INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE SÃO PAULO

MATHEUS LOPES DE OLIVEIRA

Este documento se trata da entrega projeto final da disciplina de Programação Orientada a Objetos, do curso de Tecnologia em Análise e Desenvolvimento de Sistemas, do Instituto Federal de Educação, Ciência e Tecnologia de São Paulo - Campus Campos do Jordão (IFSP-CJO).

PROFESSOR: Paulo Giovani de Faria Zeferino.

Jogo de Star Wars com Raylib C++

CAMPOS DO JORDÃO

2024

RESUMO

O presente trabalho descreve o desenvolvimento de um jogo no estilo de "Space Invaders", utilizando a biblioteca Raylib e a linguagem de programação C++. O objetivo principal do jogo é controlar uma nave X-Wing enquanto o jogador enfrenta ondas de inimigos e obstáculos, com a missão de sobreviver e acumular pontos. O projeto explora conceitos de programação de jogos, como manipulação de objetos gráficos, controle de entrada do usuário e detecção de colisões. O jogo inclui elementos como lasers disparados pela nave, obstáculos no cenário, naves inimigas com inteligência artificial simples e uma grande ameaça representada pela Estrela da Morte. A dinâmica de jogo é estruturada em torno da destruição de inimigos e obstáculos, mantendo o jogador engajado através do aumento de dificuldade conforme o progresso. O sistema de pontuação e de "highscore" é implementado, permitindo ao jogador verificar seu desempenho ao longo do jogo. O trabalho também sugere diversas melhorias para enriquecer a experiência do usuário, como a adição de mais níveis, melhorias gráficas e sonoras, e uma sincronização mais eficiente entre os efeitos sonoros e visuais.

Palavras-chave: Jogo, Raylib, C++, Inteligência Artificial, Detecção de Colisões.

ABSTRACT

This project describes the development of a "Space Invaders"-style game, using the Raylib library and the C++ programming language. The main objective of the game is to control an X-Wing fighter as the player faces waves of enemies and obstacles, with the goal of surviving and accumulating points. The project explores game programming concepts such as graphic object manipulation, user input control, and collision detection. The game includes elements like lasers shot by the ship, obstacles in the environment, enemy ships with simple artificial intelligence, and a major threat represented by the Death Star. The gameplay dynamics are structured around the destruction of enemies and obstacles, keeping the player engaged through increasing difficulty as the game progresses. A scoring and "highscore" system is implemented, allowing the player to track their performance throughout the game. The project also suggests several improvements to enhance the user experience, such as adding more levels, improving graphics and audio elements, and achieving better synchronization between visual and sound effects.

Keywords: Game, Raylib, C++, Artificial Intelligence, Collision Detection.

LISTA DE ILUSTRAÇÕES

SUMÁRIO

1 INTRODUÇÃO

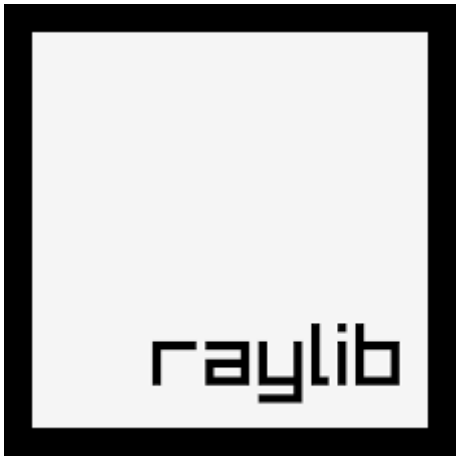

Figura 1 - Raylib

O desenvolvimento de jogos eletrônicos combina criatividade, tecnologia e lógica de programação, oferecendo um campo desafiador e dinâmico para estudantes e iniciantes na área. Este projeto tem como principal objetivo criar um jogo inspirado no clássico *Space Invaders*, utilizando a linguagem C++ e a biblioteca gráfica Raylib.

A proposta do trabalho está centrada no aprendizado e aplicação prática de conceitos essenciais de programação orientada a objetos (POO), além da implementação de técnicas fundamentais no desenvolvimento de jogos, como manipulação gráfica, detecção de colisões e gerenciamento de eventos. A Raylib foi escolhida por ser uma biblioteca leve e acessível, facilitando a criação de elementos gráficos e sonoros com simplicidade.

