INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE SÃO PAULO

VINÍCIUS SILVA DE SAMPAIO

DESLIZE APEX – O JOGO

CAMPOS DO JORDÃO

2025

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE SÃO PAULO

VINÍCIUS SILVA DE SAMPAIO

Projeto de desenvolvimento de um jogo em C++ apresentado como parte dos requisitos para aprovação na disciplina de Programação Orientada a Objetos, do curso de Análise e Desenvolvimento de Sistemas, no Instituto Federal de São Paulo no Campus de Campos do Jordão, sob orientação do professor Paulo Giovani de Faria Zeferino.

CAMPOS DO JORDÃO

2025

RESUMO

O presente trabalho apresenta o desenvolvimento de "Deslize Apex", um jogo 2D de ação em labirinto, construído na linguagem C++ com a biblioteca gráfica *Raylib*. O ambiente de desenvolvimento foi composto pelo *IDE Visual Studio* e pelo gerenciador de pacotes *vcpkg*. Inspirado em jogos como *Tomb of the Mask* e *Pac-Man*, o projeto teve como objetivo principal a aplicação prática do paradigma de Programação Orientada a Objetos para criar um jogo completo e funcional. Foram explorados conceitos como a lógica de movimento em grade, detecção de colisão, gerenciamento de estados e a implementação de inteligência artificial para diferentes inimigos. A metodologia adotada foi iterativa e incremental, com a arquitetura do *software* inteiramente modelada em classes, o que garantiu a organização e a modularidade do código. O processo abrangeu desde o planejamento das mecânicas, passando pela implementação progressiva das três fases do jogo, até a realização de testes funcionais e de jogabilidade. Os resultados demonstraram que o jogo final é estável e atinge os desafios propostos, evidenciando a eficácia da abordagem orientada a objetos no desenvolvimento de jogos.

Palavras-Chave: Programação Orientada a Objetos; C++; Raylib; Design de Jogos; Detecção de Colisão; Inteligência Artificial.

ABSTRACT

This paper presents the development of "Deslize Apex," a 2D action-maze game built using the C++ language and the *Raylib* graphics library. The development environment consisted of the *Visual Studio IDE* and the *vcpkg* package manager. Inspired by games such as *Tomb of the Mask* and *Pac-Man*, the project's main objective was the practical application of the Object-Oriented Programming (*OOP*) paradigm to create a complete and functional game. The project explored concepts such as tile-based movement logic, collision detection, game state management, and the implementation of artificial intelligence for different enemies. An iterative and incremental methodology was adopted, with the software architecture modeled entirely using classes, which ensured the code's organization and modularity. The development process ranged from planning the game mechanics and progressively implementing the three game phases, to conducting functional and gameplay tests. The results showed that the final game is stable and meets the proposed challenges, demonstrating the effectiveness of the object-oriented approach in game development.

Keywords: Object-Oriented Programming; C++; Raylib; Game Design; Collision Detection; Artificial Intelligence.

LISTA DE ILUSTRAÇÕES

LISTA DE SIGLAS

API	*Application Programming Interface* (Interface de Programação de Aplicações)
FPS	*Frames Per Second* (Quadros por segundo)
IA	Inteligência Artificial
IDE	*Integrated Development Environment* (Ambiente de Desenvolvimento Integrado)
POO	Programação Orientada a Objetos

SUMÁRIO

1 INTRODUÇÃO

O desenvolvimento de jogos eletrônicos é uma área que integra programação com campos criativos como *design* de interação, artes visuais e sonoras. Dentro desse universo, projetos que equilibram uma jogabilidade acessível com uma dificuldade crescente costumam gerar experiências cativantes, o que serviu como um dos pilares para a construção deste trabalho.

Este relatório apresenta o processo de desenvolvimento do "Deslize Apex", um jogo 2D de ação em labirinto. O projeto foi integralmente construído na linguagem C++, com o suporte da biblioteca gráfica *Raylib* para a renderização e o gerenciamento de interações. O ambiente utilizado foi o *Visual Studio* (*IDE*), em conjunto com o gerenciador de pacotes *vcpkg* para a administração das bibliotecas necessárias.

O "Deslize Apex" foi projetado para oferecer uma experiência que testa tanto os reflexos do jogador quanto seu planejamento estratégico. Sua mecânica central envolve o controle de um personagem que desliza em alta velocidade pelo cenário até colidir com um obstáculo. Para progredir, o jogador precisa coletar todas as moedas de cada fase, desviando de armadilhas e enfrentando diferentes tipos de inimigos.

Essa estrutura de jogo proporcionou um cenário ideal para a aplicação prática de conceitos fundamentais de programação. Entre eles, destacam-se a lógica para detecção de colisão, o gerenciamento de múltiplos estados de jogo como menu, partida ativa e telas de resultado e a renderização de elementos gráficos e sonoros em tempo real.

1.1 Objetivos

Este trabalho apresenta o desenvolvimento do jogo 2D "Deslize Apex", um projeto focado na aplicação prática da Programação Orientada a Objetos (POO) em C++. Toda a arquitetura do jogo foi estruturada com essa abordagem, com o suporte da biblioteca *Raylib* para os elementos gráficos, de áudio e de interação.

Para a sua realização, o projeto foi orientado pelos seguintes objetivos:

- Pesquisar as mecânicas de jogos do gênero ação em labirinto, analisando

sistemas de movimento em grade (*tiles*), coleta de itens e comportamentos de inimigos para fundamentar o *design* do jogo.

- Modelar e implementar as funcionalidades centrais do jogo utilizando classes e objetos com a biblioteca gráfica *raylib*, utilizando C++, distribuindo as seguintes responsabilidades:
    - O movimento de deslize do jogador e sua interação com os obstáculos.
    - A lógica para coleta de itens moedas e o sistema de progressão entre as fases.
    - O comportamento de diferentes tipos de perigos, como armadilhas estáticas e inimigos.
    - O gerenciamento do fluxo de jogo, controlando os estados de menu, partida ativa, tela de vitória e de *game over*.
    - A funcionalidade de um *timer* (cronômetro) para fases com limite de tempo.
- Projetar três fases com um nível de dificuldade progressivo, buscando criar uma experiência que seja ao mesmo tempo desafiadora e recompensadora.
- Integrar recursos visuais e sonoros para enriquecer a atmosfera do jogo e fornecer *feedback* claro ao jogador.
- Validar o funcionamento do jogo através de testes contínuos, garantindo a estabilidade da aplicação e a qualidade da experiência final.

1.2 Justificativa

A escolha de desenvolver o "Deslize Apex" foi motivada pelo seu alto potencial de aprendizado. O gênero de ação em labirinto é um excelente cenário para aplicar na prática conceitos essenciais de desenvolvimento de jogos, como a criação de sistemas de colisão, o gerenciamento de estados, a implementação de inimigos e o tratamento de comandos do jogador em tempo real. Este projeto se tornou, portanto, uma oportunidade ideal para consolidar os conhecimentos em C++ através de um desafio prático e bem definido.

A decisão de estruturar todo o código com base na Programação Orientada a Objetos foi um pilar do projeto. Essa abordagem permitiu que as diversas mecânicas

fossem desenvolvidas como classes independentes e modulares. Isso facilitou o desenvolvimento de forma incremental, permitindo focar na qualidade de cada sistema antes de integrá-los, o que foi fundamental para gerenciar a complexidade de um jogo com três fases de dificuldade crescente.

