INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE SÃO PAULO

LUIGI WASCHENSHIKY LUZ

PESQUISA RAYLIB

CAMPOS DO JORDÃO

2024

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE SÃO PAULO

PESQUISA RAYLIB

Entrega da pesquisa da biblioteca RayLib da disciplina de Programação Orientada a Objetos apresentado ao Instituto Federal de São Paulo (IFSP), em cumprimento a exigência da disciplina de Programação Orientada a Objetos, do curso de Análise e Desenvolvimento de Sistemas.

PROFESSOR: Paulo Giovani de Faria Zeferine

CAMPOS DO JORDÃO

2024

LISTA DE ALGORITMOS

LISTA DE IMAGENS

SUMÁRIO

1 INTRODUÇÃO

Raylib é uma biblioteca de desenvolvimento de software de código aberto e multiplataforma. A biblioteca foi criada para desenvolver aplicações gráficas 2D e 3D e jogos. O RayLib foi criado com o intuito de apoiar um curso de programação de jogos para programadores que não possuíam uma expêriencia previa. A biblioteca é projetada para ser adequada para prototipagem, ferramentas, aplicações gráficas, sistemas embarcados e educação. O código-fonte é escrito em C puro (C99) e é distribuído sob uma licença de código aberto certificada pela OSI.

O RayLib tem como principal recurso de aprendizagem colocar a mão na massa, esta biblioteca foi criada com o intuito de ser simples, minimalista eles dizem que nãoénecessáriaumavastadocumentaçãosomenteumafolhacomoqueérealmentenecessário e que a melhor maneira de se aprender é lendo código, o raylib também possui vários exemplos de código para que possa ver como utilizar determinada funcionalidade também possui exemplos de jogos nos quais você pode testa-los e ver seus códigos.

2 ONDE É UTILIZADO?

O RayLib é utilizado para o desenvolvimento de jogos e aplicações gráficas, ele suporta compilação para várias plataformas, incluindo Windows, Linux, macOS, FreeBSD, Android, Raspberry Pi,Raspberry Pi Desktop e HTML5, mas no site do RayLib é dito que se a plataforma suportar a linguagem C e o gráficos OpenGl ou similares também podem executar o raylib . O raylib foi portada para mais de 50 linguagens de programação na forma de bindings e wrappers, mas muitas dessas portas não são estáveis.