Baseado no universo de *Star Wars*, o jogo desenvolvido apresenta uma nave controlada pelo jogador, inimigos temáticos, obstáculos, sistema de pontuação e mecânicas que incluem vidas e níveis de dificuldade progressivos. Este projeto busca integrar aprendizado teórico e prático em uma experiência completa de desenvolvimento, aliando diversão e desafio.

Durante a construção do jogo, foram explorados conceitos como organização de classes, reaproveitamento de código, uso de estruturas como vetores e manipulação de arquivos para armazenamento de dados. Além disso, o projeto serviu como base para consolidar conhecimentos de lógica de programação e design de interação.

Este relatório detalha as etapas do desenvolvimento do jogo, desde a concepção e implementação até os testes finais, oferecendo uma visão crítica sobre os resultados obtidos e os conhecimentos adquiridos ao longo do processo.

1.1 Objetivo

O objetivo deste projeto é consolidar os conhecimentos adquiridos em programação, utilizando a linguagem **C++** e o paradigma de orientação a objetos (POO). Além disso, o desenvolvimento busca aprimorar habilidades na criação de aplicações gráficas interativas, com o uso da biblioteca **Raylib** para construção de jogos 2D.

Os objetivos específicos incluem:

1. **Desenvolvimento de um jogo inspirado em *Space Invaders***: Criar um jogo com temática baseada no universo de *Star Wars*, onde o jogador controla uma nave X-Wing e enfrenta naves inimigas, obstáculos, e a poderosa Estrela da Morte. O jogo inclui sistemas de pontuação, vidas, níveis de dificuldade e mecânicas de colisão.
2. **Aprofundar o conhecimento na linguagem C++**: Aplicar os conceitos da linguagem, como manipulação de vetores, estruturas condicionais e laços, além do uso de arquivos para salvar e carregar dados, como o *highscore*.
3. **Praticar conceitos de orientação a objetos**: Estruturar o projeto de forma modular, implementando classes para representar os diferentes componentes do jogo (naves, lasers, obstáculos, etc.). O uso de métodos e atributos encapsulados promove uma melhor organização do código e facilita a manutenção.
4. **Aprender a utilizar a biblioteca Raylib**: Explorar os recursos de **Raylib**, incluindo renderização de imagens, detecção de colisão, manipulação de sons e gerenciamento de entrada do usuário.
5. **Desenvolver uma aplicação com foco em usabilidade e diversão**: Criar um jogo funcional e envolvente, que apresente uma experiência fluida ao usuário, com elementos de desafio progressivo e recompensas (como aumento de pontuação e exibição do *highscore*).

Este projeto também serve como base para futuros desenvolvimentos, tanto no aprimoramento das mecânicas já implementadas quanto na adição de novas funcionalidades. É uma oportunidade de aplicar conceitos teóricos em um contexto prático e criativo, contribuindo para o aprendizado contínuo na área de programação de jogos.

1.2 Justificativa

A escolha deste tema, que envolve o desenvolvimento de um jogo inspirado no clássico *Space Invaders* com elementos do universo *Star Wars*, é motivada por diversos aspectos que abrangem tanto interesses pessoais quanto objetivos educacionais relacionados ao aprendizado da programação e ao domínio de ferramentas específicas para o desenvolvimento de jogos. O gênero de tiro espacial, popularizado por *Space Invaders*, é caracterizado por mecânicas simples e diretas, mas que permitem explorar uma ampla gama de conceitos fundamentais de desenvolvimento de jogos, como o controle de objetos, detecção de colisões e sistemas de pontuação.

A adoção desse tema se justifica, em primeiro lugar, pela oportunidade de aplicar conhecimentos adquiridos em programação orientada a objetos (POO) em um contexto prático, utilizando a linguagem C++ e a biblioteca Raylib.

1.3 Aporte Teórico

O desenvolvimento de jogos eletrônicos envolve a integração de diversos conhecimentos, como programação, design gráfico, interação do usuário e áudio. Este projeto, focado no desenvolvimento de um jogo 2D utilizando C++ e a biblioteca Raylib, visa explorar aspectos importantes da criação de jogos, como a programação orientada a objetos, o uso de ferramentas eficientes para a criação de jogos simples e a construção de uma experiência imersiva para o jogador.