As ferramentas foram escolhidas para criar um ambiente de desenvolvimento produtivo e focado. A linguagem C++ foi selecionada por sua performance e controle de baixo nível, características importantes para jogos. A biblioteca *Raylib* foi usada para toda a parte gráfica e de áudio por ser leve, simples e de fácil integração, permitindo que a maior parte do tempo fosse dedicada à lógica do jogo em si. Por fim, a combinação do *IDE Visual Studio* com o gerenciador de pacotes *vcpkg* se mostrou muito eficiente: o primeiro ofereceu um ambiente robusto para codificação e depuração (*debugging*), enquanto o segundo simplificou drasticamente a instalação e o gerenciamento de dependências como a *Raylib*.

1.3 Aspectos Metodológicos

O desenvolvimento do "Deslize Apex" seguiu uma abordagem iterativa e incremental. O processo começou com uma etapa de planejamento e pesquisa, na qual foram analisadas as mecânicas de jogos de ação em labirinto, como "*Tomb of the Mask*", para definir os requisitos e o escopo do projeto. O ambiente de desenvolvimento foi então configurado com C++, a biblioteca *Raylib* e o *IDE Visual Studio*, utilizando o *vcpkg* para gerenciar as dependências.

Com o planejamento definido, o ciclo de desenvolvimento foi guiado pelos princípios de POO. Cada sistema do jogo, como o movimento do jogador, a lógica das colisões, a inteligência artificial dos inimigos e o gerenciamento de fases, foi modelado como uma classe específica.

A implementação ocorreu de forma incremental: cada classe era codificada e testada de forma isolada antes de ser integrada ao projeto principal, garantindo a funcionalidade de cada parte do sistema.

Ao longo de todo o processo, foram realizados ciclos de testes contínuos. Esses testes serviram para validar a estabilidade do jogo, a precisão das mecânicas e para fazer ajustes no balanceamento da dificuldade das três fases. Essa abordagem

iterativa de construir, testar e refinar foi fundamental para chegar ao produto final de forma organizada e coesa.

1.4 Aporte Teórico

Para detalhar o desenvolvimento do projeto, o relatório segue as seguintes seções: A seção 2 aborda as bases do projeto. Nela, serão detalhadas as ferramentas e tecnologias utilizadas, como a linguagem C++, a biblioteca *Raylib* e o *IDE Visual Studio*. Também será apresentado o jogo "Deslize Apex", com foco em sua arquitetura, na abordagem de Programação Orientada a Objetos adotada e em suas principais mecânicas.

A seção 3 descreve o processo de desenvolvimento. Esta seção detalha as etapas do trabalho, desde o planejamento inicial até a implementação incremental de todas as funcionalidades, e a integração dos elementos audiovisuais, tudo seguindo a metodologia orientada a objetos.

A seção 4 apresenta e analisa os resultados obtidos. Aqui, o funcionamento das mecânicas, a estabilidade da aplicação e a experiência de jogabilidade do produto final serão avaliados e comparados com os objetivos estabelecidos para o projeto.

Por fim, a seção 5 traz a conclusão do trabalho. Serão discutidos os aprendizados obtidos, os pontos fortes do projeto e as dificuldades encontradas.

2 FUNDAMENTAÇÃO TEÓRICA

O *design* do "Deslize Apex" foi influenciado por mecânicas de jogos de sucesso, tanto modernos quanto clássicos do gênero *arcade*. Analisar essas referências foi um passo importante para definir a estrutura e a experiência do jogo.

A principal inspiração para a jogabilidade veio do título *mobile "Tomb of the Mask"* (2016). Desse jogo foi adaptada a mecânica central de movimento, que é o deslize rápido do personagem em uma direção até colidir com um obstáculo. Para complementar essa base, foi desenvolvido um sistema de progressão próprio, no qual o jogador precisa coletar todas as moedas da fase para desbloquear a saída, combinando a agilidade do movimento com um objetivo claro de exploração.

Figura 1 – *Tomb of the Mask* (2016)

Para o inimigo “Perseguidor”, a referência foi o clássico *“Pac-Man”* (1980). A inteligência artificial desse adversário foi projetada para simular o comportamento de perseguição dos fantasmas do jogo, fazendo com que ele navegue pelo labirinto de forma estratégica para alcançar o jogador, o que adiciona uma camada de tensão e desafio.

Figura 2 – *Pac-Man* (1980)

A implementação dessas mecânicas exigiu uma arquitetura de *software* bem definida, que foi construída sobre os princípios da Programação Orientada a Objetos. Conceitos como o movimento em grade (*tiles*) e a detecção de colisões foram tratados por métodos específicos dentro das classes de cada entidade do jogo. Da mesma forma, o gerenciamento dos estados de jogo foi encapsulado por um objeto controlador, e a lógica de cada inimigo foi isolada em sua própria classe, permitindo comportamentos distintos. Essa abordagem foi essencial para organizar e integrar as diversas regras e sistemas do jogo de forma coesa.

2.1 Ferramentas e Bibliotecas Utilizadas

O projeto foi desenvolvido na linguagem C++, escolhida pela sua performance e controle sobre os recursos do sistema, que são fatores importantes para jogos. Toda a parte gráfica, de áudio e de interação foi gerenciada pela biblioteca *Raylib*, selecionada por sua simplicidade e por ter uma direta, o que agilizou a prototipação e a implementação das mecânicas visuais. Para o ambiente de trabalho, foi utilizado o *IDE Visual Studio* em conjunto com o gerenciador de pacotes *vcpkg*. O *Visual Studio* serviu como a principal ferramenta para a codificação e depuração do código, sendo seu *debugger* essencial para a identificação e correção de erros. O *vcpkg* complementou o ambiente ao automatizar a instalação e configuração de dependências, garantindo que a biblioteca *Raylib* fosse integrada ao projeto de forma estável e consistente.

Figura 3 – *Visual Studio* (2022)

2.2 Abordagem de Programação Orientada a Objetos

A arquitetura do "Deslize Apex" foi construída com base na Programação Orientada a Objetos, em C++. Essa abordagem foi escolhida para organizar o código de forma modular, o que facilita sua manutenção e futuras expansões. As principais entidades do jogo, como o personagem do jogador, os diferentes tipos de adversários e os projéteis, foram modeladas como classes distintas.

Cada classe foi projetada para encapsular os dados e os comportamentos de uma entidade específica. Nesse modelo, a classe que representa o jogador é responsável por armazenar informações sobre sua posição e estado, e também por conter os métodos que processam seus movimentos e o desenham na tela. O mesmo princípio de agrupar dados e lógica foi aplicado às outras classes, que gerenciam de forma autônoma seus próprios ciclos de vida e interações.

A dinâmica do jogo acontece através da interação entre esses objetos. No *loop* principal, a cada *frame*, o sistema instrui cada objeto a executar suas lógicas de atualização e de renderização. Para gerenciar coleções de entidades, notadamente os inimigos e projéteis em tela, foram utilizadas estruturas de dados dinâmicas da biblioteca padrão do C++, permitindo um manejo eficiente desses elementos.

3 PROJETO PROPOSTO

Seguindo a abordagem iterativa e incremental, o projeto "Deslize Apex" iniciou-se com uma fase aprofundada de concepção e planejamento. Nesta etapa, foram detalhadas as regras e a experiência do jogador. A mecânica de movimento foi definida como um deslize unidirecional que só é interrompido por uma colisão, criando uma jogabilidade com elementos de quebra-cabeça. O sistema de progressão foi estabelecido com um objetivo claro: a coleta de todas as moedas seria a condição para desbloquear a saída da fase, incentivando a exploração completa do mapa.

Foi também planejada a curva de dificuldade ao longo de três fases distintas. A primeira fase serviria para introduzir as mecânicas básicas. A segunda adicionaria o desafio de inimigos com ataques à distância e um limite de tempo. A terceira fase, por sua vez, introduziria o adversário com inteligência artificial de perseguição, representando o clímax do desafio.

