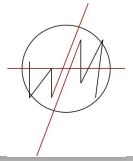
INSTITUTO FEDERAL
SÃO PAULO
Câmpus Campos do Jordão

Análise Técnica do Jogo Ping Pong WM

Wedyner Rodrigo Maciel - CJ-3019462

Conteúdo:

- Objetivo do projeto: Criar um jogo de ping pong funcional com física, renderização e lógica de jogo.
- Tecnologias:
- Linguagem: C++ (padrão C++11 ou superior)
- Biblioteca gráfica: Raylib (5.0+)
- Compilação: GCC/G++ (Linux) ou MinGW (Windows)
- Público-alvo: Desenvolvedores de jogos, estudantes de programação e entusiastas de C++.

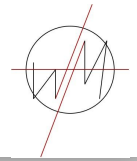


Estrutura do Projeto

Arquitetura do Jogo
Fluxo principal:

```
1. Inicialização (janela, objetos, variáveis)
2. Game Loop (atualização + renderização)
3. Entrada do usuário (teclado)
4. Lógica de física (colisões, movimento da bola)
5. Renderização (gráficos, texto, formas)
```

Dependências:
- Raylib.h (para gráficos, input e áudio)
- string.h (para formatação de texto)

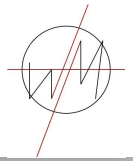


Inicialização e Configuração

Título: Setup do Jogo
Código relevante:

```
const int screenWidth = 800;
const int screenHeight = 600;
InitWindow(screenWidth, screenHeight, "Ping Pong WM");
SetTargetFPS(60);  // Controle de framerate
```

Variáveis importantes:
- `player1Pos`, `player2Pos` (posições das raquetes)
- `ballPos`, `ballSpeed` (movimento da bola)
- `player1Score`, `player2Score` (placar)



Game Loop e Input Handling

Processamento de Entradas

Código:

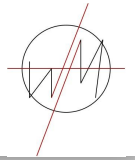
```
while (!WindowShouldClose()) {
    // Controles do jogador 1 (W/S)
    if (IsKeyDown(KEY_W)) player1Speed = -paddleSpeed;
    else if (IsKeyDown(KEY_S)) player1Speed = paddleSpeed;
    else player1Speed = 0;

    // Controles do jogador 2 (setas)
    if (IsKeyDown(KEY_UP)) player2Speed = -paddleSpeed;
    // (...)
}
```

Explicação:

- `IsKeyDown()` verifica teclas pressionadas.

- Velocidade (`paddleSpeed`) controla movimento suave.

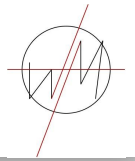


Física e Colisões

Detecção de Colisões e Movimento
Código de colisão

```
if (CheckCollisionCircleRec(ballPos, ballSize/2,
    { player1Pos.x, player1Pos.y, paddleWidth, paddleHeight })) {
    ballSpeed.x *= -1.1f; // Inverte e acelera
}
```

Física implementada:
- Rebote em bordas (`ballSpeed.y *= -1`).
- Reinício da bola após gol (`ballPos = { screenWidth/2, screenHeight/2 }`).



Renderização Gráfica

Desenho na Tela

Elementos renderizados:

1. Raquetes: `DrawRectangle()`

2. Bola: `DrawCircleV()`

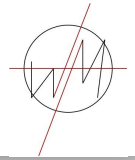
3. Placar: `DrawText(TextFormat(...))`

4. WM no centro: `DrawText("WM", ...)`

5. Linha divisória: `DrawRectangle()`

Otimização:

- Tudo desenhado no mesmo buffer (`BeginDrawing()` → `EndDrawing()`).

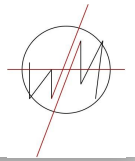


Gerenciamento de Placar

Sistema de Pontuação
Lógica

```
if (ballPos.x < 0) {  // Jogador 2 marca ponto
    player2Score++;
    ResetBall();
}
```

Reset da bola:
- Posição central (`screenWidth/2, screenHeight/2`).
- Velocidade reiniciada (`ballSpeed = {5, 5}`).



Compilação e Execução

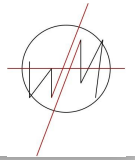
Como Compilar o Projeto
Comando no Linux

```
g++ pingpong.cpp -o pingpong -lraylib -lGL -lm -lpthread -ldl -lrt -lX11
```

Comando no Windows (MinGW):

```
g++ pingpong.cpp -o pingpong.exe -lraylib -lopengl32 -lgdi32 -lwinmm
```

Requisitos:
- Raylib instalado (`sudo apt install libraylib-dev` no Linux).



Possíveis Melhorias

Roadmap de Evolução

Sugestões técnicas:

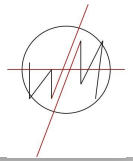
1. IA para single-player:

```
// Movimento automático da raquete do oponente
if (ballPos.y < player2Pos.y) player2Speed = -paddleSpeed;
```

2. Efeitos sonoros: `PlaySound()` (Raylib).

3. **Menu interativo: Estados do jogo (MENU, PLAYING, GAME_OVER).

4. **Shader effects: Pós-processamento com GLSL.**



Conclusão e Código Fonte

Próximos Passos e Links

Resumo:

- Jogo funcional com mecânicas básicas.

- Fácil extensão para novos recursos.

Repositório:

- GitHub: [https://github.com/Wedyner/Trabalhos_CJOPROO]

- Documentação Raylib: https://www.raylib.com

Obrigado?