INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE SÃO PAULO

RAISSA PEREIRA MIRANDA

JOGO HAMSTORM

CAMPOS DO JORDÃO
2025

RAISSA PEREIRA MIRADA

JOGO HAMSTORM

Relatório apresentado ao curso de Tecnologia em Análise e Desenvolvimento de Sistemas, da Faculdade de Campos do Jordão, para projeto proposto à disciplina de Programação Orientada a Objetos.

Orientador: Paulo Giovani de Faria Zeferino.

CAMPOS DO JORDÃO
2025

RESUMO

Este relatório apresenta o processo de desenvolvimento do jogo *Hamstorm*, um projeto produzido na linguagem C++ utilizando a biblioteca gráfica *Raylib*. Trata-se de um jogo do gênero *top-down arena shooter* com elementos clássicos de *roguelike*, no qual o jogador assume o papel de um hamster armado e deve sobreviver em uma área limitada enquanto enfrenta sucessivas ondas de inimigos. Ao longo da partida, o jogador precisa derrotar cinco ondas progressivamente mais difíceis até chegar ao confronto decisivo contra o chefe final (boss), encerrando o ciclo principal da gameplay. Durante o desenvolvimento, foram implementados sistemas para a construção da experiência de jogo, incluindo uma loja interna para aprimoramento de atributos do personagem, um sistema de ondas cronometradas, *drop* de cristais utilizados como moeda, além de animações em *pixel art*, gerenciamento de fases e transição entre telas do jogo. O projeto busca integrar mecânicas inspiradas em jogos populares do gênero, aplicando conceitos estudados ao longo da disciplina e explorando práticas de programação orientada a objetos e design de jogos.

Palavras-Chave: Hamstorm; Raylib; Jogo.

SUMÁRIO

1 INTRODUÇÃO

Este trabalho apresenta o desenvolvimento do jogo *Hamstorm*, um projeto criado em C++ utilizando a biblioteca *Raylib*. O jogo foi pensado no estilo *top-down arena shooter roguelike*, onde o jogador controla um hamster armado e precisa sobreviver a ondas de inimigos até enfrentar o *boss* final. A seguir, são apresentados os objetivos do projeto, a justificativa da escolha do tema, os aspectos metodológicos utilizados e o aporte teórico que serviu de base para a construção do jogo.

1.1 Objetivos

O objetivo principal deste trabalho é demonstrar o processo de criação de um jogo digital simples, porém completo, aplicando conhecimentos de programação orientada a objetos, lógica de jogos, animação e organização de sistemas interativos. Além disso, busca-se desenvolver um protótipo funcional que inclua mecânicas de combate, progressão por ondas, loja de upgrades e animações em *pixel art*.

1.2 Justificativa

O desenvolvimento do jogo *Hamstorm* foi escolhido por ser uma forma prática de aprofundar os estudos em programação e criação de jogos. O gênero *arena shooter roguelike* foi selecionado por sua estrutura dinâmica, que permite trabalhar conceitos como aleatoriedade, dificuldade progressiva, colisões, controle de personagem, IA simples para inimigos e gerenciamento de estados do jogo. Além disso, o projeto foi proposto com base em referências de jogos do mesmo gênero, como *Brotato*, servindo como inspiração para a adaptação de mecânicas semelhantes às necessidades e limitações do escopo disponível.

1.3 Aspectos Metodológicos

Para desenvolver o jogo, foi utilizado a linguagem C++ no editor *Visual Studio Code*, juntamente com a biblioteca *Raylib*, responsável pela parte gráfica, controle de input, colisões e gerenciamento das telas. A arte foi criada no *Aseprite*, utilizando *pixel art* e animações feitas com sprites em frames. O desenvolvimento ocorreu de forma incremental, começando pelo movimento do jogador, depois os tiros, inimigos, sistema de ondas, loja, animações e telas do jogo. As ondas foram programadas com tempos definidos e aumento gradual de dificuldade, enquanto os inimigos são gerados por funções aleatórias.

