INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE SÃO PAULO

LEANDRO TORRES MOCELIN

JOGO PONG EM RAYLIB E C++

CAMPOS DO JORDÃO

2024

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE SÃO PAULO

Leandro Torres Mocelin

Entrega de trabalho final apresentado ao Instituto Federal de São Paulo (IFSP-CJO), em cumprimento a exigência da disciplina de (CJOPROO) - PROGRAMAÇÃO ORIENTADA A OBJETOS, do curso de Analise e Desenvolvimento de Sistema (ADS).

Me. Paulo Giovani de Faria Zeferine

CAMPOS DO JORDÃO

2024

RESUMO

Este documento apresenta o desenvolvimento de um jogo Pong em C++ utilizando a biblioteca Raylib. O objetivo principal do projeto é ilustrar o processo de criação de um jogo interativo, aplicando conceitos fundamentais de programação, como movimentação de objetos, detecção de colisões, inteligência artificial básica e gerenciamento de pontuação. A metodologia envolveu a configuração de um ambiente de desenvolvimento com Visual Studio Code e a Raylib, seguido pela criação de classes para estruturar os componentes principais do jogo, incluindo a bola e as raquetes controladas pelo jogador e pela CPU. A implementação foi realizada em etapas, desde a construção de um loop de jogo até a configuração dos elementos gráficos, movimentação e lógica de pontuação. Adicionalmente, foi criada uma interface de usuário que exibe telas de início e fim de jogo, além de um sistema de pontuação que encerra a partida quando um dos jogadores atinge cinco pontos. Como resultado, o jogo oferece uma experiência interativa onde o jogador desafia a CPU em um cenário clássico de Pong, com feedback visual para vitórias e derrotas. Conclui-se que o projeto atingiu seus objetivos, proporcionando uma compreensão sólida dos conceitos de desenvolvimento de jogos e programação orientada a objetos. Para aprimoramentos futuros, são sugeridas adições como níveis de dificuldade, sons e aprimoramentos gráficos, visando uma experiência de jogo mais completa e imersiva.

Palavras-Chave: Raylib; Pong; C++; Inteligência Artificial; Desenvolvimento de Jogos.

ABSTRACT

This document presents the development of a Pong game in C++ using the Raylib library. The main objective of the project is to demonstrate the process of creating an interactive game, applying fundamental programming concepts such as object movement, collision detection, basic artificial intelligence, and score management. The methodology involved setting up a development environment with Visual Studio Code and Raylib, followed by creating classes to structure the main game components, including the ball and the paddles controlled by the player and CPU. The implementation was carried out in stages, from constructing a game loop to configuring the graphical elements, movement, and scoring logic. Additionally, a user interface was created to display start and game over screens, as well as a scoring system that ends the game when one of the players reaches five points. As a result, the game offers an interactive experience where the player challenges the CPU in a classic Pong setting, with visual feedback for wins and losses. The project successfully achieved its objectives, providing a solid understanding of game development concepts and object-oriented programming. Future improvements may include the addition of difficulty levels, sounds, and graphical enhancements to create a more complete and immersive game experience.

Keywords: Raylib; Pong; C++; Artificial Intelligence; Game Development.

LISTA DE ILUSTRAÇÕES

LISTA DE QUADROS

LISTA DE TABELAS

LISTA DE ALGORITMOS

LISTA DE SIGLAS

IA	Inteligência Artificial
POO	Programação Orientada a Objetos
CPU	*Central Processing Unit* (Unidade Central de Processamento)
API	*Application Programming Interface* (Interface de Programação de Aplicações)
GUI	*Graphical User Interface* (Interface Gráfica do Usuário)
IDE	*Integrated Development Environment* (Ambiente de Desenvolvimento Integrado)
FPS	*Frames Per Second* (Quadros por Segundo)
VSC	*Visual Studio Code*

SUMÁRIO

1 INTRODUÇÃO

Neste capítulo serão introduzidos os principais temas abordados ao longo deste documento, que trata do desenvolvimento de um jogo Pong utilizando C++ e a biblioteca gráfica Raylib. O trabalho explora conceitos fundamentais de programação orientada a objetos aplicados ao desenvolvimento de jogos, abordando temas como controle de movimento, detecção de colisões, inteligência artificial básica e gerenciamento de pontuação.

A motivação para este projeto surge da necessidade de aprimorar habilidades práticas em programação, utilizando uma estrutura de desenvolvimento modular e organizada para simular um ambiente de jogo interativo. A escolha do Pong, um jogo clássico e de mecânica simples, permite a aplicação eficiente de técnicas de lógica de programação, facilitando o aprendizado e a assimilação de conceitos complexos de forma progressiva.

Além de apresentar o desenvolvimento do jogo, este documento também discute os objetivos específicos do projeto, como a criação de uma interface de usuário intuitiva, a implementação de inteligência artificial para o controle da raquete do CPU, e a estruturação de um sistema de pontuação que possibilita o término do jogo com mensagens de vitória ou derrota.

Por fim, a organização do texto é explicada em detalhes, com cada capítulo focando em uma etapa do desenvolvimento do projeto.

1.1 Objetivos

O objetivo deste projeto é desenvolver uma versão interativa do jogo Pong utilizando C++ e a biblioteca gráfica Raylib. O projeto visa aplicar e consolidar conceitos fundamentais de programação orientada a objetos e de desenvolvimento de jogos, como controle de movimento, detecção de colisões, inteligência artificial básica e controle de pontuação.

1.2 Justificativa

A criação do jogo Pong permite ao desenvolvedor compreender melhor os conceitos de lógica de programação e estrutura de classes, fundamentais para a criação de jogos e aplicações interativas. A escolha do jogo Pong é justificada por sua simplicidade e relevância histórica, servindo como um ótimo exercício de prática e aplicação de técnicas básicas de desenvolvimento de jogos.

1.3 Aspectos Metodológicos

O desenvolvimento do projeto foi dividido em várias etapas, seguindo uma abordagem modular para facilitar a implementação. Cada elemento do jogo, como a bola e as raquetes, foi desenvolvido em uma classe independente, permitindo a atualização e o controle de cada objeto de forma organizada e modular.

1.4 Aporte Teórico

Este projeto se apoia em conceitos de programação orientada a objetos, como encapsulamento, herança e modularidade. Além disso, utiliza lógica de programação aplicada à detecção de colisões e inteligência artificial básica, conceitos essenciais para o desenvolvimento de jogos.

2 FUNDAMENTAÇÃO TEÓRICA

2.1 Conceitos de Programação Orientada a Objetos

A programação orientada a objetos (POO) é a base teórica para a estruturação do código deste jogo. Com o uso de classes, métodos e encapsulamento, o projeto organiza e controla os elementos do jogo de maneira eficiente.

2.2 Lógica de Detecção de Colisões

A detecção de colisões é essencial para o funcionamento do jogo, pois define a interação entre a bola e as raquetes, além das bordas da tela. A lógica foi implementada usando a função **`CheckCollisionCircleRec()`** da Raylib, permitindo que a bola mude de direção ao colidir com os objetos.