- **Linguagem C++ no Desenvolvimento de Jogos:** A linguagem C++ é amplamente utilizada no desenvolvimento de jogos devido à sua alta performance e controle sobre os recursos do sistema. Esse controle direto sobre a memória é crucial em jogos de alto desempenho, como os jogos de ação, que exigem a otimização dos recursos. A linguagem também oferece suporte à programação orientada a objetos, o que permite a organização modular do código, essencial para o desenvolvimento de jogos complexos.

Em jogos como o proposto, a programação orientada a objetos facilita a criação de classes que representam entidades do jogo, como personagens, inimigos e objetos interativos. Cada classe pode ser responsável por comportamentos específicos, o que organiza e torna o código mais eficiente e de fácil manutenção.

- **Programação Orientada a Objetos (POO):** A Programação Orientada a Objetos (POO) é um paradigma que organiza o código em torno de "objetos", instâncias de "classes" que possuem atributos e métodos. No contexto de jogos, a POO permite modelar as diversas entidades, como naves e lasers, de maneira estruturada, facilitando sua manipulação e interação.

  Por exemplo, neste jogo, a classe `Laser` controla os disparos do personagem e dos inimigos, enquanto a classe `Xwing` representa a nave do jogador e a classe `Nave` modela os inimigos. Esses objetos interagem por meio de métodos específicos, como o movimento e a detecção de colisões, o que torna a estrutura do jogo mais flexível e escalável.
- **Biblioteca Raylib:** Raylib é uma biblioteca de desenvolvimento de jogos C simples e eficiente, que facilita a criação de jogos 2D e 3D. Sua API foi projetada para ser acessível, permitindo que os desenvolvedores se concentrem na lógica do jogo em vez de lidar com detalhes complexos de baixo nível. Raylib oferece funções para manipulação gráfica, entrada do usuário e áudio, tornando-se uma ferramenta poderosa para criar jogos interativos.

  Neste projeto, Raylib é utilizada para gerenciar gráficos, como o controle de sprites e texturas, e para lidar com a entrada do usuário por meio do teclado e mouse. Além disso, a biblioteca fornece funcionalidades para a detecção de colisões, essencial para a interação realista entre objetos do jogo.
- **Design de Jogos:** O design de jogos envolve a criação de mecânicas que garantem uma experiência divertida e desafiadora para o jogador. Em um jogo como o proposto, o jogador controla uma nave e deve eliminar inimigos, enquanto o jogo avalia a pontuação e as vidas restantes. A definição de regras claras, como a perda de vidas ao colidir com inimigos ou disparos, é essencial para a jogabilidade.

  Além disso, o design de jogos envolve a criação de uma experiência visual e sonora imersiva. O uso de gráficos atrativos e efeitos sonoros dinâmicos contribui para o ambiente do jogo, tornando-o mais envolvente. A música de fundo

e os efeitos de colisão, por exemplo, são recursos que ajudam a manter o jogador imerso na experiência.

- **Mecânicas de Colisão e Inteligência Artificial:** A mecânica de colisão é crucial para garantir a interação entre os objetos do jogo de forma realista. A biblioteca Raylib oferece funções que facilitam a detecção de colisões entre objetos retangulares, permitindo que o jogo reaja corretamente a eventos como disparos que atingem inimigos ou obstáculos.

  A inteligência artificial (IA) dos inimigos, embora simples, é um componente importante. As naves inimigas se movem horizontalmente e disparam lasers de forma aleatória, criando um desafio para o jogador. A movimentação coordenada das naves inimigas, juntamente com os disparos, torna o jogo mais dinâmico e interessante.

2 METODOLOGIA

A metodologia empregada neste trabalho foi fundamentada em uma combinação de pesquisa bibliográfica e aplicação prática dos conhecimentos adquiridos, com o objetivo de desenvolver um jogo 2D baseado no universo de *Star Wars*, utilizando a biblioteca **Raylib**. Através da utilização da Programação Orientada a Objetos (POO) e da biblioteca mencionada, foi possível criar um sistema modular e eficiente para o gerenciamento das diversas funcionalidades do jogo, como movimentação, disparo, detecção de colisões, pontuação e interação com obstáculos.

