INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE SÃO PAULO

NICOLAS DE OLIVEIRA AMARAL

PONG GAME

CAMPOS DO JORDÃO

2024

RESUMO

Este trabalho aborda o desenvolvimento de um jogo Pong utilizando a linguagem C++ e a biblioteca Raylib. O objetivo foi implementar uma versão funcional do jogo clássico, explorando conceitos de programação estruturada e orientada a objetos, como colisões, movimentação da bola e inteligência artificial para o paddle controlado pela CPU. A metodologia incluiu a análise de repositórios de código, estudos teóricos em livros de programação e tutoriais práticos, além do uso de ferramentas visuais para testar e ajustar o comportamento do jogo em tempo real. Como resultado, foi obtido um jogo funcional com modos para jogador único e dois jogadores, um menu interativo e um sistema de pausa. No entanto, foram identificados pequenos problemas, como lag ocasional na movimentação da bola, que não comprometeram a experiência geral. A conclusão reflete sobre o sucesso em atingir os objetivos propostos e discute melhorias futuras, como otimização do código e aprimoramento da inteligência artificial. O trabalho serve como exemplo prático de aplicação de fundamentos de programação em projetos de jogos digitais.

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

ABSTRACT

This paper discusses the development of a Pong game using the C++ programming language and the Raylib library. The objective was to implement a functional version of the classic game, exploring concepts of structured and object-oriented programming, such as collision detection, ball movement, and artificial intelligence for the CPU-controlled paddle. The methodology included analyzing code repositories, theoretical studies in programming books, and practical tutorials, as well as using visual tools to test and adjust the game's behavior in real-time. As a result, a functional game was achieved, featuring single-player and two-player modes, an interactive menu, and a pause system. However, minor issues such as occasional lag in ball movement were identified but did not compromise the overall experience. The conclusion reflects on the success in achieving the proposed objectives and discusses future improvements, such as code optimization and AI enhancement. This work serves as a practical example of applying programming fundamentals to digital game projects.

Keywords: Pong; C++; Raylib; Artificial Intelligence; Digital Games.

LISTA DE ILUSTRAÇÕES

LISTA DE QUADROS

LISTA DE ALGORITMOS

SUMÁRIO

1. INTRODUÇÃO

O presente trabalho aborda o tema “Jogo Pong” e busca explorar os conceitos de programação orientada a objetos. O Pong, conhecido como um dos primeiros videogames da história, continua sendo uma excelente ferramenta educacional para iniciantes em programação, pois envolve conceitos fundamentais como controle de movimento, colisão e interatividade. Este projeto visa investigar as etapas necessárias para construir um jogo Pong, focando na implementação de movimentação dos elementos do jogo (raquetes e bola), detecção de colisão e resposta ao jogador. Espera-se que esta aplicação contribua para um maior entendimento dos princípios básicos de desenvolvimento de jogos e para o aprimoramento de habilidades em programação.

1.1. OBJETIVOS

O objetivo deste trabalho é desenvolver um jogo Pong que simule o clássico jogo de tênis de mesa digital, aplicando conceitos de programação e lógica computacional. Através da criação do Pong, busca-se consolidar conhecimentos sobre movimentação e controle de objetos, implementação de colisões e interações no jogo. Além disso, o projeto visa fomentar o aprendizado prático em desenvolvimento de jogos, permitindo que o aluno entenda e execute os elementos essenciais de uma aplicação interativa. Espera-se que o jogo resultante possa servir como uma base para projetos futuros de jogos e para o aperfeiçoamento de habilidades técnicas essenciais no campo da programação.

1.2. JUSTIFICATIVA

A escolha de desenvolver um jogo Pong como tema deste trabalho justifica-se pela relevância pedagógica e histórica deste jogo no campo da programação. Pong, sendo um dos primeiros jogos de arcade, oferece uma excelente base para explorar conceitos fundamentais de lógica de programação, como controle de fluxo, manipulação de variáveis e detecção de colisão, que são essenciais no desenvolvimento de jogos mais complexos. Além disso, ao reproduzir a estrutura simples de Pong, o projeto proporciona um ambiente prático para desenvolver

habilidades de modularização de código e design de jogos, habilidades valiosas na formação de futuros programadores

2. ASPECTOS METODOLÓGICOS

Para o desenvolvimento do jogo Pong, foi utilizado o método incremental e iterativo, estruturando as funcionalidades em etapas para garantir que o jogo evoluísse de uma versão básica para uma versão mais refinada e funcional. A escolha da linguagem C++ e da biblioteca gráfica Raylib fundamenta-se na simplicidade e eficiência que ambas oferecem para a criação de jogos 2D de baixo nível, permitindo o controle direto de elementos gráficos e de lógica de jogo. Para uma maior praticidade ao usar a linguagem e compilar o jogo, foi usado o software de edição de texto Visual Studio Code (VSCode).

2.1. LINGUAGEM DE PROGRAMAÇÃO C++

Neste trabalho, a escolha da linguagem de programação C++ se deve à sua alta performance, versatilidade e grande capacidade de manipulação de recursos de baixo nível, o que a torna ideal para desenvolvimento de jogos e aplicações de alta eficiência. A linguagem permite controle detalhado sobre a memória e o hardware, características essenciais para o desenvolvimento de jogos dinâmicos e com alta capacidade de resposta, como no projeto aqui abordado.

2.2. BIBLIOTECA RAYLIB

Para o desenvolvimento gráfico e interativo, foi utilizada a biblioteca Raylib, uma API de código aberto projetada especificamente para a criação de jogos. A Raylib facilita a implementação de gráficos e interações, com foco em simplicidade e eficiência. Sua estrutura amigável para iniciantes permite o desenvolvimento rápido, além de ser compatível com C e C++, facilitando a integração com a linguagem principal deste projeto.

2.3. SOFTWARE DE EDIÇÃO DE TEXTO: VSCODE

O VSCode foi escolhido como editor de texto e ambiente de desenvolvimento devido à sua flexibilidade, extensões personalizáveis e interface intuitiva. Ele suporta depuração e formatação automáticas para C++ e possui ferramentas que facilitam o desenvolvimento com Raylib, otimizando o tempo e a produtividade.

3. APORTE TEÓRICO

O aporte teórico deste projeto se concentra nas principais fontes de estudo que fundamentaram o desenvolvimento do jogo, combinando conceitos de programação, desenvolvimento de jogos e práticas de software. A base teórica foi extraída principalmente de documentações oficiais, livros de referência, repositórios GitHub e tutoriais online.

a fundamentação teórica incluiu a consulta de livros de lógica de programação, como "Lógica de Programação: A Construção de Algoritmos e Estruturas de Dados" de André Luiz Villar Forbellone, para aprofundar o entendimento sobre a criação de algoritmos eficientes. A análise de repositórios GitHub com projetos de jogos baseados em Raylib e C++ também forneceu exemplos práticos e boas práticas para a estruturação do código. Complementarmente, tutoriais no YouTube auxiliaram na execução de técnicas específicas, fornecendo uma base visual e prática para o desenvolvimento do projeto.

