INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE SÃO PAULO

VINÍCIUS SILVA DE SAMPAIO

JOGO BRICK BREAKER

CAMPOS DO JORDÃO

2024

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE SÃO PAULO

JOGO BRICK BREAKER

Entrega do projeto de Desenvolvimento de um jogo em C++ à disciplina Programação Orientada a Objetos, do curso de Análise e Desenvolvimento de Sistemas no Instituto Federal de São Paulo de Campos do Jordão como requisito parcial para aprovação na disciplina, sob a orientação do professor Paulo Giovani de Faria Zeferino

CAMPOS DO JORDÃO

2024

RESUMO

O presente trabalho apresenta o desenvolvimento de um jogo eletrônico do tipo Brick Breaker em C++, utilizando a biblioteca gráfica raylib e o ambiente Visual Studio com suporte do gerenciador de dependências vcpkg. Inspirado nos clássicos Breakout e Arkanoid, o projeto visa proporcionar uma experiência nostálgica e educativa, explorando conceitos fundamentais de programação, como detecção de colisões, manipulação de eventos e renderização gráfica. A metodologia adotada foi iterativa, com ênfase na modularização e boas práticas de codificação, garantindo um produto funcional e expansível. O desenvolvimento incluiu a configuração do ambiente, a implementação de funcionalidades como movimentação e colisões, e testes contínuos para validação. A escolha das ferramentas foi fundamentada em sua simplicidade e eficiência, proporcionando um aprendizado prático e aprofundado. Os resultados demonstraram o cumprimento dos objetivos propostos, evidenciando a relevância do projeto como ferramenta pedagógica no estudo de programação.

Palavras-Chave: Programação; C++; Raylib; Desenvolvimento de jogos; Visual Studio; Educação; Breakout; Arkanoid.

ABSTRACT

This paper presents the development of a Brick Breaker-type video game in C++, using the raylib graphics library and the Visual Studio environment with support from the vcpkg package manager. Inspired by the classic Breakout and Arkanoid games, the project aims to provide a nostalgic and educational experience while exploring fundamental programming concepts such as collision detection, event handling, and graphical rendering. The methodology followed an iterative approach, emphasizing modularization and coding best practices to ensure a functional and extensible product. Development steps included setting up the environment, implementing features like movement and collision detection, and conducting continuous testing for validation. The choice of tools was based on their simplicity and efficiency, enabling practical and in-depth learning. The results demonstrated that the objectives were achieved, highlighting the project's relevance as an educational tool for programming studies.

Keywords: Programming; C++; Raylib; Game development; Visual Studio; Education; Breakout; Arkanoid;

LISTA DE ILUSTRAÇÕES

LISTA DE SIGLAS

IDE	*Integrated Development Environment* (Ambiente de Desenvolvimento Integrado)
OOP	*Object-Oriented Programming* (Programação Orientada a Objetos)
FPS	*Frames Per Second* (Quadros Por Segundo)

SUMÁRIO

1 INTRODUÇÃO

O desenvolvimento de jogos eletrônicos tem se consolidado como uma área multidisciplinar que alia conceitos de programação, design, e interação humano-computador. Com o avanço das tecnologias e a popularização dos dispositivos de entretenimento, jogos simples e desafiadores como o *Brick Breaker* mantêm seu apelo atemporal. Este trabalho tem como objetivo principal apresentar o processo de criação de um jogo denominado *Brick Breaker*, desenvolvido em C++ com a utilização da biblioteca gráfica *raylib*, a partir do *Visual Studio* como IDE e com suporte do gerenciador de dependências *vcpkg*.

o *Brick Breaker* desenvolvido neste projeto visa capturar essa essência e proporcionar uma experiência nostálgica e educativa. A simplicidade do design, que envolve a interação do jogador com uma plataforma para rebater uma bola e destruir blocos, oferece um estudo eficaz de conceitos de programação como manipulação de eventos, detecção de colisões e renderização de gráficos em tempo real.

