INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE SÃO PAULO – CAMPUS CAMPOS DO JORDÃO

ANDREY DE SOUZA SETÚBAL DESTRO

DENTE DEFENSOR: UM SHOOTER 2D COM RAYLIB E C++

CAMPOS DO JORDÃO
2025

RESUMO

Este relatório detalha o processo de concepção, desenvolvimento e implementação do jogo eletrônico "Dente Defensor", um shooter 2D de estilo arcade. O projeto foi desenvolvido na linguagem de programação C++ com o auxílio da biblioteca Raylib, focada na criação de gráficos e interatividade de forma simplificada. O objetivo principal do jogo é controlar uma nave defensora para proteger uma fileira de dentes contra ondas de bactérias invasoras. A jogabilidade envolve movimentação, disparo de projéteis, diferentes tipos de inimigos e um sistema de pontuação. O documento aborda a estrutura de classes orientada a objetos, a gestão de estados de jogo (título, gameplay, game over), a lógica de colisão e o gerenciamento de recursos, como texturas e áudio. Por fim, são apresentados os resultados visuais do projeto e sugestões para trabalhos futuros, consolidando o aprendizado prático em desenvolvimento de jogos.

Palavras-chave: Desenvolvimento de Jogos. C++. Raylib. Shooter 2D. Computação Gráfica.

ABSTRACT

This report details the conception, development, and implementation process of the electronic game "Dente Defensor" (Tooth Defender), an arcade-style 2D shooter. The project was developed in the C++ programming language with the support of the Raylib library, which focuses on simplified graphics and interactivity creation. The main objective of the game is to control a defender ship to protect a row of teeth against waves of invading bacteria. The gameplay involves movement, projectile shooting, different enemy types, and a scoring system. The document addresses the object-oriented class structure, game state management (title, gameplay, game over), collision logic, and resource management, such as textures and audio. Finally, the visual results of the project are presented, along with suggestions for future work, consolidating practical learning in game development.

Keywords: Game Development. C++. Raylib. 2D Shooter. Computer Graphics.

SUMÁRIO

1. INTRODUÇÃO

O desenvolvimento de jogos digitais é uma área multidisciplinar que combina conceitos de programação, computação gráfica, design de interação e arte. A criação de um projeto completo, mesmo que de escopo reduzido, oferece uma oportunidade valiosa para aplicar conhecimentos teóricos em um ambiente prático e desafiador.

Objetivo: O objetivo principal deste trabalho foi desenvolver um jogo 2D funcional do gênero *shooter* (jogo de tiro), intitulado "Dente Defensor". Os objetivos específicos incluem:

- Aplicar os conceitos de Programação Orientada a Objetos (POO) em C++ para modelar as entidades do jogo.
- Utilizar a biblioteca Raylib para renderização gráfica, gerenciamento de janelas, entrada de usuário e áudio.
- Implementar uma máquina de estados finitos para controlar os diferentes fluxos do jogo (tela de título, jogo principal e tela de fim de jogo).
- Desenvolver mecânicas de jogo essenciais, como movimento do jogador, disparo de projéteis, IA simples para inimigos e detecção de colisão.

Justificativa: A escolha do C++ se deve à sua relevância na indústria de jogos, sendo a linguagem base para grandes motores como a Unreal Engine. A biblioteca Raylib foi selecionada por sua simplicidade e baixo nível de abstração, permitindo um contato mais direto com os fundamentos da computação gráfica, sem a complexidade de um motor de jogo completo. O tema lúdico de "defender dentes" foi escolhido para criar uma identidade visual única e divertida, tornando o projeto mais engajador.

Aporte Teórico: O desenvolvimento baseou-se em conceitos fundamentais da ciência da computação, incluindo: estruturas de dados (como o std::vector para listas dinâmicas), algoritmos (como o std::remove_if para limpeza de vetores), princípios de POO (encapsulamento, herança e polimorfismo, embora de forma implícita na estrutura das classes) e matemática vetorial para cálculo de movimento e direção.

2. METODOLOGIA

Esta seção detalha as ferramentas e os processos adotados para a construção do jogo "Dente Defensor", desde a concepção até a implementação final do código.

2.1 Considerações Iniciais

O projeto foi concebido como um jogo de arcade inspirado em clássicos como *Space Invaders*. A premissa é simples: o jogador deve impedir que inimigos (bactérias) cheguem à base (dentes). A partida termina se um número específico de dentes for destruído. Para dar mais dinâmica ao jogo, foram planejados diferentes tipos de inimigos e um sistema de tiro aprimorado (power-up).