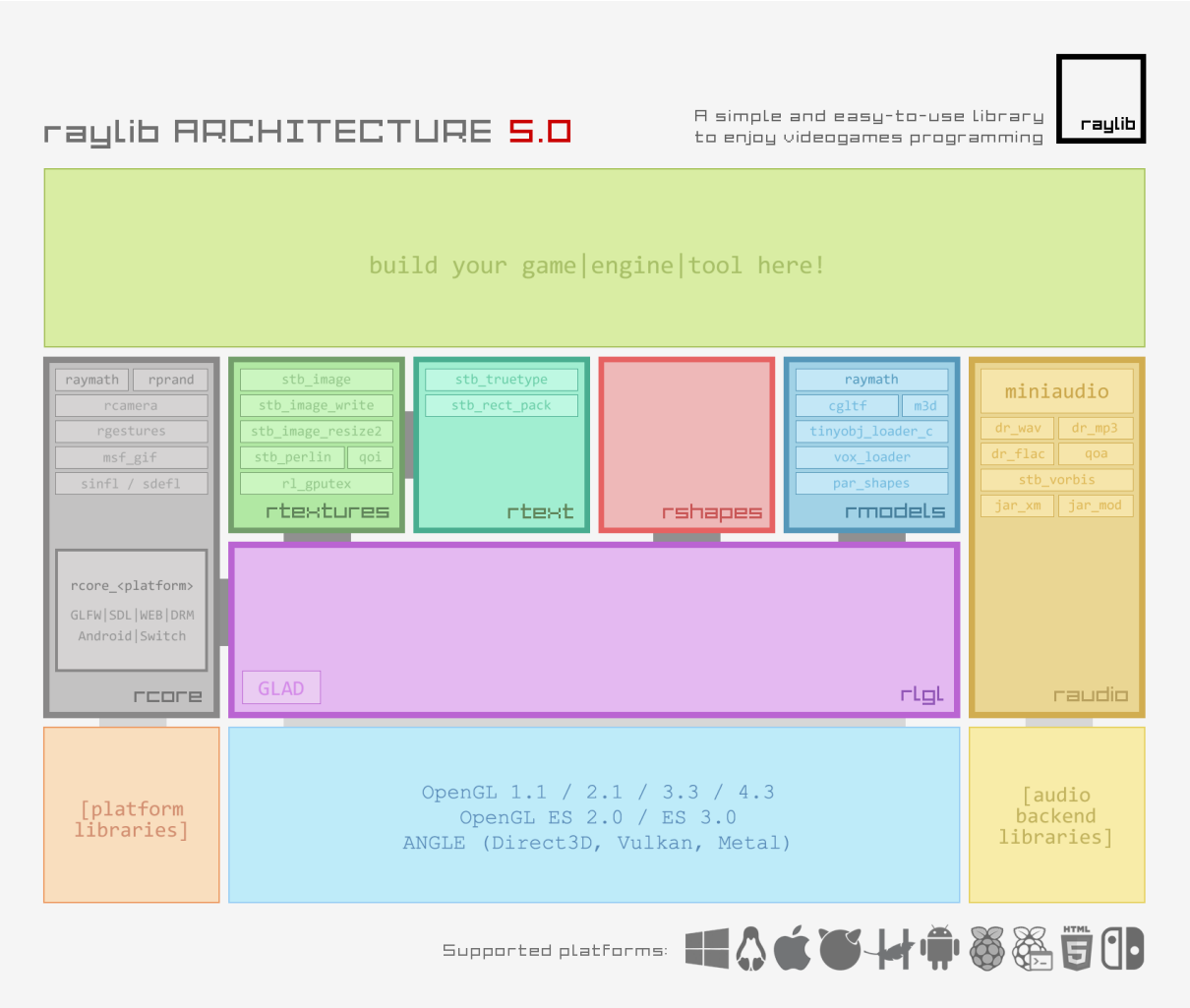

Imagem 1 – Arquitetura RayLib

3 QUEM DESENVOLVEU?

O desenvolvimento da raylib foi iniciado em agosto de 2013 por Ramon Santamaria com o intuito de apoiar um curso de programação de jogos para programadores que não possuíam uma expêriencia previa e com perfil artístico. Durante o curso, a raylib foi desenvolvida com base no feedback dos alunos e, em junho de 2014, a biblioteca começou a ser apresentada em vários eventos de desenvolvimento de jogos em Barcelona.

A raylib 1.0 foi lançada em novembro de 2013 e continha cerca de 80 funções para gerenciamento de janelas e entradas, desenho básico de formas 2D e 3D, carregamento e desenho de texturas, carregamento de fontes, desenho de textos, gerenciamento de sistemas de áudio e carregamento e reprodução de arquivos de áudio. A

primeira versão da raylib teve oito lançamentos menores subsequentes (de raylib 1.1 a raylib 1.8) ao longo de cinco anos, cada um introduzindo novos recursos. Alguns dos aprimoramentos mais notáveis foram o suporte para Android, WebAssembly e Raspberry Pi, múltiplos backends OpenGL, suporte a realidade virtual e dez exemplos.

A raylib 2.0 foi lançada em julho de 2018 e removeu todas as dependências ex ternas do sistema de compilação. Também expôs várias opções de configuração no sistema de compilação para minimizar o tamanho e aumentar o suporte, além de suportar vários sistemas de integração contínua. Nos dois anos seguintes, partes da biblioteca foram revisadas, atualizadas e o ecossistema foi desenvolvido. Durante este período, foi lançada uma versão menor, a raylib 2.5.

A raylib 3.0 foi lançada em abril de 2020, reformulando muitas partes do código para melhorar a portabilidade e os bindings. Isso incluiu a movimentação de variáveis globais para contextos, suporte para alocadores de memória personalizados, um sistema de arquivos para carregar assets e mais de 115 exemplos de código. Ela recebeu uma atualização menor, a raylib 3.5, em dezembro de 2020.