A primeira etapa do desenvolvimento envolveu uma pesquisa bibliográfica sobre a biblioteca Raylib. A pesquisa anterior, entregue para a disciplina de Programação, foi essencial para a fundamentação do uso da biblioteca, permitindo o entendimento de suas funcionalidades, como a renderização de gráficos, controle de entrada do usuário e manipulação de áudio. Essa pesquisa serviu como base para o estudo da documentação oficial da Raylib, que foi fundamental para a integração da biblioteca ao projeto. Além disso, a documentação ofereceu insights sobre a utilização de funções como LoadTexture, DrawRectangl, PlayMusicStream, e CheckCollisionRecs, as

quais foram aplicadas diretamente no desenvolvimento do jogo, oferecendo uma estrutura gráfica simples, mas eficiente. Por fim, foi realizada a instalação das ferramentas para o desenvolvimento.

2.1 Ferramentas Utilizadas

O desenvolvimento deste projeto foi realizado utilizando um conjunto de ferramentas poderosas e amplamente utilizadas para a criação de jogos 2D. A seguir, será apresentada uma descrição detalhada de cada uma dessas ferramentas: Raylib, C++, GCC Compiler e VSCode.

2.1.1 Raylib

Raylib é uma biblioteca gráfica de código aberto desenvolvida em C, projetada para facilitar a criação de jogos e aplicações multimídia. Ela fornece uma série de funções simples e eficientes para trabalhar com gráficos 2D e 3D, áudio e entrada do usuário. Raylib é particularmente popular entre iniciantes devido à sua sintaxe intuitiva e à vasta documentação, o que torna a criação de jogos acessível e sem a complexidade de outras bibliotecas mais robustas. No contexto deste projeto, a Raylib foi utilizada para lidar com as operações de renderização gráfica, como o desenho de objetos na tela, manipulação de sons e músicas, e o gerenciamento de eventos de entrada do usuário, como o movimento da nave e o disparo de lasers. Além disso, a biblioteca também foi responsável pela detecção de colisões, uma funcionalidade essencial para o funcionamento do jogo. A utilização da Raylib facilitou o desenvolvimento do jogo, proporcionando ferramentas prontas para uso e reduzindo a necessidade de escrever código complexo para tarefas comuns.

2.1.2 C++

C++ é uma das linguagens de programação mais populares e poderosas, amplamente utilizada no desenvolvimento de jogos, sistemas de alto desempenho, e aplicações que exigem manipulação direta de hardware. A escolha do C++ para este projeto foi motivada pela sua performance e pela facilidade com que se pode trabalhar

com a Programação Orientada a Objetos (POO), um paradigma que facilita a modularização e a organização do código. A utilização de C++ permitiu que o desenvolvimento do jogo fosse realizado de forma estruturada, com a criação de classes que encapsulam as diversas entidades do jogo, como a nave X-Wing, as naves inimigas, os lasers e os obstáculos. Além disso, o C++ também foi fundamental para a implementação de funcionalidades avançadas de controle de fluxo, como a verificação de colisões e a manipulação de eventos de entrada.

2.1.3 GCC Compiler

O GCC (GNU Compiler Collection) é um compilador de código aberto que suporta diversas linguagens de programação, incluindo C e C++. Ele é amplamente utilizado na comunidade de desenvolvedores devido à sua robustez, performance e capacidade de compilar código para diferentes plataformas. O GCC Compiler foi utilizado neste projeto para compilar o código C++ e gerar o arquivo executável do jogo. Sua integração com sistemas operacionais como Linux, Windows e macOS garantiu que o projeto fosse facilmente portável entre diferentes plataformas, sem a necessidade de ajustes significativos no código. O GCC também possui recursos avançados de otimização, permitindo que o jogo seja executado de forma eficiente, mesmo em máquinas com recursos limitados.

2.1.4 Visual Studio Code

O Visual Studio Code (VSCode) é um editor de código-fonte leve, mas poderoso, que oferece suporte a uma ampla variedade de linguagens de programação, incluindo C++.

