**INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE SÃO PAULO**

**Marcos Benner Diniz Moreira**

# SNAKE GAME UTILIZANDO A BIBLIOTECA RAYLIB

**CAMPOS DO JORDÃO**

**2024**

# RESUMO

Neste trabalho foi desenvolvido um jogo 2D chamado Snake Game (Jogo da cobra), cujo objetivo é conduzir uma cobra a se alimentar em um ambiente representado por uma grade, sem colidir com as bordas ou com seu próprio corpo. Este jogo foi desenvolvido com base em princípios de programação gráfica e lógica, utilizando a biblioteca Raylib para criar uma experiência visual simples e funcional. O jogador controla a direção da cobra usando as teclas direcionais do teclado, guiando-a para coletar alimentos que aparecem aleatoriamente na tela. Cada vez que a cobra consome um alimento, ela cresce em tamanho, aumentando a dificuldade e desafiando o jogador a planejar suas movimentações com precisão. Espera-se que o Jogo da Cobrinha proporcione uma experiência lúdica, nostálgica e desafiadora, ao mesmo tempo em que introduz conceitos de lógica de programação e manipulação de estruturas de dados como filas (deque). Além disso, o projeto serviu como uma excelente porta de entrada para o aprendizado de desenvolvimento de jogos com gráficos simples, mas envolventes, oferecendo tanto entretenimento quanto aprendizado técnico.

**Palavras-Chave**: Snake Game, Raylib, Programação Orientada a Objetos.

# ABSTRACT

In this work, a 2D game called Snake Game was developed, the objective of which is to guide a snake to feed in an environment represented by a grid, without colliding with the edges or its own body. This game was developed based on graphical and logical programming principles, using the Raylib library to create a simple and functional visual experience. The player controls the snake's direction using the directional keys on the keyboard, guiding it to collect food that appears randomly on the screen. Each time the snake consumes food, it grows in size, increasing the difficulty and challenging the player to plan their movements with precision. The Snake Game is expected to provide a playful, nostalgic and challenging experience, while introducing concepts of programming logic and manipulation of data structures such as queues (deque). Furthermore, the project served as an excellent gateway to learning game development with simple but engaging graphics, offering both entertainment and technical learning.



**Keywords**: Snake Game, Raylib, Object Oriented Programming.

# 1 INTRODUÇÃO

Os jogos em video são uma porta de entrada para o mundo da tecnologia estando entre os meios mais diretos de acesso de crianças e jovens. A maioria das crianças ocidentais brincam com consoles de videogames e seu primeiro contato com computadores é por meio de um jogo de computador (GROS, 2003). Os jogos sempre fizeram parte da vida do ser humano, não apenas na infância mas por todas as fases da vida.

Por suas características, os jogos podem possuir aplicações eficientes no processo de ensino e aprendizagem, pois eles são motivadores e divertidos ao mesmo tempo, facilitando o aprendizado e retenção daquilo que foi aprendido, exercitando as funções mentais e intelectuais daquele que o joga. Além disso, para o desenvolvimento de um jogo faz-se necessário conhecimentos de programação tanto estrutural quanto orientada a objetos,

Neste contexto, o trabalho visa desenvolver um jogo em 2D utilizando a biblioteca Raylib, onde o jogador controla uma cobra em um ambiente bidimensional, com o objetivo de coletar alimentos, evitando colisões consigo mesma e com as bordas da tela. Durante o desenvolvimento do jogo, foram aplicados diversos conceitos da programação orientada a objetos (POO), como encapsulamento, através da definição de classes para a cobra, comida e o jogo em si; herança, ao estruturar e organizar funcionalidades comuns; e polimorfismo, ao sobrepor métodos para criar comportamentos específicos. Essa abordagem não apenas facilitou a organização do código, mas também permitiu uma melhor modularidade e reutilização dos componentes, elementos fundamentais no desenvolvimento de aplicações escaláveis. Além de oferecer uma experiência lúdica e nostálgica, o projeto serve como uma ferramenta educacional que alia aprendizado técnico com a criação de uma aplicação funcional e interativa.

