
*   raylib [core] example - 2d camera split screen
*
```

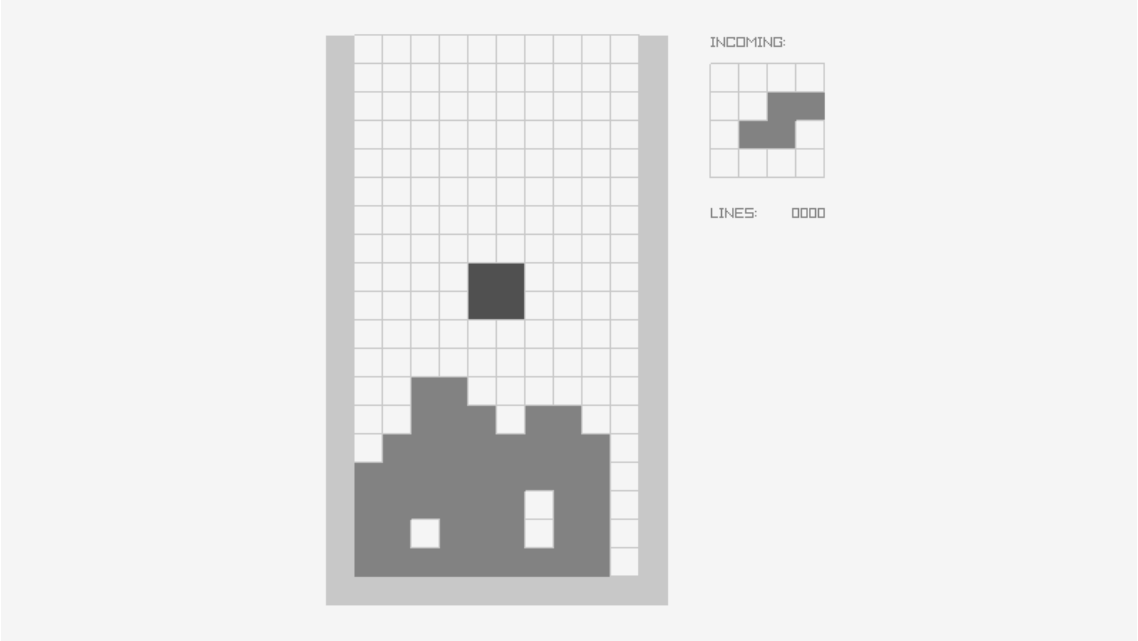
```
*   Addapted from the core_3d_camera_split_screen example:
*
https://github.com/raysan5/raylib/blob/master/examples/core/core_3d_camera_split_screen.c
*
*   Example originally created with raylib 4.5, last time updated with raylib 4.5
*
*   Example contributed by Gabriel dos Santos Sanches (@gabrielssanches) and reviewed by
Ramon Santamaria (@raysan5)
*
*   Example licensed under an unmodified zlib/libpng license, which is an OSI-certified,
*   BSD-like license that allows static linking with closed source software
*
*   Copyright (c) 2023 Gabriel dos Santos Sanches (@gabrielssanches)
*
********************************************************************************************
**/

#include "raylib.h"

#define PLAYER_SIZE 40

//----------------------------------------------------------------------------------
// Program main entry point
//----------------------------------------------------------------------------------
int main(void)
{
    // Initialization
    //--------------------------------------------------------------------------------------
--
    const int screenWidth = 800;
    const int screenHeight = 440;

    InitWindow(screenWidth, screenHeight, "raylib [core] example - 2d camera split
screen");

    Rectangle player1 = { 200, 200, PLAYER_SIZE, PLAYER_SIZE };
    Rectangle player2 = { 250, 200, PLAYER_SIZE, PLAYER_SIZE };

    Camera2D camera1 = { 0 };
    camera1.target = (Vector2){ player1.x, player1.y };
    camera1.offset = (Vector2){ 200.0f, 200.0f };
    camera1.rotation = 0.0f;
    camera1.zoom = 1.0f;