2.2 Ferramentas Utilizadas

- **Linguagem de Programação:** C++, por seu desempenho e controle de baixo nível.
- **Biblioteca Gráfica:** Raylib (versão 4.x), uma biblioteca simples e poderosa para o desenvolvimento de jogos e aplicações gráficas em C/C++.
- **Ambiente de Desenvolvimento Integrado (IDE):** Code::Blocks, um IDE de código aberto e multiplataforma.
- **Compilador:** MinGW (Minimalist GNU for Windows), um port do GCC (GNU Compiler Collection) para o ambiente Windows, utilizado em conjunto com o Code::Blocks.
- **Criação de Ativos (Assets):** As imagens (sprites) e as músicas foram geradas com o auxílio de ferramentas de Inteligência Artificial, agilizando o processo de criação de conteúdo visual e sonoro.

2.3 Descrição do Projeto

O código-fonte foi estruturado utilizando o paradigma de Programação Orientada a Objetos para organizar e encapsular a lógica de cada elemento do jogo. A seguir, descreve-se a arquitetura e as principais mecânicas implementadas.

- **Estrutura Principal e Máquina de Estados:** O fluxo do jogo é controlado por um enum GameScreen que define três estados: TITLE (tela inicial), GAMEPLAY (jogo principal) e GAMEOVER (tela de fim de jogo). Um switch-case dentro do loop principal do jogo (while !WindowShouldClose()) gerencia a lógica e a

renderização de cada tela, garantindo que apenas os elementos relevantes para o estado atual sejam processados e desenhados.

- **Classe** Tooth **(Dente):** Representa os objetos que o jogador deve proteger. Cada dente possui atributos como posição (Rectangle), saúde (health) e textura. Um método Damage() reduz sua vida, e sua cor muda visualmente (de WHITE para YELLOW e BROWN) para indicar o dano recebido.

- **Classe** Enemy **(Inimigo):** Modela as bactérias. Possui atributos de posição, velocidade e tipo (NORMAL ou FAST). O método Update() calcula a direção em que o inimigo deve se mover, que é sempre em direção ao dente com a menor quantidade de vida restante, criando uma IA simples e direcionada.

- **Classe** Projectile **(Projétil):** Representa os tiros disparados pelo jogador. Possui posição, velocidade e um status booleano (active). O projétil é desativado ao sair da tela para otimizar o processamento.

- **Classe** Player **(Jogador):** Controla a entidade do jogador. Gerencia a posição, a movimentação (controlada pelas setas do teclado) e o sistema de tiro. O jogador pode alternar entre um tiro único (SINGLE_SHOT) e um tiro triplo (TRIPLE_SHOT) pressionando a tecla 'C', adicionando um elemento estratégico à jogabilidade.

- **Detecção de Colisão:** A biblioteca Raylib oferece funções prontas para detecção de colisão. Foram utilizadas:

    - CheckCollisionCircles(): Para colisões entre projéteis e inimigos (tratados como círculos).
    - CheckCollisionCircleRec(): Para colisões entre inimigos (círculos) e dentes (retângulos).
- **Gerenciamento de Recursos:** Todas as texturas e músicas são carregadas no início do programa e descarregadas no final (UnloadTexture, UnloadMusicStream) para evitar vazamentos de memória. A música é tratada como *stream* para não sobrecarregar a RAM, exigindo atualização manual a cada quadro com UpdateMusicStream().

3. RESULTADOS OBTIDOS

As imagens a seguir ilustram as diferentes telas e momentos do jogo "Dente Defensor", demonstrando a aplicação prática dos conceitos descritos.

Figura 1 – Tela de Título Descrição: A tela inicial do jogo, exibindo o título "DENTE DEFENSOR" e a instrução para o jogador iniciar a partida. A música de introdução toca neste momento.

Figura 2 – Gameplay em Ação Descrição: Uma captura da tela de jogo principal. Observa-se o jogador na parte inferior, a fileira de dentes a ser protegida, inimigos (bactérias) descendo do topo e projéteis em trajetória. A interface (HUD) exibe a pontuação e o tipo de tiro ativo.