2.2 Desenvolvimento

A segunda etapa foi a aplicação prática do conhecimento adquirido. Para isso, foi necessário planejar a estrutura do jogo, identificando os elementos principais, como a nave do jogador (X-Wing), os inimigos (naves imperiais), obstáculos e lasers. A escolha de Programação Orientada a Objetos foi decisiva para modularizar o código,

tornando-o mais legível e flexível. Cada componente do jogo foi implementado como uma classe independente, o que permitiu a criação de objetos com responsabilidades bem definidas e a implementação de interações entre eles.

A seguir, será apresentada uma descrição detalhada das principais classes e suas funcionalidades dentro do jogo:

- **Game:** é a principal responsável pela lógica central do jogo, funcionando como o controlador geral das mecânicas e das interações entre os objetos. Entre suas funções principais, destacam-se a inicialização dos elementos do jogo, a atualização do estado do jogo a cada quadro, a renderização dos objetos na tela e o tratamento das entradas do jogador. Adicionalmente, a classe é responsável por verificar as colisões entre os diversos elementos do jogo, calcular a pontuação e gerenciar as vidas do jogador. A função Update() é a responsável pela atualização do estado dos objetos, enquanto a **Draw()** lida com a renderização dos componentes gráficos. Além disso, a classe implementa métodos de controle de colisões e de reinício do jogo, como ChecarColisoes() e Reiniciar(), que garantem o bom funcionamento da dinâmica de jogo.
- **Xwing:** representa a nave do jogador, a icônica X-Wing. Esta classe controla a movimentação da nave, que é realizada horizontalmente, além de gerenciar o disparo dos lasers. A nave pode ser movimentada para a esquerda ou direita utilizando os métodos MoverEsquerda() e MoverDireita(), enquanto o método AtirarLaser() permite que o jogador dispare lasers na tela. A classe também possui a função Reiniciar(), que reseta as propriedades da nave quando o jogo é reiniciado. Para a renderização, o método Draw() é utilizado para desenhar a nave na tela.
- **Laser:** representa os projéteis disparados tanto pela nave do jogador quanto pelas naves inimigas. Cada laser possui propriedades como posição, velocidade e um estado de atividade, que determina se o projétil está ativo ou não. O método Update() atualiza a posição do laser com base em sua velocidade, enquanto o Draw() é responsável por desenhá-lo na tela. Já o método getRect() retorna um retângulo que representa a área de colisão do laser, permitindo a verificação de interações com outros objetos no jogo.
- **Obstaculo:** é responsável por representar as barreiras que protegem a nave do jogador, formando uma espécie de escudo. Cada obstáculo é composto por pequenos blocos, que podem ser destruídos quando atingidos por lasers. O

método Draw() renderiza os blocos na tela, enquanto o getRect() retorna o retângulo de colisão de cada bloco, necessário para a detecção de colisões com outros objetos, como lasers.

- **Blocos:** compõem os obstáculos são representados pela classe Bloco, que define um objeto simples com posição e dimensão, e que pode ser destruído pelos lasers disparados. Cada bloco é representado por um pequeno retângulo, e a classe contém os métodos Draw(), que renderiza o bloco na tela, e getRect(), que fornece o retângulo de colisão para a verificação de interações.
- **Nave:** representa as naves inimigas, que se movem em grupos e disparam lasers periodicamente. Cada nave possui um tipo, que varia em termos de valor de pontuação quando destruída. O método Update(int direcao) é responsável pela movimentação das naves, enquanto o Draw() realiza a renderização gráfica na tela. A classe também possui o método getRect(), que retorna o retângulo de colisão da nave, utilizado para detectar interações com outros objetos do jogo.
- **EstrelaDaMorte:** representa um elemento especial do jogo, que aparece aleatoriamente e é um dos principais inimigos a ser destruído pelo jogador. A Estrela da Morte se move horizontalmente pela tela e pode ser destruída ao ser atingida por lasers. O método Spawn() a posiciona em um ponto aleatório da tela, enquanto o Update() atualiza sua posição conforme ela se move. A função Draw() é responsável por renderizar a Estrela da Morte, e o método getRect() fornece o retângulo de colisão, permitindo a detecção de impactos.