    Camera2D camera2 = { 0 };
    camera2.target = (Vector2){ player2.x, player2.y };
    camera2.offset = (Vector2){ 200.0f, 200.0f };
    camera2.rotation = 0.0f;
    camera2.zoom = 1.0f;

    RenderTexture screenCamera1 = LoadRenderTexture(screenWidth/2, screenHeight);
    RenderTexture screenCamera2 = LoadRenderTexture(screenWidth/2, screenHeight);

    // Build a flipped rectangle the size of the split view to use for drawing later
    Rectangle splitScreenRect = { 0.0f, 0.0f, (float)screenCamera1.texture.width, (float)-
screenCamera1.texture.height };

    SetTargetFPS(60);               // Set our game to run at 60 frames-per-second
    //--------------------------------------------------------------------------------------
--

    // Main game loop
    while (!WindowShouldClose())    // Detect window close button or ESC key
    {
        // Update
        //----------------------------------------------------------------------------------
--
        if (IsKeyDown(KEY_S)) player1.y += 3.0f;
        else if (IsKeyDown(KEY_W)) player1.y -= 3.0f;
        if (IsKeyDown(KEY_D)) player1.x += 3.0f;
```

```
        else if (IsKeyDown(KEY_A)) player1.x -= 3.0f;

        if (IsKeyDown(KEY_UP)) player2.y -= 3.0f;
        else if (IsKeyDown(KEY_DOWN)) player2.y += 3.0f;
        if (IsKeyDown(KEY_RIGHT)) player2.x += 3.0f;
        else if (IsKeyDown(KEY_LEFT)) player2.x -= 3.0f;

        camera1.target = (Vector2){ player1.x, player1.y };
        camera2.target = (Vector2){ player2.x, player2.y };
        //----------------------------------------------------------------------------
--

        // Draw
        //----------------------------------------------------------------------------
--
        BeginTextureMode(screenCamera1);
            ClearBackground(RAYWHITE);

            BeginMode2D(camera1);

                // Draw full scene with first camera
                for (int i = 0; i < screenWidth/PLAYER_SIZE + 1; i++)
                {
                    DrawLineV((Vector2){(float)PLAYER_SIZE*i, 0}, (Vector2){
(float)PLAYER_SIZE*i, (float)screenHeight}, LIGHTGRAY);
                }

                for (int i = 0; i < screenHeight/PLAYER_SIZE + 1; i++)
                {
                    DrawLineV((Vector2){0, (float)PLAYER_SIZE*i}, (Vector2){
(float)screenWidth, (float)PLAYER_SIZE*i}, LIGHTGRAY);
                }

                for (int i = 0; i < screenWidth/PLAYER_SIZE; i++)
                {
                    for (int j = 0; j < screenHeight/PLAYER_SIZE; j++)
                    {
                        DrawText(TextFormat("[%i,%i]", i, j), 10 + PLAYER_SIZE*i, 15 +
PLAYER_SIZE*j, 10, LIGHTGRAY);
                    }
                }

                DrawRectangleRec(player1, RED);
                DrawRectangleRec(player2, BLUE);
            EndMode2D();

            DrawRectangle(0, 0, GetScreenWidth()/2, 30, Fade(RAYWHITE, 0.6f));
            DrawText("PLAYER1: W/S/A/D to move", 10, 10, 10, MAROON);

        EndTextureMode();

        BeginTextureMode(screenCamera2);
            ClearBackground(RAYWHITE);

            BeginMode2D(camera2);

                // Draw full scene with second camera
                for (int i = 0; i < screenWidth/PLAYER_SIZE + 1; i++)
                {
                    DrawLineV((Vector2){ (float)PLAYER_SIZE*i, 0}, (Vector2){
(float)PLAYER_SIZE*i, (float)screenHeight}, LIGHTGRAY);
                }

                for (int i = 0; i < screenHeight/PLAYER_SIZE + 1; i++)
                {
                    DrawLineV((Vector2){0, (float)PLAYER_SIZE*i}, (Vector2){
(float)screenWidth, (float)PLAYER_SIZE*i}, LIGHTGRAY);
                }