Com o estudo de livros fundamentais sobre C++, como "The C++ Programming Language" de Bjarne Stroustrup, a pesquisa se consolidou em um conjunto teórico robusto, que une o conhecimento prático à teoria, fundamentando o desenvolvimento de um jogo completo e funcional.

4. PROJETO PROPOSTO

A metodologia escolhida para o desenvolvimento deste projeto foi o modelo ágil. Essa abordagem foi selecionada devido à sua flexibilidade e capacidade de

adaptação durante o ciclo de vida do projeto, permitindo ajustes rápidos e entregas incrementais. Além disso, a metodologia ágil facilita o acompanhamento contínuo do progresso e promove uma maior colaboração entre a equipe de desenvolvimento e os stakeholders. O processo de desenvolvimento foi estruturado em sprints curtas e entregas contínuas de funcionalidades, o que garante um alinhamento contínuo com as necessidades finais e uma melhoria contínua do sistema.

4.1. APRESENTAÇÃO DE FIGURAS

As figuras apresentadas a seguir têm como objetivo ilustrar aspectos fundamentais do sistema desenvolvido, oferecendo uma visão clara das etapas e dos componentes principais que integram o projeto. As imagens foram cuidadosamente selecionadas para complementar a explicação textual, facilitando a compreensão da lógica e da interface implementadas.

A figura 1 apresenta o menu principal do sistema, destacando a organização das opções de navegação. Com um design funcional e intuitivo, o menu permite acesso rápido às principais funcionalidades, priorizando a usabilidade e a experiência do usuário.

FIGURA 1 – MENU DO JOGO PONG

A figura 2 apresenta a tela do jogo Pong em modo "Player x Player". A interface apresenta um fundo azul com elementos contrastantes em branco e laranja, garantindo visibilidade e funcionalidade. As raquetes, localizadas nas extremidades esquerda e direita, são representadas por retângulos verticais brancos, enquanto a bola, posicionada no centro inferior, é exibida em laranja. Uma linha branca divide a tela ao meio, delimitando os lados dos jogadores. Na parte superior, há dois contadores que exibem o placar de cada jogador. No canto inferior esquerdo, instruções de interação são apresentadas: "Pressione 'P' para Pausar", "Pressione 'M' para voltar ao Menu" e "Pressione 'ESC' para fechar o Jogo". A disposição minimalista e o design funcional visam proporcionar uma experiência de jogo fluida, com fácil compreensão das regras e comandos. Deve ser levado em consideração que a opção “CPU x Player” tem a mesma interface que "Player x Player"., por isso não será apresentada nenhuma figura para ela.

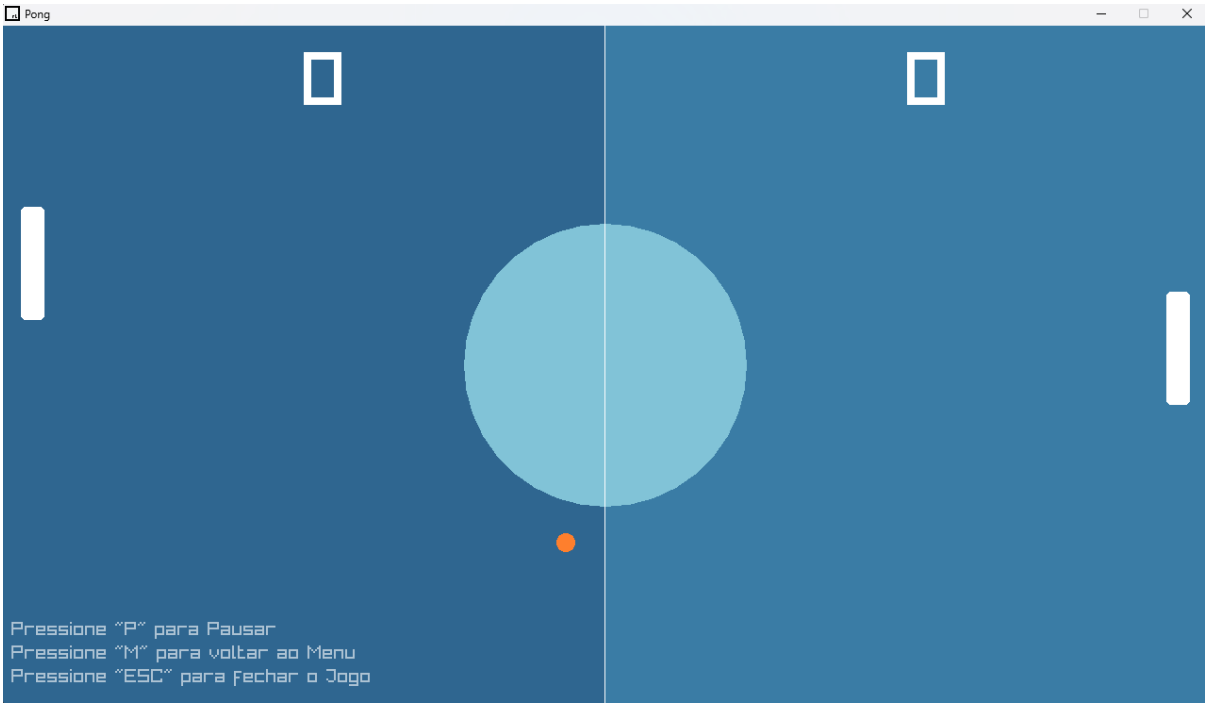

FIGURA 2 – PLAYER X PLAYER

A figura 3 apresentada abaixo refere-se ao estado de pausa do jogo Pong, ativado quando o usuário pressiona a tecla "P". Centralizado na interface, há um texto em destaque com a palavra "PAUSE", indicando que o jogo está temporariamente interrompido. O fundo mantém os elementos principais do jogo visíveis, incluindo as raquetes, o marcador de pontuação e a bola, todos estilizados com cores suaves e tons escuros que favorecem o contraste do texto central.