Todo esse planejamento foi feito levando em conta as tecnologias que seriam empregadas. A escolha prévia da linguagem C++ e da biblioteca *Raylib* permitiu que o *design* das funcionalidades fosse realizado já com uma visão clara de como seriam tecnicamente implementadas, garantindo que o escopo do projeto se mantivesse realista e executável.

O ponto central desta fase foi o desenho da arquitetura do *software*, planejada em torno dos princípios da Programação Orientada a Objetos. Foi feita uma especificação inicial das principais classes do sistema e suas responsabilidades. Definiu-se uma classe para o jogador, responsável por gerenciar os comandos de entrada e seu estado. Para os inimigos, planejou-se uma estrutura com uma classe base e classes derivadas para os comportamentos específicos. Além disso, foi projetada uma classe controladora para gerenciar o estado geral do jogo, como menus e transições de fase. Ter esse planejamento detalhado da arquitetura e das mecânicas antes do início da codificação foi fundamental. Essa especificação inicial criou um roteiro de desenvolvimento claro, minimizou ambiguidades e permitiu que a implementação de cada sistema ocorresse de forma organizada, servindo como um alicerce sólido para o processo de desenvolvimento.

3.1 Considerações Iniciais

A ideia central do projeto foi criar um jogo de ação em labirinto, "Deslize Apex". A escolha foi motivada por razões didáticas e pelo desafio técnico que o gênero oferece. Diferente de jogos com mecânicas mais simples, este projeto permitiu explorar uma gama mais ampla de conceitos fundamentais da programação de jogos, como: lógica de movimento não-contínuo, gerenciamento de múltiplos estados de jogo, implementação de diferentes comportamentos para inimigos e design de níveis com desafios progressivos, proporcionando uma experiência de aprendizado prática e abrangente.

3.2 Ferramentas Utilizadas

Para o desenvolvimento do projeto, foram selecionadas e configuradas ferramentas específicas para garantir um processo de trabalho eficiente e estável:

- **Linguagem C++:** Foi a linguagem de programação principal, escolhida por sua performance, robustez e controle de memória.
- **Biblioteca *Raylib*:** Utilizada para toda a parte de renderização gráfica, áudio e gerenciamento de entradas. Sua simplicidade e otimização para jogos 2D foram decisivas para a escolha. Sua instalação foi feita por meio do comando "*vcpkg install raylib:x64-windows*", que configurou a biblioteca para uso no projeto.
- ***Visual Studio 2022*:** Serviu como o ambiente de desenvolvimento integrado (*IDE*) para a edição, compilação e depuração (*debugging*) de todo o código. Sua instalação foi feita com o *workload* "Desenvolvimento de *Desktop* com C++", que já inclui o compilador e as bibliotecas padrão necessárias.

***vcpkg*:** Utilizado para gerenciar as dependências do projeto. A ferramenta foi obtida de seu repositório no *GitHub* e integrada ao *IDE* através do comando "*vcpkg integrate install"*, o que garantiu a correta localização da biblioteca *Raylib*.

3.3 Apresentação de Equações e Fórmulas

O processo de desenvolvimento do "Deslize Apex" foi dividido em quatro etapas principais, seguindo uma abordagem organizada e incremental.

1. **Configuração do Ambiente de Desenvolvimento:**
    - A primeira etapa consistiu em baixar e instalar o *Visual Studio 2022 Community Edition*, selecionando a carga de trabalho (*workload*) específica para desenvolvimento com C++.
    - Em seguida, foi realizado o *download* e a instalação do gerenciador de pacotes *vcpkg* para facilitar a integração da *Raylib*.
    - A integração do *vcpkg* com o *Visual Studio* foi feita utilizando o comando vcpkg integrate install, garantindo que todas as bibliotecas necessárias fossem reconhecidas automaticamente pelo *IDE*.
2. **Implementação da Estrutura Orientada a Objetos:**

Com o ambiente pronto, a fase seguinte foi a criação da arquitetura do *software*. Um projeto C++ foi iniciado e, seguindo o paradigma de Programação Orientada a Objetos, a estrutura inicial das classes foi definida. Para cada entidade principal do jogo, como o jogador, os inimigos e os projéteis, uma classe correspondente foi criada. O princípio do encapsulamento foi aplicado desde o início: cada classe tornou-se responsável por gerenciar seus próprios dados e seus comportamentos essenciais, que foram definidos em métodos para atualização de estado e renderização gráfica.

3. **Desenvolvimento Incremental das Funcionalidades:**

A construção do jogo ocorreu de forma incremental, adicionando camadas de funcionalidades sobre a estrutura de classes já estabelecida. A primeira mecânica implementada foi o sistema de movimento de deslize do jogador, incluindo a lógica de colisão com as paredes do mapa. Logo após, foram adicionados os elementos de interação, como as moedas e o portão de saída. Com a base da jogabilidade pronta, as três fases foram construídas, introduzindo progressivamente os desafios: as armadilhas, os inimigos atiradores com seus padrões de tiro e, por fim, o inimigo perseguidor com sua IA de navegação na Fase 3. Os toques finais incluíram o desenvolvimento das telas de menu, vitória e derrota, e a integração dos recursos de áudio e texturas.

4. **Testes e Depuração:**

Os testes foram uma atividade constante e integrada ao longo de todo o ciclo de desenvolvimento. Cada nova funcionalidade era testada de forma isolada e integrada assim que implementada, o que permitiu a identificação e correção de erros de maneira ágil. Além dos testes funcionais para garantir a ausência de *bugs*, foram realizadas diversas iterações de testes de jogabilidade. Nesses ciclos, foram feitos ajustes no balanceamento da dificuldade, na velocidade do jogador, na cadência de tiro dos inimigos e na resposta geral do jogo, refinando a experiência para que o produto final fosse coeso e polido.

A escolha das ferramentas e a organização do processo metodológico permitiram um desenvolvimento eficaz e fluido. A abordagem orientada a objetos facilitou a gestão da complexidade crescente do jogo, enquanto o uso da *Raylib* agilizou a implementação dos elementos gráficos e de interação, resultando em um produto final alinhado aos objetivos propostos.

4 AVALIAÇÃO

A mecânica central de movimento de deslize em grade (tiles) foi o primeiro ponto avaliado. Os testes confirmaram que os comandos são precisos, e que a parada do personagem nos obstáculos é consistente, permitindo que o jogador planeje suas trajetórias. O sistema de progressão, que condiciona a abertura da saída à coleta de todas as moedas, provou ser uma regra eficaz, pois incentiva a exploração completa dos mapas e adiciona uma camada de estratégia ao percurso.

Os elementos de desafio foram avaliados individualmente e em conjunto. As armadilhas estáticas e os inimigos com padrões de tiro fixos funcionaram bem como obstáculos que exigem ritmo e atenção. A inteligência artificial do inimigo Perseguidor, na Fase 3, foi um destaque: sua capacidade de navegar pelo cenário de forma autônoma para caçar o jogador criou uma fonte de tensão dinâmica e constante, que elevou significativamente o desafio. A inclusão do timer nas fases 2 e 3 também foi bem-sucedida, cumprindo seu objetivo de pressionar o jogador a tomar decisões mais rápidas.