2.3 Trabalhos Relacionados

A implementação de jogos clássicos como o Pong é um exercício comum em cursos de desenvolvimento de software, pois proporciona a aplicação prática de conceitos de lógica, estrutura de dados e interface de usuário.

3 PROJETO PROPOSTO (METODOLOGIA)

Neste capítulo será apresentada detalhadamente a metodologia utilizada para o desenvolvimento do jogo Pong em C++ com a biblioteca Raylib. A escolha dessa metodologia foi baseada na simplicidade e na capacidade de segmentação das etapas de desenvolvimento, o que facilitou a implementação modular e o controle de cada componente do jogo. Essa abordagem permitiu criar um projeto organizado, onde cada elemento tem uma função específica e é desenvolvido de forma independente, o que facilita testes, manutenção e futuras expansões.

A seguir, são detalhadas as etapas principais, os artefatos utilizados, e as diretrizes para apresentação de figuras, tabelas, quadros, equações e algoritmos:

3.1 Considerações Iniciais

A estruturação inicial do projeto incluiu a escolha das ferramentas e do ambiente de desenvolvimento, sendo o Visual Studio Code a plataforma para edição e compilação do código C++, e a Raylib a biblioteca gráfica selecionada para renderização dos elementos do jogo. Essas ferramentas foram escolhidas por oferecerem uma integração prática, com ampla documentação e suporte a gráficos 2D, adequados ao desenvolvimento de um jogo simples e dinâmico como o Pong.

3.2 Requisitos

Para atingir os objetivos propostos, foram definidos os seguintes requisitos:

- **Funcionalidade de Controle de Jogo**: Tela inicial com botão de “Iniciar”, controle da raquete do jogador por meio do teclado e detecção de colisões.
- **Inteligência Artificial Básica**: A raquete do CPU deve seguir a bola para simular um oponente.
- **Sistema de Pontuação e Condição de Vitória**: A pontuação é incrementada a cada vez que a bola ultrapassa o limite da tela do oponente, e o jogo termina quando um dos jogadores atinge 5 pontos.
- **Interface Intuitiva**: Uma interface com tela de Game Over, que apresenta o resultado e oferece opção de reiniciar o jogo.

3.3 Etapas de Desenvolvimento

O desenvolvimento foi dividido em várias etapas para facilitar a implementação modular e garantir uma abordagem incremental. As principais etapas foram:

1. **Configuração do Ambiente**: Preparação do ambiente de desenvolvimento, incluindo a instalação do compilador C++ e da biblioteca Raylib.
2. **Desenvolvimento de Classes e Estrutura do Código**: Implementação das classes Ball, Paddle e CpuPaddle, cada uma encapsulando a lógica específica de cada elemento do jogo.
3. **Movimentação e Controle de Colisões**: Desenvolvimento de métodos que controlam a movimentação da bola e das raquetes, bem como a detecção de colisões com as bordas e entre os objetos.
4. **Inteligência Artificial do CPU**: Implementação da lógica de IA para a raquete do CPU, que segue a posição da bola.
5. **Interface de Usuário**: Adição de uma interface com tela inicial, botão de início e tela de Game Over.

3.4 Arquitetura

A arquitetura do jogo foi projetada para ser modular e clara, com cada elemento encapsulado em uma classe. A classe *Ball* controla o comportamento da bola, enquanto *Paddle* e *CpuPaddle* representam as raquetes do jogador e do CPU, respectivamente. O uso da programação orientada a objetos permitiu separar a lógica de cada componente, facilitando o entendimento e manutenção do código.

3.5 Projeto de Dado

As principais variáveis de dados incluem a posição e velocidade da bola, posições das raquetes, pontuações do jogador e do CPU, e o estado atual do jogo. Esses dados são atualizados durante o loop principal do jogo, permitindo que o sistema mantenha controle da partida e exiba as informações corretas ao usuário.

3.6 Interface

A interface do jogo inclui elementos simples e intuitivos para o usuário, como a tela inicial, um botão de “Iniciar”, uma tela de Game Over e a exibição da pontuação atual. Esses elementos foram organizados para tornar o jogo fácil de usar e compreender, proporcionando uma experiência de usuário eficiente.

3.7 Apresentação de Figuras

As figuras a seguir ilustram aspectos importantes do jogo Pong, desde a interface inicial até o diagrama da estrutura de classes que compõem a lógica do jogo. As figuras foram escolhidas para representar visualmente as etapas e os componentes principais do projeto, tornando mais claras as funcionalidades e a organização do sistema.

A Figura 1 exibe a tela inicial do jogo Pong, onde o jogador é recebido com uma interface simplificada e um botão centralizado de “Iniciar”. Esta tela permite que o jogador comece a partida de forma intuitiva e direta, com um design minimalista para foco exclusivo no botão de início.

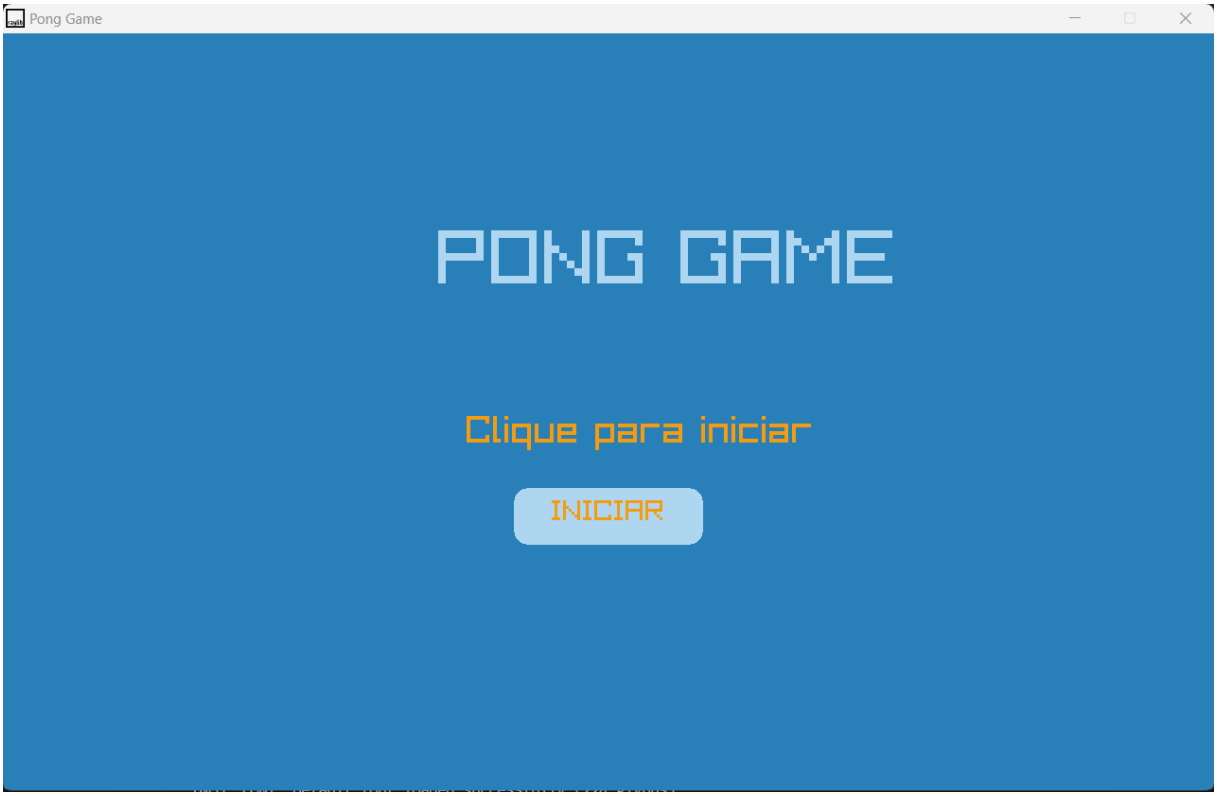


Figura 1: Tela inicial do jogo com o botão “Iniciar”

