INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE SÃO PAULO

ANDRE LUIZ LIMA RODRIGUES

PROJETO JOGO RAYLIB: TETRIS

CAMPOS DO JORDÃO

2024

RESUMO

A indústria de jogos digitais é uma das maiores forças culturais e econômicas no mundo atual, movimentando bilhões de dólares e superando até outros setores de entretenimento como cinema e música. Este trabalho visa reviver o clássico *Tetris*, um dos jogos mais influentes da história dos videogames, utilizando ferramentas modernas de desenvolvimento. O objetivo é criar uma versão do jogo que resgate a experiência dos arcades, ao mesmo tempo em que utilize as metodologias e tecnologias contemporâneas para envolver tanto jogadores antigos quanto novos. A linguagem escolhida para o desenvolvimento é o *C++*, com o auxílio da biblioteca *Raylib* para facilitar a criação gráfica, e o editor de código *Visual Studio Code*. O projeto também explora os princípios de *Programação Orientada a Objetos* (POO), como abstração e herança, aplicados na construção do código do jogo. A escolha do *Tetris* é justificada por sua simplicidade aparente, mas que exige raciocínio lógico e uma boa compreensão de programação, sendo ideal para o desenvolvimento de habilidades em lógica computacional e design de jogos. A pesquisa e o desenvolvimento do projeto também consideram o legado do *Tetris* na indústria de jogos e sua resiliência ao longo das décadas, permanecendo relevante mesmo com o avanço das tecnologias e dos gráficos no mercado de games.

Palavras-Chave: Desenvolvimento de Jogos; Tetris; Raylib; C++; Jogos Digitais; Programação Orientada a Objetos(POO).

ABSTRACT

The digital gaming industry is one of the largest cultural and economic forces in the modern world, generating billions of dollars and surpassing even other entertainment sectors like cinema and music. This project aims to revive the classic Tetris, one of the most influential games in the history of video games, using modern development tools. The goal is to create a version of the game that captures the arcade experience, while also incorporating contemporary methodologies and technologies to engage both old and new players. The chosen programming language for development is C++, with the aid of the Raylib library to simplify graphical creation, and the Visual Studio Code code editor. The project also explores the principles of Object-Oriented Programming (OOP), such as abstraction and inheritance, applied in the game's code structure. The choice of Tetris is justified by its apparent simplicity, which, in reality, requires logical thinking and a solid understanding of programming, making it ideal for developing skills in computational logic and game design. The research and development of the project also take into account Tetris' legacy in the gaming industry and its resilience over the decades, remaining relevant even as technologies and graphics in the gaming market continue to advance.

Keywords: Game Development; Tetris; Raylib; C++; Digital Games; Object Oriented Programming (OOP).

LISTA DE ILUSTRAÇÕES

LISTA DE ALGORITMOS

LISTA DE SIGLAS

IFSP Instituto Federal de Educação, Ciência e Tecnologia de São Paulo

XML *Extensible Markup Language*

POO *Programação Orientada a Objeto*

SUMÁRIO

1 INTRODUÇÃO

A indústria de jogos digitais, também conhecida como videogames, é, sem dúvida, uma das maiores forças culturais e econômicas do mundo moderno. Nos últimos anos, ela se consolidou não apenas como uma forma de entretenimento, mas como um fenômeno global que transcende barreiras geográficas, demográficas e sociais. Estima-se que a indústria de jogos movimenta bilhões de dólares anualmente, ultrapassando até mesmo outros setores tradicionais de entretenimento, como o cinema e a música (RIBEIRO, 2023).

Partindo desse contexto, os jogos digitais sempre estiveram na vanguarda do desenvolvimento tecnológico global, impulsionando a criação e adoção de tecnologias cada vez mais sofisticadas, poderosas e acessíveis. Esse ciclo contínuo de inovação tem permitido que o mercado de jogos cresça de forma exponencial, alcançando novas fronteiras e levando os jogos a lugares cada vez mais distantes, tanto em termos geográficos quanto em sua diversidade de plataformas e experiências.

De forma a dar continuidade ao legado dos desenvolvedores de jogos, este trabalho busca criar e desenvolver um jogo, embora não com a complexidade dos títulos modernos, mas sim com a proposta de reviver um clássico que muitos já jogaram em algum momento de suas vidas: o Tetris. Apesar de sua aparência simples e de ser frequentemente considerado fácil, o Tetris apresenta um grau de complexidade interessante, especialmente quando se considera a dinâmica de seu design, a lógica de programação envolvida e o desafio que ele oferece ao jogador.