                for (int i = 0; i < screenWidth/PLAYER_SIZE; i++)
```

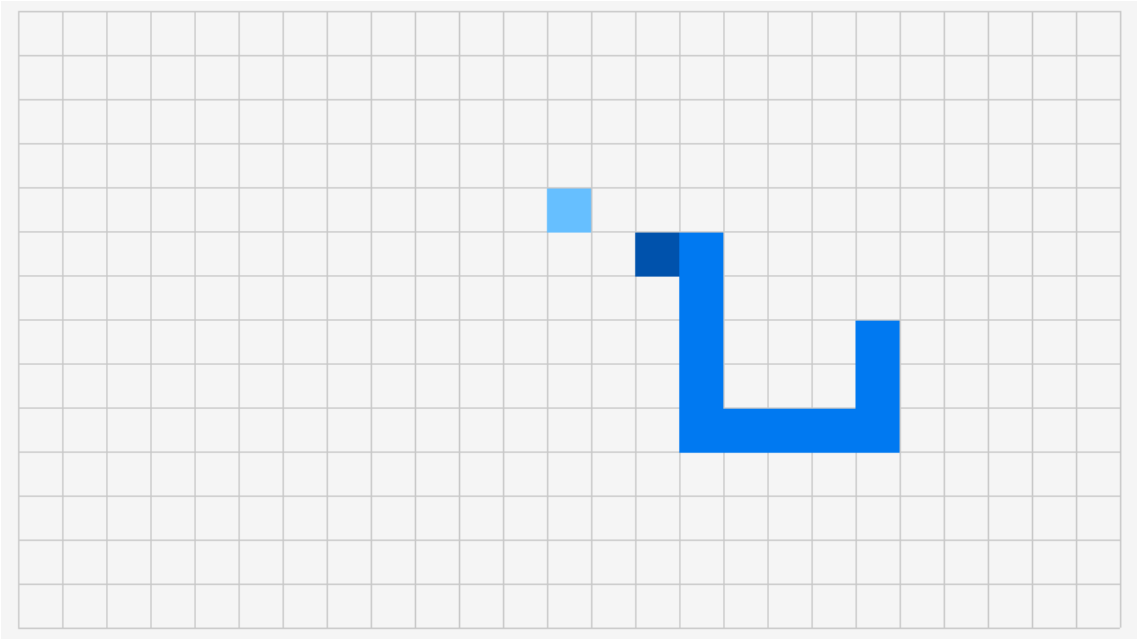
```
                {
                    for (int j = 0; j < screenHeight/PLAYER_SIZE; j++)
                    {
                        DrawText(TextFormat("[%i,%i]", i, j), 10 + PLAYER_SIZE*i, 15 +
PLAYER_SIZE*j, 10, LIGHTGRAY);
                    }
                }

                DrawRectangleRec(player1, RED);
                DrawRectangleRec(player2, BLUE);

            EndMode2D();

            DrawRectangle(0, 0, GetScreenWidth()/2, 30, Fade(RAYWHITE, 0.6f));
            DrawText("PLAYER2: UP/DOWN/LEFT/RIGHT to move", 10, 10, 10, DARKBLUE);

        EndTextureMode();

        // Draw both views render textures to the screen side by side
        BeginDrawing();
            ClearBackground(BLACK);

            DrawTextureRec(screenCamera1.texture, splitScreenRect, (Vector2){ 0, 0 },
WHITE);
            DrawTextureRec(screenCamera2.texture, splitScreenRect, (Vector2){
screenWidth/2.0f, 0 }, WHITE);

            DrawRectangle(GetScreenWidth()/2 - 2, 0, 4, GetScreenHeight(), LIGHTGRAY);
        EndDrawing();
    }

    // De-Initialization
    //--------------------------------------------------------------------------------------
--
    UnloadRenderTexture(screenCamera1); // Unload render texture
    UnloadRenderTexture(screenCamera2); // Unload render texture