A Figura 2 mostra a tela principal do jogo durante uma partida, onde é possível visualizar os elementos essenciais do jogo: a bola, as raquetes do jogador e do CPU, e o placar. A disposição dos elementos permite ao jogador acompanhar seu progresso e a movimentação da bola, enquanto compete contra o oponente controlado pela inteligência artificial.

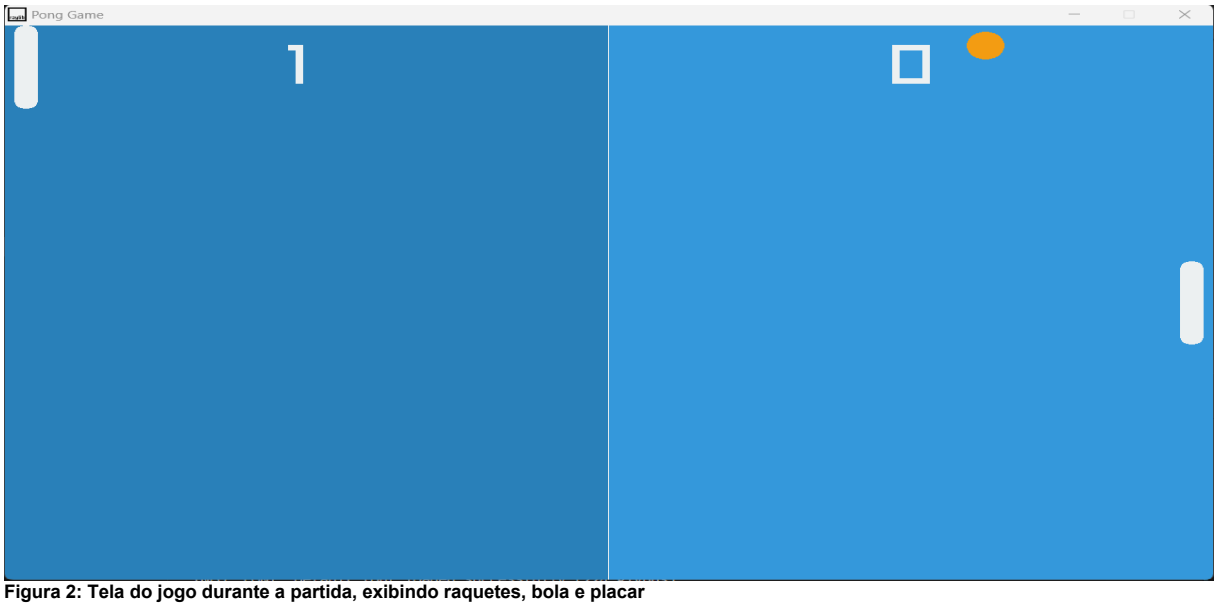


Figura 2: Tela do jogo durante a partida, exibindo raquetes, bola e placar

A Figura 3 apresenta a tela de “Game Over”, que é exibida ao final de uma partida. Nessa tela, uma mensagem de vitória ou derrota aparece com base no resultado do jogo, informando se o jogador atingiu o limite de pontos necessário para vencer ou se foi superado pelo CPU. Além disso, há uma opção de reiniciar a partida, incentivando o jogador a tentar novamente.

Figura 3: Tela de “Game Over” com mensagens de vitória ou derrota

A Figura 4 exibe a interface de entrada para o nome do jogador, que aparece logo após o botão de início. Nessa tela, o jogador pode inserir seu nome, personalizando a experiência. Essa entrada de nome é um diferencial que permite identificar o jogador nas mensagens de vitória exibidas posteriormente.

Figura 4: Interface de entrada para o nome do jogador antes do início da partida

A Figura 5 apresenta o diagrama da estrutura de classes do jogo, destacando as relações entre as classes `Ball`, `Paddle` e `CpuPaddle`. Esse diagrama ajuda a entender a organização do código e a interação entre os objetos que compõem a lógica do jogo. A classe `Ball` representa a bola que se movimenta e interage com as raquetes; a classe `Paddle` define a raquete do jogador, enquanto `CpuPaddle` define a raquete controlada pela inteligência artificial.