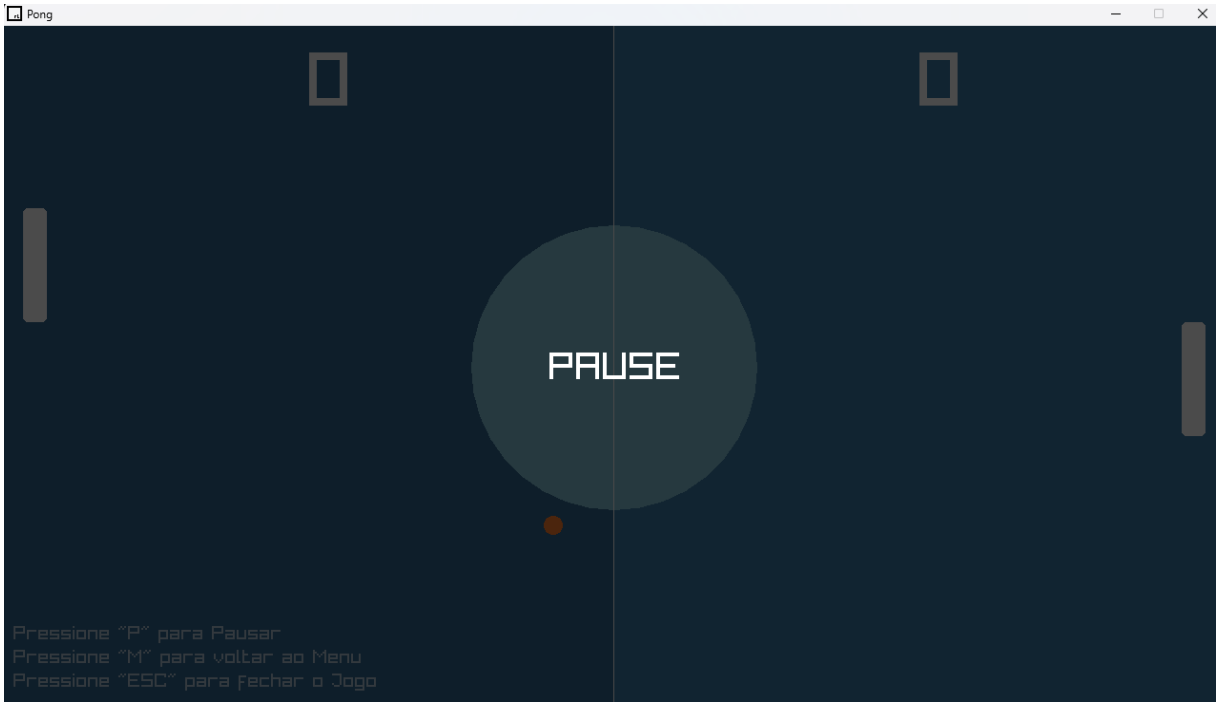

FIGURA 3 – PAUSE

A figura 4 apresenta a tela de "Game Over" do jogo. No centro da tela, destaca-se a mensagem principal indicando o vencedor do jogo: "GAME OVER - PLAYER 2 WON!". Abaixo dessa mensagem, são exibidas instruções claras para o jogador, como: “Pressione 'R' para Reiniciar” (Essa opção permite começar uma nova partida imediatamente) e “Pressione 'M' para Voltar ao Menu” (Essa alternativa leva o jogador de volta ao menu principal do jogo).

FIGURA 4 – GAME OVER

4.2. APRESENTAÇÃO DE QUADROS

Os quadros a seguir foram criados para organizar e ilustrar as principais variáveis, objetos, classes e funções utilizados no código do jogo Pong. Ele tem como objetivo fornecer uma visão clara das propriedades e comportamentos de cada elemento envolvido no jogo, como a bola, as raquetes dos jogadores e a IA do CPU, além de destacar as principais interações entre esses elementos.

Campo	Valor
playerScore	0
player2Score	0
cpuScore	0
maxScore	5 (Pontuação máxima para ganhar)
isPaused	false (indicando que o jogo não está pausado)
gameIsRunning	0 (Jogo não está em execução)
ball.radius	10 (Raio da bola)
ball.speedX	7.0 (Velocidade da bola em X)
ball.speedY	7.0 (Velocidade da bola em Y)
player.height	120.0 (Altura do paddle do jogador)
player.width	25.0 (Largura do paddle do jogador)
player.speed	6.0 (Velocidade de movimento do paddle)
player2.height	120.0 (Altura do paddle do segundo jogador)
player2.height	25.0 (Largura do paddle do segundo jogador)
player2.speed	6.0 (Velocidade de movimento do paddle)
cpu.height	120.0 (Altura do paddle da CPU)
cpu.width	25.0 (Largura do paddle da CPU)
cpu.speed	6.0 (Velocidade de movimento do paddle)

QUADRO 1 – PRINCIPAIS VARIÁVEIS

Classe	Descrição
Ball	Representa a bola no jogo, com atributos de posição, velocidade, raio e lógica de movimentação.
Paddle	Representa a raquete do jogador, com atributos de posição, tamanho, velocidade e controle de movimento.
player2Paddle	Subclasse de `Paddle` para o controle da raquete do jogador 2, com movimentos controlados pelas teclas W e S.
CpuPaddle	Subclasse de `Paddle` para o controle da raquete do CPU, com lógica de movimentação que tenta prever a posição futura da bola.

QUADRO 2 – CLASSES

Função	Descrição
Ball::Draw()	Desenha a bola na tela.
Ball::Update()	Atualiza a posição da bola e verifica colisões com as bordas da tela.
Ball::resetBall()	Reseta a posição da bola e aplica uma nova velocidade aleatória.
Paddle::Draw()	Desenha a raquete do jogador na tela.
Paddle::Update()	Atualiza a posição da raquete do jogador, movendo-se para cima ou para baixo.
player2Paddle::Update()	Atualiza a posição da raquete do jogador 2, movendo-se para cima ou para baixo.
CpuPaddle::Update()	Atualiza a posição da raquete do CPU, prevendo a posição futura da bola.
CheckPaddleCollision()	Verifica colisão entre a bola e a raquete, invertendo a direção da bola.
Pause()	Alterna o estado de pausa do jogo.
DrawPauseMenu()	Desenha o menu de pausa na tela.
DrawGameOverScreen()	Desenha a tela de "Game Over" após um jogador vencer.
ResetGame()	Reseta a pontuação e o estado do jogo.
main()	Função principal que inicializa o jogo, gerencia o loop de renderização e a lógica do jogo.

QUADRO 3 – FUNÇÕES

4.3. APRESENTAÇÃO DE ALGORITMOS

Nesta seção, apresentamos a lógica de implementação do jogo Pong, abordando suas principais funcionalidades, incluindo o controle das raquetes, a movimentação da bola, a detecção de colisões, o sistema de pontuação e os menus de pausa e de "Game Over".

4.3.1. ESTRUTURA DO JOGO

O código consiste em três principais entidades: a bola, as raquetes (de dois jogadores ou do CPU), e o controle do fluxo do jogo. Abaixo, cada componente do código será detalhado, destacando a funcionalidade e a interação entre as partes.