## 2 - OBJETIVO PRINCIPAL

Desenvolver um jogo 2D no estilo clássico da cobrinha, utilizando a biblioteca Raylib, para aplicar e consolidar conceitos de programação orientada a objetos e técnicas de desenvolvimento de jogos, promovendo aprendizado técnico e habilidades práticas em lógica de programação.

### 2.1 - Objetivos Específicos

Desenvolver a estrutura do jogo utilizando POO, Implementar classes que representam os elementos fundamentais do jogo, como a *cobra*, a *comida* e a lógica geral do jogo aplicar  os seguintes os pilares:

- **Encapsulamento:** Proteger os dados internos das classes, permitindo acesso e manipulação apenas através de métodos públicos.
- **Abstração:** Simplificar a lógica do jogo ao encapsular detalhes complexos em métodos reutilizáveis.
- **Composição:** Integrar as classes de forma que trabalhem harmonicamente, como a interação entre a cobra e a comida.

Deste modo promovendo a modularidade no código para facilitar a manutenção e expansão do jogo em projetos futuros.

Além de melhorar  a experiência de desenvolvimento e aprendizado técnico ao utilizar a biblioteca gráfica Raylib para renderizar elementos visuais, como a cobra, a comida e o tabuleiro, criando um ambiente estético e funcional. Assim consolidando conhecimentos em estruturas de dados aplicadas, como o uso do deque para manipular a lógica do corpo da cobra.

# 3 - FUNDAMENTAÇÃO TEÓRICA

Neste capítulo são apresentados com maior detalhamento, os conceitos utilizados durante a criação deste trabalho, assim como as técnicas utilizadas para alcançar o objetivo proposto.

## 3.1 - Video Game

Os videogames são uma das formas de entretenimento mais populares e acessíveis na sociedade contemporânea, atraindo públicos de diferentes faixas etárias e contextos sociais. Além de sua função lúdica, os videogames são plataformas interativas que estimulam habilidades cognitivas, como raciocínio lógico, criatividade e resolução de problemas. Com o avanço da tecnologia, os jogos deixaram de ser apenas ferramentas de diversão, transformando-se em ferramentas educacionais, culturais e sociais. Eles possibilitam simulações realistas, exploram narrativas imersivas e oferecem experiências interativas que facilitam a aquisição de novos conhecimentos, tornando-se cada vez mais relevantes tanto no contexto de lazer quanto no de aprendizado.

## 3.2 - Programação Orientada a Objetos(POO)

A POO é um paradigma de programação que organiza o código em torno de "objetos", que são instâncias de classes que agrupam dados e comportamentos relacionados. Este paradigma foi amplamente utilizado para estruturar o jogo da cobrinha, garantindo modularidade, reutilização e clareza no código.

- **Encapsulamento:** Cada classe criada (como *Cobra* e *Comida*) possui atributos e métodos que protegem seus dados internos, permitindo interações seguras e controladas. Por exemplo, o tamanho e a posição da cobra são manipulados apenas através de métodos da classe *Cobra*.
- **Abstração:** Elementos complexos, como a movimentação da cobra e a geração de comida em locais aleatórios, foram abstraídos em métodos de fácil reutilização, como `movimenta()` e `aleatorio()`, tais elementos serão apresentados na sessão criação do jogo.
- **Composição:** As classes trabalham juntas para construir a lógica do jogo. Por exemplo, a classe *Jogo* integra as classes *Cobra* e *Comida* para gerenciar as interações entre elas.
- **Polimorfismo:** Apesar de pouco explorado neste projeto, abre a possibilidade para evoluções futuras, como heranças ou especializações de comportamento.

## 4.0 - METODOLOGIA

O presente trabalho utilizou uma abordagem prática e teórica para o desenvolvimento do jogo Snake Game. Na parte teórica, foram explorados conceitos fundamentais de programação orientada a objetos, como encapsulamento, herança e uso de classes, bem como fundamentos relacionados à criação de jogos digitais. Na parte prática, foi implementado o jogo utilizando a biblioteca gráfica Raylib, conhecida por sua simplicidade e eficiência na criação de interfaces 2D. Durante o processo, foram consultadas a documentação oficial da Raylib, materiais acadêmicos relacionados ao desenvolvimento de jogos e tutoriais no YouTube, que auxiliaram no entendimento das funcionalidades da biblioteca e na superação de desafios técnicos durante a implementação.