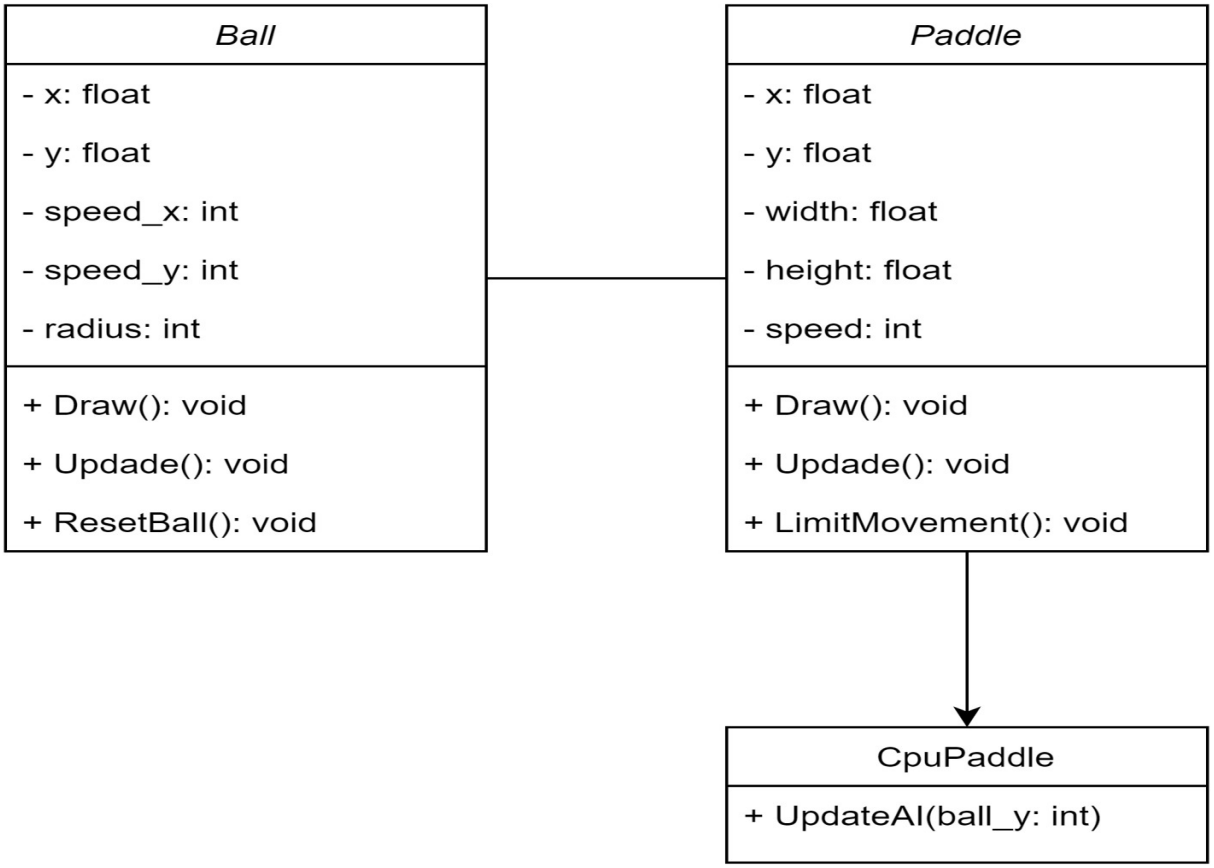


Figura 5: Diagrama da estrutura de classes do jogo, mostrando as relações entre Ball, Paddle e CpuPaddle

3.8 Apresentação de Tabelas e Quadros

As tabelas e quadros a seguir organizam informações essenciais sobre a estrutura e o desenvolvimento do jogo Pong. As tabelas foram utilizadas para apresentar dados numéricos e configurações específicas, enquanto os quadros ilustram informações textuais relevantes para o entendimento do projeto. Essas representações visuais facilitam a compreensão das características, da estrutura e da lógica implementada no jogo.

A Tabela 1 detalha o sistema de pontuação do jogo. Nela, cada jogador (Player e CPU) ganha 1 ponto ao fazer com que a bola ultrapasse o lado oposto da tela. A tabela descreve a condição de vitória, que é alcançada quando um dos jogadores atinge 5 pontos. Essa estrutura simples de pontuação contribui para uma mecânica de jogo competitiva e intuitiva.

Elemento	Descrição
Pontuação do Jogador	O jogador recebe 1 ponto quando a bola cruza o lado direito da tela controlado pelo CPU.
Pontuação do CPU	O CPU recebe 1 ponto quando a bola cruza o lado esquerdo da tela controlado pelo jogador.
Condição de Vitória	A partida termina quando um dos jogadores alcança 5 pontos, exibindo uma mensagem de vitória ou derrota.

TABELA 1 – Estrutura de Pontuação do Jogo

A Tabela 2 apresenta os parâmetros de configuração inicial dos principais objetos do jogo, como a velocidade da bola, o tamanho das raquetes, e as cores utilizadas para os elementos visuais (background, bola, e raquetes). Esses valores foram definidos para garantir uma experiência de jogo equilibrada e visualmente agradável.

Objeto	Parâmetro	Valor Inicial	Descrição
Bola	Velocidade X	5	Define a velocidade inicial da bola no eixo X.
Bola	Velocidade Y	5	Define a velocidade inicial da bola no eixo Y.
Bola	Raio	10	Define o tamanho da bola em pixels.
Raquete do Jogador	Largura	25	Largura da raquete controlada pelo jogador.
Raquete do Jogador	Altura	120	Altura da raquete controlada pelo jogador.
Raquete do CPU	Largura	25	Largura da raquete controlada pela inteligência artificial do jogo.
Raquete do CPU	Altura	20	Altura da raquete controlada pelo CPU.
Cor do Fundo	Cor	Azul Escuro	Define a cor de fundo do jogo (Dark Blue).
Cor da Bola	Cor	Laranja	Define a cor da bola (Orange).
Cor das Raquetes	Cor	Branco	Define a cor das raquetes do jogador e do CPU (White).

TABELA 2 – Parâmetros de Configuração dos Objetos do Jogo

A Tabela 3 descreve as principais fases do desenvolvimento do jogo Pong, como “Configuração do Ambiente”, “Implementação da Lógica de Movimentação e Colisão”, e “Criação da Interface de Usuário”. Cada fase é listada em ordem cronológica, permitindo visualizar o processo de criação e as etapas seguidas para a conclusão do projeto.

Fase do Desenvolvimento	Descrição
Configuração do Ambiente	Configuração do Visual Studio Code e Raylib, instalação das bibliotecas e definição das variáveis iniciais.
Implementação da Lógica de Movimentação e Colisão	Desenvolvimento das classes `Ball`, `Paddle` e `CpuPaddle`, com movimentação da bola e lógica de colisão com os limites da tela e raquetes.
Criação da Interface de Usuário	Adição de elementos visuais, como botões de início e mensagens de vitória ou derrota, e configuração do layout do jogo.

TABELA 3 – Fases do Desenvolvimento do Jogo Pong

O Quadro 1 detalha a estrutura das principais classes utilizadas no jogo: Ball, Paddle, e CpuPaddle. Ele apresenta os atributos e métodos específicos de cada classe, como movimentação, colisão e controle das raquetes. A Ball é responsável pela movimentação da bola; Paddle define a raquete do jogador e CpuPaddle implementa a raquete controlada pela IA. Esse quadro permite uma visão clara e organizada da lógica de orientação a objetos aplicada ao desenvolvimento do jogo.

Classe	Descrição	Principais Atributos	Principais Métodos
Ball	Representa a bola do jogo, responsável por sua posição e movimento na tela.	`x`, `y`, `speed_x`, `speed_y`	`Draw()`, `Update()`, `ResetBall()`
Paddle	Representa a raquete controlada pelo jogador, permitindo movimentação vertical e detecção de limites.	`x`, `y`, `width`, `height`, `speed`	`Draw()`, `Update()`, `LimitMovement()`
CpuPaddle	Raquete controlada pela CPU, que segue a posição da bola usando uma lógica de inteligência simples.	Herda de `Paddle`	`UpdateAI()` (método adicional para IA)

QUADRO 1 – Estrutura de Classes do Jogo

O Quadro 2, lista os comandos de controle do jogador, incluindo as teclas para movimentar a raquete (cima e baixo) e o botão de início. Esse quadro facilita a compreensão dos controles de entrada do usuário, garantindo uma experiência intuitiva e acessível para o jogador.

Comando	Função
Seta para Cima	Move a raquete do jogador para cima.
Seta para Baixo	Move a raquete do jogador para baixo.
Botão de Iniciar	Inicia o jogo quando pressionado na tela inicial.
Tecla Enter	Confirma o nome do jogador após a entrada.

QUADRO 2 – Controles do Jogador

O Quadro 3 descreve a estrutura do loop principal do jogo, incluindo as etapas de verificação de eventos, atualização de posições, e detecção de colisões. Este loop controla o fluxo contínuo do jogo, gerenciando a lógica necessária para manter o movimento dos objetos, a resposta aos comandos do jogador e a exibição dos elementos visuais.