1.1 Objetivos

Este trabalho tem como objetivo resgatar a experiência de jogar jogos de arcades clássicos, relembrando os jogadores mais velhos e apresentando aos mais jovens o charme e a nostalgia dos primeiros videogames e jogos do passado. Com esse propósito, busca-se a criação e o desenvolvimento de um jogo de Tetris que mantenha os elementos clássicos que tornaram o jogo icônico, ao mesmo tempo em que se comunica com as novas gerações. Além disso, o projeto utiliza técnicas modernas de desenvolvimento de jogos, com foco em um processo rápido e flexível, capaz de se

adaptar às novas tecnologias e práticas do mercado atual, garantindo uma experiência envolvente e acessível a todos.

1.2 Justificativa

A escolha de realizar este trabalho surge da necessidade de preservar a história dos videogames clássicos, ao mesmo tempo em que aprofunda o entendimento sobre a origem dos jogos e aqueles que marcaram a história. Além disso, busca-se explorar as técnicas de desenvolvimento e criação de jogos contemporâneos, que aprimoram a forma de produzir. O objetivo é, assim, desenvolver o jogo clássico Tetris ao que conecta o legado dos jogos antigos com as inovações tecnológicas atuais, mostrando como as metodologias e as ferramentas de desenvolvimento evoluíram ao longo do tempo.

Dessa forma, o desenvolvimento do Tetris não apenas contextualiza o passado, por ser um jogo que marcou o mundo com sua simplicidade, mas também por apresentar camadas de complexidade que exigem pensamento lógico e tomadas de decisão rápidas na forma como se joga. Além disso, o jogo faz alusão à sua resistência e adaptabilidade no decorrer das décadas, mantendo-se relevante e influente até hoje, um testemunho de sua atemporalidade e impacto duradouro na indústria de games.

1.3 Aspectos Metodológicos

Este trabalho foi fundamentado em pesquisas bibliográficas, que forneceram a base teórica necessária para o desenvolvimento do projeto. Após a construção da parte teórica, foi iniciado o processo de criação do jogo.

O objetivo deste estudo é o desenvolvimento do clássico jogo Tetris, utilizando ferramentas e metodologias modernas que estabelecem uma conexão entre a história dos videogames e as inovações tecnológicas atuais. A seguir, apresentamos um resumo das metodologias adotadas no desenvolvimento deste projeto.

Para a implementação, utilizou-se o Visual Studio Code, uma ferramenta popular e de fácil instalação, que oferece suporte a diversas linguagens de programação. A linguagem escolhida para o desenvolvimento foi o C++, reconhecida por sua eficiência

e flexibilidade, sendo uma opção ideal para a criação de jogos. Para a parte gráfica, foi utilizada a biblioteca Raylib, amplamente adotada por desenvolvedores iniciantes devido à sua simplicidade e facilidade de integração, o que facilitou a implementação dos recursos visuais do jogo.

1.4 Aporte Teórico

No embasamento teórico, destacam-se as obras C++: How to Program, de Deitel (2006), e Programação Orientada a Objetos, de Maitino Neto (2018). O primeiro livro foi fundamental para fornecer uma base sólida sobre a linguagem C++, com um foco teórico que permitiu uma compreensão aprofundada de seus conceitos e estruturas. Já o livro de Maitino Neto contribuiu com uma abordagem mais prática, facilitando a aplicação dos princípios de Programação Orientada a Objetos (POO) no desenvolvimento do projeto e proporcionando uma melhor integração dos conceitos de POO no código.

Além desses livros, artigos acadêmicos também desempenharam um papel importante na formação do embasamento teórico. Destaca-se o estudo de Medeiros e Martins (2023), que analisa o impacto histórico do Tetris. Segundo os autores, o jogo não só revolucionou o mercado de videogames, mas também se consolidou como uma excelente ferramenta de aprendizado em várias áreas da computação. O Tetris pode ser utilizado para explorar desde o uso de novas tecnologias até o desenvolvimento de habilidades em lógica computacional e design visual, áreas essenciais para o avanço do campo dos jogos digitais e da programação.

Essas fontes teóricas forneceram uma base robusta para a execução do projeto, permitindo uma abordagem mais técnica e fundamentada no desenvolvimento do jogo permitindo uma rápida evolução nas partes de absorção de conteúdos necessárias para a realização do projeto.

2 FUNDAMENTAÇÃO TEÓRICA

Nesta seção, serão apresentadas de forma detalhada a metodologia adotada no desenvolvimento do projeto, o projeto em si e as etapas realizadas até a conclusão do jogo. Também serão discutidos os principais desafios enfrentados e as ferramentas utilizadas ao longo do processo.

2.1 Considerações Iniciais

O registro mais antigo de um videogame remonta a 1958, quando William Higinbotham, com o objetivo de fazer uma demonstração para o público na exposição permanente do Brookhaven National Laboratory, em Long Island, Estados Unidos, criou o jogo "Tennis for Two". Esse jogo, desenvolvido para um osciloscópio, é considerado um dos primeiros exemplos de entretenimento interativo eletrônico (KLÜSER; SALAZAR HERRERA; CARMO; GAZZIRO, 2023) e (LUZ, 2010).