A comunicação de informações ao jogador foi considerada clara e funcional. As telas de interface para os estados de início, vitória e derrota, com as mensagens "FIM DE JOGO!" e "PARABÉNS, VOCÊ VENCEU!!!" são exibidas nos momentos corretos e comunicam o resultado da partida de forma inequívoca. O sistema de áudio complementou bem o feedback visual: a trilha sonora ajuda a ditar o ritmo, e os efeitos sonoros para ações importantes, como coletar moedas ou ser derrotado, reforçam o que acontece na tela, tornando a experiência mais imersiva.

Do ponto de vista técnico, a versão final do jogo alcançou um alto nível de estabilidade. Embora diversos *bugs* e erros tenham sido identificados e corrigidos durante as fases de desenvolvimento e teste, o produto final não apresentou travamentos ou falhas inesperadas durante a avaliação. A performance manteve-se fluida e com uma taxa de quadros constante, mesmo em momentos com múltiplos inimigos e projéteis na tela, o que valida a eficiência da arquitetura orientada a objetos e da biblioteca *Raylib*.

4.1 Condução

Antes de iniciar os testes, foram definidos os critérios que guiariam a avaliação. Os principais pontos observados foram a estabilidade da performance do jogo, a precisão das colisões entre o jogador e todos os elementos do cenário (paredes, moedas, armadilhas e inimigos), a resposta imediata aos comandos, o funcionamento correto da inteligência artificial dos adversários e a lógica de progressão entre as fases, incluindo as condições de vitória e derrota.

Os testes foram conduzidos no ambiente de desenvolvimento do *Visual Studio*, onde o jogo foi executado e depurado repetidamente. Diversos cenários foram explorados para garantir a cobertura de todas as mecânicas:

- A mecânica de deslize foi testada em diferentes tipos de corredores e obstáculos para verificar sua consistência.
- A interação com todos os perigos foi forçada para assegurar que as condições de fim de jogo fossem acionadas corretamente.
- O comportamento dos inimigos, como a frequência dos tiros e a rota de navegação do Perseguidor, foi observado para confirmar que estava de acordo com o planejado para cada fase.
- Finalmente, o ciclo completo do jogo foi executado várias vezes, do início ao fim, para garantir que a transição entre as fases e as telas de resultado funcionasse de forma coesa.

Além dos testes técnicos, houve uma atenção especial à experiência do jogador. Foi avaliado se os controles eram intuitivos e se os objetivos eram comunicados de forma clara. O feedback visual (como a mudança de cor do portão) e sonoro (efeitos de coleta de moeda, derrota e vitória) também foi analisado para confirmar se contribuía positivamente para a imersão e a jogabilidade. Durante todo o processo, foram feitos ajustes iterativos em parâmetros como a velocidade dos inimigos e os limites de tempo, buscando um desafio progressivo e equilibrado.

4.2 Resultados

Nesta seção, são apresentados os resultados obtidos durante o desenvolvimento e os testes do jogo "Deslize Apex", com base nas funcionalidades implementadas e na análise da experiência de jogo. As evidências coletadas e as principais características observadas serão descritas a seguir, ilustradas por capturas de tela representativas dos diferentes momentos do jogo, a tela de início, a jogabilidade e a progressão de desafios nas três fases, e as telas de conclusão da partida.

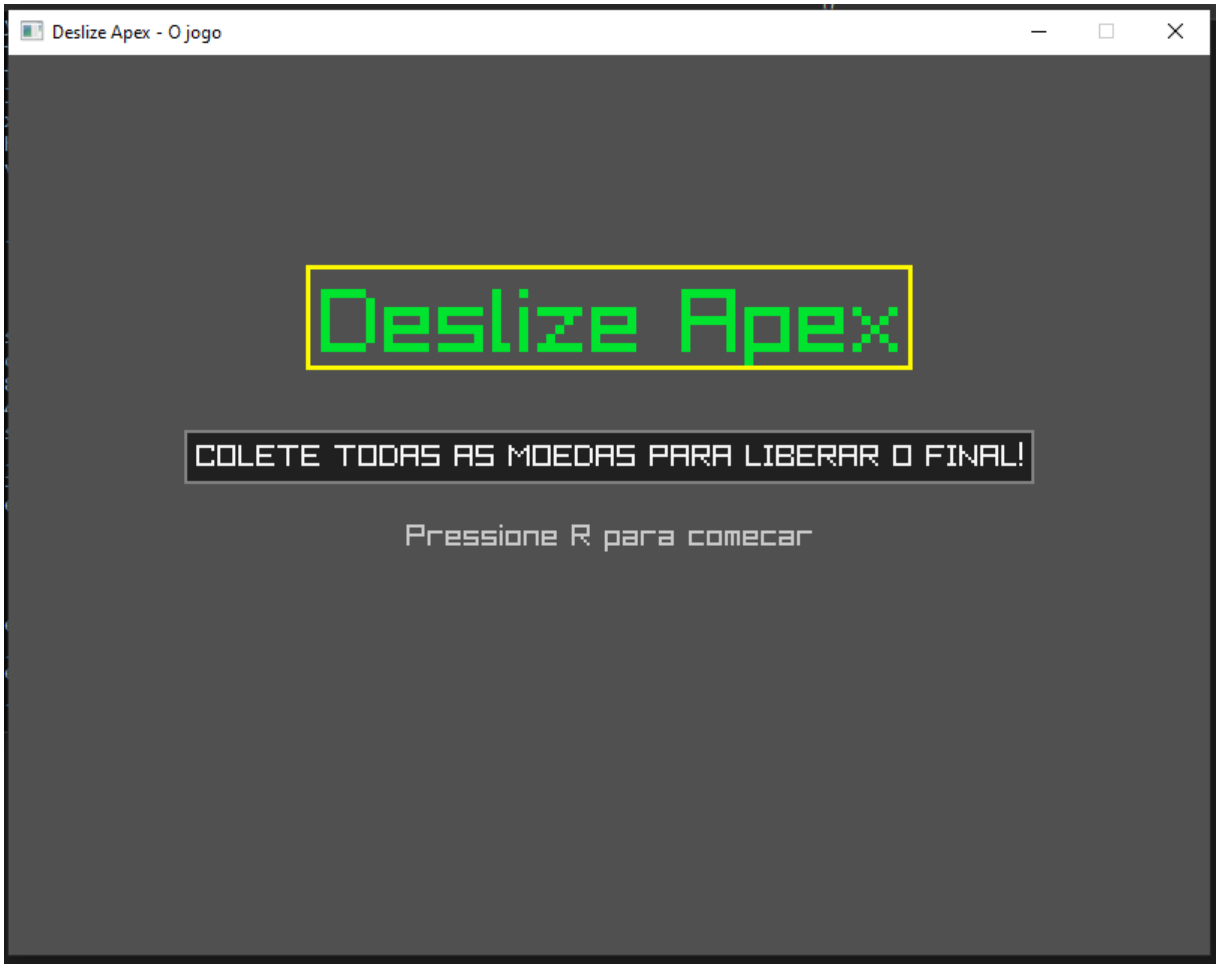

Figura 4 – Tela de início (2025)

A Figura 4 apresenta a tela inicial de "Deslize Apex". o primeiro contato do jogador com o jogo, exibindo o título do jogo de forma destacada e colorida para chamar a atenção. Logo abaixo, são fornecidas as instruções essenciais para o jogador: "COLETE TODAS AS MOEDAS PARA LIBERAR O FINAL!" e o comando para iniciar a partida, "Pressione R para começar". Testes validaram que esta tela

cumpre seu objetivo de apresentar o jogo e aguardar a ação do jogador para dar início à jogabilidade, acompanhada da música de fundo que estabelece o clima do jogo.