Etapa do Loop	Descrição
Verificação de Eventos	Identifica entradas do usuário, como movimentação e botões pressionados.
Atualização de Posições	Calcula novas posições da bola e das raquetes com base em suas velocidades e colisões.
Detecção de Colisões	Verifica colisões entre a bola e as raquetes, além das bordas da tela.
Desenho dos Objetos	Renderiza a bola, raquetes e elementos visuais na tela com base nas posições atualizadas.

QUADRO 3 – Estrutura do Loop de Jogo

3.9 Apresentação de Algoritmos

Os algoritmos apresentados a seguir documentam a lógica central do jogo Pong, incluindo o loop principal de controle, a movimentação da bola e a detecção de colisões, além do movimento automatizado da raquete do CPU. Esses algoritmos foram implementados para garantir um funcionamento fluido e responsivo do jogo, com estrutura modular e clara, permitindo uma fácil compreensão das principais operações realizadas em cada iteração do jogo.

O Algoritmo 1 apresenta o loop principal que controla o fluxo do jogo. A cada iteração, ele desenha e atualiza os objetos, verifica o estado do jogo e processa as ações do jogador e do CPU. Esse loop também inclui a lógica para iniciar o jogo, verificar o Game Over e reiniciar a partida conforme necessário.

```
while (!WindowShouldClose()) {                // Loop principal do jogo
    BeginDrawing();                           // Início do desenho
    ClearBackground(Dark_Blue);               // Define a cor de fundo

    if (!game_started) {                      // Verifica se o jogo foi iniciado
        // Exibe a tela inicial e verifica clique para iniciar o jogo
    } else if (!name_entered) {               // Verifica se o nome foi inserido
        // Entrada do nome do jogador e exibição
    } else if (game_over) {                   // Verifica se o jogo terminou
        // Exibe a tela de Game Over e verifica se o jogador quer reiniciar
    } else {
        // Atualiza as posições da bola e das raquetes
        ball.Update();
        player.Update();
        cpu.UpdateAI(ball.y);                 // Atualiza a posição do CPU com base na bola

        // Verifica colisões e desenha todos os elementos
    }

    EndDrawing();                             // Fim do desenho
}
```

Algoritmo 1 – Estrutura do Loop de Jogo

Descrição: Este algoritmo é responsável por controlar o fluxo contínuo do jogo, atualizando os objetos e respondendo às ações do jogador. Dependendo do estado do jogo (inicial, jogando, Game Over), ele exibe diferentes telas e funcionalidades, garantindo que o jogador tenha uma experiência coesa e interativa.

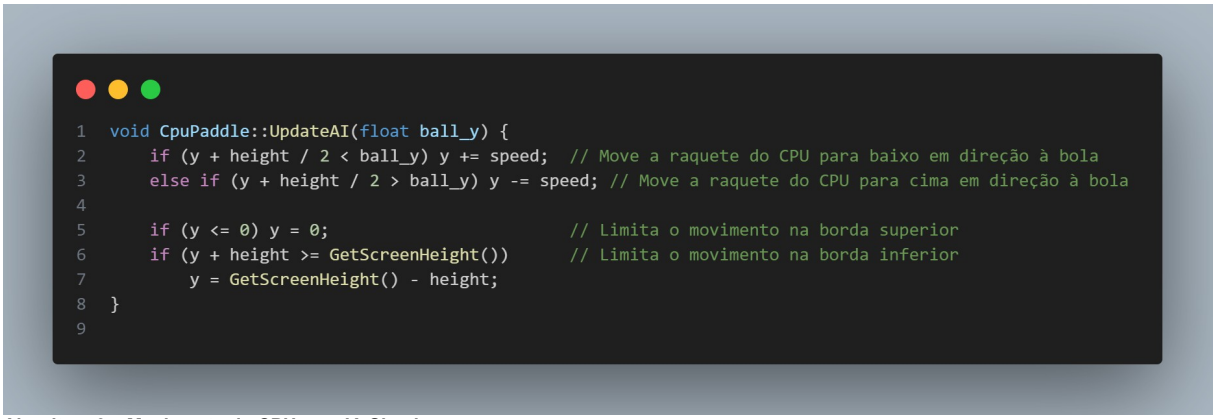
O Algoritmo 2, implementa a lógica de movimentação da bola e a detecção de colisões com as bordas e as raquetes. Ele atualiza a posição da bola em cada iteração e, ao detectar colisões com as bordas superior e inferior, inverte a direção vertical da bola. Colisões com as raquetes também são tratadas, e o algoritmo reinicia a posição da bola quando ela ultrapassa o limite da tela de um dos jogadores, atualizando o placar.

```
void Ball::Update() {
    x += speed_x;                             // Atualiza a posição horizontal
    y += speed_y;                             // Atualiza a posição vertical

    if (y + radius >= GetScreenHeight() || y - radius <= 0) {
        speed_y *= -1;                        // Inverte a direção vertical ao colidir com bordas superior/inferior
    }

    if (x + radius >= GetScreenWidth()) {
        cpu_score++;                          // Incrementa a pontuação do CPU
        ResetBall();                          // Reseta a posição da bola
    }

    if (x - radius <= 0) {
        player_score++;                       // Incrementa a pontuação do jogador
        ResetBall();                          // Reseta a posição da bola
    }
}
```

Algoritmo 2 – Atualização da Bola e Detecção de Colisões

Descrição: Esse algoritmo gerencia a movimentação da bola e sua interação com o ambiente de jogo. Ele detecta as colisões nas bordas superior e inferior e realiza a troca de direção, além de verificar quando a bola ultrapassa a raquete do jogador ou do CPU para atualizar a pontuação e resetar a bola.

O Algoritmo 3, implementa o movimento da raquete do CPU com uma lógica simples de inteligência artificial. A raquete do CPU ajusta sua posição com base na posição vertical da bola, seguindo-a para interceptá-la. Além disso, o movimento da raquete do CPU é limitado pelas bordas superior e inferior, garantindo que ela não ultrapasse o campo de jogo.



```
void CpuPaddle::UpdateAI(float ball_y) {
    if (y + height / 2 < ball_y) y += speed;  // Move a raquete do CPU para baixo em direção à bola
    else if (y + height / 2 > ball_y) y -= speed; // Move a raquete do CPU para cima em direção à bola

    if (y <= 0) y = 0;                        // Limita o movimento na borda superior
    if (y + height >= GetScreenHeight())      // Limita o movimento na borda inferior
        y = GetScreenHeight() - height;
}

```

Algoritmo 3 – Movimento do CPU com IA Simples

Descrição: Este algoritmo define o comportamento da raquete do CPU, aplicando uma lógica básica de IA que ajusta a posição da raquete para acompanhar a bola. Ele também assegura que a raquete do CPU permaneça dentro dos limites da tela, proporcionando um oponente competitivo ao jogador.

3.10 Implementação

A implementação do jogo seguiu o planejamento inicial, com a organização das classes e a criação de métodos para cada função essencial. O código foi estruturado de forma modular para garantir clareza e facilidade de manutenção, permitindo que cada funcionalidade do jogo seja isolada e testada individualmente.