A raylib 4.0 foi lançada em novembro de 2021, com uma revisão completa de nomenclatura para consistência e coerência da biblioteca: nomes de funções, parâmetros, descrições, comentários e mensagens de log foram revisados. Ela adicionou um sistema interno de automação de eventos e expôs o controle do loop de jogo para o usuário. Também passou a contar com algumas de suas bibliotecas internas que podem ser usadas como módulos independentes: rlgl e raymath. As linguagens de programação Zig e Odin suportam oficialmente a raylib.

A raylib 4.2 foi lançada em agosto de 2022.

A raylib 4.5 foi lançada em março de 2023, 7 meses após o último lançamento. Essa atualização trouxe suporte para ANGLE em plataformas desktop, um novo módulo de câmera, suporte para modelos M3D e animações M3D/GLTF, compatibilidade com o formato de arquivo de áudio QOA, um novo módulo para carregamento de texturas comprimidas (rl_gputex), revisões nos módulos rlgl e rshapes, validação de es-

truturas de dados para muitas estruturas da raylib e muitas outras melhorias. Foi a maior atualização da biblioteca até o momento.

A raylib 5.0 foi lançada em novembro de 2023, melhorando o suporte para futu ras portas de plataforma.

4 EXEMPLOS DE CÓDIGOS E JOGOS COM RAYLIB

4.1 Exemplo de Código

Os exemplos de Raylib são organizados por cores, dependendo do módulo Raylib que eles focam. Os exemplos de Raylib são classificados por complexidade com estrelas. Uma estrela (☆) para os exemplos básicos e quatro estrelas ( ) para os mais complexos.

Este exemplo de código criado com o **raylib**, demonstra como implementar um efeito de rolagem de fundo com paralaxe em um jogo ou simulação gráfica. O objetivo principal do exemplo é desenhar três camadas de fundo (background, midground e foreground) que se movem em velocidades diferentes, criando uma sensação de profundidade e movimento contínuo.

```
/*******************************************************************
*
* Exemplo de raylib [texturas] - Rolagem em segundo plano
*
* Exemplo criado originalmente com raylib 2.0, última atualização com raylib 2.5
*
* Exemplo licenciado sob uma licença zlib/libpng não modificada, que é uma
licença certificada pela OSI, semelhante à BSD, que permite vinculação estática
com software de código fechado
*
* Copyright (c) 2019-2024 Ramon Santamaria (@raysan5)
*
************************************************/

#incluir "raylib.h"

//------------------------------------------------------------------
// Ponto de entrada principal do programa
//------------------------------------------------------------------
int main ( void )
{
```

```
    // Inicialização
    //--------------------------------------------------------------
    const int screenWidth = 800 ;
     const int screenHeight = 450 ;

    InitWindow(screenWidth, screenHeight, "exemplo de raylib [texturas] - rola-
gem de fundo" );

    // OBSERVAÇÃO: Tenha cuidado, a largura do fundo deve ser igual ou maior que
a largura da tela
    // caso contrário, a textura deve ser desenhada mais de duas vezes para
efeito de rolagem
    Texture2D background = LoadTexture( "resources/cyberpunk_street_back-
ground.png" );
    Texture2D midground = LoadTexture( "recursos/cyberpunk_street_midground.png"
);
    Texture2D primeiro plano = LoadTexture( "recursos/cyberpunk_street_fore-
ground.png" );

    rolagem flutuanteBack = 0,0f ;
     rolagem flutuanteMid = 0,0f ;
     rolagem flutuanteFore = 0,0f ;

    SetTargetFPS( 60 );               // Defina nosso jogo para rodar a 60 qua-
dros por segundo
    //--------------------------------------------------------------

    // Loop do jogo principal
    while ( ! WindowShouldClose())     // Detectar botão de fechar janela ou
tecla ESC
    {
        // Atualizar
        //----------------------------------------------------------
        scrollingBack -= 0.1f ;
        rolagemMid -= 0,5f ;
        rolagemFore -= 1.0f ;

        // NOTA: A textura é dimensionada duas vezes seu tamanho, então ela deve
ser considerada na rolagem
        if (scrollingBack <= - background.width * 2 ) scrollingBack = 0 ;
         if (scrollingMid <= - midground.width * 2 ) scrollingMid = 0 ;
         if (scrollingFore <= - foreground.width * 2 ) scrollingFore = 0 ;
          //----------------------------------------------------------

        // Empate
        //------------------------------------------------------------------------
        ComeçarDesenho();

            ClearBackground(ObterCor( 0x052c46ff ));

            // Desenhar imagem de fundo duas vezes
            // NOTA: A textura é dimensionada duas vezes seu tamanho
            DrawTextureEx(background, (Vector2){ scrollingBack, 20 }, 0.0f ,
2.0f , WHITE);
            DrawTextureEx(fundo, (Vetor2){ fundo.largura * 2 + scrollingBack, 20
}, 0.0f , 2.0f , BRANCO);

            // Desenhar a imagem do meio do plano duas vezes
            DrawTextureEx(midground, (Vector2){ scrollingMid, 20 }, 0.0f ,
2.0f , WHITE);
            DrawTextureEx(meio-terreno, (Vetor2){ meio-terreno.largura * 2 +
```

```
rolagemMid, 20 }, 0,0f , 2,0f , BRANCO);

            // Desenhar a imagem do primeiro plano duas vezes
            DrawTextureEx(foreground, (Vector2){ scrollingFore, 70 }, 0.0f ,
2.0f , WHITE);
            DrawTextureEx(primeiro plano, (Vetor2){ primeiro plano. largura * 2
+ scrollingFore, 70 }, 0,0f , 2,0f , BRANCO);

            DrawText( "ROLAGEM DE FUNDO E PARALAXE" , 10 , 10 , 20 , VERMELHO);
            DrawText( "(c) Ambiente de rua Cyberpunk por Luis Zuno (@ansimuz)" ,
screenWidth - 330 , screenHeight - 20 , 10 , RAYWHITE);

        FimDesenho();
        //---------------------------------------------- --------------------
    }

    // Desinicialização
    //--------------------------------------------------------------------------
    UnloadTexture(background);   // Descarregar textura de fundo
    UnloadTexture(midground);    // Descarregar textura de meio-campo
    UnloadTexture(foreground);   // Descarregar textura de primeiro plano

    CloseWindow();               // Fecha a janela e o contexto OpenGL
    //--------------------------------------------------------------------------

    retornar 0 ;
}
```