1.1 Objetivos

O principal objetivo deste projeto foi o desenvolvimento de um jogo interativo que reproduzisse fielmente a dinâmica dos jogos *Brick Breaker* clássicos. Nesse processo, buscou-se aplicar conceitos fundamentais de programação orientada a objetos, além de aprofundar conhecimentos em estruturas de dados e gestão de eventos. Além disso, o projeto foi uma oportunidade de explorar a integração de bibliotecas gráficas. Para alcançar esse objetivo geral, foram definidos os seguintes objetivos específicos:

- Realizar uma investigação sobre os elementos fundamentais de jogos do gênero arcade, com foco em mecânicas de colisão e jogabilidade interativa;
- Integrar ferramentas como o *Visual Studio* e o *vcpkg* para facilitar o processo de desenvolvimento e gerenciamento de dependências;
- Aplicar conceitos de programação em C++ para implementar mecânicas de jogo, como movimentação e colisão;

- Propor um design de jogo que mantenha a simplicidade, mas incentive a estratégia e a persistência do jogador;
- Implementar as funcionalidades principais do jogo utilizando a biblioteca gráfica *raylib*;
- Integrar técnicas de programação modular e boas práticas de codificação para garantir a manutenção e expansibilidade do projeto;
- Testar e validar a funcionalidade e a experiência de usuário do jogo criado.

1.2 Justificativa

A escolha por desenvolver um jogo do tipo *Brick Breaker* foi motivada por sua relevância pedagógica no contexto do aprendizado de programação. Este gênero de jogo proporciona uma abordagem prática para explorar conceitos fundamentais, como lógicas de colisão, manipulação de eventos de entrada do usuário. Essas características tornam o projeto uma oportunidade rica para aplicar e consolidar conceitos de programação, especialmente em C++.

Além disso, a simplicidade estrutural do jogo possibilita que o foco esteja na qualidade da implementação, incentivando o domínio das ferramentas utilizadas e uma compreensão mais profunda dos elementos técnicos. O *Brick Breaker* também se destaca por ser um projeto acessível, adequado tanto para introduzir novos conceitos quanto para aperfeiçoar habilidades existentes.

No desenvolvimento do projeto, a utilização de ferramentas como o *Visual Studio* e o *vcpkg* desempenhou um papel central. O *Visual Studio*, uma IDE amplamente reconhecida por sua robustez e suporte a diversas linguagens, foi escolhido por oferecer uma interface intuitiva e recursos avançados, como depuração, autocompletar e sugestões de código, que agilizam o processo de desenvolvimento. Sua flexibilidade permitiu que a codificação fosse realizada de forma eficiente, ao mesmo tempo que facilitou a identificação e correção de erros ao longo do desenvolvimento.

Já o *vcpkg*, uma ferramenta de gerenciamento de pacotes, foi essencial para a instalação e manutenção das dependências do projeto, como a biblioteca gráfica *raylib*.

A utilização do *vcpkg* não apenas simplificou a configuração do ambiente de desenvolvimento, mas também garantiu a compatibilidade e integridade das ferramentas utilizadas. Essa combinação de tecnologias permitiu a criação de um ambiente produtivo e confiável para o desenvolvimento do jogo, reforçando ainda mais a escolha deste projeto como uma experiência enriquecedora e formativa.

1.3 Aspectos Metodológicos

O desenvolvimento do jogo *Brick Breaker* seguiu uma metodologia iterativa, iniciando com uma pesquisa sobre o jogo original para entender suas mecânicas e como aprimorá-las com tecnologias atuais. A escolha do *Visual Studio* 2022 como ambiente de desenvolvimento, junto com o *vcpkg* para gerenciar dependências, garantiu um processo eficiente. A biblioteca *raylib* foi selecionada pela sua simplicidade e alto desempenho em jogos 2D, facilitando a integração de gráficos e física.