3.11 Testes e Falhas Conhecidas

Durante o desenvolvimento, foram realizados testes para validar o funcionamento de cada componente. Testes de movimentação, detecção de colisões e controle de IA foram realizados para garantir uma experiência de jogo satisfatória. Falhas conhecidas incluem possíveis travamentos da bola nas bordas em situações extremas e limitações na resposta da raquete do CPU em altas velocidades.

3.12 Implantação

O jogo foi implementado e testado em ambiente Windows, utilizando o Visual Studio Code e a Raylib. A arquitetura e o código foram desenvolvidos para garantir compatibilidade com sistemas que suportem compilação em C++.

3.13 Manual do Usuário

Para jogar, o usuário deve:

1. Iniciar o jogo a partir da tela inicial clicando no botão “Iniciar”.
2. Utilizar as teclas de seta para cima e para baixo para controlar a raquete.
3. Atingir uma pontuação de 5 pontos para vencer, ou reiniciar o jogo após o término da partida.

3.14 Resultados Esperados

O jogo está completamente funcional, com movimentação da bola, controle de IA do CPU e pontuação. O objetivo é proporcionar uma experiência fluida e interativa, onde o usuário possa competir contra o CPU em um ambiente intuitivo. Expansões futuras incluem aprimoramentos na IA, adição de efeitos sonoros e personalização de níveis de dificuldade.

4 AVALIAÇÃO

Nesta seção, são apresentados os resultados do desenvolvimento do jogo Pong, acompanhados de uma análise sobre a execução do projeto e os aspectos identificados ao longo do processo. A condução da implementação, os resultados obtidos e uma discussão sobre as melhorias e possíveis avanços futuros serão detalhados a seguir.

4.1 Condução

A condução do desenvolvimento seguiu o planejamento inicial, com as etapas organizadas de forma modular para facilitar a implementação e garantir a correta funcionalidade dos elementos do jogo. A divisão do projeto em etapas – como a criação das classes principais, o loop de jogo e a detecção de colisões – possibilitou o desenvolvimento e testes independentes de cada parte. O processo incluiu também o uso de figuras e tabelas para documentar as configurações e parâmetros de objetos, facilitando o ajuste da velocidade, movimentação e lógica de colisões.

Para avaliar o desempenho do jogo, foram realizados testes com foco nos seguintes aspectos:

- **Funcionalidade da Lógica de Colisões:** Testes para garantir que a bola interage corretamente com as raquetes e bordas da tela.
- **Desempenho da IA do CPU**: Verificação da movimentação da raquete controlada pela IA, avaliando sua capacidade de seguir a bola.
- **Experiência de Usuário**: Avaliação da fluidez e responsividade dos controles do jogador, além da transição entre telas (tela inicial, tela de jogo, e tela de “Game Over”).

Os dados coletados durante esses testes permitiram identificar áreas de aprimoramento e validar o funcionamento esperado do jogo.

4.2 Resultados

Os resultados do desenvolvimento foram satisfatórios, com o jogo Pong apresentando as funcionalidades esperadas de forma eficiente. A lógica de detecção de colisões, implementada na classe `Ball`, mostrou-se robusta ao identificar corretamente os pontos de impacto com as bordas e raquetes. Esse comportamento permitiu que o jogo mantivesse uma jogabilidade precisa e interativa.

Os principais achados incluem:

- **Controle e Movimentação Fluídos**: O controle do jogador, realizado por meio das teclas direcionais, proporcionou uma experiência responsiva, com o movimento da raquete ocorrendo de forma contínua e sem falhas.
- **Inteligência Artificial do CPU**: A IA básica da raquete do CPU mostrou-se funcional, com movimentação direcionada pela posição da bola. Isso permitiu uma interação competitiva, mas ainda deixou espaço para aprimoramentos na precisão e velocidade da resposta da IA.
- **Transições de Tela e Feedback ao Usuário**: As telas de início, jogo e "Game Over" foram adequadamente implementadas, oferecendo uma estrutura de feedback que facilita a navegação e indica claramente o status atual do jogo.

Esses resultados foram organizados em figuras e tabelas, que documentam o desempenho e os parâmetros utilizados no desenvolvimento, servindo como referência para possíveis ajustes futuros.

4.3 Discussão

A análise dos resultados revela que o jogo cumpriu com os objetivos de funcionalidade e jogabilidade, oferecendo uma experiência fluida e acessível ao usuário. A lógica de colisões e a movimentação da raquete do jogador foram bem implementadas, mas identificou-se que a raquete do CPU pode ter uma resposta aprimorada, especialmente em termos de velocidade e posicionamento. Essa melhora pode ser obtida através de um ajuste fino nos parâmetros de movimentação ou até da introdução de

um algoritmo de IA mais complexo, que antecipe a trajetória da bola de maneira mais eficaz.

Os principais aspectos a considerar para aprimoramentos futuros incluem:

- **Aprimoramento da IA do CPU**: Implementação de uma lógica mais avançada para o CPU, capaz de prever melhor a trajetória da bola e oferecer um desafio maior ao jogador.
- **Adição de Efeitos Sonoros e Visuais**: A inclusão de efeitos sonoros ao colidir a bola com as raquetes e bordas, além de aprimoramentos visuais, pode tornar o jogo mais envolvente e atrativo.
- **Ajustes na Interface de Usuário**: Embora funcional, a interface do jogo pode ser aprimorada com elementos visuais mais detalhados e opções adicionais, como a seleção de níveis de dificuldade.

Em conclusão, o desenvolvimento do jogo Pong atingiu os objetivos propostos, criando uma base sólida para uma experiência de jogo divertida e interativa. A organização modular e o foco na implementação dos elementos essenciais permitiram a criação de um projeto bem estruturado, que pode ser expandido com funcionalidades adicionais no futuro, enriquecendo a experiência de usuário e ampliando as possibilidades de interação.

5 CONCLUSÃO

Este projeto teve como objetivo desenvolver um jogo Pong funcional e completo utilizando C++ e a biblioteca Raylib. Através deste trabalho, foi possível obter uma experiência prática em todas as etapas de criação de um jogo, desde a concepção e implementação da lógica básica até a integração de elementos gráficos e de interface, proporcionando uma visão abrangente do processo de desenvolvimento de software orientado a objetos.

Os objetivos propostos foram alcançados com êxito. A lógica de colisão, a movimentação da bola e a implementação de uma IA básica para a raquete do CPU foram eficazmente desenvolvidas, garantindo uma jogabilidade fluida e desafiadora. A Raylib demonstrou-se uma ferramenta valiosa ao simplificar a criação de elementos gráficos e a interação entre objetos no ambiente de jogo, permitindo um desenvolvimento mais intuitivo e organizado.