Anos mais tarde veio o Tetris, um videogame conhecido como um dos jogos mais conhecidos do mundo. Criado pelo programador russo Alexey Pajitnov em 1984, enquanto trabalhava no centro de Computação da Academia de Ciências da União Soviética. O Tetris rapidamente se espalhou pelo mundo angariando milhares de jogadores devido a jogabilidade única para a época ao mesmo tempo que exigia capacidades cognitivas e raciocínio logico para a escolha de decisões no decorrer do jogo (MEDEIROS; MARTINS, 2024).

Neste ano, o Tetris completa 40 anos e, mesmo após quatro décadas, continua a ser um fenômeno no cenário competitivo, atraindo milhares de fãs que acompanham suas competições. Esse fato pode parecer curioso, especialmente considerando que a indústria de jogos atualmente oferece títulos mais complexos, viciantes e com gráficos altamente sofisticados. No entanto, o Tetris permanece relevante e conquistando novos jogadores, o que demostra sua resiliência e apelo atemporal no mundo dos videogames.

No contexto deste trabalho, o Tetris oferece uma abordagem simples do ponto de vista computacional, facilitando a compreensão de conceitos fundamentais, como o

uso de estruturas de classe e herança. Além do aprimoramento da lógica de programação em sistemas modulares. Vale ressaltar que, embora pareça simples à primeira vista, o Tetris representa um desafio significativo para iniciantes, devido aos complexos cálculos envolvidos na detecção de colisões e na gestão das interações entre as peças.

2.2 Ferramentas Utilizadas

Para a criação do jogo Tetris, foram utilizadas as seguintes ferramentas: o Visual Studio Code para a codificação do projeto, a linguagem de programação C++ e a biblioteca Raylib configurada para C++. Além disso, o draw.io foi empregado para a criação de um diagrama de classes, que ilustra a estrutura e as relações entre as classes utilizadas no projeto. A seguir, será apresentado um detalhamento sobre cada uma dessas ferramentas.

Visual Studio Code: O Visual Studio Code é um editor de código-fonte desenvolvido pela Microsoft, projetado para ser leve, rápido e compatível com os principais sistemas operacionais, como Linux, Windows e macOS. Esses pontos contribuíram para sua popularidade, se tornando um dos editores de código mais utilizados. Além disso, sua vasta gama de atalhos, extensões e a possibilidade de personalizar componentes na instalação, oferecem alta flexibilidade, permitindo uma rápida adaptação às necessidades dos mais diversos tipos de profissionais (MICROSOFT, 2024).

Linguagem C++: “O C++ desenvolveu-se a partir do C, que se desenvolveu a partir de duas linguagens de programação anteriores o B e o BCPL.” (DEITEL; 2006, p.6). Dennis Ritchie foi o responsável pelo desenvolvimento da linguagem C na *Bell Laboratories*, aproveitando conceitos e técnicas das linguagens B e BCPL. A linguagem C rapidamente se popularizou e foi disseminada nas mais diversas áreas da computação, sendo adotada por inúmeras empresas e organizações até os dias de hoje. Em1980, ainda na *Bell Laboratories*, Bjarne Stroustrup criou uma extensão da linguagem C chamada de C++, introduzindo recursos e aprimoramentos, sobretudo acerca do paradigma da Programação Orientada a Objetos (POO). Hoje em dia o C++ considerado uma linguagem diferente da linguagem C, perdeu relevância no cenário atual das linguagens de computação devido a popularização de outras linguagens,

entretanto o C++ continua sendo uma das linguagens mais rápidas e mais flexíveis, permitindo o desenvolvimento de uma ampla variedade de projetos (DEITEL; 2006).

Biblioteca Raylib: O raylib é uma biblioteca de código aberto e multiplataforma, que foi desenvolvido por Ramon Santamaria para facilitar criação de jogos 2D e 3D. O raylib não fornece uma documentação típica, mas sim uma lista de funcionalidades que o usuário pode ir testando e aprendendo mais sobre a mesma. Uma das principais razoes pela raylib ser tão utilizada é pela sua facilidade de combinar com outras bibliotecas externas para construir o projeto. A arquitetura do raylib é uma biblioteca altamente modular. Tudo está contido dentro dele em um pequeno número de módulos bem definidos. Um dos principais fatos que tornam o raylib uma plataforma utilizada é o público de sua comunidade que continuamente continua criando jogos e cada vez mais melhorando e adicionando funcionalidades na biblioteca (RAYLIB, 2024).