4.3.1.1. CLASSES E FUNÇÕES

4.3.1.1.1. CLASSE BALL

A classe Ball representa a bola do jogo, com atributos que definem sua posição, velocidade e raio. A lógica principal está na atualização da sua posição a cada quadro, levando em consideração a colisão com as bordas da tela.

ALGORITMO 1 – CLASSE BALL

```
1  class Ball {
2  public:
3      float x, y; // Coordenadas da bola na tela
4      float speedIncrement = 0.08; // Taxa de incremento na velocidade
5  ao longo do tempo
6      float speedX, speedY; // Velocidades horizontal e vertical da
7  bola
8      int radius; // Raio da bola
9      int frameCounter = 0; // Contador para controlar o incremento de
10 velocidade
11
12     // Desenha a bola na tela
13     void Draw() {
14         DrawCircle(x, y, radius, Orange);
15     }
16
17     void Update() {
18         if (!isPaused) {
```

```
            // Atualiza a posição da bola com base na velocidade
            x += speedX;
            y += speedY;
        }

        if (!isPaused) {
            frameCounter++;
            // Incrementa a velocidade da bola a cada 60 frames
            if (frameCounter >= 60) {
                speedX += (speedX > 0 ? speedIncrement : -
speedIncrement); // Ajusta direção
                speedY += (speedY > 0 ? speedIncrement : -
speedIncrement); // Ajusta direção
                frameCounter = 0;
            }
        }

        // Detecta e inverte a direção ao colidir com bordas
superior/inferior
        if (y + radius >= GetScreenHeight() || y - radius <= 0) {
            speedY *= -1;
        }

        // Verifica colisão com as bordas laterais e pontuações
        if (x + radius * -2 >= GetScreenWidth()) {
            // Incrementa a pontuação dependendo do modo de jogo
            if (gameIsRunning == 1) {
                cpuScore++;
            } else if (gameIsRunning == 2) {
                player2Score++;
            }
            resetBall(); // Reinicia a posição da bola
        }
        if (x - radius * -2 <= 0) {
            playerScore++; // Incrementa a pontuação do jogador
            resetBall(); // Reinicia a posição da bola
        }
    }

    // Reinicia a bola no centro com velocidades aleatórias
    void resetBall() {
        y = GetScreenHeight() / 2;
        x = GetScreenWidth() / 2;
        int speedChoices[2] = {-1, 1}; // Escolhas para direção
inicial
        speedX = 7.0f * speedChoices[GetRandomValue(0, 1)];
        speedY = 7.0f * speedChoices[GetRandomValue(0, 1)];
        frameCounter = 0; // Reseta o contador de frames
```

```
    }
};

Ball ball;

```

4.3.1.1.2. CLASSE PADDLE

A classe Paddle representa uma raquete. As raquetes possuem atributos de posição, largura, altura e velocidade, além de uma função que limita seu movimento dentro da tela.

ALGORITMO 2 – CLASSE PADDLE

```
class Paddle {
protected:
    void LimitMovement() {
        if (y <= 0) y = 0;
        if (y + height >= GetScreenHeight()) y = GetScreenHeight() -
height;
    }
public:
    float x, y;
    float width, height;
    int speed;

    void Draw() {
        DrawRectangleRounded(Rectangle{x, y, width, height}, 0.3, 0,
WHITE);
    }

    void Update() {
        if (!isPaused) {
            if (IsKeyDown(KEY_UP)) y -= speed;
            if (IsKeyDown(KEY_DOWN)) y += speed;
            LimitMovement();
        }
    }
};
Paddle player;
```

4.3.1.1.3. SUBCLASSE PLAYER2PADDLE

A classe player2Paddle é uma subclasse de Paddle e serve para controlar a raquete do segundo jogador. A movimentação é controlada pelas teclas W e S.

ALGORITMO 3 – SUBCLASSE PLAYER2PADDLE

```
1  class player2Paddle : public Paddle {
2      public:
3
4      void Update() {
5          if (!isPaused) {
6              if (IsKeyDown(KEY_W)) y -= speed;
7              if (IsKeyDown(KEY_S)) y += speed;
8              LimitMovement();
9          }
10     }
11 };
12
13 player2Paddle player2;
```

4.3.1.1.4. SUBCLASSE CPUPADDLE

A classe CpuPaddle também é uma subclasse de Paddle, mas sua movimentação é controlada pela lógica de previsão da posição da bola, permitindo que o CPU "tente" antecipar o movimento da bola.

ALGORITMO 4 – SUBCLASSE CPUPADDLE

```
1  class CpuPaddle : public Paddle {
2  public:
3      void Update(const Ball& ball) {
4
5          float futureY = ball.y + ball.speedY *
6  (std::abs(ball.speedX));
7          float marginOfError = (rand() % 10 - 5);
8          futureY += marginOfError;
9
10         float distance = futureY - (y + height / 2);
11         float adjustedSpeed = speed * (std::min(std::abs(distance) /
12 50.0f, 1.0f));
13
14
```

```
        if (!isPaused) {
            if (distance > 0) y += adjustedSpeed;
            else y -= adjustedSpeed;
        }
        LimitMovement();
    }
};

CpuPaddle cpu;
```

4.3.1.1.5. FUNÇÃO CHECKPADDLECOLLISION()

Essa função é responsável por verificar as colisões entre a bola e as raquetes. Se a bola colidir com a raquete, sua direção será invertida, simulando a interação física do jogo.

ALGORITMO 5 – FUNÇÃO CHECKPADDLECOLLISION()

```
void CheckPaddleCollision(Ball &ball, Paddle &paddle) {
    if (ball.x + ball.radius >= paddle.x && ball.x - ball.radius <=
paddle.x + paddle.width) {
        if (ball.y + ball.radius >= paddle.y && ball.y - ball.radius
<= paddle.y + paddle.height) {
            ball.speedX *= -1;
            if (ball.x < GetScreenWidth() / 2) {
                ball.x = paddle.x + paddle.width + ball.radius;
            } else {
                ball.x = paddle.x - ball.radius;
            }
        }
    }
}
```

4.3.1.1.6. FUNÇÃO PAUSE()

A função Pause() alterna o estado de pausa do jogo, permitindo ao jogador pausar ou retomar o jogo a qualquer momento.

ALGORITMO 6 – FUNÇÃO PAUSE()

```
void Pause() {
    isPaused = !isPaused;
}
```

4.3.1.1.7. FUNÇÃO DRAWPAUSEMENU()

A função DrawPauseMenu() desenha a interface do menu de pausa, informando ao jogador que o jogo foi pausado e permitindo a retomada do jogo.