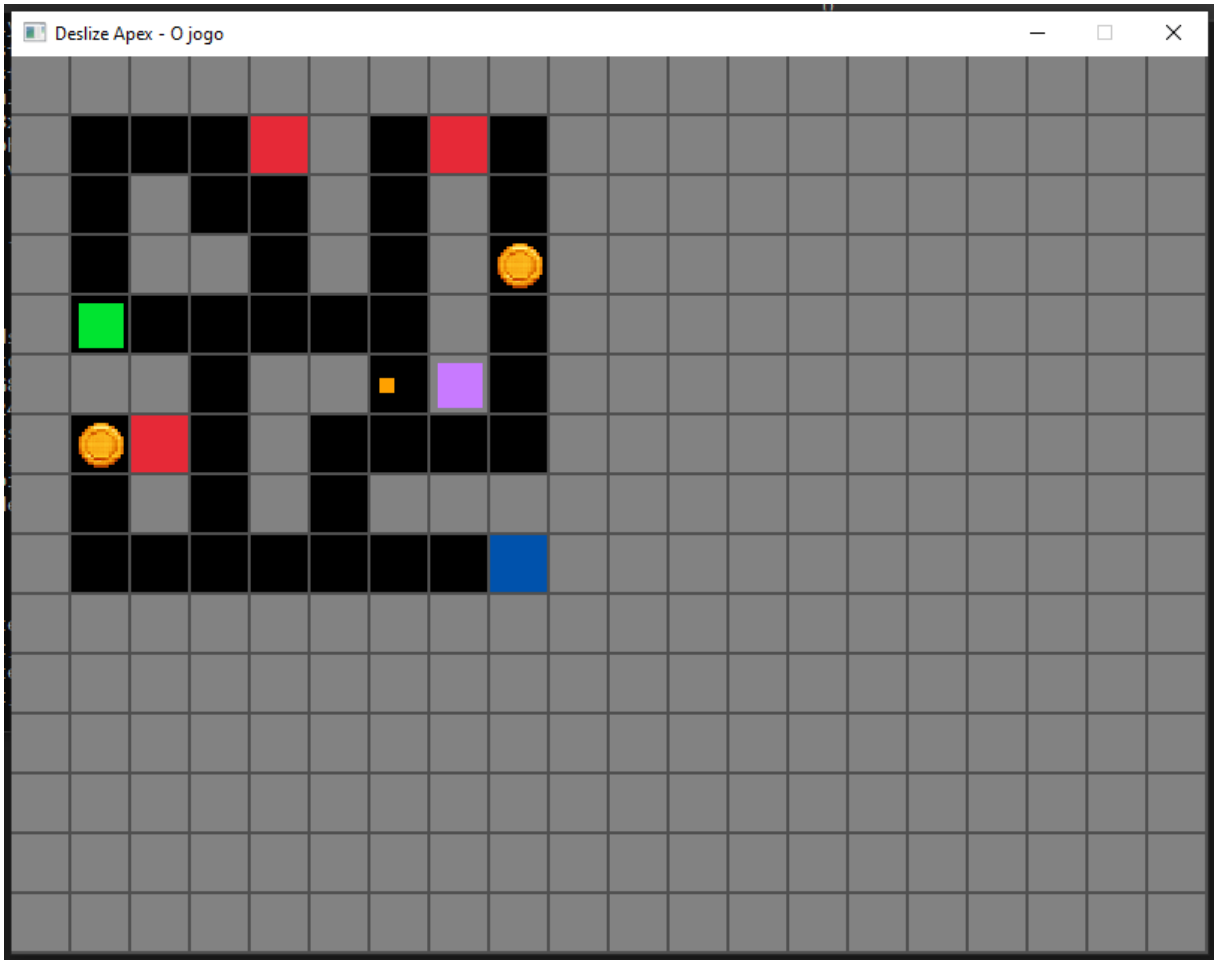

Figura 5 – Fase 1 (2025)

A Figura 5 ilustra a jogabilidade na Fase 1 de "Deslize Apex". Nesta fase introdutória, o jogador aprende a mecânica central de movimento, onde o personagem desliza até encontrar um obstáculo. O cenário é composto por *tiles* que definem caminhos, paredes e armadilhas. O objetivo principal, coletar todas as moedas para habilitar o portão de saída, é introduzido, acompanhada pelo efeito sonoro específico.

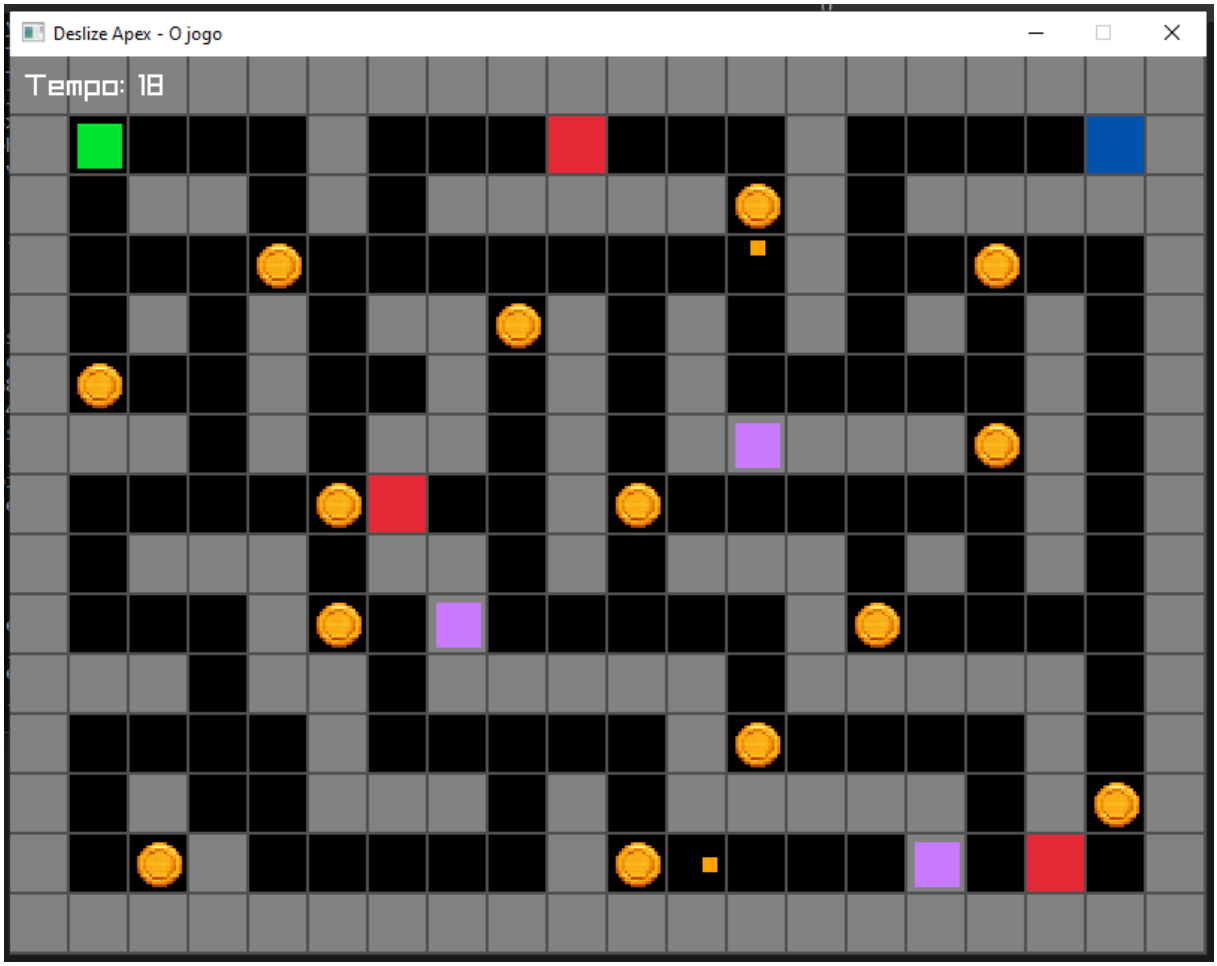

Figura 6 – Fase 2 (2025)

Na Fase 2, como demonstrado na Figura 6, a complexidade do jogo aumenta com a introdução de novos desafios. Além das mecânicas presentes na Fase 1, esta fase implementa um limite de tempo, exibido na interface do usuário, que adiciona um elemento de urgência. Mais inimigos que disparam projéteis em direções pré-definidas são introduzidos, exigindo que o jogador não apenas colete moedas e navegue pelo labirinto, mas também desvie ativamente de ameaças. A captura de tela evidencia um design de nível mais elaborado. Testes nesta fase confirmaram o funcionamento adequado do temporizador, a lógica de comportamento dos inimigos atiradores e a interação dos projéteis.