Draw.io: O draw.io é uma ferramenta online que oferece soluções para a criação e manipulação de diagramas diretamente na web. Com uma ampla gama de funcionalidades, permite a criação de fluxogramas, mapas mentais, diagramas de rede, plantas baixas, entre outros. Sua popularidade se deve à sua versatilidade, especialmente nas áreas de desenvolvimento de software e infraestrutura de TI, oferecendo diversos modelos prontos para facilitar a diagramação e organização visual de processos e sistemas (DRAW.IO, 2024).

2.3 Projeto Desenvolvido

O projeto desenvolvido teve como objetivo a criação de um jogo utilizando a biblioteca Raylib, com o código implementado na linguagem C++. Durante o desenvolvimento, o foco foi criar um jogo simples, mas que incorporasse conceitos e técnicas da disciplina de Programação Orientada a Objetos (POO), propondo aprimorar os conhecimentos sobre o tema e demonstrar domínio na aplicação da lógica de programação.

O jogo escolhido foi o Tetris, que à primeira vista, pode parecer simples o suficiente para ser implementado em um único arquivo de código. No entanto, sua estrutura revela um certo grau de complexidade, especialmente quando se adota o paradigma

da POO. Essa abordagem não só facilita a organização e a manutenção do código, como também permite uma melhor modularização e escalabilidade do projeto.

Todo jogo é composto por duas partes principais: a primeira envolve a definição de variáveis e a criação dos objetos que compõem o jogo. A segunda, o Game Loop, responsável por atualizar as posições dos objetos, calcular colisões e gerenciar as interações entre eles. Esse loop funciona de maneira semelhante a um moinho de rio, girando continuamente até que o jogo seja encerrado.

```c
// Exemplo Game Loop
#include "raylib.h"

int main() {
    // Inicializa a janela com largura 800 e altura 600, e o título "Raylib"
    InitWindow(800, 600, "Raylib");

    // Verifica se a janela foi inicializada corretamente
    while (!WindowShouldClose()) {
        // Começa o desenho na tela
        BeginDrawing();
        // Limpa a tela com a cor de fundo (neste caso, branco)
        ClearBackground(RAYWHITE);
        // Termina o desenho na tela
        EndDrawing();
    }
    // Fecha a janela
    CloseWindow();
    return 0;
}
```

Algoritmo 01: Exemplo Game Loop (2024).

No Tetris, a cada iteração do jogo, os objetos são desenhados na tela com suas novas posições, juntamente com os cálculos necessários para verificar colisões e executar outras funcionalidades, como limpar a tela quando uma linha é completamente preenchida e mover as peças para baixo quando uma linha é apagada.

O jogo é baseado em uma grade (grid) onde blocos compostos por quatro pequenas peças caem continuamente. O jogador deve posicionar esses blocos de forma estratégica para completar linhas na grade. Quando uma linha é completamente preenchida, ela é apagada, e as linhas acima dela descem, gerando pontos para o jogador. No entanto, se a pilha de blocos atingir o limite superior da tela, onde as peças

começam a cair, o jogo é encerrado e o jogador perde.

Para a criação do jogo, foi elaborado um diagrama de classes que ilustra todas as classes presentes na estrutura do jogo. Utilizando o conceito de abstração, o diagrama representa o mundo do jogo em termos de objetos, incluindo seus nomes, propriedades e métodos. Além disso, foi aplicada a herança, um princípio da Programação Orientada a Objetos (POO), onde uma classe filha herda características e métodos de uma classe pai (MAITINO NETO, 2018).