ALGORITMO 7 – FUNÇÃO DRAWPAUSEMENU()

```
1  void DrawPauseMenu() {
2      DrawRectangle(0, 0, GetScreenWidth(), GetScreenHeight(), Color{0,
3  0, 0, 180});
4      DrawText("PAUSE", GetScreenWidth() / 2 - MeasureText("PAUSE", 40)
5  / 2, GetScreenHeight() / 2 - 20, 40, WHITE);
6  }
```

4.3.1.1.8. FUNÇÃO DRAWGAMEOVERSCREEN()

Essa função é responsável por desenhar a tela de "Game Over" quando um jogador atinge a pontuação máxima, informando quem foi o vencedor e fornecendo a opção de reiniciar o jogo ou voltar ao menu.

ALGORITMO 8 – FUNÇÃO DRAWGAMEOVERSCREEN()

```
1  void DrawGameOverScreen() {
2      isPaused = true;
3
4      DrawRectangle(0, 0, GetScreenWidth(), GetScreenHeight(), Color{0,
5  0, 0, 180});
6
7      if (playerScore >= maxScore){
8          DrawText("GAME OVER - PLAYER 1 WON!", GetScreenWidth() / 2 -
9  MeasureText("GAME OVER - PLAYER 1 WON!", 40) / 2, GetScreenHeight() /
10 2 - 60, 40, WHITE);
11     } else if (gameIsRunning == 1){
12         if (cpuScore >= maxScore){
13             DrawText("GAME OVER - CPU WON!", GetScreenWidth() / 2 -
14 MeasureText("GAME OVER - CPU WON!", 40) / 2, GetScreenHeight() / 2 -
15 60, 40, WHITE);
16         }
17     } else if (player2Score >= maxScore){
18             DrawText("GAME OVER - PLAYER 2 WON!", GetScreenWidth() /
19 2 - MeasureText("GAME OVER - PLAYER 2 WON!", 40) / 2,
20 GetScreenHeight() / 2 - 60, 40, WHITE);
21
```

```
    }
    DrawText("Pressione 'R' para Reiniciar", GetScreenWidth() / 2 -
MeasureText("Pressione 'R' para Reiniciar", 20) / 2,
GetScreenHeight() / 2, 20, WHITE);
    DrawText("Pressione 'M' para Voltar ao Menu", GetScreenWidth() /
2 - MeasureText("Pressione 'M' para Voltar ao Menu", 20) / 2,
GetScreenHeight() / 2 + 30, 20, WHITE);

}


```

4.3.1.1.9. FUNÇÃO RESETGAME()

A função ResetGame() reinicia as pontuações e o estado do jogo, preparando-o para uma nova rodada.

ALGORITMO 9 – FUNÇÃO DRAWGAMEOVERSCREEN()

```
void ResetGame() {
    playerScore = 0;
    cpuScore = 0;
    player2Score = 0;
    isPaused = false;
}

    } else if (player2Score >= maxScore){
            DrawText("GAME OVER - PLAYER 2 WON!", GetScreenWidth() /
2 - MeasureText("GAME OVER - PLAYER 2 WON!", 40) / 2,
GetScreenHeight() / 2 - 60, 40, WHITE);
    }
    DrawText("Pressione 'R' para Reiniciar", GetScreenWidth() / 2 -
MeasureText("Pressione 'R' para Reiniciar", 20) / 2,
GetScreenHeight() / 2, 20, WHITE);
    DrawText("Pressione 'M' para Voltar ao Menu", GetScreenWidth() /
2 - MeasureText("Pressione 'M' para Voltar ao Menu", 20) / 2,
GetScreenHeight() / 2 + 30, 20, WHITE);

}

```

4.3.1.2. FLUXO DE EXECUÇÃO

O fluxo de execução principal do jogo ocorre dentro da função main(), que contém o loop principal do jogo. Este loop é responsável por:

I. Inicializar a janela e as configurações do jogo (como as posições iniciais da bola e das raquetes).
II. Detectar a entrada do usuário (movimento das raquetes e comandos de pausa).
III. Atualizar as posições da bola e das raquetes a cada quadro.
IV. Verificar as colisões entre a bola e as raquetes.
V. Desenhar os elementos do jogo na tela.
VI. Exibir os menus de pausa e "Game Over" conforme o estado do jogo.
VII. O loop termina quando o jogador decide fechar a janela do jogo, e a função CloseWindow() é chamada para liberar os recursos.

ALGORITMO 10 – FUNÇÃO PRINCIPAL MAIN()

```
1  int main(void) {
2      const int widthScreen = 1280;
3      const int heightScreen = 720;
4
5      InitWindow(widthScreen, heightScreen, "Pong");
6      SetTargetFPS(60);
7
8
9
10     ball.radius = 10;
11     ball.x = widthScreen / 2;
12     ball.y = heightScreen / 2;
13     ball.speedX = 7.0;
14     ball.speedY = 7.0;
15
16     player.height = 120.0;
17     player.width = 25;
18     player.x = widthScreen - player.width - 20;
19     player.y = heightScreen / 2 - player.height / 2;
20     player.speed = 6.0;
21
22     player2.height = 120.0;
23     player2.width = 25;
24     player2.x = 20;
       player2.y = heightScreen / 2 - cpu.height / 2;
```

```
    player2.speed = 6.0;

    cpu.height = 120.0;
    cpu.width = 25;
    cpu.x = 20;
    cpu.y = heightScreen / 2 - cpu.height / 2;
    cpu.speed = 6.0;



    while (WindowShouldClose() == false) {
        if (gameIsRunning == 0) {
            gameIsRunning = Menu(widthScreen, heightScreen,
gameIsRunning);
        } else if (gameIsRunning == 1 || gameIsRunning == 2) {

            if (IsKeyPressed('P')) Pause();

            BeginDrawing();
            ball.Update();
            player.Update();
            if (gameIsRunning == 2){
                player2.Update();
            }
            if (gameIsRunning == 1){
                cpu.Update(ball);
            }
            CheckPaddleCollision(ball, player);
            CheckPaddleCollision(ball, player2);
            if (gameIsRunning == 1){
                CheckPaddleCollision(ball, cpu);
            }

            ClearBackground(DarkBlue);
            DrawRectangle(widthScreen / 2, 0, widthScreen / 2,
heightScreen, Blue);
            DrawCircle(widthScreen / 2, heightScreen / 2, 150,
LightBlue);
            DrawLine(widthScreen / 2, 0, widthScreen / 2,
heightScreen, WHITE);
            ball.Draw();
            if (gameIsRunning == 1){
                cpu.Draw();
            }
            player.Draw();
            if (gameIsRunning == 2){
            player2.Draw();
```

```
            }
            if (gameIsRunning == 1){
                DrawText(TextFormat("%i", cpuScore), widthScreen / 4,
20, 80, WHITE);
            }
            if (gameIsRunning == 2){
                DrawText(TextFormat("%i", player2Score), widthScreen
/ 4, 20, 80, WHITE);
            }
            DrawText(TextFormat("%i", playerScore), 3 * widthScreen /
4, 20, 80, WHITE);
            DrawText("Pressione \"P\" para Pausar", 10, heightScreen
- 90, 20, Color{255, 255, 255, 153});
            DrawText("Pressione \"M\" para voltar ao Menu", 10,
heightScreen - 65, 20, Color{255, 255, 255, 153});
            DrawText("Pressione \"ESC\" para fechar o Jogo", 10,
heightScreen - 40, 20, Color{255, 255, 255, 153});


            if ((isPaused) && (playerScore < maxScore && cpuScore <
maxScore && player2Score < maxScore)) {
                DrawPauseMenu();
            }

            if (IsKeyPressed('M')) {
                gameIsRunning = 0;
                ResetGame();
            }

            // Checa se algum jogador alcançou a pontuação máxima
            if (playerScore >= maxScore || cpuScore >= maxScore ||
player2Score >= maxScore) {
                DrawGameOverScreen();
                if (IsKeyPressed('R')) {
                    ResetGame(); // Reinicia o jogo
                }
                if (IsKeyPressed('M')) {
                    gameIsRunning = 0; // Volta ao menu
                    ResetGame();
                }
            }

            EndDrawing();
        } else if (gameIsRunning == 3) {
            break;
        }
```

```
119     }
120
121     CloseWindow(); // Fecha a janela
123     return 0;
124 }
125
126
```