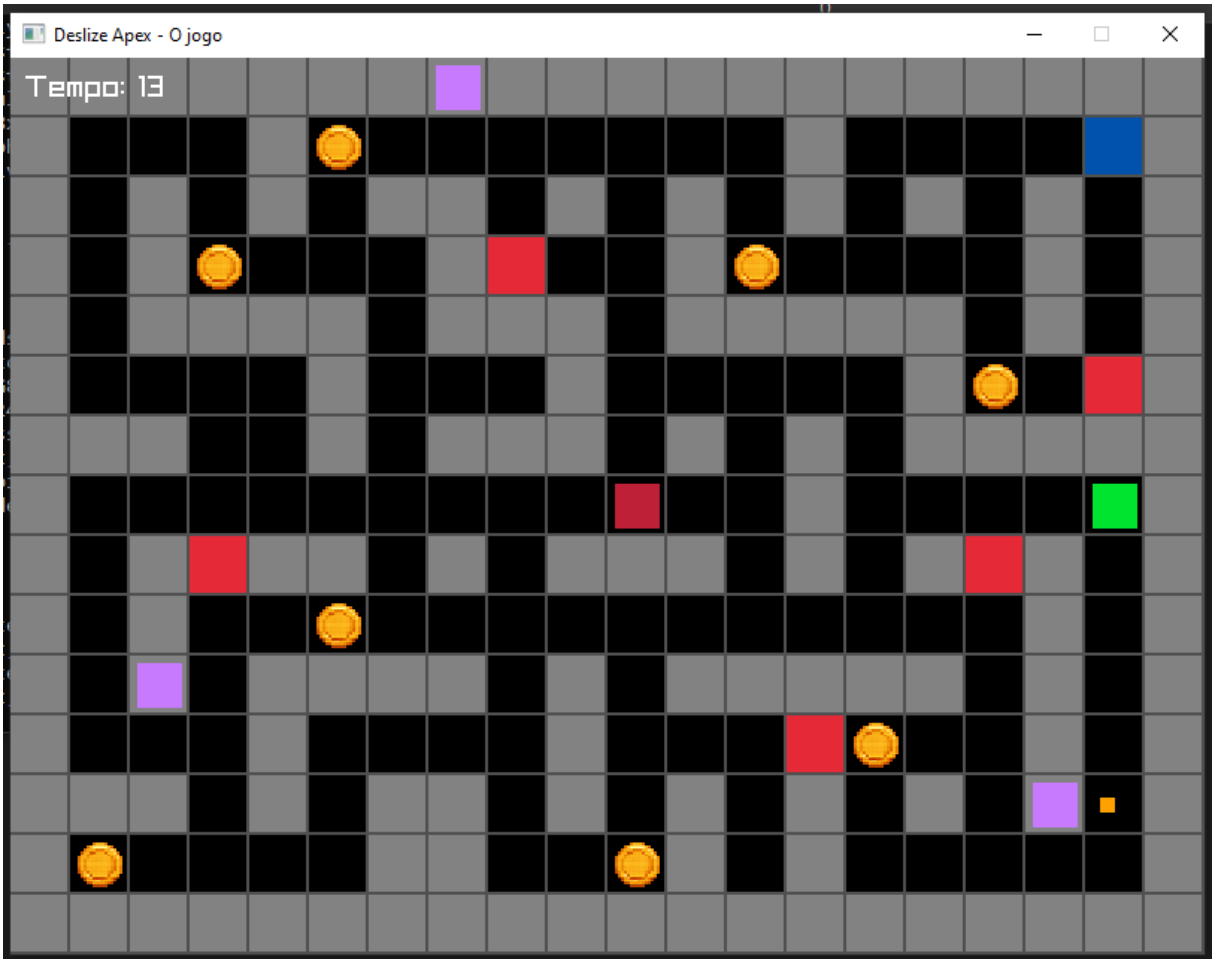

Figura 7 – Fase 3 (2025)

A Figura 7 exibe a Fase 3, que representa o maior desafio em "Deslize Apex". Nesta etapa, é introduzido o "Perseguidor", um inimigo que persegue o jogador ativamente pelo mapa. O limite de tempo, mais curto que na fase anterior, intensifica a dificuldade. Na imagem é possível o Perseguidor se aproximando do jogador. Os testes realizados validaram a eficácia do Perseguidor em sua navegação e perseguição, assim como a correta interação de todos os elementos de jogo em um cenário de alta pressão.