A interação entre essas classes é essencial para o funcionamento do jogo. A classe **Game** gerencia a criação e atualização das instâncias das classes **Xwing, Laser, Obstaculo, Bloco, Nave e EstrelaDaMorte,** garantindo a interação entre elas e o bom funcionamento da dinâmica do jogo. A utilização de objetos de diferentes classes permite modularizar as funcionalidades, facilitando a manutenção e a expansão do código.

2.3 Estudos Realizados

A pesquisa também envolveu o estudo de técnicas acesso a arquivos de texto, como a leitura e escrita de arquivos para salvar o highscore do jogo. A função **SalvaHighscoreNoArquivo()** foi implementada com base em conceitos aprendidos sobre manipulação de arquivos em C++, permitindo que o jogador consiga registrar a maior pontuação alcançada.

Adicionalmente, foi realizada uma pesquisa sobre o design de jogos, focando na criação de uma experiência de jogo desafiadora e divertida. O equilíbrio entre o tempo de spawn dos inimigos, a velocidade das naves e a dificuldade dos obstáculos foi ajustado através de testes contínuos durante o desenvolvimento.

Por fim, a integração de todas as classes e funcionalidades foi realizada com testes contínuos e ajustes, de modo a garantir o correto funcionamento da detecção de colisões, movimentação e renderização. O processo de depuração e otimização foi conduzido utilizando as ferramentas de debugging da IDE utilizada, além de observações em tempo real da execução do jogo.

Em resumo, a metodologia aplicada envolveu uma combinação de pesquisa bibliográfica e desenvolvimento prático, utilizando a biblioteca Raylib como ferramenta central. A pesquisa anterior sobre a Raylib, juntamente com a aplicação dos conceitos de POO, foi fundamental para o sucesso do desenvolvimento do jogo. Além disso, as fontes de pesquisa forneceram os fundamentos necessários para a criação de um jogo funcional, intuitivo e agradável de jogar.

3 RESULTADOS OBTIDOS

Os resultados obtidos neste projeto são demonstrados por meio de capturas de tela (prints) do jogo em funcionamento. Essas imagens ilustram os diferentes aspectos e etapas do jogo, incluindo a interface gráfica, as interações do jogador, e as dinâmicas de colisões e movimentação. O fluxo do jogo, desde a inicialização até o encerramento, é visível nas capturas, proporcionando uma visão detalhada de como o sistema de pontuação, a interação com os inimigos e a destruição dos obstáculos são realizados.

Os prints também mostram os momentos em que o jogador interage com o jogo, como quando o X-Wing atira lasers ou quando ocorre uma colisão entre a nave inimiga e um laser. A tela de "Game Over" e as transições de fase podem ser observadas, fornecendo uma visão clara do estado do jogo em diferentes pontos de progresso.