Algoritmo 1: Rolagem em Segundo Plano

4.2 Jogos com RayLib

Por ser extremamente simples e acessível, o RayLib atrai tanto desenvolvedores iniciantes quanto experientes, permitindo criar jogos de forma rápida e eficiente. Devido à sua flexibilidade e suporte a múltiplas plataformas, diversos jogos foram desenvolvidos utilizando a RayLib, demonstrando sua versatilidade e facilidade de uso.

Exemplos de jogos criados com RayLib:

1. Drift: Um jogo de corrida onde os carros deslizam pelas curvas, desafiando os jogadores a dominar a arte de derrapar.
2. Retro Snaker: Uma versão minimalista do clássico jogo Snake, com gráficos simples e jogabilidade fluida.
3. Pong: Um remake do clássico Pong, onde dois jogadores controlam barras para rebater uma bola, testando reflexos e precisão.

4. Asteroids: Um jogo de tiro espacial, em que o jogador deve destruir asteroides enquanto manobra sua nave em um ambiente sem gravidade.
5. Floppy: Uma recriação do popular Flappy Bird, onde o jogador guia um pássaro por uma série de obstáculos.

Esses exemplos mostram como a RayLib pode ser utilizada para criar desde jogos simples até experiências mais complexas, sendo uma excelente ferramenta para quem deseja aprender e criar jogos.

