Raylib C++ - Referência Completa

INICIALIZAÇÃO E JANELA

```
InitWindow(width, height, title)           // Primeira função obrigatória
CloseWindow()                              // Última função obrigatória
WindowShouldClose()                        // true se ESC ou fechar janela
SetTargetFPS(fps)                          // Define FPS alvo (ex: 60)
SetWindowSize(width, height)               // Redimensiona janela
ToggleFullscreen()                         // Alterna tela cheia
GetScreenWidth() / GetScreenHeight()       // Retorna dimensões
IsWindowResized()                          // Detecta redimensionamento
```

DESENHO (DRAWING)

```
BeginDrawing() / EndDrawing()              // Delimita bloco de renderização
(obrigatório)
ClearBackground(color)                     // Limpa tela com cor

// Formas básicas
DrawRectangle(x, y, width, height, color)
DrawRectangleRec(Rectangle rec, color)
DrawRectangleLines(x, y, width, height, color)
DrawRectangleLinesEx(rec, thickness, color)
DrawCircle(x, y, radius, color)
DrawCircleV(Vector2 center, radius, color)
DrawCircleLines(x, y, radius, color)
DrawLine(x1, y1, x2, y2, color)
DrawLineEx(start, end, thickness, color)
DrawTriangle(v1, v2, v3, color)
DrawPoly(center, sides, radius, rotation, color) // Polígono regular

// Texto
DrawText(text, x, y, fontSize, color)      // Fonte padrão
MeasureText(text, fontSize)                // Largura do texto
LoadFont(fileName)                         // Carrega fonte TTF/OTF
UnloadFont(font)
DrawTextEx(font, text, pos, fontSize, spacing, color)
DrawTextPro(font, text, pos, origin, rotation, fontSize, spacing, color)
```

TEXTURAS (IMAGENS)

```
LoadTexture(fileName)                      // Formatos: PNG, JPG, BMP, TGA, GIF
UnloadTexture(texture)                     // Libera memória (obrigatório)
DrawTexture(texture, x, y, tint)           // tint = WHITE para original
```

```
DrawTextureEx(texture, pos, rotation, scale, tint)
DrawTextureRec(texture, source, pos, tint) // Desenha região (spritesheet)
DrawTexturePro(texture, source, dest, origin, rotation, tint) // Controle total

// Render Texture (off-screen)
LoadRenderTexture(width, height)
UnloadRenderTexture(target)
BeginTextureMode(target) / EndTextureMode()
```

INPUT - TECLADO

```
IsKeyPressed(key)                          // true apenas no frame que pressiona
IsKeyDown(key)                             // true enquanto mantém pressionado
IsKeyReleased(key)                         // true no frame que solta
GetKeyPressed()                            // Código da última tecla
GetCharPressed()                           // Caractere Unicode

// Constantes: KEY_SPACE, KEY_ENTER, KEY_A-Z, KEY_UP, KEY_DOWN, KEY_LEFT,
KEY_RIGHT
```

INPUT - MOUSE

```
IsMouseButtonPressed(button)               // Evento único de clique
IsMouseButtonDown(button)                  // Mantém pressionado
IsMouseButtonReleased(button)
GetMousePosition()                         // Retorna Vector2
GetMouseDelta()                            // Movimento desde último frame
GetMouseWheelMove()                        // Scroll: + cima, - baixo

// Constantes: MOUSE_LEFT_BUTTON, MOUSE_RIGHT_BUTTON, MOUSE_MIDDLE_BUTTON
```

INPUT - GAMEPAD

```
IsGamepadAvailable(gamepad)                // gamepad: 0-3
IsGamepadButtonPressed(gamepad, button)
IsGamepadButtonDown(gamepad, button)
GetGamepadAxisMovement(gamepad, axis)      // Retorna -1.0 a 1.0

// Botões: GAMEPAD_BUTTON_RIGHT_FACE_DOWN (A), GAMEPAD_BUTTON_RIGHT_FACE_RIGHT (B)
// Eixos: GAMEPAD_AXIS_LEFT_X, GAMEPAD_AXIS_LEFT_Y, GAMEPAD_AXIS_RIGHT_X,
GAMEPAD_AXIS_RIGHT_Y
```

ÁUDIO

```
InitAudioDevice()                           // Chamar após InitWindow
CloseAudioDevice()                          // Chamar antes CloseWindow

// Sons curtos (efeitos)
LoadSound(fileName)                         // WAV, OGG, MP3, FLAC
UnloadSound(sound)
PlaySound(sound)
StopSound(sound)
PauseSound(sound) / ResumeSound(sound)
IsSoundPlaying(sound)
SetSoundVolume(sound, volume)               // 0.0 a 1.0
SetSoundPitch(sound, pitch)                 // 1.0 = normal

// Músicas (streaming)
LoadMusicStream(fileName)
UnloadMusicStream(music)
PlayMusicStream(music)
UpdateMusicStream(music)                    // Chamar todo frame no loop
StopMusicStream(music)
PauseMusicStream(music) / ResumeMusicStream(music)
IsMusicStreamPlaying(music)
SetMusicVolume(music, volume)
GetMusicTimePlayed(music)                   // Tempo em segundos
```