No entanto, algumas atividades e melhorias, como o aprimoramento da IA, a inclusão de efeitos sonoros e a introdução de diferentes níveis de dificuldade, foram identificadas como potenciais expansões para o projeto. Esses elementos podem ser incorporados em versões futuras para tornar o jogo mais envolvente e proporcionar ao usuário uma experiência mais rica e desafiadora. Essas sugestões apontam para o potencial do projeto em evoluir, destacando a flexibilidade e modularidade da arquitetura desenvolvida.

Em resumo, o desenvolvimento deste jogo não apenas cumpriu os objetivos iniciais, mas também proporcionou uma base sólida para a expansão e o aprendizado contínuo, consolidando conhecimentos em lógica de programação, estrutura de classes, uso de bibliotecas gráficas e desenvolvimento de jogos em C++.

REFERÊNCIAS

LIVROS:

DEITEL, Harvey. C++ Como Programar. 5. ed. São Paulo: Pearson, 2006.

MANZANO, José Augusto Navarro Garcia. Programação de Computadores com C++: guia prático de orientação e desenvolvimento. 1. ed. São Paulo: Érica, 2010. 302 p.

ONLINE:

SANTAMARIA, Ramon. Raylib Documentation. Disponível em: https://www.raylib.com/. Acesso em: 02 nov. 2024.

PROGRAMMING WITH NICK. Get Started in raylib in 20 minutes! Disponível em: https://www.youtube.com/watch?v=RGzj-PF7D74. Acesso em: 02 nov. 2024.

RAYLIB. Documentação Raylib. Disponível em: https://www.raylib.com. Acesso em: 02 nov. 2024.

RAYLIB. Raylib Wiki. Disponível em: https://github.com/raysan5/raylib/wiki. Acesso em: 02 nov. 2024.

YOUTUBE. Tutorial de introdução ao Raylib. Disponível em: https://www.youtube.com/watch?v=PPrY54Evkx8. Acesso em: 02 nov. 2024.

GLOSSÁRIO

Algoritmo: conjunto de instruções sequenciais e definidas que resolvem um problema ou executam uma tarefa específica. No contexto deste projeto, os algoritmos definem a lógica de movimentação dos objetos, a inteligência artificial básica da raquete do CPU e a atualização de estados do jogo.

Biblioteca Raylib: biblioteca gráfica em C voltada para o desenvolvimento de jogos e aplicações gráficas interativas. Utilizada para renderizar elementos gráficos e capturar as entradas do usuário de forma simplificada, ideal para jogos 2D e 3D.

C++: linguagem de programação orientada a objetos utilizada no desenvolvimento de sistemas de alto desempenho. Neste projeto, foi escolhida para implementar a estrutura de classes, controle de fluxo e lógica de jogo.

CPU: abreviação de “Central Processing Unit” (Unidade Central de Processamento). No contexto deste jogo, refere-se à raquete controlada pelo sistema, jogando automaticamente contra o usuário.

IA (Inteligência Artificial): comportamento programado para simular a inteligência humana em jogos. Neste projeto, uma IA básica controla a raquete do CPU, ajustando sua posição em relação à bola para competir contra o jogador.