Imagem 2 - Jogo Floppy

```
/
*******************************************************************************
***********
*
*   raylib - classic game: floppy
*
*   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
```

```c
//----------------------------------------------------------------------------------
// Some Defines
//----------------------------------------------------------------------------------
#define MAX_TUBES 100
#define FLOPPY_RADIUS 24
#define TUBES_WIDTH 80

//----------------------------------------------------------------------------------
// Types and Structures Definition
//----------------------------------------------------------------------------------
typedef struct Floppy {
    Vector2 position;
    int radius;
    Color color;
} Floppy;

typedef struct Tubes {
    Rectangle rec;
    Color color;
    bool active;
} Tubes;

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

static bool gameOver = false;
static bool pause = false;
static int score = 0;
static int hiScore = 0;

static Floppy floppy = { 0 };
static Tubes tubes[MAX_TUBES*2] = { 0 };
static Vector2 tubesPos[MAX_TUBES] = { 0 };
static int tubesSpeedX = 0;
static bool superfx = false;

//----------------------------------------------------------------------------------
// 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
    //---------------------------------------------------------
    InitWindow(screenWidth, screenHeight, "classic game: floppy");

    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)
{
    floppy.radius = FLOPPY_RADIUS;
    floppy.position = (Vector2){80, screenHeight/2 - floppy.radius};
    tubesSpeedX = 2;

    for (int i = 0; i < MAX_TUBES; i++)
    {
        tubesPos[i].x = 400 + 280*i;
        tubesPos[i].y = -GetRandomValue(0, 120);
    }

    for (int i = 0; i < MAX_TUBES*2; i += 2)
    {
        tubes[i].rec.x = tubesPos[i/2].x;
        tubes[i].rec.y = tubesPos[i/2].y;
        tubes[i].rec.width = TUBES_WIDTH;
        tubes[i].rec.height = 255;

        tubes[i+1].rec.x = tubesPos[i/2].x;
        tubes[i+1].rec.y = 600 + tubesPos[i/2].y - 255;
        tubes[i+1].rec.width = TUBES_WIDTH;
        tubes[i+1].rec.height = 255;

        tubes[i/2].active = true;
    }

    score = 0;

    gameOver = false;
    superfx = false;
    pause = false;
}
```

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

        if (!pause)
        {
            for (int i = 0; i < MAX_TUBES; i++) tubesPos[i].x -= tubesSpeedX;

            for (int i = 0; i < MAX_TUBES*2; i += 2)
            {
                tubes[i].rec.x = tubesPos[i/2].x;
                tubes[i+1].rec.x = tubesPos[i/2].x;
            }

            if (IsKeyDown(KEY_SPACE) && !gameOver) floppy.position.y -= 3;
            else floppy.position.y += 1;

            // Check Collisions
            for (int i = 0; i < MAX_TUBES*2; i++)
            {
                if (CheckCollisionCircleRec(floppy.position, floppy.radius, tu-
bes[i].rec))
                {
                    gameOver = true;
                    pause = false;
                }
                else if ((tubesPos[i/2].x < floppy.position.x) && tubes[i/2].ac-
tive && !gameOver)
                {
                    score += 100;
                    tubes[i/2].active = false;

                    superfx = true;

                    if (score > hiScore) hiScore = score;
                }
            }
        }
    }
    else
    {
        if (IsKeyPressed(KEY_ENTER))
        {
            InitGame();
            gameOver = false;
        }
    }
}

// Draw game (one frame)
void DrawGame(void)
{
    BeginDrawing();

        ClearBackground(RAYWHITE);

        if (!gameOver)
        {
```

```
            DrawCircle(floppy.position.x, floppy.position.y, floppy.radius,
DARKGRAY);

            // Draw tubes
            for (int i = 0; i < MAX_TUBES; i++)
            {
                DrawRectangle(tubes[i*2].rec.x, tubes[i*2].rec.y,
tubes[i*2].rec.width, tubes[i*2].rec.height, GRAY);
                DrawRectangle(tubes[i*2 + 1].rec.x, tubes[i*2 + 1].rec.y,
tubes[i*2 + 1].rec.width, tubes[i*2 + 1].rec.height, GRAY);
            }

            // Draw flashing fx (one frame only)
            if (superfx)
            {
                DrawRectangle(0, 0, screenWidth, screenHeight, WHITE);
                superfx = false;
            }

            DrawText(TextFormat("%04i", score), 20, 20, 40, GRAY);
            DrawText(TextFormat("HI-SCORE: %04i", hiScore), 20, 70, 20, LIGHT-
GRAY);

            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 - Measu-
reText("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 2: Código Jogo Floppy

REFERÊNCIAS

A. ONLINE:

Raylib-A-simple-and-easy-to-use-library-to-enjoy-videogames-programming Disponível em:<raylib | A simple and easy-to-use library to enjoy videogames programming. Acesso em: 05 out 2024.

Raylib-Wikipedia Disponível em:<raylib - Wikipedia Acesso em: 05 out 2024.

TerminalRoot Disponível em:<Crie Jogos para Windows, Linux e Web com Raylib C/C++ (terminalroot.com.br). Acesso em: 05 out 2024.