COLISÃO

```
CheckCollisionRecs(rec1, rec2)              // Retângulos AABB
CheckCollisionCircles(center1, r1, center2, r2)
CheckCollisionCircleRec(center, radius, rec)
CheckCollisionPointRec(point, rec)          // Ponto em retângulo
CheckCollisionPointCircle(point, center, radius)
CheckCollisionLines(start1, end1, start2, end2, *collisionPoint)
GetCollisionRec(rec1, rec2)                 // Retorna área de interseção
```

CÂMERA 2D

```
typedef struct Camera2D {
    Vector2 offset;      // Centro da tela (width/2, height/2)
    Vector2 target;      // Ponto que câmera segue (posição jogador)
    float rotation;      // Rotação em graus
    float zoom;          // 1.0 = normal, >1.0 = zoom in, <1.0 = zoom out
} Camera2D;

BeginMode2D(camera) / EndMode2D()           // Aplica transformações
GetScreenToWorld2D(screenPos, camera)       // Converte tela → mundo
GetWorldToScreen2D(worldPos, camera)        // Converte mundo → tela
```

MATEMÁTICA VETORIAL

```
typedef struct Vector2 { float x, y; } Vector2;

Vector2Add(v1, v2)                          // Soma
Vector2Subtract(v1, v2)                     // Subtração (direção entre pontos)
Vector2Scale(v, scale)                      // Multiplica por escalar
Vector2Length(v)                            // Magnitude/comprimento
Vector2Normalize(v)                         // Vetor unitário (comprimento 1)
Vector2Distance(v1, v2)                     // Distância entre pontos
Vector2DotProduct(v1, v2)                   // Produto escalar
Vector2Angle(v1, v2)                        // Ângulo entre vetores (radianos)
```

TEMPO

```
GetFrameTime()                              // Tempo do último frame (segundos)
GetTime()                                   // Tempo desde InitWindow
WaitTime(seconds)                           // Pausa execução (evitar no loop)

// Movimento independente de FPS:
position.x += speed * GetFrameTime();       // speed em pixels/segundo
```

UTILITÁRIOS

```
GetRandomValue(min, max)                    // Inteiro aleatório [min, max]
SetRandomSeed(seed)                         // Define semente
FileExists(fileName)                        // Verifica arquivo
DirectoryExists(dirPath)                    // Verifica diretório
GetFileExtension(fileName)
GetFileName(filePath)
```

ESTRUTURAS

```
typedef struct Color { unsigned char r, g, b, a; } Color;
// Cores: RED, GREEN, BLUE, WHITE, BLACK, YELLOW, GRAY, RAYWHITE, etc.

typedef struct Rectangle { float x, y, width, height; } Rectangle;
typedef struct Texture2D { unsigned int id; int width, height; } Texture2D;
typedef struct Font { int baseSize; Texture2D texture; } Font;
```

TEMPLATE DE PROGRAMA

```cpp
int main() {
    InitWindow(800, 600, "Game");
    SetTargetFPS(60);
    InitAudioDevice();

    // Carregar recursos (LoadTexture, LoadSound, etc)

    while (!WindowShouldClose()) {
        // ATUALIZAÇÃO
        float dt = GetFrameTime();
        // UpdateMusicStream(music); // se usar música

        // DESENHO
        BeginDrawing();
            ClearBackground(RAYWHITE);

            // Mundo (com câmera)
            BeginMode2D(camera);
                // DrawTexture, DrawRectangle, etc
            EndMode2D();

            // UI (sem câmera)
            DrawText("FPS: ...", 10, 10, 20, BLACK);
        EndDrawing();
    }

    // Descarregar recursos (UnloadTexture, UnloadSound, etc)
    CloseAudioDevice();
    CloseWindow();
    return 0;
}
```

LIMITAÇÕES CRÍTICAS

1. **Ordem obrigatória**: InitWindow → InitAudioDevice → Load recursos
2. **Memória**: Todo Load* precisa de Unload* correspondente
3. **Loop**: BeginDrawing/EndDrawing obrigatórios todo frame
4. **Música**: UpdateMusicStream() obrigatório todo frame
5. **Coordenadas**: Y cresce para baixo (0,0 = canto superior esquerdo)
6. **Thread**: Não é thread-safe, usar apenas thread principal
7. **Câmera vs UI**: UI deve ser desenhada fora de BeginMode2D
8. **Normalização**: Vector2Normalize retorna (0,0) se vetor for (0,0)
9. **FPS**: SetTargetFPS não garante FPS fixo em hardware lento
10. **Textura**: DrawTexturePro é mais lento que DrawTexture simples

PADRÕES DE USO

Movimento suave: `position += velocity * GetFrameTime()`
Seguir jogador: `camera.target = player.position`
Spritesheet: `DrawTextureRec(texture, {x, y, w, h}, position, WHITE)`

Detecção mouse: `CheckCollisionPointRec(GetMousePosition(), buttonRec)`
Distância: `Vector2Distance(pos1, pos2) < range`
Direção normalizada: `Vector2Normalize(Vector2Subtract(target, origin))`