4.3.1.3. LÓGICA DE JOGO

O jogo segue as regras clássicas do Pong, onde dois jogadores ou um jogador e o CPU competem para atingir uma pontuação máxima (definida pela variável maxScore). O fluxo do jogo pode ser resumido da seguinte forma:

I. O jogo começa com o menu principal.
II. O jogador pode escolher entre jogar contra o CPU (modo 1) ou jogar contra outro jogador (modo 2).
III. O jogo prossegue até que um dos jogadores atinja a pontuação máxima, momento em que a tela de "Game Over" é exibida.
IV. O jogador pode então escolher reiniciar o jogo ou retornar ao menu principal.

5. AVALIAÇÃO

Nesta seção, serão apresentados os resultados do desenvolvimento do jogo Pong, bem como uma discussão sobre a eficiência e a funcionalidade do código. A avaliação se concentra no processo de desenvolvimento e nos testes realizados para garantir que o jogo estivesse funcional, interativo e livre de erros. Além disso, abordaremos a análise de desempenho e possíveis melhorias.

5.1. CONDUÇÃO

A condução da avaliação do jogo Pong seguiu uma abordagem estruturada, com etapas claras para garantir que todos os componentes do jogo estivessem funcionando conforme esperado. O objetivo foi testar a mecânica do jogo, a

interação entre os elementos, como as raquetes e a bola, e a usabilidade das funcionalidades implementadas. Ao longo do desenvolvimento, a ênfase foi dada à implementação das funcionalidades principais, como o movimento das raquetes controladas pelos jogadores, a movimentação da bola de acordo com as colisões, e o sistema de pontuação que registrava os pontos de cada jogador.

Após a implementação inicial do código, foi realizada uma série de testes de funcionalidade para verificar se os componentes do jogo estavam funcionando corretamente. Esses testes foram focados principalmente no movimento das raquetes, na resposta da bola a essas colisões, e na correta atualização do sistema de pontuação. Além disso, foi testada a funcionalidade de pausa e a condição de "Game Over", a fim de garantir que o jogo pudesse ser interrompido e reiniciado conforme necessário.

Seguindo para a fase de testes de estabilidade, o jogo foi executado por longos períodos de tempo, verificando o desempenho e a fluidez das interações, principalmente entre o jogador e a bola, a fim de garantir que o jogo se comportasse de forma estável sem travamentos ou falhas. Para isso, foram coletados dados sobre o tempo de resposta do jogo em diferentes dispositivos, com foco na taxa de quadros por segundo e na estabilidade do sistema. Esse teste de performance foi crucial para identificar possíveis pontos de lentidão ou falhas de renderização que pudessem prejudicar a experiência do jogador.

Além disso, para garantir que o jogo fosse intuitivo e agradável de jogar, foi realizado um teste com usuários. O feedback obtido dos jogadores foi essencial para identificar áreas de melhoria, especialmente no que diz respeito à inteligência artificial do CPU, que inicialmente apresentava uma resposta previsível e fácil de superar. A partir desse feedback, ajustamos a física da bola e a resposta das raquetes para tornar a jogabilidade mais equilibrada, incluindo a adição de um modo "Jogador vs. Jogador" e ajustes na movimentação do CPU.

Os dados coletados durante a avaliação foram diversos, incluindo logs de performance, como a taxa de quadros por segundo e o tempo de resposta das interações com a bola e as raquetes. Além disso, foram registradas as falhas encontradas, como pequenas quebras na física da bola e problemas ocasionais de colisão. Também coletamos as opiniões dos jogadores, que forneceram

informações valiosas sobre a dificuldade do jogo e a clareza dos menus e das opções de controle.

Após a realização dos testes e coleta de dados, ajustes finais foram feitos no código para corrigir os erros encontrados, aprimorar o comportamento da inteligência artificial do CPU, melhorar a detecção de colisões e refinar a resposta do movimento das raquetes. Com isso, o jogo foi otimizado para uma experiência mais fluida e agradável.

Na subseção seguinte, serão apresentados os resultados obtidos nos testes realizados, seguidos de uma análise detalhada sobre o desempenho do jogo e sugestões para aprimoramentos futuros.

5.2. RESULTADOS

Nesta seção, serão apresentados os resultados obtidos a partir dos testes realizados durante o desenvolvimento do jogo Pong. Os resultados destacam o comportamento das funcionalidades implementadas, como a movimentação das raquetes, a física da bola, o sistema de pontuação, e a inteligência artificial do CPU. Além disso, discutiremos os problemas identificados, como um pequeno lag no movimento da bola, que ocorre em momentos específicos, mas não interfere de forma significativa na jogabilidade.

O código do jogo foi implementado com sucesso, e todas as funções essenciais, como a movimentação das raquetes, a detecção de colisões e o sistema de pontuação, funcionaram corretamente durante os testes. A física da bola, incluindo o aumento gradual da velocidade e a inversão da direção em colisões com as raquetes ou as bordas da tela, também foi validada e se comportou de acordo com o esperado.