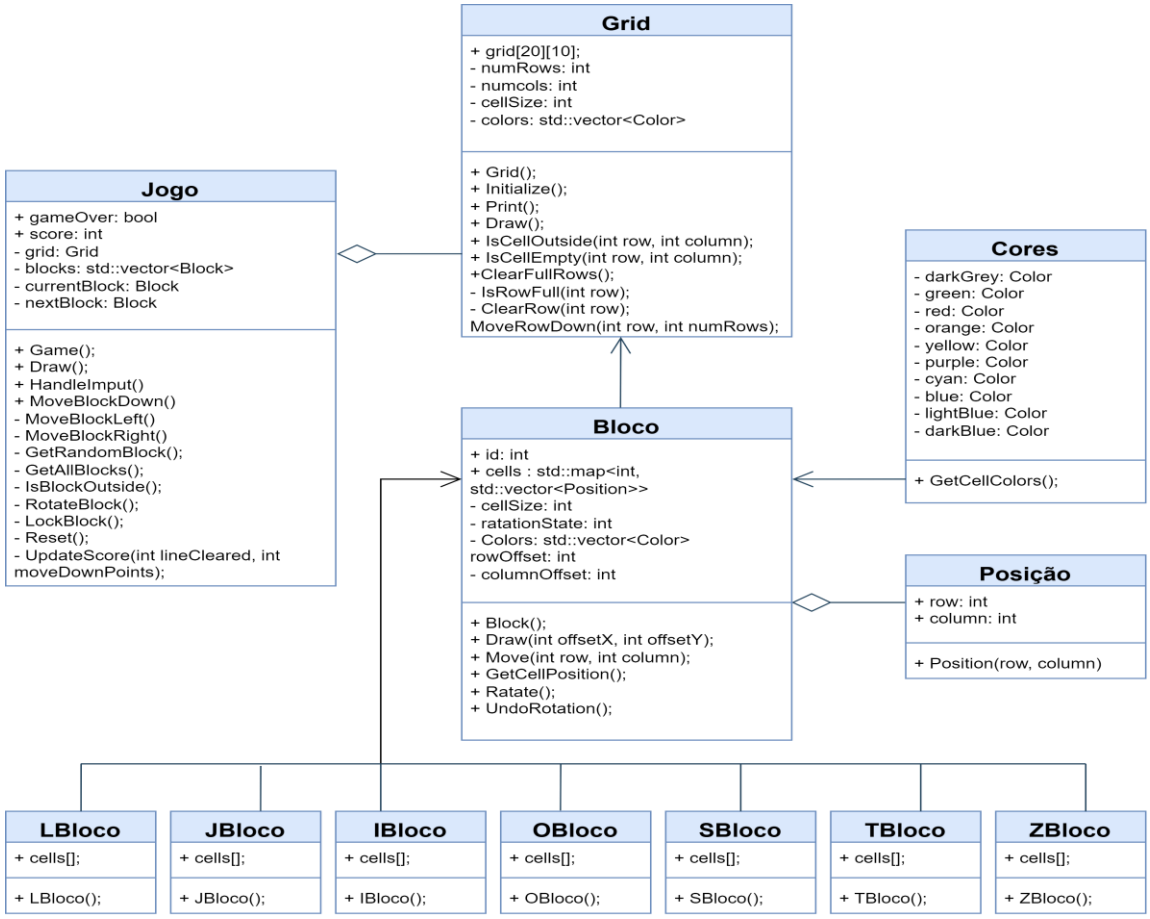

Figura 01: Diagrama de Classes do Jogo Tetris (2024).

O diagrama de classes representa os principais componentes do jogo, começando com a classe **Jogo**, que, ao ser inicializada, chama a classe **Grid**. Esta, por sua vez, cria o vetor "malha", onde o jogo ocorre. Em seguida, a classe **Bloco** cria os blocos, e suas classes filhas — **Lbloco, Jbloco, Ibloco, Obloco, Tbloco, Sbloco e Zbloco** — definem os tipos específicos de blocos a serem utilizados. A cor de cada

bloco é atribuída pela classe **Cores**, enquanto a **Posição** determina a localização de cada bloco na grade.

Em C++, as classes não são apenas cabeçalhos, mas sim estruturas que definem tanto os dados quanto os comportamentos (métodos) dos objetos. Os cabeçalhos (arquivos.h) contêm as declarações das classes, enquanto os arquivos de implementação (normalmente com a extensão arquivo.cpp) contêm o código dos métodos e funções que operam sobre esses dados. A seguir, apresentamos um breve resumo de cada uma das classes e seus construtores:

Construtores do Jogo:

Função: Gerencia a lógica do jogo, como os movimentos dos blocos, o controle do fluxo do jogo (início, fim, pontuação) e a interação com o usuário.

Métodos:

- **Game():** Inicializa o grid, os blocos e o estado do jogo.
- **Block GetRandomBlock():** Retorna um bloco aleatório da lista de blocos disponíveis.
- **std::vector<Block> GetAllBlocks():** Retorna todos os tipos de blocos.
- **void Draw():** Desenha o grid e os blocos na tela.
- **void HandleInput():** Lida com as entradas do teclado (movimento e rotação dos blocos).
- **void MoveBlockLeft():** Move o bloco atual para a esquerda.
- **void MoveBlockRight():** Move o bloco atual para a direita**.**
- **void MoveBlockDown():** Move o bloco para baixo e verifica se deve ser travado.
- **bool IsBlockOutside():** Verifica se o bloco está fora dos limites do grid.
- **void RotateBlock():** Rotaciona o bloco, revertendo se não couber no grid.
- **void LockBlock():** "Trava" o bloco atual no grid e prepara o próximo bloco.
- **bool BlockFits():** Verifica se o bloco pode ser colocado na posição atual do grid.
- **void Reset():** Reinicia o jogo (grid, blocos e pontuação).
- **void UpdateScore(int lineCleared, int moveDownPoints):** Atualiza a pontuação com base nas linhas limpas ou movimento do bloco.