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

    return 0;
}
```

Exemplo de Jogos

Tetris: um jogo de quebra-cabeça onde peças caem e você precisa encaixá-las para completar linhas. Quando consegue formar uma linha, ela some e você ganha os pontos.

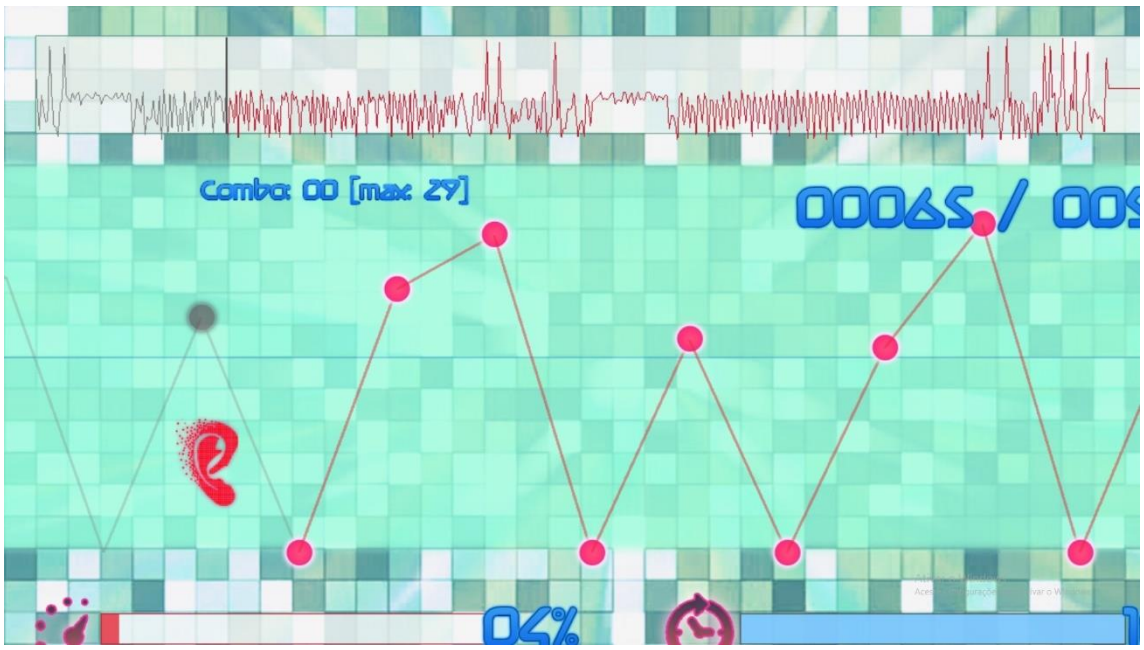


Jogo da “Cobrinha”: você controla uma cobra que se move pra pegar alimentos. Cada vez que come, a cobra cresce, e não se deve bater nas paredes ou nela mesma.



Além desses jogos clássicos, também existem muitos mais elaborados, como o Wave Collector e o Skull Escape.

Wave Collector: você controla uma orelha que tenta coletar ondas sonoras enquanto desvia de obstáculos. O objetivo é chegar até o final da música sem errar os picos das ondas sonoras.



Skull Escape: você é uma caveira tentando sair de uma casa. Mas, dependendo de suas escolhas, você pode perder vidas ao ser atacado por monstros. O jogo tem uma mecânica de escolhas que influencia o seu progresso e a sua sobrevivência.

REFERÊNCIAS

raylib | A simple and easy-to-use library to enjoy videogames programming. Disponível em: https://www.raylib.com. Acesso em 05 out. 2024.

RAYSAN5, raylib. GitHub. Disponível em: https://github.com/raysan5/raylib. Acesso em: 05 out. 2024.

SANTAMARIA, Ramon. raylib: Teaching Videogames Programming. LinkedIn. Disponível em: https://www.linkedin.com/pulse/raylib-teaching-videogames-programming-ramon-santamaria/. Acesso em: 06 out. 2024.

SANTAMARIA, Ramon. LinkedIn. Disponível em: https://www.linkedin.com/feed/update/urn:li:activity:7245090867131109376/. Acesso em: 06 out. 2024.

SANTAMARIA, Ramon. GitNation. Disponível em: https://gitnation.com/person/ramon_santamaria.. Acesso em: 06 out. 2024.