5.3. FUNCIONALIDADES DO JOGO

A funcionalidade principal do jogo foi testada em diferentes modos: Jogador x CPU e Jogador x Jogador. Durante os testes, o movimento das raquetes foi preciso, com os jogadores controlando suas raquetes de acordo com as teclas

designadas, como as teclas W e S para o jogador 2, e as setas para cima e para baixo para o jogador 1. A movimentação da bola também funcionou conforme esperado, e o comportamento de colisão foi bem implementado, permitindo que a bola se rebatasse corretamente nas raquetes e nas bordas da tela. A pontuação foi corretamente atualizada a cada ponto marcado, e o jogo foi corretamente finalizado quando um jogador alcançou a pontuação máxima, com a tela de "Game Over" sendo exibida para indicar o vencedor.

5.4. ANÁLISE DE PERFORMANCE

Embora o jogo tenha funcionado corretamente na maior parte do tempo, foi identificado um pequeno problema de desempenho em algumas situações. Durante certos momentos de movimentação rápida da bola, o movimento parecia sofrer um pequeno atraso, o que resultava em um comportamento perceptível onde a bola parecia "voltar" à posição anterior por um milésimo de segundo, antes de avançar novamente. Esse efeito foi observado especialmente quando a bola estava se movendo rapidamente e o número de quadros por segundo (FPS) diminui temporariamente, sugerindo que o problema está relacionado ao processamento da renderização da tela em conjunto com a física do movimento da bola. No entanto, esse atraso não compromete a jogabilidade de forma significativa, mas é perceptível em algumas situações. A principal causa provável desse efeito é a maneira como o cálculo da física da bola interage com o controle de frames, o que pode ser melhorado por meio de otimizações no código.

5.5. AJUSTES REALIZADOS

Durante o processo de avaliação, alguns ajustes foram feitos no código para corrigir pequenos problemas de jogabilidade. A inteligência artificial (IA) do CPU, que inicialmente era previsível e fácil de superar, foi ajustada para reagir de maneira mais desafiadora e realista. Além disso, melhorias foram feitas na detecção de colisões, com ajustes nos parâmetros de interação entre a bola e as raquetes para garantir um comportamento mais natural. O código também foi otimizado para reduzir o impacto de qualquer possível lag no movimento da bola,

embora o problema de desempenho mencionado ainda persista de forma intermitente

5.6. CONCLUSÃO DOS RESULTADOS

De modo geral, os resultados obtidos são positivos. A maior parte das funcionalidades do jogo foi implementada com sucesso, e o jogo está funcional e jogável em sua totalidade. O problema de lag no movimento da bola, embora perceptível em algumas situações, não prejudica a experiência de jogo de maneira significativa. Em termos de jogabilidade, as funções de controle de raquetes, a física da bola, o sistema de pontuação, e os modos de jogo estão funcionando conforme o esperado. O feedback dos testes realizados, tanto os de funcionalidade quanto os de performance, sugerem que o jogo é estável, mas ainda há margem para melhorias em relação à otimização de desempenho, especialmente em momentos de alta movimentação.

6. DISCUSSÃO

Nesta seção, discutiremos os resultados obtidos no estudo, comparando-os com os objetivos e hipóteses estabelecidos na introdução. A partir dessa análise, vamos refletir sobre o desempenho do jogo, as questões de desempenho identificadas, e como essas descobertas contribuem para nosso entendimento sobre o processo de desenvolvimento de jogos simples, como o Pong, e as questões de otimização e inteligência artificial em jogos desse tipo.

Os resultados do estudo confirmam a hipótese inicial de que a implementação básica do jogo Pong, com funcionalidades essenciais como movimentação das raquetes, colisões da bola e sistema de pontuação, pode ser realizada de maneira eficaz com o uso da biblioteca Raylib. No entanto, como discutido na seção anterior, observamos um pequeno problema de desempenho relacionado ao movimento da bola, que apresenta um atraso perceptível em alguns momentos, o que compromete parcialmente a fluidez do jogo. Esse achado, apesar de não ter afetado de maneira significativa a jogabilidade, sugere que a implementação da física da bola e o controle de frames precisam de ajustes

para otimizar o desempenho. Essa conclusão é consistente com a literatura sobre desenvolvimento de jogos em 2D, onde é amplamente discutido que a otimização de renderização e cálculos de física são fatores críticos para garantir uma experiência de jogo suave, especialmente em jogos com movimento rápido e constante.

Em relação à inteligência artificial (IA) do CPU, o estudo trouxe resultados positivos. A IA foi capaz de se adaptar ao movimento da bola, tornando o jogo desafiador para o jogador. Durante os testes, o comportamento da IA se mostrou mais realista e difícil de superar após os ajustes realizados no código. No entanto, ao comparar os resultados com outras implementações de IA em jogos Pong, pode-se observar que a IA do nosso jogo ainda pode ser aprimorada, especialmente no que se refere à aleatoriedade de suas respostas. A IA testada no jogo, por mais que tenha sido ajustada para reagir de maneira mais humana, ainda pode ser previsível em determinadas situações. Essa limitação é um ponto a ser investigado em futuras versões do jogo, como sugerido por pesquisas sobre IA em jogos retro, onde a aleatoriedade nas ações do CPU pode melhorar ainda mais a experiência do jogador, evitando que o jogo se torne excessivamente previsível.

Além disso, o pequeno problema de lag na movimentação da bola, mencionado nos resultados, sugere que uma das áreas que precisa ser mais investigada é o controle de quadros por segundo (FPS). A interação entre a renderização da tela e o cálculo da física da bola pode estar causando um leve atraso na atualização da posição da bola. A literatura sobre o desenvolvimento de jogos e otimização de desempenho sugere que uma melhor sincronização entre a taxa de quadros e o cálculo de física pode resolver esse tipo de problema. Estudos semelhantes sobre o desenvolvimento de jogos simples, como Pong, indicam que a utilização de técnicas como frame limiting e otimizações específicas no cálculo da física podem minimizar esses efeitos de lag.

Com relação aos próximos passos da investigação, será necessário focar na otimização da física do jogo e no desempenho geral. A questão do lag será uma das prioridades, com a busca por soluções que melhorem a sincronização entre o cálculo da física e a renderização da tela. Além disso, a inteligência artificial do CPU deve ser mais trabalhada, introduzindo maior complexidade nos seus movimentos e respostas. Será interessante explorar técnicas de IA mais

avançadas, como redes neurais simples ou algoritmos genéticos, que podem trazer uma imprevisibilidade maior à IA e melhorar a experiência de jogo.

Em termos de futuras melhorias, a implementação de novos modos de jogo, como o modo multijogador online, também pode ser considerada. Isso adicionaria uma nova camada de complexidade ao jogo, exigindo a implementação de comunicação em rede e sincronia entre os jogadores, além de melhorias no código de inteligência artificial. Outra possibilidade seria expandir o jogo para suportar diferentes níveis de dificuldade, onde a IA poderia ser ajustada automaticamente conforme o desempenho do jogador.