O ciclo de desenvolvimento começou com a análise de requisitos e o desenvolvimento de um protótipo focado nas funcionalidades principais, como movimentação da bola e interação com os blocos. Testes contínuos, com ajustes iterativos, asseguraram a precisão das colisões, responsividade e estabilidade do jogo. A modularização da lógica central possibilitou fácil manutenção e expansão futura, resultando em um produto final funcional e estável.

1.4 Aporte Teórico

Na seção 2, serão apresentadas as ferramentas e tecnologias utilizadas ao longo do desenvolvimento do projeto, incluindo a escolha da linguagem de programação, da biblioteca gráfica e do ambiente de desenvolvimento. Além disso, será descrito o jogo criado, com foco em sua estrutura e nas principais mecânicas implementadas. Na seção 3, o processo de desenvolvimento será detalhado, abordando as etapas seguidas, desde a pesquisa inicial até a implementação das funcionalidades. A seção

4 trará uma análise dos resultados obtidos, comparando-os com os objetivos inicialmente estabelecidos para o projeto. Serão avaliados os aspectos técnicos e jogabilidade. Por fim, na seção 5, será apresentada a conclusão do trabalho, com uma reflexão sobre os pontos fortes e as áreas para melhorias no projeto.

2 FUNDAMENTAÇÃO TEÓRICA

Inspirado nos clássicos *Breakout* (1976) e *Arkanoid* (1986), que marcaram a história dos jogos de arcade com sua mecânica simples, mas altamente envolvente, o gênero *Brick Breaker* se consolidou com jogos onde os jogadores utilizam uma bola para destruir gradualmente uma parede de blocos ou elementos similares. Desde o lançamento do jogo original pela *Atari* em 1976, inúmeros clones e versões atualizadas foram desenvolvidos para diversas plataformas, mantendo a essência do desafio.

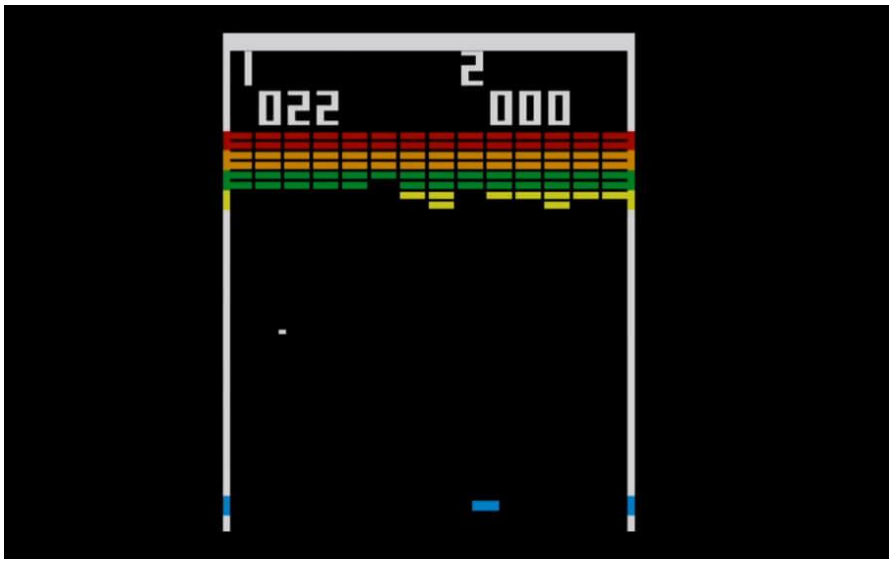

Figura 1 – *Breakout* (1976)

Figura *2* **–** *Arkanoid* (1986)

O desenvolvimento de jogos nesse estilo envolve conceitos essenciais, como gráficos, física de colisão, movimentação de objetos e eventos de entrada. No caso deste projeto, o jogo *Brick Breaker* foi escolhido como base, oferecendo uma mecânica simples, mas desafiadora, em que o jogador controla uma plataforma para rebater a bola e destruir os blocos.