1.4 Aporte Teórico

O aporte teórico do projeto foi baseado principalmente na documentação oficial da *Raylib*, que serviu como fonte para entender o funcionamento das funções gráficas e estruturais da biblioteca. Também foram analisados exemplos e tutoriais da comunidade e estudadas características do jogo *Brotato*, utilizado como referência de design e mecânicas. Esses materiais ajudaram a compreender como estruturar o gameplay, a progressão e a lógica geral do jogo.

2 METODOLOGIA

2.1 Como o Projeto Começou

No início do projeto, ocorreu a definição dos elementos centrais do jogo, que serviram como base para as próximas etapas de implementação. Sendo os elementos:

- Uma arena fixa;
- Um personagem principal;
- Três tipos de armas;
- Inimigos aparecendo de forma aleatória;
- Cinco ondas com tempo definido;
- Uma loja entre as ondas para o jogador evoluir.

A seguir, foi desenvolvido cada parte aos poucos: primeiro o movimento do jogador, depois o sistema de tiro, depois a movimentação e geração aleatória dos inimigos, depois as ondas e as telas (menu, opções do jogo, partida em si, loja, tela de vitória e derrota), por último foi realizada a implementação das *sprites* e animações.

2.2 Ferramentas Utilizadas

- Linguagem: C++;
- Biblioteca: *Raylib*;
- Editor: *Visual Studio Code*;
- Arte: *Aseprite*;
- Extensões: Suporte para C++ e *Raylib* no *VS Code.*

2.3 Descrição do Jogo Hamstorm

O jogador controla um hamster de óculos armado. Ele precisa sobreviver às cinco ondas de inimigos para vencer o jogo. Cada onda tem um tempo específico:

- Onda 1: 30s;
- Onda 2: 45s;
- Onda 3: 50s;

- Onda 4: 60s;
- Onda 5: 90s (onda do boss).

Os inimigos são gerados com uma função aleatória, e a cada onda aparecem tipos novos, ficando mais rápidos, com mais dano e mais vida. No total são cinco tipos, contando com o boss. Quando os inimigos morrem, eles dropam cristais, que o jogador usa na loja para comprar:

- Vida máxima;
- Dano;
- Velocidade;
- Recuperação de vida;
- Dash.

A arte do jogo é toda feita no *Aseprite*, em *pixel art*, usando tons mais escuros. O fundo da arena tem dois tons, que ajudam a dar contraste com o personagem e os inimigos.

3 RESULTADOS OBTIDOS

A seguir, são apresentados os resultados obtidos com o desenvolvimento do projeto que possui sistema completo de ondas com tempos e dificuldade crescente.

- Sistema de dano e colisões funcionando.
- *Drop* de cristais com valores diferentes dependendo do inimigo.
- Sistema de troca de telas (menu, *settings*, loja, vitória e derrota).
- Loja funcionando corretamente e levando as novas estatísticas para o jogo.
- Movimentação do jogador com *dash* e limites da arena.
- Animações em *pixel art* feitas com frames para o jogador, inimigos e cristais.