## 5.0 - CRIAÇÃO DO JOGO

Capítulo  no qual, será apresentada as principais etapas do desenvolvimento do jogo, detalhando-as desde a concepção inicial até a implementação final. Serão discutidos o planejamento da lógica do jogo, a estruturação do código em classes e métodos, e a utilização da biblioteca gráfica Raylib para renderização e controle de eventos, bem como a integração de elementos gráficos e mecânicas interativas que fazem com que os conceitos ensinados em sala de aula fossem aplicados.

### 5.1 - Classe Cobra

A classe Cobra representa a entidade principal do jogo, responsável pela mecânica de movimentação, crescimento e desenho visual da cobra na tela. Essa classe utiliza uma estrutura dinâmica, a *deque,* para armazenar os segmentos do

corpo da cobra como uma sequência de posições bidimensionais no espaço da grade. Cada posição é representada por um objeto do tipo Vector2, que guarda as coordenadas x e y. Além disso, a classe gerencia a direção de movimento da cobra por meio de outro Vector2, permitindo atualizações contínuas durante o jogo. Métodos como “desenhar” e “movimenta” garantem que a cobra seja corretamente exibida e movimentada na tela. O método “crescer”, por sua vez, adiciona um novo segmento ao corpo da cobra, aumentando seu tamanho sempre que ela consome uma comida. Essa classe encapsula toda a lógica associada à cobra, promovendo uma implementação modular e de fácil manutenção.

```
// Classe que representa a Cobra
class Cobra {
public:
    // Corpo da cobra representado como uma lista de segmentos
    deque<Vector2> corpo = {Vector2{6, 9}, Vector2{5, 9}, Vector2{4, 9}};
    // Direção atual da cobra (1, 0) significa que está indo para a direita
    Vector2 direcao{1, 0};

    // Desenha a cobra na tela
    void desenhar() {
        for (unsigned int i = 0; i < corpo.size(); i++) {
            float x = corpo[i].x;
            float y = corpo[i].y;
            Rectangle pedaco = Rectangle{x * tamanhoCelulas, y * tamanhoCelulas, (float)tamanhoCelulas, (float)tamanhoCelulas};
            DrawRectangleRounded(pedaco, 0.5, 6, darkBlue); // Desenha segmentos arredondados
        }
    }

    // Move a cobra na direção atual
    void movimenta() {
        corpo.pop_back(); // Remove o último segmento (cauda)
        corpo.push_front(Vector2Add(corpo[0], direcao)); // Adiciona um novo segmento na cabeça
    }

    // Faz a cobra crescer adicionando um segmento na cauda
    void crescer() {
        corpo.push_back(corpo.back()); // Duplica o último segmento para simular crescimento
    }
};
```

### 5.2 - Classe Comida

A classe Comida é responsável por gerenciar a lógica associada ao alimento que a cobra deve consumir para crescer e pontuar no jogo. A posição da comida é representada por um objeto do tipo Vector2, que indica as coordenadas x e y dentro da grade do jogo. Inicialmente, a posição da comida é definida de maneira aleatória utilizando o método “aleatorio”, que garante que a comida seja colocada em uma célula válida da grade. O método desenhar é responsável por exibir visualmente a comida na tela, representando-a como um pequeno retângulo colorido. Sempre que

a cobra consome a comida, uma nova posição é gerada aleatoriamente, mantendo o desafio e a dinâmica do jogo. Assim, a classe Comida desempenha um papel fundamental na progressão do jogo, proporcionando o elemento que impulsiona o crescimento da cobra e a evolução da pontuação.

```
// Classe que representa a Comida
class Comida {
public:
    Vector2 position; // Posição da comida no grid

    // Construtor que posiciona a comida aleatoriamente
    Comida() {
        position = aleatorio();
    }

    // Desenha a comida na tela
    void desenhar() {
        DrawRectangle(position.x * tamanhoCelulas, position.y * tamanhoCelulas, tamanhoCelulas, tamanhoCelulas, darkBlue);
    }

    // Gera uma nova posição aleatória para a comida
    Vector2 aleatorio() {
        float x = GetRandomValue(0, contagemCelulas - 1);
        float y = GetRandomValue(0, contagemCelulas - 1);
        return Vector2{x, y};
    }
};
```