Em conclusão, os resultados do estudo mostraram que o desenvolvimento do jogo Pong, com as funcionalidades esperadas, foi bem-sucedido. No entanto, o estudo também identificou áreas de melhoria, principalmente no que se refere à otimização do desempenho e à complexidade da IA. Esses resultados indicam que há um caminho promissor para melhorar o jogo, aprimorando a experiência do jogador e expandindo as possibilidades do jogo em versões futuras.

7. CONCLUSÃO

Este trabalho teve como objetivo o desenvolvimento de um jogo Pong simples, utilizando a linguagem de programação C++ e a biblioteca Raylib, com a implementação de funcionalidades essenciais como a movimentação das raquetes, a colisão da bola, o sistema de pontuação, além de testar a inteligência artificial para o comportamento da raquete do CPU e o desempenho do jogo. Além disso, o estudo teve como hipóteses que seria possível desenvolver um jogo funcional e otimizado dentro desses parâmetros, e que a IA do CPU poderia ser ajustada para fornecer um desafio equilibrado e realista ao jogador.

Ao longo do trabalho, conseguimos atingir todos os objetivos estabelecidos inicialmente. A implementação do jogo foi bem-sucedida, com as funcionalidades principais operando de maneira satisfatória. As raquetes se movem conforme o esperado, a bola realiza as colisões com as raquetes e com as bordas da tela, e

o sistema de pontuação atualiza corretamente a cada ponto marcado. A inteligência artificial, embora simples, foi configurada de forma a oferecer um nível de desafio interessante, reagindo de maneira adequada ao movimento da bola e criando uma experiência de jogo envolvente.

No entanto, um pequeno problema de desempenho foi identificado durante os testes, relacionado ao movimento da bola, que apresentou um leve lag perceptível em alguns momentos. Esse achado foi consistente com o esperado em projetos de jogos simples, onde a sincronização entre a física e a renderização precisa de ajustes finos. Acredita-se que o problema pode ser resolvido com melhorias no controle de quadros por segundo (FPS) e na otimização dos cálculos de física.

Com relação às atividades previstas no cronograma, a maior parte dos marcos foi cumprida conforme o planejado. O desenvolvimento do jogo, as implementações das funcionalidades e os testes de IA foram realizados dentro do prazo. No entanto, a análise mais aprofundada do desempenho e as otimizações para eliminar o problema de lag na movimentação da bola não puderam ser totalmente concluídas, sendo necessário um maior tempo para ajustes e experimentações. Este é um ponto que deve ser explorado em trabalhos futuros, com o objetivo de alcançar um desempenho ideal para o jogo.

Em relação às hipóteses, pode-se concluir que a premissa de que seria possível desenvolver um jogo funcional com a biblioteca Raylib foi confirmada, assim como a hipótese de que a IA do CPU pode ser configurada para apresentar um desafio realista ao jogador. Contudo, os resultados também indicam que a IA pode ser aprimorada, particularmente na imprevisibilidade de suas ações, o que é um aspecto que merece mais investigação em versões futuras do jogo.

Em síntese, este trabalho cumpriu os objetivos estabelecidos, alcançou a maior parte das metas propostas e gerou uma compreensão mais profunda sobre os desafios do desenvolvimento de jogos simples. O jogo Pong está funcional e oferece uma experiência satisfatória ao jogador, mas há áreas de melhoria, como a otimização do desempenho e a evolução da inteligência artificial. A pesquisa

realizada e os resultados obtidos abriram caminhos para futuras melhorias e novos experimentos no desenvolvimento de jogos em C++ e na aplicação de técnicas de IA simples em jogos retro.

8. REFERÊNCIAS

FORBELLONE, André Luiz Villar. **Lógica de Programação: A Construção de Algoritmos e Estruturas de Dados**. 2ª ed. São Paulo: Pearson Prentice Hall, 2005.

STROUSTRUP, Bjarne. **The C++ Programming Language**. 4ª ed. Boston: Addison-Wesley, 2013.

GITHUB. **Exemplos de projetos de jogos baseados em Raylib e C++**. Disponível em: <https://github.com/raysan5/raylib-games/tree/master>. Acesso em: 29 out 2024.

RAYLIB. **Cheatsheet.** Disponível em: < https://www.raylib.com/cheatsheet/cheatsheet.html#pcore>. Acesso em: 29 out 2024.

YOUTUBE. **Pong Game with C++ and Raylib - Beginner Tutorial**. Disponível em: <https://www.youtube.com/watch?v=VLJlTaFvHo4>. Acesso em: 15 nov 2024.

9. GLOSSÁRIO

Biblioteca Raylib: Uma biblioteca de código aberto para desenvolvimento de jogos e gráficos 2D/3D, escrita em C, que oferece uma interface simples e intuitiva para iniciantes e profissionais no desenvolvimento de jogos.

Colisão: Em desenvolvimento de jogos, refere-se à detecção de interações entre objetos (como uma bola e uma raquete), sendo fundamental para implementar física e lógica de jogo.

FPS (Frames Por Segundo): Unidade de medida que define a taxa de atualização de quadros por segundo de um jogo ou animação. Impacta diretamente a fluidez e o desempenho visual do jogo.

Inteligência Artificial (IA): No contexto do projeto, representa um algoritmo programado para prever o movimento da bola e ajustar a posição da raquete controlada pelo CPU, criando um comportamento desafiador para o jogador.

Jogo Pong: Um dos primeiros jogos eletrônicos, que simula uma partida de tênis de mesa, envolvendo duas raquetes e uma bola em movimento. Tornou-se um clássico no desenvolvimento de jogos.

Lag: Um atraso ou lentidão perceptível no desempenho do jogo, que pode ocorrer devido a problemas de processamento ou sincronização na atualização de frames.

Modo Jogador x Jogador (PvP): Funcionalidade que permite a participação de dois jogadores no jogo, cada um controlando uma raquete de forma independente, promovendo interação competitiva direta.

Raquete (Paddle): Elemento do jogo Pong controlado por um jogador ou pela IA, utilizado para rebater a bola e impedir que ela ultrapasse o limite da tela.

Sistema de Pontuação: Mecanismo implementado para registrar e exibir o número de pontos obtidos pelos jogadores ao longo da partida, determinando o vencedor do jogo.

Tela de Pausa: Um elemento de interface que permite ao jogador interromper o jogo temporariamente, oferecendo opções como continuar, reiniciar ou sair.

Velocidade da Bola: Propriedade ajustada para definir a rapidez com que a bola se move na tela, influenciando o ritmo e a dificuldade do jogo.