2.1 Ferramentas e Bibliotecas

O *raylib*, uma biblioteca gráfica 2D de código aberto, foi escolhida devido à sua simplicidade e alta performance, facilitando a implementação das mecânicas de jogo. A integração do *raylib* foi feita de forma eficiente utilizando o *vcpkg*, que simplifica o gerenciamento de dependências. O *Visual Studio* 2022 foi a IDE escolhida, fornecendo recursos avançados de depuração e integração com controle de versões, essenciais para o desenvolvimento de jogos.

2.2 Programação Orientada a Objetos

A programação orientada a objetos foi aplicada para organizar e modularizar o código. Isso facilitou a manutenção do jogo e a possível expansão, como a adição de novos níveis ou funcionalidades. A OOP é uma prática recomendada em projetos de software, promovendo escalabilidade e organização.

3 PROJETO PROPOSTO

A metodologia adotada para o desenvolvimento do jogo em C++ seguiu uma abordagem sistemática e estruturada, com o objetivo de garantir a eficiência do processo e o cumprimento das diretrizes acadêmicas estabelecidas. O desenvolvimento foi conduzido de maneira faseada, com cada etapa cuidadosamente planejada para assegurar o alinhamento do projeto com os requisitos técnicos e acadêmicos.

O processo teve início com a fase de concepção do projeto, onde foram definidos os objetivos principais do jogo, as funcionalidades essenciais e os elementos de design que seriam implementados. Durante essa fase, foi essencial realizar um levantamento detalhado das mecânicas do jogo, dos requisitos de desempenho e das características que deveriam ser priorizadas na implementação, como a jogabilidade, os gráficos e a interação com o usuário. A definição clara desses elementos foi fundamental para garantir uma visão coesa e bem estruturada de todo o desenvolvimento, evitando ambiguidades e permitindo a adaptação das fases seguintes às necessidades do projeto.

Além disso, nesta fase inicial, também foram selecionadas as ferramentas e as tecnologias que seriam utilizadas, como o C++ para a programação, o *Visual Studio* para o ambiente de desenvolvimento e o *raylib* para as funcionalidades gráficas. Essas escolhas técnicas foram feitas com base na análise das características do jogo e das necessidades de desempenho, garantindo que a execução do projeto fosse realizada de maneira otimizada.

Essa fase de concepção foi fundamental, pois serviu como base para todas as etapas subsequentes do projeto. Ela permitiu que o desenvolvimento fosse conduzido de forma organizada e eficiente, com uma clara compreensão do que seria necessário em termos de código, design e testes, assegurando a criação de um jogo funcional e coeso.

3.1 Considerações Iniciais

A ideia central do projeto era criar um jogo *Brick Breaker*, escolha motivada por razões didáticas e pela simplicidade do gênero, que permite explorar conceitos fundamentais de programação gráfica, manipulação de objetos e eventos em tempo real de maneira prática e acessível.

3.2 Ferramentas Utilizadas

Para o desenvolvimento, foram selecionadas ferramentas específicas que garantiram a eficiência e a qualidade do projeto:

- Linguagem C++: Escolhida por sua robustez e alto desempenho em aplicações gráficas e de jogos.
- Biblioteca *raylib*: Utilizada por sua simplicidade e capacidade de oferecer uma interface amigável para desenvolvimento de jogos 2D. Sua instalação foi feita por meio do comando "*vcpkg install raylib:x64-windows*", que configurou a biblioteca para uso no projeto.
- *Visual Studio 2022 Community Edition*: Ferramenta de desenvolvimento integrada (IDE) que facilitou a codificação, depuração e testes do projeto. Sua edição Community, que é gratuita, foi baixada e instalada por meio do *Visual Studio Installer*, garantindo a instalação do *workload* "Desenvolvimento de *Desktop* com C++". Este *workload* inclui ferramentas essenciais, como o compilador C++ e a Biblioteca Padrão, que foram fundamentais para a compilação e execução do projeto.