APÊNDICE A: Código Completo do Jogo Pong

A seguir, apresentamos o código completo do jogo Pong desenvolvido em C++ utilizando a biblioteca Raylib. O código foi organizado para ser modular e compreensível, com funções específicas para a atualização e renderização dos principais elementos do jogo.

```
#include <raylib.h>
#include <iostream>
#include <string>

// Definição de cores
Color Blue = Color{52, 152, 219, 255};
Color Dark_Blue = Color{41, 128, 185, 255};
Color Light_Blue = Color{174, 214, 241, 255};
Color Orange = Color{243, 156, 18, 255};
Color White = Color{236, 240, 241, 255};

// Variáveis globais para pontuação, estado do jogo e nome do jogador
int player_score = 0;
int cpu_score = 0;
bool game_started = false;     // Indica se o jogo foi iniciado
bool game_over = false;        // Indica se o jogo terminou
std::string player_name = ""; // Armazena o nome do jogador
bool name_entered = false;     // Indica se o nome do jogador foi digitado

// Classe da bola
class Ball {
public:
  float x, y;              // Posição da bola
  int speed_x, speed_y;    // Velocidade da bola em X e Y
  int radius;              // Raio da bola

  // Função para desenhar a bola
  void Draw() {
    DrawCircle(x, y, radius, Orange);
  }

  // Função para atualizar a posição da bola e verificar colisões com as bor-
das
  void Update() {
    x += speed_x;
    y += speed_y;
```

```
    // Inverte a direção em Y caso a bola colida com o topo ou a base da tela
    if (y + radius >= GetScreenHeight() || y - radius <= 0) {
        speed_y *= -1;
    }

    // CPU ganha ponto se a bola atravessa o lado direito da tela
    if (x + radius >= GetScreenWidth()) {
        cpu_score++;
        ResetBall();
    }

    // Jogador ganha ponto se a bola atravessa o lado esquerdo da tela
    if (x - radius <= 0) {
        player_score++;
        ResetBall();
    }
  }

  // Função para reposicionar a bola no centro e mudar a direção aleatoriamente
  void ResetBall() {
    x = GetScreenWidth() / 2;
    y = GetScreenHeight() / 2;
    int speed_choices[2] = {-1, 1};
    speed_x *= speed_choices[GetRandomValue(0, 1)];
    speed_y *= speed_choices[GetRandomValue(0, 1)];
  }
};

// Classe da raquete do jogador
class Paddle {
protected:
  // Função para limitar o movimento da raquete aos limites da tela
  void LimitMovement() {
    if (y <= 0) {
        y = 0;
    }
    if (y + height >= GetScreenHeight()) {
        y = GetScreenHeight() - height;
    }
  }

public:
  float x, y;       // Posição da raquete
  float width, height; // Largura e altura da raquete
  int speed;        // Velocidade da raquete

  // Função para desenhar a raquete
  void Draw() {
```

```
    DrawRectangleRounded(Rectangle{x, y, width, height}, 0.8, 0, White);
  }

  // Função para atualizar o movimento da raquete conforme teclas de seta
  void Update() {
    if (IsKeyDown(KEY_UP)) {
      y -= speed;
    }
    if (IsKeyDown(KEY_DOWN)) {
      y += speed;
    }
    LimitMovement();
  }
};

// Classe da raquete do CPU, herda da classe Paddle
class CpuPaddle : public Paddle {
public:
  // Função para atualizar o movimento da raquete do CPU, seguindo a posição
da bola
  void Update(int ball_y) {
    if (y + height / 2 > ball_y) {
      y -= speed;
    }
    if (y + height / 2 <= ball_y) {
      y += speed;
    }
    LimitMovement();
  }
};

// Instância das classes Ball, Paddle e CpuPaddle
Ball ball;
Paddle player;
CpuPaddle cpu;

// Função para reiniciar o jogo
void ResetGame() {
  player_score = 0;
  cpu_score = 0;
  game_over = false;
  ball.ResetBall();
}

// Função para capturar o nome do jogador
void GetPlayerName() {
  int key = GetKeyPressed();
  while (key > 0) {
    // Adiciona caracteres ao nome do jogador, limitando a 20 caracteres
```

```
    if ((key >= 32) && (key <= 125) && (player_name.size() < 20)) {
      player_name.push_back((char)key);
    }
    // Remove o último caractere se a tecla BACKSPACE for pressionada
    if (key == KEY_BACKSPACE && !player_name.empty()) {
      player_name.pop_back();
    }
    key = GetKeyPressed();
  }
}

int main() {
  // Configurações iniciais da tela
  const int screen_width = 1280;
  const int screen_height = 800;
  InitWindow(screen_width, screen_height, "Pong Game");
  SetTargetFPS(60);

  // Inicialização dos parâmetros da bola e das raquetes
  ball.radius = 20;
  ball.x = screen_width / 2;
  ball.y = screen_height / 2;
  ball.speed_x = 7;
  ball.speed_y = 7;

  player.width = 25;
  player.height = 120;
  player.x = screen_width - player.width - 10;
  player.y = screen_height / 2 - player.height / 2;
  player.speed = 6;

  cpu.height = 120;
  cpu.width = 25;
  cpu.x = 10;
  cpu.y = screen_height / 2 - cpu.height / 2;
  cpu.speed = 6;

  // Loop principal do jogo
  while (!WindowShouldClose()) {
    BeginDrawing();
    ClearBackground(Dark_Blue);

    if (!game_started) {
      // Tela inicial com botão "Iniciar"
      DrawText("PONG GAME", screen_width / 2 - 180, screen_height / 4, 80,
Light_Blue);
      DrawText("Clique para iniciar", screen_width / 2 - 150, screen_height /
2, 40, Orange);
```

```
        Rectangle startButton = {screen_width / 2 - 100, screen_height / 2 + 80,
200, 60};
        DrawRectangleRounded(startButton, 0.5, 0, Light_Blue);
        DrawText("INICIAR", screen_width / 2 - 60, screen_height / 2 + 90, 30,
Orange);

        if (IsMouseButtonPressed(MOUSE_LEFT_BUTTON) && CheckCollisionPoin-
tRec(GetMousePosition(), startButton)) {
            game_started = true;
        }
    } else if (!name_entered) {
        // Entrada do nome do jogador
        DrawText("Digite seu nome:", screen_width / 2 - 180, screen_height / 4,
40, White);
        DrawText(player_name.c_str(), screen_width / 2 - 100, screen_height / 2,
30, Light_Blue);

        GetPlayerName();

        if (IsKeyPressed(KEY_ENTER) && !player_name.empty()) {
            name_entered = true;
            ResetGame();
        }
    } else if (game_over) {
        // Tela final com o resultado
        if (player_score >= 5) {
            DrawText(TextFormat("Parabéns %s, você venceu!", player_name.c_str()),
screen_width / 2 - 250, screen_height / 2, 40, Orange);
        } else if (cpu_score >= 5) {
            DrawText("Você perdeu! Tente novamente!", screen_width / 2 - 200,
screen_height / 2, 40, Orange);
        }

        Rectangle restartButton = {screen_width / 2 - 100, screen_height / 2 +
80, 200, 60};
        DrawRectangleRounded(restartButton, 0.5, 0, Light_Blue);
        DrawText("REINICIAR", screen_width / 2 - 75, screen_height / 2 + 90, 30,
Orange);

        if (IsMouseButtonPressed(MOUSE_LEFT_BUTTON) && CheckCollisionPoin-
tRec(GetMousePosition(), restartButton)) {
            game_started = false;
            player_name.clear();
            name_entered = false;
            ResetGame();
        }
    } else {
        // Jogo em andamento
        ball.Update();
```

```
        player.Update();
        cpu.Update(ball.y);

        if (CheckCollisionCircleRec({ball.x, ball.y}, ball.radius, {player.x,
player.y, player.width, player.height})) {
            ball.speed_x *= -1; // Inverte direção ao colidir com a raquete do
jogador
        }
        if (CheckCollisionCircleRec({ball.x, ball.y}, ball.radius, {cpu.x,
cpu.y, cpu.width, cpu.height})) {
            ball.speed_x *= -1; // Inverte direção ao colidir com a raquete do CPU
        }

        if (player_score >= 5 || cpu_score >= 5) {
            game_over = true; // Termina o jogo se algum jogador alcança 5 pontos
        }

        // Desenha a bola, as raquetes e o placar
        ball.Draw();
        player.Draw();
        cpu.Draw();
        DrawText(TextFormat("%i", player_score), screen_width / 2 + 60, 50, 40,
Orange);
        DrawText(TextFormat("%i", cpu_score), screen_width / 2 - 60, 50, 40,
Orange);
        DrawText(player_name.c_str(), screen_width - 150, screen_height - 30,
20, White);
    }
    EndDrawing();
  }

  // Fecha a janela do jogo
  CloseWindow();
  return 0;
}
```

Este código apresenta a estrutura completa do jogo Pong, incluindo a inicialização, o loop de jogo, as classes `Ball`, `Paddle` e `CpuPaddle`, além das lógicas de movimentação, colisão e controle do jogador e CPU. A implementação permite que o jogo funcione de maneira autônoma e modular, facilitando possíveis melhorias e manutenções futuras.

ANEXO A: DOCUMENTAÇÃO DA BIBLIOTECA RAYLIB

Este anexo apresenta uma seleção da documentação oficial da biblioteca Raylib, que serviu de base para o desenvolvimento do jogo Pong. A Raylib é uma biblioteca gráfica desenvolvida para facilitar a criação de jogos e aplicações gráficas em C e C++. As informações a seguir foram extraídas do site oficial da Raylib e fornecem uma visão geral das principais funções e características utilizadas no projeto.

Trechos da Documentação

1. **Função InitWindow**
   A função InitWindow inicializa e abre uma janela para o jogo. É a função principal para definir o título, largura e altura da janela.
   Exemplo: InitWindow(screen_width, screen_height, "Pong Game");
2. **Função BeginDrawing e EndDrawing**
   BeginDrawing inicia o processo de renderização, enquanto EndDrawing finaliza a renderização e exibe o conteúdo na tela.
3. **Função DrawText**
   DrawText permite exibir textos na tela, com personalização de tamanho, cor e posição. No jogo Pong, essa função é usada para exibir a pontuação, mensagens de início e "Game Over".
4. **Função DrawCircle e DrawRectangleRounded**
   Utilizadas para desenhar formas na tela, DrawCircle é responsável pela bola do jogo, enquanto DrawRectangleRounded define as raquetes.
5. **Função CheckCollisionCircleRec**
   Esta função verifica a colisão entre um círculo e um retângulo, essencial para determinar o contato da bola com as raquetes e com as bordas.
6. **Função WindowShouldClose**
   WindowShouldClose verifica se a janela deve ser encerrada, facilitando o controle do loop principal do jogo.

Fonte

Toda a documentação da Raylib pode ser encontrada no site oficial: https://www.raylib.com/

Este anexo serve como referência complementar para desenvolvedores interessados em entender as funcionalidades básicas da Raylib e como elas foram aplicadas no projeto. A consulta à documentação oficial foi essencial para a implementação das funcionalidades visuais e lógicas do jogo Pong.