Construtor Grid:

Função: Gerencia o grid onde os blocos caem, armazena os blocos já travados e cuida da limpeza das linhas completas.

Métodos:

- **Grid():** Inicializa o grid com 20 linhas e 10 colunas, e chama Initialize() para limpar o grid.
- **void Initialize():** Limpa o grid, configurando todas as células como vazias.
- **void Print():** Imprime o estado do grid no console.
- **void Draw():** Desenha o grid e as células preenchidas na tela.
- **bool IsCellOutside(int row, int column):** Verifica se uma célula está fora dos limites do grid.
- **bool IsCellEmpty(int row, int column):** Verifica se uma célula está vazia.
- **int ClearFullRows():** Limpa as linhas completas e move as linhas superiores para baixo.

Construtor Bloco:

Função: Representa os blocos que caem no jogo, controla sua rotação e movimentação no grid.

Métodos: l

- **Block():** Inicializa o bloco com o tamanho das células, cores e estado de rotação.
- **void Draw(int offsetX, int offsetY)**: Desenha o bloco na tela com o deslocamento especificado.
- **void Move(int rows, int columns):** Move o bloco no grid ajustando seus offsets de linha e coluna.
- **std::vector<Position> GetCellPositions():** Retorna as posições das células do bloco, considerando os deslocamentos.
- **void Rotate():** Rotaciona o bloco para o próximo estado.
- **void UndoRotation():** Reverte a rotação do bloco para o estado anterior.

Construtor Posição:

Função: Estrutura simples para armazenar as coordenadas de uma célula no grid(Linha, Coluna).

Métodos:

- **Position(int row, int column)**: Construtor que define as coordenadas de linha e coluna

Construtor Cores:
Função: Definir as cores usadas para representar os blocos no jogo.
Métodos:

- **std::vector<Color> GetCellColors():** Retorna um vetor com as cores definidas para os blocos.

Por fim, vale ressaltar que, embora o código seja pequeno e modularizado em várias partes, sua complexidade reside em fazer com que cada componente funcione de maneira integrada. Cada módulo complementa os demais, trabalhando em conjunto para garantir que as funcionalidades do jogo sejam executadas corretamente. A interação entre essas partes deve ser precisa, para que o jogo seja executado de forma coesa, sem falhas que comprometam a experiência do jogador.

3 RESULTADOS OBTIDOS

Nesta seção, são apresentados os resultados obtidos no desenvolvimento e criação do jogo Tetris em C++, utilizando a biblioteca Raylib. Como principais métricas da finalização do projeto, foram incluídas capturas de tela do jogo, além de imagens que ilustram outros aspectos do processo de desenvolvimento.

3.1 Apresentação de Figuras

Antes de apresentar as imagens dos resultados que ilustram o jogo criado, é importante apresentar alguns conceitos fundamentais para a compreensão do jogo como um todo. O primeiro e mais crucial é o grid ou malha (também chamado de "tabuleiro") do Tetris, pois é nele que o jogo ocorre. Por baixo das cores e blocos em movimento, é essa estrutura que define o espaço onde as peças caem, se posicionam e formam as linhas que serão completadas. A seguir a figura 2:

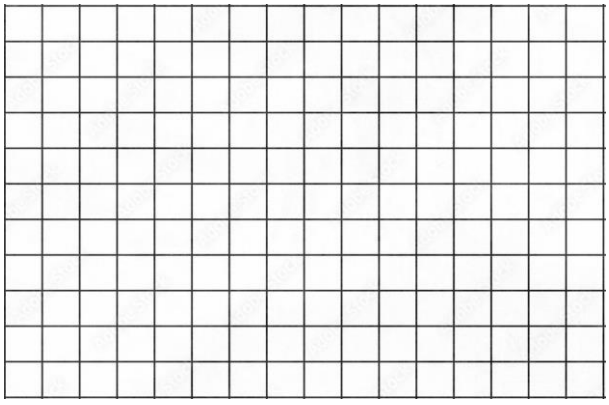

Figura 02: Exemplo de Grid (ADOBE, 2024).

Esta figura ilustra o grid do Tetris, que ao ser criado é inicializado com valores zero em todos os campos, indicando que estão vazios. À medida que as peças são adicionadas ao jogo, cada campo ocupado por um bloco recebe o valor um, representando que a posição está preenchida. Essa mudança de 0 para 1 é fundamental para o controle das colisões e para a verificação de linhas completas, como demonstrado na Figura 03.