### 5.3 Classe Jogo

A classe Jogo é a responsável por integrar todas as funcionalidades do jogo da cobrinha, funcionando como um controlador central que organiza e coordena as interações entre a cobra, a comida e o ambiente. Essa classe contém instâncias das classes Cobra e Comida, além de métodos que gerenciam o fluxo do jogo, como o desenho dos elementos na tela, a movimentação da cobra, e a verificação de colisões.Os métodos de verificação de colisão são essenciais para manter as regras do jogo, garantindo que a cobra reaja ao consumir a comida (crescendo e aumentando a pontuação), e encerrando o jogo caso a cobra colida com as bordas ou com seu próprio corpo. O método “desenhaPontuacao” exibe a pontuação atual, fornecendo feedback visual ao jogador sobre seu desempenho.Essa classe centraliza a lógica principal do jogo, organizando a interação entre os componentes e garantindo que todas as ações sejam executadas no momento certo, permitindo uma ótima experiência de jogo.

```
// Classe principal do jogo
class Jogo {
public:
    Cobra cobra = Cobra();  // Instância da cobra
    Comida comida = Comida();  // Instância da comida

    // Desenha todos os elementos do jogo
    void desenhar() {
        comida.desenhar();    // Desenha a comida
        cobra.desenhar();     // Desenha a cobra
        desenhaPontuacao();   // Desenha a pontuação na tela
    }

    // Movimenta a cobra
    void movimentar() {
        cobra.movimenta();
    }

    // Verifica colisões entre a cobra e a co
    void verificaColisao() {
        if (CheckCollisionRecs(
                Rectangle{cobra.corpo[0].x * tamanhoCelulas, cobra.corpo[0].y * tamanhoCelulas, tamanhoCelulas, tamanhoCelulas},
                Rectangle{comida.position.x * tamanhoCelulas, comida.position.y * tamanhoCelulas, tamanhoCelulas, tamanhoCelulas})) {
            comida.position = comida.aleatorio(); // Reposiciona a comida
            cobra.crescer();                      // Faz a cobra crescer
            pontuacao++;                          // Incrementa a pontuação
        }
    }
```