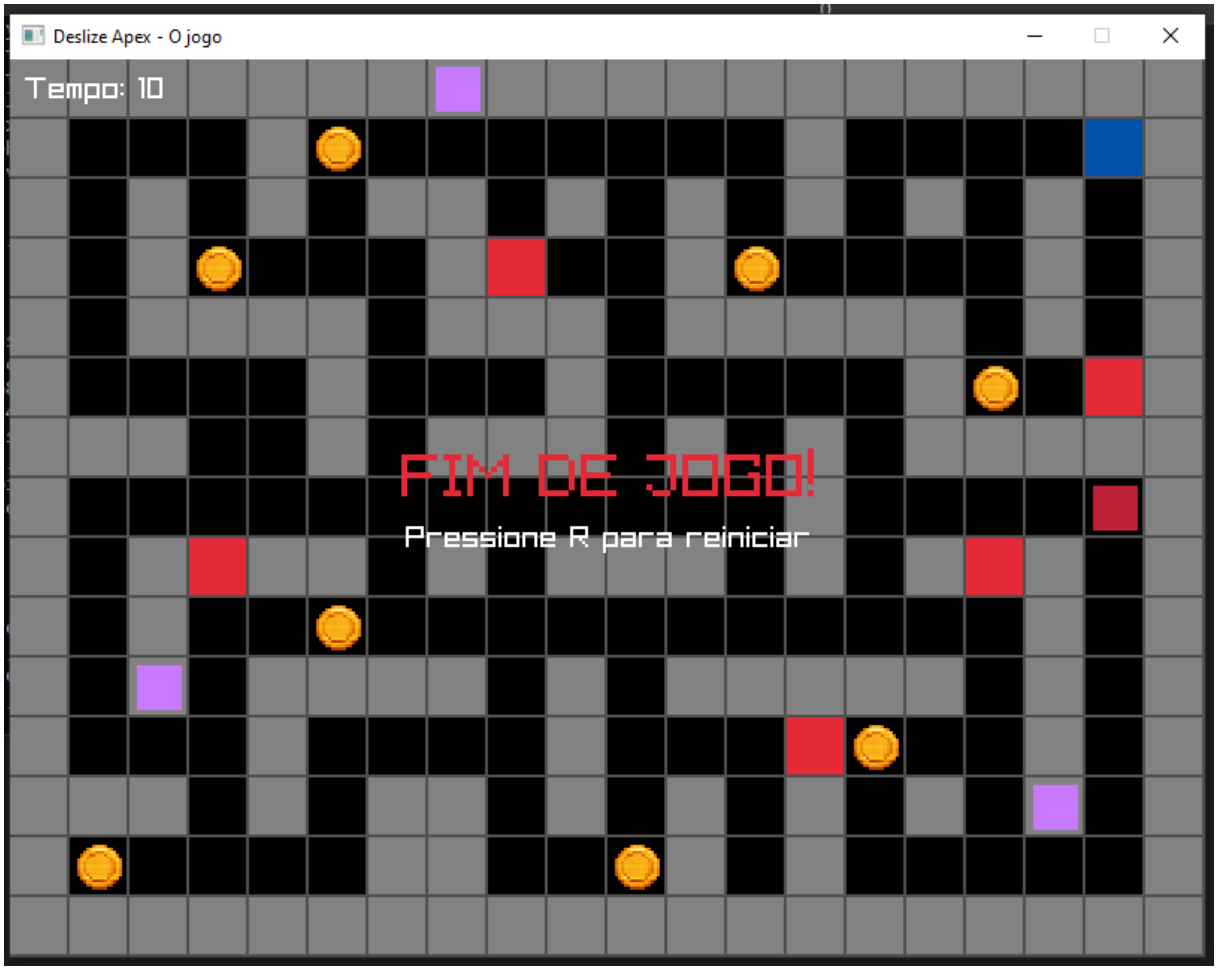

Figura 8 – Tela de Fim de Jogo (2025)

A Figura 8 ilustra a tela de "FIM DE JOGO!", que é apresentada quando o jogador não consegue superar os desafios. As condições para o fim de jogo incluem ser atingido por um projétil, colidir com o Perseguidor, cair em uma armadilha ou o esgotamento do tempo nas fases cronometradas. No momento em que o jogo termina, a música de fundo é interrompida e um som característico de derrota é tocado. A tela exibe a mensagem de encerramento e a instrução "Pressione R para reiniciar", permitindo ao jogador retornar à tela de início.

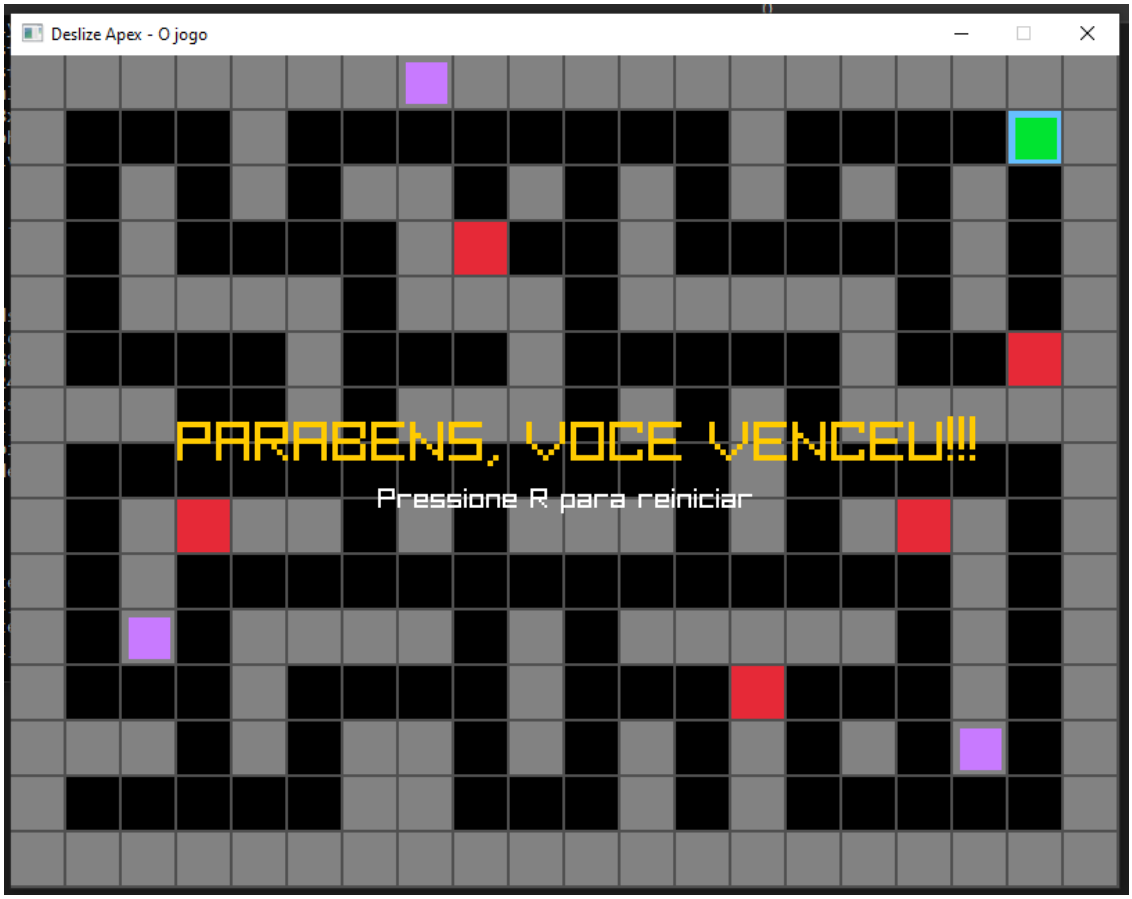

Figura 9 – Tela de Vitória (2025)

Por fim, a Figura 9 demonstra a tela de conclusão bem-sucedida do jogo, exibida quando o jogador completa todos os desafios da Fase 3, coletando todas as moedas e alcançando o portão de saída. Uma mensagem festiva, "PARABÉNS, VOCÊ VENCEU!!!", é mostrada, e, similarmente à tela de derrota, a música de fundo cessa para dar lugar a um som de vitória. A opção "Pressione R para reiniciar" também está presente, convidando o jogador a uma nova partida a partir da primeira fase.

Em resumo, os resultados obtidos demonstram que todas as mecânicas planejadas para "Deslize Apex" foram implementadas com sucesso. O fluxo de jogo, desde a tela de início, progressão pelas três fases com seus respectivos desafios, até as telas de vitória ou derrota, funciona de maneira consistente. A adição de elementos audiovisuais como texturas para moedas, música de fundo e efeitos sonoros para eventos chave enriquece a experiência do jogador, fornecendo feedback claro e aumentando a imersão.

4.3 Discussão

A análise dos resultados mostra que a jogabilidade do "Deslize Apex" é integrada e funcional. A mecânica de deslize e o sistema de colisão, que são a base da interação, se mostraram precisos e responsivos durante os testes. Sobre essa base sólida, os elementos de desafio foram bem-sucedidos em criar uma curva de dificuldade progressiva.

Os inimigos atiradores criaram zonas de perigo que exigem atenção do jogador, enquanto o Perseguidor da Fase 3 se tornou o principal fator de tensão do jogo, com sua capacidade de navegar pelo mapa e caçar o jogador ativamente. A progressão de dificuldade, com a introdução gradual de novos desafios como o *timer* e os diferentes inimigos, funcionou bem para manter o jogador engajado.