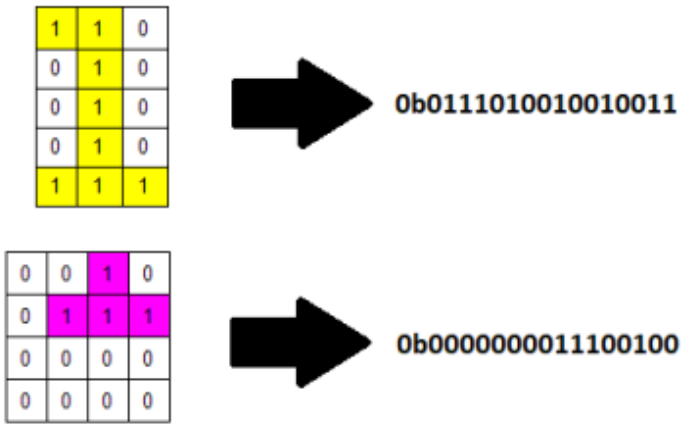

Figura 2 – Representação do número e peça

Figura 03: exemplo de grid com blocos (TETRIS RGB, 2019).

A Figura 04 exibe todas as peças do Tetris. O Tetris é um jogo no qual o objetivo é encaixar diferentes formas compostas por exatamente quatro blocos cada. Essas formas, chamadas tetrominós, são sete no total, pois, por definição, cada peça deve ter quatro componentes. Cada um desses sete tipos de tetrominós tem um formato único, o que exige do jogador diferentes estratégias de encaixe para não atingir o limite.

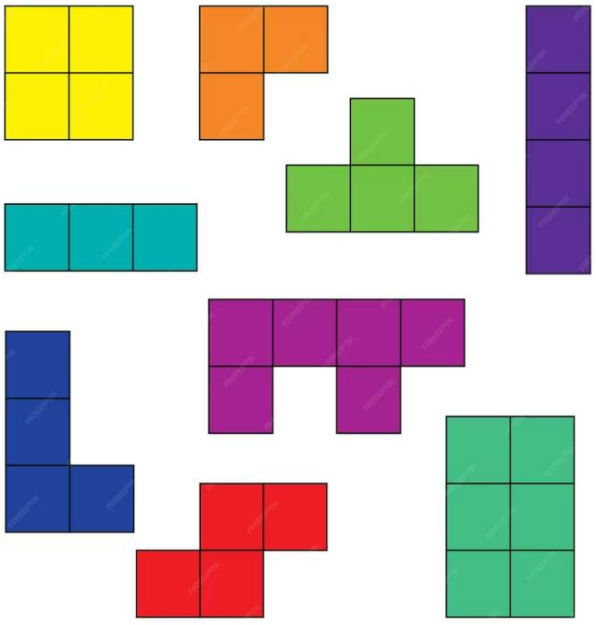

Figura 04: Todas as peças do tetris (FEEPIK,2024).

A partir deste ponto, com as devidas explicações e contextualizações sobre o funcionamento do jogo, apresentamos as imagens do projeto desenvolvido. As Figura 05 e Figura 06 ilustram duas situações distintas do jogo. A Figura 05 mostra uma sessão normal, com o jogo em andamento, exibindo o grid, as peças caindo e a interação do jogador com o tabuleiro.

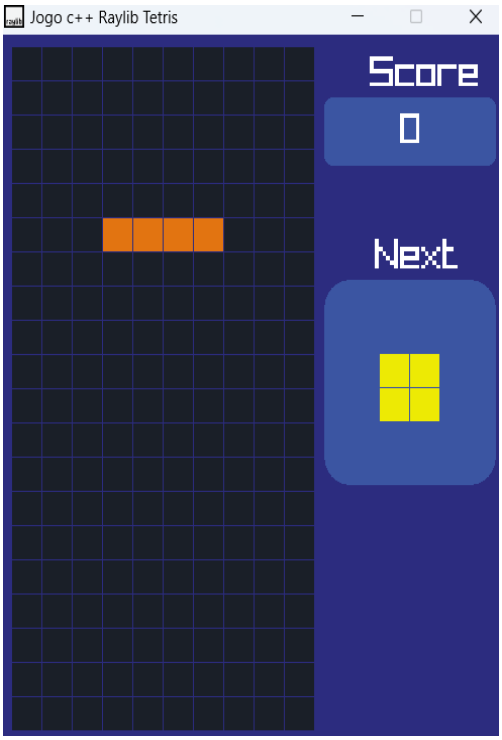

Figura 05: Demonstração do jogo Tetris (2024).

Já a Figura 06 representa o estado do jogo após o seu término, quando a pilha de blocos atinge o limite superior da tela, resultando na perda do jogador. Essas imagens ajudam a visualizar os diferentes estágios do jogo e como as interações são refletidas na interface.