Figura *3* **–** *Visual Studio* (2022)

- *vcpkg*: Gerenciador de pacotes utilizado para facilitar a integração e gerenciamento de bibliotecas externas, incluindo o *raylib*. O *vcpkg* foi obtido por meio de seu repositório no GitHub e descompactado no diretório raiz do sistema (C:). Após a extração, comandos como “boostrap-vcpkg.bat” e “*vcpkg integrate install*” foram executados para preparar a ferramenta e integrá-la ao Visual Studio, facilitando o reconhecimento das bibliotecas utilizadas.

3.3 Descrição do Processo de Desenvolvimento

1. Configuração do Ambiente de Desenvolvimento:
    - A primeira etapa consistiu em baixar e instalar o *Visual Studio 2022 Community Edition*, seguido da instalação da carga de trabalho específica para desenvolvimento com C++.
    - Em seguida, foi realizado o download e a instalação do gerenciador de pacotes *vcpkg* para facilitar a integração do *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 pela IDE.
2. Implementação do Código Base:
    - O projeto foi iniciado com a criação de uma aplicação de console em C++, para a qual foram importadas as bibliotecas necessárias, como #include “*raylib.h*”.
    - Estruturas básicas foram codificadas para representar elementos do jogo, como a bola, os blocos e a plataforma, juntamente com suas propriedades (posição, velocidade, cores).
3. Desenvolvimento das Funcionalidades Principais:
    - A lógica de movimentação da plataforma e da bola foi implementada utilizando cálculos de posição e velocidade, além de manipulação de

eventos de entrada, como o pressionamento de teclas, para controlar o movimento da plataforma.

- As colisões entre a bola e os blocos foram tratadas de forma eficiente, utilizando princípios de detecção de colisão e resposta, garantindo que a bola refletisse corretamente ao colidir com os objetos da tela e que os blocos fossem removidos ao serem atingidos.

4. Testes e Depuração:

    - Cada funcionalidade foi testada de forma incremental para garantir que os erros fossem detectados e corrigidos rapidamente.
    - Foram realizadas várias iterações de testes para refinar a jogabilidade e a resposta visual do jogo.

A escolha das ferramentas e a organização do processo metodológico permitiram um desenvolvimento eficaz e fluido. Cada etapa foi pensada para maximizar a produtividade e assegurar que o projeto final estivesse alinhado aos objetivos propostos. O uso do *raylib* facilitou a implementação de gráficos e a gestão de eventos, enquanto o Visual Studio e o *vcpkg* agilizaram o processo de desenvolvimento e integração das bibliotecas necessárias.

4 AVALIAÇÃO

O jogo *Brick Breaker* demonstrou que todas as mecânicas essenciais, como a movimentação da bola, colisões com os blocos e o controle da plataforma, funcionaram corretamente. O jogo apresentou boa responsividade aos comandos do usuário e manteve a estabilidade durante as partidas. No entanto, pequenos ajustes podem ser feitos na dificuldade, e a implementação de novos recursos, como mais níveis e efeitos visuais, pode aprimorar a experiência do jogador. De modo geral, o jogo atendeu aos objetivos propostos, oferecendo uma jogabilidade divertida e funcional.

4.1 Condução

A condução da avaliação foi realizada por meio de uma série de testes focados na validação das funcionalidades e da qualidade do jogo. Inicialmente, foram definidos os critérios principais de avaliação: fluidez do jogo, precisão das colisões, resposta aos comandos do usuário e estabilidade geral da aplicação.

Os testes foram conduzidos diretamente no ambiente de desenvolvimento, utilizando o *Visual Studio* 2022. Diversos cenários de jogo foram simulados para verificar as interações entre a bola, a plataforma e os blocos, além de monitorar o desempenho da aplicação, especialmente a taxa de quadros por segundo (FPS) e a resposta aos comandos. Também foram realizados testes de jogabilidade, com ajustes na dificuldade, para garantir que o jogo oferecesse desafios progressivos, mantendo a diversão.