O jogo apresentou a estabilidade e a performance esperadas. A taxa de quadros se manteve fluida mesmo em cenas com mais ação, o que valida a eficiência da biblioteca *Raylib*. Esse bom desempenho também é um reflexo positivo da arquitetura orientada a objetos adotada. A organização do código em classes modulares foi fundamental para gerenciar a complexidade do projeto, permitindo que cada sistema fosse desenvolvido e testado de forma independente, o que contribuiu para a estabilidade do produto final.

Em resumo, a discussão dos resultados confirma que o "Deslize Apex" é um produto completo e funcional, que cumpre o que foi planejado. A metodologia e as ferramentas escolhidas provaram ser adequadas para o escopo do trabalho. Embora os testes tenham validado o jogo, reconhece-se que avaliações de usabilidade com mais jogadores poderiam fornecer dados adicionais para um balanceamento ainda mais refinado, um ponto que pode ser explorado em versões futuras.

5 CONCLUSÃO

Com base no desenvolvimento e nos resultados apresentados, conclui-se que o projeto do jogo "Deslize Apex" atendeu plenamente aos objetivos estabelecidos. A implementação foi realizada com sucesso, utilizando as ferramentas descritas na metodologia e, principalmente, aplicando de forma prática os conceitos da Programação Orientada a Objetos para estruturar o *software*.

O desenvolvimento do projeto permitiu um aprofundamento significativo em técnicas de lógica de jogos. A modelagem das entidades em classes, o gerenciamento das interações entre os objetos e a implementação da inteligência artificial para os inimigos foram os principais pontos de aprendizado técnico. Todos os objetivos planejados foram alcançados, e as funcionalidades foram implementadas de forma a garantir a jogabilidade desafiadora que foi proposta, como comprovado nas seções de Avaliação e Resultados.

Para trabalhos futuros, o projeto serve como uma base sólida para diversas melhorias. Sugere-se a adição de mais conteúdo, como novas fases com mecânicas e desafios diferentes, ou a inclusão de *power-ups* para o jogador.

Em resumo, o desenvolvimento do "Deslize Apex" consolidou na prática os conceitos de programação e *design* de jogos. O projeto não apenas resultou em um produto de *software* interativo e funcional, mas também serviu como uma valiosa experiência formativa na área de desenvolvimento de jogos com uma arquitetura orientada a objetos.

REFERÊNCIAS

A. ONLINE:

MICROSOFT. Vcpkg. Disponível em:<https://github.com/microsoft/vcpkg>. Acesso em: 9 jun. 2025

MICROSOFT. Visual Studio Community 2022. Disponível em:< https://visualstudio.microsoft.com/pt-br/vs/community/>. Acesso em: 9 jun. 2025.

PAC-MAN. *Pac-Man*. Disponível em:<https://pt.wikipedia.org/wiki/Pac-Man>. Acesso em: 10 jun. 2025.

PATEL, Amit. Introduction to A*. Red Blob Games. Disponível em:<https://www.redblobgames.com/pathfinding/a-star/introduction.html>. Acesso em: 10 jun. 2025.

RAYLIB. Examples. Disponível em:<https://www.raylib.com/examples.html>. Acesso em: 4 nov. 2024.

RAYSAN5. Raylib. Disponível em:< https://github.com/raysan5/raylib>. Acesso em: 9 nov. 2024

TOMB OF THE MASK. *Tomb of the Mask*. Disponível em:< https://en.wikipedia.org/wiki/Tomb_of_the_Mask>. Acesso em: 10 jun. 2025.

B. Livro:

DEITEL, H. M.; DEITEL, P. J. C++: como programar. 10. ed. São Paulo: Pearson, 2016.

GLOSSÁRIO

Bugs: Termo informal usado na computação para se referir a um erro, falha ou defeito no código de um programa. Um *bug* faz com que o *software* se comporte de maneira inesperada, como travar, exibir um gráfico incorreto ou não responder a um comando.

Debugging: É o processo de encontrar e corrigir erros no código de um programa. O *debugging* é realizado com a ajuda de uma ferramenta, o *debugger*, que permite ao programador executar o código passo a passo e inspecionar variáveis para identificar a origem de um problema.

Frame: Literalmente "quadro". Em jogos e computação gráfica, um *frame* é uma única imagem estática. Um jogo cria a ilusão de movimento ao exibir dezenas de *frames* por segundo, uma medida conhecida como *FPS*.

Loop: Em programação, é uma estrutura que repete um conjunto de instruções continuamente. No contexto de jogos, o *Game Loop* é o ciclo principal que mantém o jogo funcionando, repetindo rapidamente três passos: processar os comandos do jogador, atualizar a lógica do jogo (movimento, colisões, etc.) e desenhar o novo *frame* na tela.

Tiles: "ladrilhos". Em jogos 2D, *tiles* são pequenas imagens, geralmente quadradas, que se encaixam para construir um mapa ou cenário, como um mosaico. Um mapa baseado em *tiles* organiza o mundo do jogo em uma estrutura de grade, o que simplifica o design de níveis e a programação de colisões.

vcpkg: Um gerenciador de pacotes da Microsoft para a linguagem C++. Sua função é automatizar o processo de baixar, compilar e instalar bibliotecas externas (como a *Raylib*), facilitando sua integração a um projeto no *Visual Studio*.

Workload: Termo utilizado pelo *Visual Studio Installer* para se referir a um pacote de ferramentas e componentes pré-configurados para um tipo específico de desenvolvimento. No projeto, foi instalado o *workload* “Desenvolvimento de Desktop com C++”, que agrupa o compilador, as bibliotecas e outros itens necessários para criar aplicações em C++.

APÊNDICE A: RECURSOS UTILIZADOS

Áudios

- **Música de Fundo:** SKIFF, Eric. **A Night of Dizzy Spells**. 2008. Disponível em: <https://www.youtube.com/watch?v=qNRf5u1XoBM>. Acesso em: 6 jun. 2025.

- **Efeitos Sonoros:**

    - Efeito sonoro de coleta de moeda. YOUTUBE. Disponível em: <https://www.youtube.com/shorts/oQuYrD-jgWQ>. Acesso em: 6 jun. 2025.
    - Efeito sonoro de derrota do jogador. YOUTUBE. Disponível em: <https://www.youtube.com/shorts/QR9KJqM1Iys>. Acesso em: 6 jun. 2025.
    - Efeito sonoro de vitória. YOUTUBE. <https://www.youtube.com/watch?v=Zb-90sKtS5o>. Acesso em: 6 jun. 2025.

Gráficos

- **Textura da Moeda:** Gerada pelo autor utilizando ferramenta de Inteligência Artificial**.** Ferramenta: ChatGPT (OpenAI). 6 jun. 2025

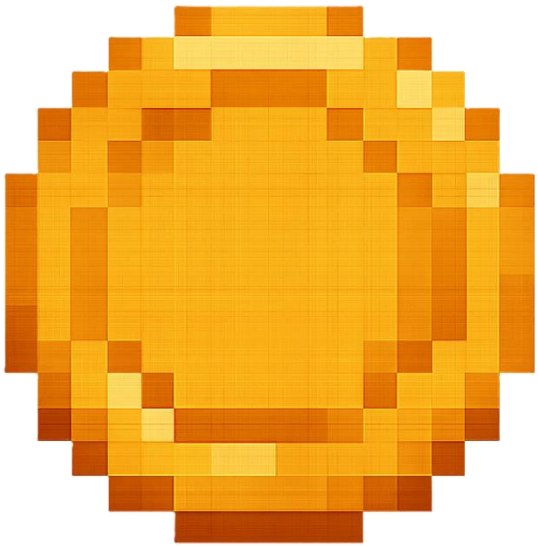

Figura 10 – Moeda de “Deslize Apex” (2025)