int tamanhoCelulas

Tamanho de cada célula no grid

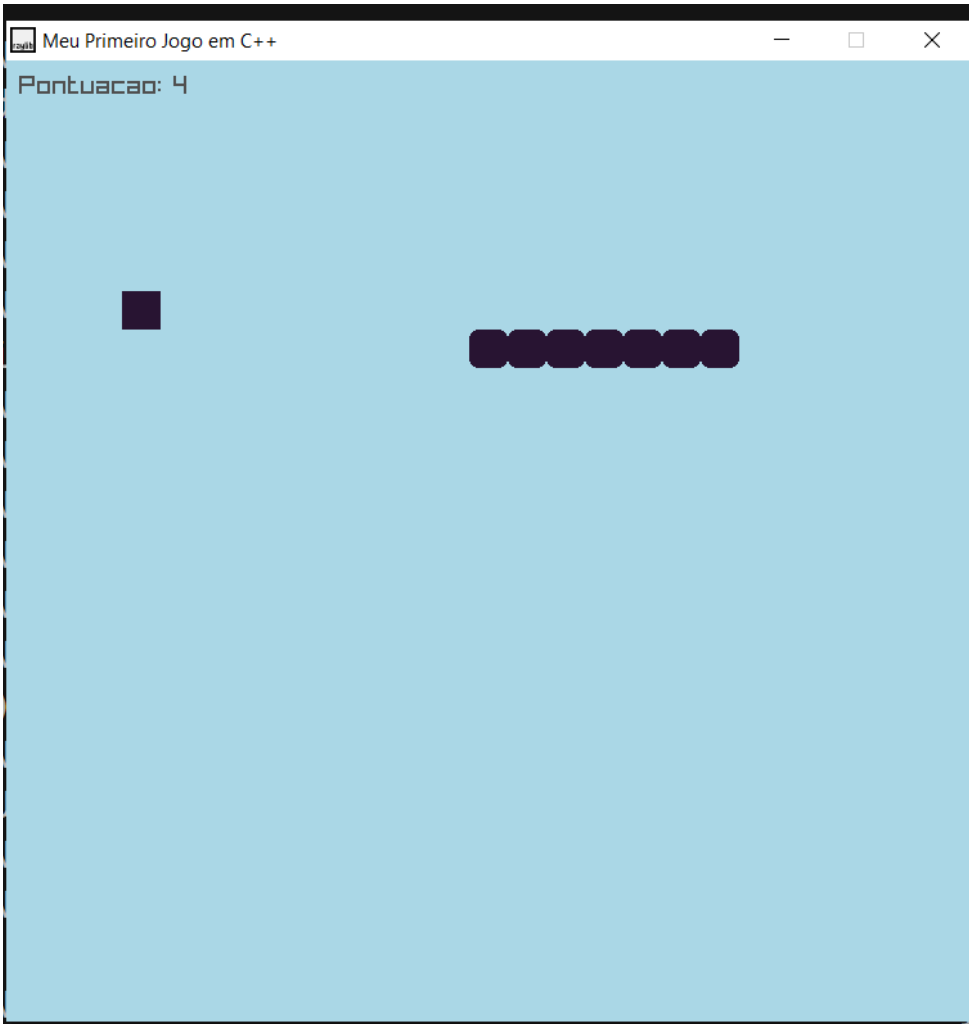
```
    // Verifica se a cobra colidiu com as bordas do jogo
    void verificaColisaoBorda() {
        if (cobra.corpo[0].x < 0 || cobra.corpo[0].x >= contagemCelulas ||
            cobra.corpo[0].y < 0 || cobra.corpo[0].y >= contagemCelulas) {
            CloseWindow(); // Encerra o jogo em caso de colisão
        }
    }

    // Verifica se a cobra colidiu consigo mesma
    void verificaColisaoCorpo() {
        for (size_t i = 1; i < cobra.corpo.size(); i++) {
            if (cobra.corpo[0].x == cobra.corpo[i].x && cobra.corpo[0].y == cobra.corpo[i].y) {
                CloseWindow(); // Encerra o jogo em caso de colisão
            }
        }
    }

    // Exibe a pontuação atual do jogador
    void desenhaPontuacao() {
        DrawText(TextFormat("Pontuacao: %d", pontuacao), 10, 10, 20, DARKGRAY);
    }
};
```

### 5.4 - Convergência das Classes

A interação entre as classes Cobra, Comida e Jogo é fundamental para o funcionamento do jogo, demonstrando como a programação orientada a objetos facilita a separação de responsabilidades e dividir o código em módulos. A classe Jogo atua como o elo central, organizando a comunicação entre os componentes: ela gerencia o comportamento da Cobra, garantindo que sua movimentação e crescimento sejam realizados corretamente, e coordena a interação com a Comida, verificando quando ocorre uma colisão que gera a pontuação e reposiciona a comida no tabuleiro.




(Jogo em Funcionamento)

## 6 - CONCLUSÃO

Durante o desenvolvimento do jogo, foi possível consolidar conhecimentos sobre programação orientada a objetos, estruturação de classes, manipulação de bibliotecas gráficas como a Raylib assuntos que foram abordados em sala de aula, bem como a organização de código para atender a objetivos específicos do jogo. A experiência reforçou a importância de separar responsabilidades entre classes e de implementar uma lógica clara para a interação entre componentes, tornando o código mais legível. No entanto, algumas melhorias podem ser implementadas em futuras versões, como a adição de níveis de dificuldade, sistemas de vidas para prolongar o jogo, animações mais dinâmicas, e melhorias na interface gráfica para tornar o jogo mais envolvente e acessível. Essas mudanças podem não só aprimorar a jogabilidade, mas também oferecer novos desafios técnicos para expandir ainda mais o aprendizado adquirido.

## REFERÊNCIAS

GROS, B. The impact of digital games in education. First Monday, v. 8, n. 7, p. 8 – 9, Jul 2003. Disponível em: . Acesso em: 15 nov. 2024. Citado na página 4.

Documentação Raylib. Site Oficial da biblioteca que está disponível em: https://www.raylib.com/cheatsheet/cheatsheet.htm

Tutorial Youtube: C++ Snake game using raylib - Beginner Tutorial 🐍 (OOP): Disponível em: https://www.youtube.com/watch?v=LGqsnM_WEK4&t=2744s