3.1 Apresentação das capturas de tela do Hamstorm

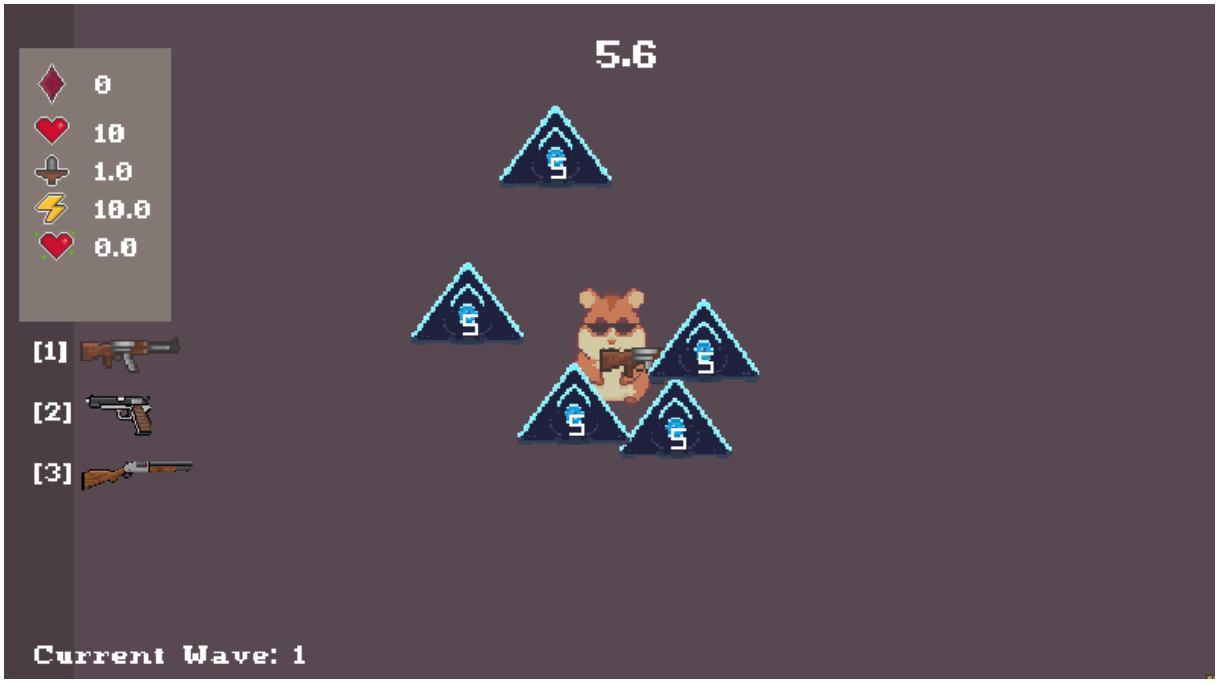
Imagem 01. Tela de Menu Principal.

Tela com opções de começar e sair do jogo.

Imagem 02. Tela de Seleção de Personagem (Opções do Jogo).

Tela com opções de escolha de personagem: *Fox*, Hamster ou *Ferret*.

Imagem 03. Tela da Partida.



Tela em que ocorre a gameplay em si.

Imagem 04. Tela da Loja.

Onde o jogador pode comprar melhorias para seus atributos.

Imagem 05. Tela de Vitória.

Tela que aparece quando o jogador venceu o *boss* final.

Imagem 06. Tela de Derrota.

Tela que aparece quando o jogador perdeu todas as suas vidas.

4 CONCLUSÃO

O desenvolvimento do *Hamstorm* foi uma experiência muito importante para aprender a criar um jogo completo usando *Raylib* e C++. Foi implementado praticamente tudo o que havia sido planejado no início, incluindo ondas, inimigos variados, sistema de loja, boss e telas de interação.

Apesar dos resultados alcançados, ainda existem diversas melhorias que podem ser feitas em versões futuras do projeto, como:

- Adicionar os personagens *Fox* e *Ferret*, cada um com skills próprias;
- Adicionar efeitos sonoros;
- Criar animações e sprites mais elaboradas;
- Criar artes em *pixel art* para todas as telas do jogo (*Game Settings*, Loja, Vitória e Derrota);
- Criar novos mapas para a partida
- Mostrar estatísticas do jogador nas telas de vitória e derrota;
- Mudar os inimigos para terem sprites e comportamentos únicos;
- Deixar o boss mais difícil e com mais personalidade;
- Fazer mapas que mudam de tamanho a cada onda e tenham elementos extras, como torretas ativáveis.

REFERÊNCIAS

Raylib is a simple and easy-to-use library to enjoy videogames programming. *Raylib*, 2025. Disponível em: https://www.raylib.com/. Acesso em: 16 ago. 2025.

Brotato Wiki. *Spells & Guns*, 2024. Disponível em: https://brotato.wiki.spellsandguns.com/Brotato_Wiki. Acesso em: 16 ago. 2025.

Pixel Joint – The Internet Pixel Art Gallery. *Pixel Joint*, 2025. Disponível em: https://pixeljoint.com/. Acesso em: 16 ago. 2025.

Palette List. *Lospec*. Disponível em: https://lospec.com/palette-list. Acesso em: 16 ago. 2025.