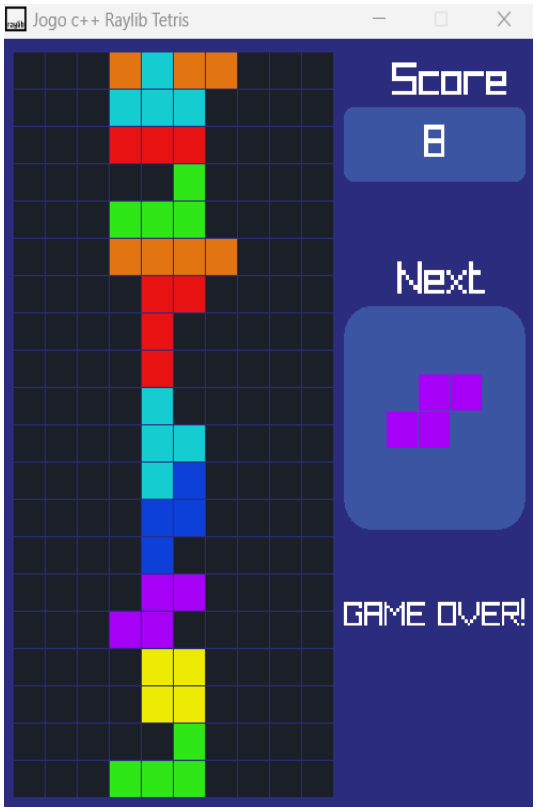

Figura 06: Demonstração de Fim de Jogo do Tetris (2024).

Com isso, concluímos a seção referente às imagens, que ilustram os principais resultados obtidos no desenvolvimento do jogo. Como demonstrado anteriormente, as figuras apresentadas refletem os diferentes estágios do jogo, desde a execução normal até o momento do término, evidenciando as funcionalidades e o comportamento esperado do jogo durante a interação com o usuário.

4 CONSIDERAÇÕES FINAIS

Portanto, para as considerações finais deste trabalho, foi desenvolvido um jogo com o objetivo de reforçar e consolidar os conceitos computacionais abordados nas disciplinas de Programação Orientada a Objetos. A intenção foi ilustrar a evolução da lógica de programação e o desenvolvimento de competências avançadas, essenciais para a criação de projetos mais complexos. Embora o projeto não tenha alcançado um nível elevado de complexidade, ele forneceu uma base sólida para a aplicação prática desses conceitos, contribuindo para o aprimoramento das habilidades no desenvolvimento de jogos e na programação em geral.

Além disso, o trabalho também buscou proporcionar uma contextualização histórica sobre o surgimento e a popularização dos jogos digitais, abordando de forma aprofundada o jogo Tetris. Ao explorar a trajetória desse clássico, foi possível analisar não apenas sua relevância histórica, mas também como ele influenciou a evolução dos jogos e contribuiu para a formação do que conhecemos hoje como indústria de jogos digitais.

REFERÊNCIAS

MICROSOFT. **Visual Studio Code: Setup Overview**. Disponível em: https://code.visualstudio.com/docs/setup/setup-overview. Acesso em: 17 nov. 2024.

RAYLIB. **Raylib: A Simple and Easy-to-Use Library to Learn Game Programming**. Disponível em: https://www.raylib.com/. Acesso em: 17 nov. 2024.

DRAW.IO. **Draw.io Documentation**. Disponível em: https://www.drawio.com/doc/. Acesso em: 17 nov. 2024.

DEITEL, Paul; DEITEL, Harvey. **C++: How to Program**. 5. ed. Upper Saddle River: Pearson Prentice Hall, 2006.

ASSUNÇÃO, J.; KLÜSSER, C.; HERRERA, V.; CARMO, J.; GAZZIRO, M. **Computadores e videogames**. 2023.

LUZ, Alan Richard da. **Video Games: História, Linguagem e Expressão Gráfica**. 1. ed. São Paulo: Editora, 2010.

MAITINO NETO, Roque. **Programação Orientada a Objetos**. 1. ed. São Paulo: Editora, 2018.

MEDEIROS, Julyana Mira; MARTINS, Ernane Rosa. **Desenvolvimento e Avaliação do Jogo Tetris em Linguagem Java: Um Estudo sobre Aprendizado de Lógica e Habilidades de Resolução de Problema**. 2023.

PINTO, Lucas Gonçalves; ELLER, Luiz; AUGUSTO, Luiz. **Tetris RGB**. 2019.

ADOBE. **Stock Photos - Grid**. Disponível em: https://stock.adobe.com/br/search/images?k=grid. Acesso em: 17 nov. 2024.

FREEPIK. **Conjunto de Blocos Coloridos para Ilustração Vetorial de Jogo Tetris**. Disponível em: https://br.freepik.com/vetores-premium/conjunto-de-blocos-coloridos-para-ilustracao-vetorial-de-jogo-tetris_29738013.htm#fromView=keyword&page=1&position=28&uuid=c1e473f0-4968-455c-8cd8-cf02a5b61769. Acesso em: 17 nov. 2024.