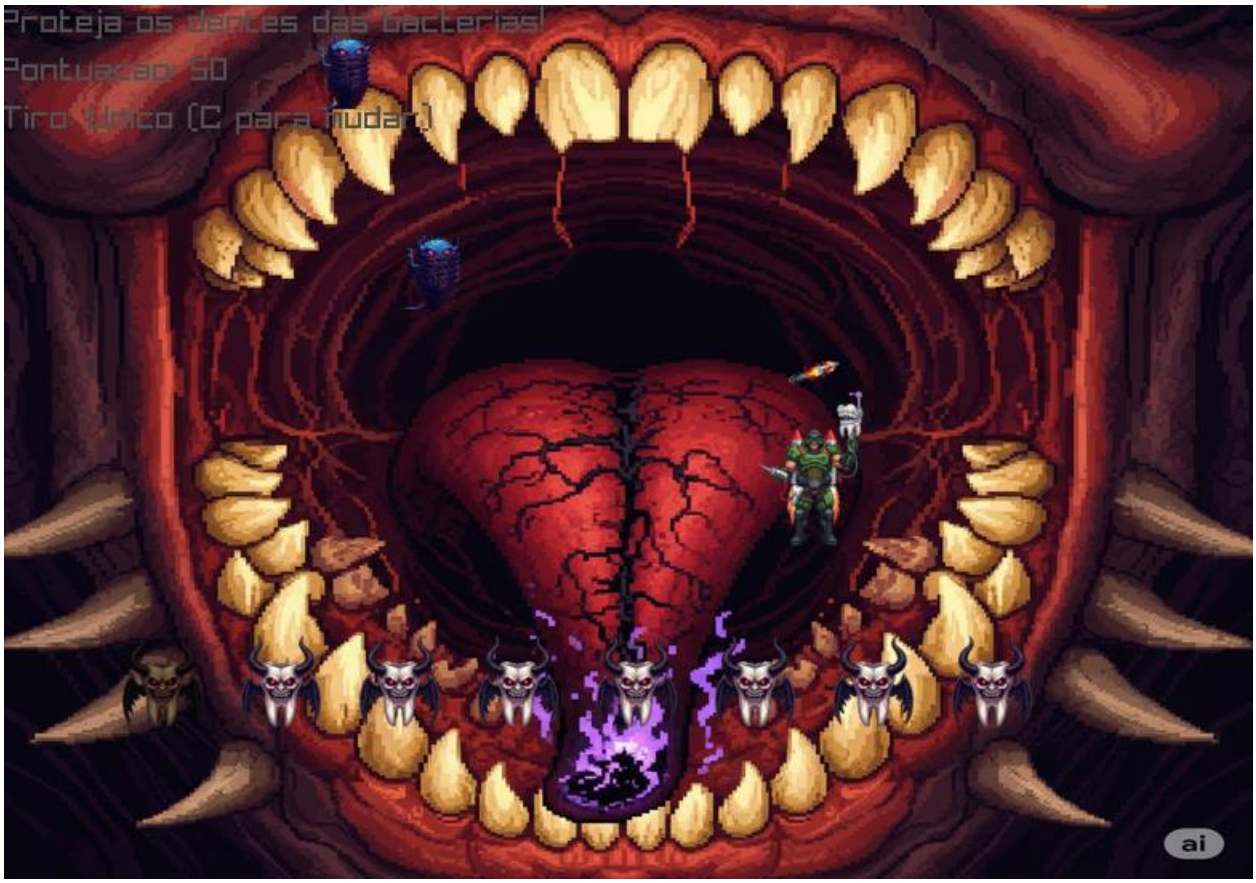

Figura 3 – Dentes Danificados e Inimigos Descrição: Detalhe da mecânica de dano. Os dentes à esquerda estão com a cor alterada para amarelo e marrom, indicando que já sofreram colisões com inimigos.