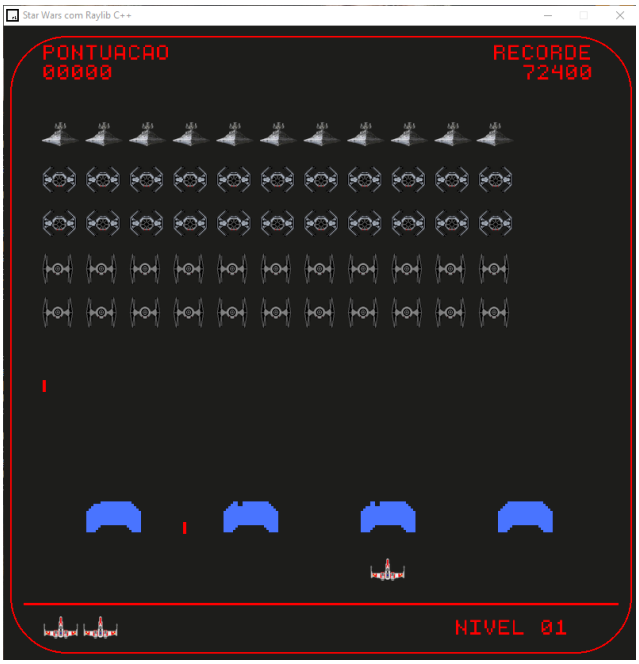

Figura 2 – Jogo Início

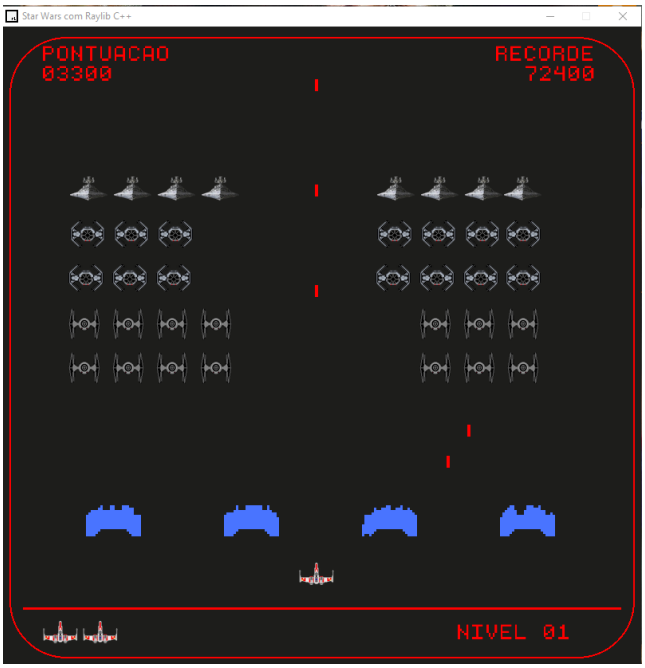

Figura 3 – Tiros de Laser

Figura 4 – Pontuação Recorde

Figura 5 – Vidas do Jogador

Figura 6 - Escudo

Figura 7 – Estrela da Morte

4 CONCLUSÃO

O projeto do jogo desenvolvido oferece uma experiência divertida e desafiadora, proporcionando uma interação dinâmica entre o jogador e os elementos do jogo, como a nave X-Wing, as naves inimigas, os lasers e os obstáculos. A utilização da biblioteca Raylib garantiu uma implementação gráfica e sonora eficaz, enquanto o código em C++ possibilitou um controle preciso dos elementos do jogo. No entanto,

como em qualquer projeto de desenvolvimento, há sempre oportunidades para melhorias, que poderiam enriquecer ainda mais a experiência do jogador e agregar mais camadas de complexidade e imersão ao jogo.

4.1 Melhorias na Sincronização dos Efeitos Sonoros

Um dos pontos que poderia ser aprimorado é a sincronização dos efeitos sonoros. Embora os sons de disparo e explosão estejam presentes, alguns jogadores podem perceber uma falta de harmonia entre os efeitos e as ações no jogo. Por exemplo, os lasers disparados pela nave X-Wing poderiam ter um som mais distinto e com melhor tempo de resposta, criando uma sensação mais imersiva de ação. Além disso, o efeito sonoro da Estrela da Morte poderia ser mais impactante, refletindo melhor a grandiosidade do evento. A adição de transições sonoras mais suaves entre os diferentes estados do jogo, como a tela de "Game Over" e a reinicialização, também ajudaria a criar uma experiência auditiva mais fluida.

4.2 Inclusão de Novos Níveis e Aumento de Complexidade

Outro ponto que pode ser melhorado é a inclusão de múltiplos níveis de dificuldade, o que proporcionaria ao jogador uma experiência mais variada e desafiadora. Atualmente, o jogo segue uma estrutura linear, onde as naves inimigas e a Estrela da Morte aparecem conforme o tempo.

A introdução de novos níveis, com diferentes tipos de inimigos, mais obstáculos e até mesmo novos desafios, como naves inimigas com comportamentos mais complexos, poderia aumentar a longevidade e o apelo do jogo. Além disso, ao finalizar um nível, o jogador poderia ser recompensado com um aumento na dificuldade, introduzindo novos elementos que exigem habilidades aprimoradas, criando assim uma progressão contínua e uma maior motivação para jogar.

4.3 Implementação de um Fundo Estrelado

Para melhorar a estética do jogo, uma das sugestões seria a implementação de um fundo estrelado que se movesse lentamente, criando a sensação de que a nave

X-Wing está se deslocando no espaço. Atualmente, o jogo carece de um cenário de fundo mais imersivo, o que pode impactar a experiência visual.