A experiência do usuário foi igualmente avaliada, com foco na fluidez das animações e na intuitividade do controle do a plataforma. O código foi revisado e ajustado continuamente para corrigir pequenos bugs e aprimorar a detecção de colisões. Durante o processo de avaliação, foram coletados dados sobre o desempenho do jogo e a interação com o usuário, com o intuito de identificar possíveis áreas de melhoria, como a otimização das colisões.

Com base nos testes realizados, ajustes iterativos foram feitos para garantir que os requisitos de qualidade fossem atendidos, oferecendo uma experiência de jogo mais precisa e estável.

4.2 Resultados

Nessa seção, são apresentados os resultados obtidos durante o desenvolvimento e testes do jogo, com base nas funcionalidades implementadas e na análise de desempenho. As evidências coletadas serão descritas a seguir.

A primeira série de testes foi realizada para verificar a funcionalidade principal do jogo: o movimento da plataforma e a interação com a bola e os blocos. As imagens

abaixo ilustram o jogo em funcionamento, validando a correta aplicação da lógica de colisão.

Os testes subsequentes foram focados na estabilidade do jogo, observando o comportamento do sistema durante longos períodos de execução e em diferentes condições de interação do usuário. As imagens mostram a resposta do jogo ao controle do usuário e a fluidez das animações, com o controle da plataforma sendo preciso e a bola reagindo corretamente ao impacto nos blocos.

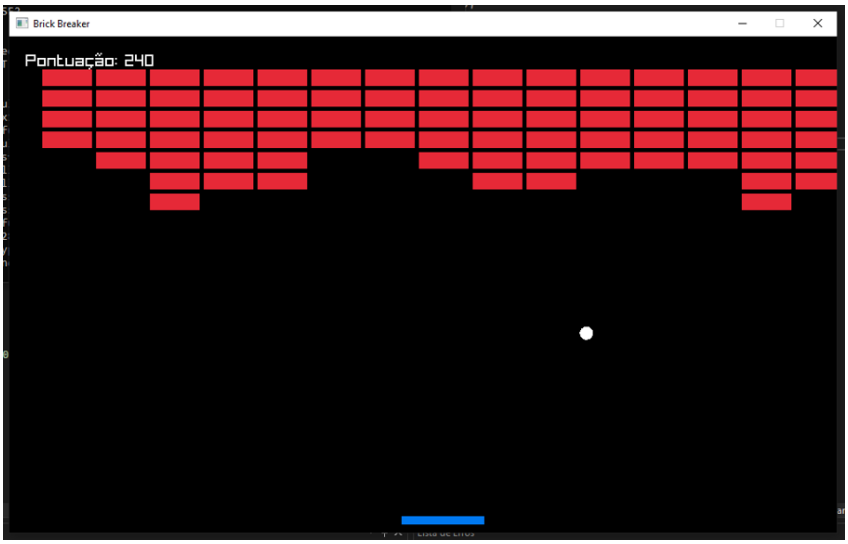

Figura *4* **–** tela do jogo em funcionamento (2024)

A Imagem acima mostra o jogo em pleno funcionamento, evidenciando a interação entre a plataforma controlada pelo jogador e a bola que destrói os blocos. É possível observar a dinâmica de movimentação e colisão, com a bola quicando e os blocos sendo destruídos gradualmente. A imagem demonstra que as mecânicas de movimento, colisão e destruição estão funcionando corretamente, garantindo uma jogabilidade fluida e sem falhas.

Figura *5* **–** tela de vitória do jogo (2024)

A Imagem acima ilustra o momento de vitória, quando o jogador destrói todos os blocos do cenário, alcançando o objetivo do jogo. Neste ponto, o sistema exibe uma tela de celebração, indicando que o jogador completou com êxito a missão proposta. A imagem reflete a mecânica de vitória, recompensando o jogador com uma mensagem de "Você Venceu!", conforme implementado no jogo.