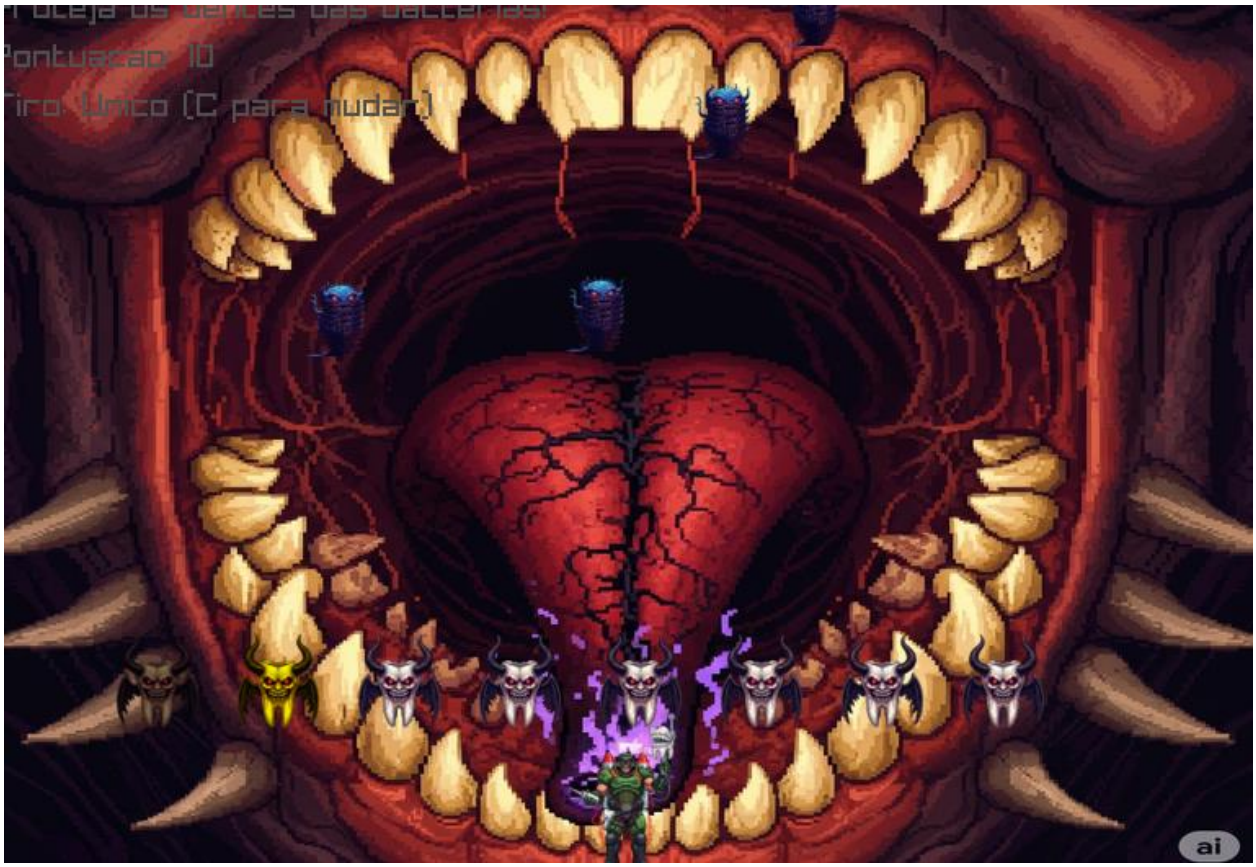

Figura 4 – Tela de Fim de Jogo (Game Over) Descrição: A tela de "FIM DE JOGO", exibida quando o jogador perde 3 ou mais dentes. Apresenta a pontuação final e a opção para reiniciar a partida.

4. CONCLUSÃO

O desenvolvimento do jogo "Dente Defensor" permitiu alcançar com sucesso todos os objetivos propostos. Foi possível criar um produto de software funcional e divertido, aplicando de maneira prática os conhecimentos em C++, Programação Orientada a Objetos e o uso da biblioteca Raylib. A estruturação do projeto em classes bem definidas e o uso de uma máquina de estados mostraram-se abordagens eficientes para gerenciar a complexidade do jogo.

O processo de desenvolvimento reforçou a importância do gerenciamento de memória e recursos, da matemática vetorial em jogos e do design de um loop de jogo coeso. A simplicidade da Raylib provou ser um excelente ponto de partida para focar nas mecânicas do jogo sem as abstrações de um motor de grande porte.

Sugestões para Melhorias e Trabalhos Futuros:

O projeto atual serve como uma base sólida que pode ser expandida de diversas maneiras. Algumas sugestões para trabalhos futuros incluem:

- **Mais Variedade:** Adicionar novos tipos de inimigos com padrões de movimento diferentes (ex: em zigue-zague) ou que também disparem projéteis.
- **Power-ups:** Implementar itens coletáveis que possam conceder ao jogador um escudo temporário, aumento de velocidade ou armas mais potentes.
- **Efeitos Sonoros:** Adicionar efeitos sonoros para ações como disparo, colisão e destruição de inimigos, enriquecendo a experiência auditiva.
- **Sistema de Dificuldade Progressiva:** Aumentar a frequência de surgimento e a velocidade dos inimigos conforme a pontuação do jogador aumenta.
- **Recordes (High Scores):** Implementar um sistema para salvar a pontuação mais alta em um arquivo local, incentivando o jogador a superar seus próprios recordes.

Conclui-se que o projeto "Dente Defensor" não só cumpriu seus requisitos técnicos, mas também proporcionou uma experiência de aprendizado significativa e abrangente no campo do desenvolvimento de jogos digitais.

5. REFERÊNCIAS BIBLIOGRÁFICAS

CHACON, R. A. (raysan5). **Raylib: a simple and easy-to-use library to enjoy videogames programming.** GitHub, 2013-2024. Disponível em: https://github.com/raysan5/raylib. Acesso em: 10 jun. 2024.

CODE::BLOCKS. **Code::Blocks: The open source, cross platform, free C, C++ and Fortran IDE.** Disponível em: https://www.codeblocks.org/. Acesso em: 10 jun. 2024.

MinGW-w64. **MinGW-w64 - for 32 and 64 bit Windows.** Disponível em: https://www.mingw-w64.org/. Acesso em: 10 jun. 2024.

STROUSTRUP, Bjarne. **A Linguagem de Programação C++.** 4. ed. Porto Alegre: Bookman, 2014.

1