A adição de estrelas piscando ao fundo poderia enriquecer a atmosfera e dar uma sensação de vastidão ao universo do jogo. Além disso, o fundo poderia ser dinamicamente alterado conforme o nível ou a situação do jogo, por exemplo, com a adição de nebulosas ou efeitos visuais especiais em momentos críticos, como quando a Estrela da Morte é destruída.

4.4 Melhoria no Design Visual dos Elementos

Embora os gráficos básicos do jogo, como as naves, lasers e obstáculos, estejam adequadamente implementados, existe espaço para melhorias estéticas. O design visual dos elementos poderia ser mais detalhado, com texturas mais refinadas e animações mais fluidas.

A nave X-Wing, por exemplo, poderia ter animações mais detalhadas ao se mover, como a simulação de aceleração e desaceleração. As explosões também poderiam ser melhoradas, com animações mais complexas, que tornariam os efeitos visuais mais impactantes.

Ademais, a inclusão de efeitos visuais ao interagir com objetos no jogo, como partículas ou efeitos de choque quando os lasers atingem obstáculos ou inimigos, poderia tornar a jogabilidade mais envolvente. Tais adições estéticas ajudariam a criar uma sensação mais intensa de ação e recompensariam o jogador visualmente, contribuindo para uma experiência mais satisfatória.

4.5 Expansão da Trilha Sonora

A trilha sonora do jogo, embora adequada para a temática de combate espacial, poderia ser expandida para incluir mais variações e faixas que mudassem de acordo com o andamento do jogo. Por exemplo, a música de fundo poderia mudar para um tema mais intenso durante o aparecimento da Estrela da Morte ou quando o jogador estiver prestes a enfrentar uma situação crítica. A diversidade de músicas ajudaria a

manter o jogador mais imerso, proporcionando uma sensação de progressão e dinamismo no jogo.

4.6 Considerações Finais

Em resumo, o jogo desenvolvido oferece uma boa base para uma experiência de combate espacial envolvente, com mecânicas interessantes e um desempenho eficiente. No entanto, como sugerido, existem várias áreas de melhoria, tanto no aspecto técnico quanto estético, que poderiam enriquecer significativamente a experiência do jogador. A implementação de mais níveis, a adição de um fundo estrelado e efeitos sonoros mais sincronizados são apenas algumas das sugestões que, se aplicadas, poderiam transformar o jogo em uma experiência ainda mais completa e imersiva. Essas melhorias contribuiriam para aumentar a longevidade e o apelo do jogo, tornando-o mais desafiador e visualmente interessante para os jogadores.

REFERÊNCIAS

Terminal Root. **"Crie Jogos para Windows, Linux e Web com Raylib C/C++".** Disponível em: https://www.youtube.com/watch?v=LZUMVdkWWrg . Acesso em: 4 setembro 2024.

DOCUMENTAÇÃO oficial da RAYLIB. Disponível em: https://raylib.com/cheatsheet.html . Acesso em: 12 out. 2024.

RAYLIB GitHub Repository. Disponível em: https://github.com/raysan5/raylib . Acesso em: 2 out. 2024.

RAYLIB Wiki. Disponível em: https://github.com/raysan5/raylib/wiki . Acesso em: 5 out. 2024.

RAYLIB Examples Repository. Disponível em: https://github.com/raysan5/raylib/tree/master/examples . Acesso em: 7 out. 2024.

DISCORD Raylib Community. Disponível em: https://discord.gg/r . Acesso em: 29 out. 2024.

STACK Overflow. "Raylib Tag". Disponível em: https://stackoverflow.com/questions/tagged/raylib . Acesso em: 6 out. 2024.

GAMEDEV.NET Fórum. "Discussions on Raylib". Disponível em: https://www.gamedev.net/forums/ . Acesso em: 30 out. 2024.

Raylib Game Programming Tutorial and Reforcing Refs. Playlist de tutoriais no YouTube. Disponível em: https://www.youtube.com/playlist?list=PL-NSm9WUHXM-FEpA3qto28UozmIxsZPX27 . Acesso em: 21 out. 2024.

Programming With Nick. "**How to install raylib with C++ on Windows and use it with Visual Studio Code**". Vídeo no YouTube. Disponível em: https://www.youtube.com/watch?v=PaAcVk5jUd8 . Acesso em: 1 out. 2024.