Figura *6* **–** tela de fim de jogo (2024)

A Imagem acima apresenta o encerramento do jogo, com a mensagem de "FIM DE JOGO" exibida, sinalizando que o jogador perdeu todas as suas vidas disponíveis. Este momento marca o término da partida, indicando que o objetivo de destruir todos os blocos não foi atingido. A interface é projetada para refletir a falha do jogador, evidenciando a mecânica de perda de vidas, elemento central para o desafio do jogo.

4.3 Discussão

A partir dos resultados apresentados, é possível analisar o desempenho e a funcionalidade do jogo, bem como a eficácia das técnicas utilizadas para a implementação. A análise das imagens e dos testes realizados mostrou que a lógica de colisão, que é um dos elementos centrais do jogo, funcionou corretamente, sem apresentar falhas durante a interação com a plataforma e os blocos. Esses achados são consistentes com o esperado, considerando a simplicidade da mecânica do jogo e a utilização do *raylib*, uma biblioteca conhecida por sua eficiência em projetos 2D.

Em relação à estabilidade do jogo, os testes indicaram que o desempenho foi satisfatório, com a taxa de FPS mantendo-se constante mesmo após longos períodos de execução. Esse comportamento está de acordo com o que seria esperado em um jogo com mecânicas simples e otimizações para a renderização de gráficos 2D.

Os resultados obtidos corroboram a eficácia do ambiente de desenvolvimento escolhido, o Visual Studio, juntamente com a integração do *raylib*, para a construção de um jogo com boa performance e estabilidade. Embora não tenha sido observada a necessidade de ajustes significativos durante os testes, a utilização de testes mais avançados de performance e a comparação com outros jogos do gênero podem fornecer insights adicionais sobre como melhorar a eficiência do código.

Com base nos resultados, é possível concluir que o jogo atendeu aos objetivos propostos, tanto em termos de funcionalidade quanto de desempenho. No entanto, para futuras versões do jogo, recomenda-se a implementação de funcionalidades adicionais, como níveis progressivos de dificuldade e maior variedade de elementos gráficos, a fim de enriquecer a experiência do usuário.

5 CONCLUSÃO

Com base no desenvolvimento e nos resultados obtidos, conclui-se que o projeto do jogo "*Brick Breaker*" atendeu aos objetivos estabelecidos. A implementação do jogo foi realizada com sucesso, utilizando as ferramentas descritas na metodologia e aplicando conceitos de programação em C++ e desenvolvimento com a biblioteca *raylib*.

A experiência de desenvolvimento permitiu o aprofundamento em técnicas de lógica de jogos, gerenciamento de colisões e manipulação de gráficos em C++.

Todos os objetivos planejados foram alcançados, conforme demonstrado nas seções anteriores. As funcionalidades foram implementadas de forma a garantir a jogabilidade desejada, e as capturas de tela apresentadas comprovam a efetividade do produto final.

Para aprimorar o projeto, são sugeridas futuras melhorias, como a adição de novos níveis, incremento de elementos visuais e sonoros e a possibilidade de multiplayer local. Essas sugestões visam expandir a experiência do usuário e aumentar o apelo do jogo.

Em resumo, o desenvolvimento do "*Brick Breaker*" serviu como uma aplicação prática de conceitos de programação e design de jogos, proporcionando uma base sólida para futuros projetos e estudos na área de desenvolvimento de software.

REFERÊNCIAS

A. ONLINE:

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

MICROSOFT. Visual Studio Community. Disponível em: <https://visualstudio.microsoft.com/pt-br/vs/community/>. Acesso em: 9 nov. 2024.

MICROSOFT. Vcpkg. Disponível em: <https://github.com/microsoft/vcpkg>. Acesso em: 9 nov. 2024.

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

HERO CONCEPT. A brief history of brick breaker video games. Disponível em: <https://www.heroconcept.com/a-brief-history-of-brick-breaker-video-games/>. Acesso em: 10 nov. 2024.

WIKIPEDIA. Brick Breaker. Disponível em: <https://en.wikipedia.org/wiki/Brick_Breaker>. Acesso em: 10 nov. 2024

B. Livro:

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