

Relatório Técnico – Jogo 2D “Nave no Espaço”

1. Identificação

Autores: Mateus Mendes dos Santos e Isabella Louzado da Silva

Disciplina: Desenvolvimento de Jogos digitais

Professor: Christien Lana Rachid

Data: 08/10/2025

2. Introdução

O presente relatório descreve o desenvolvimento de um jogo 2D em HTML5 Canvas intitulado “Nave no Espaço”, cujo objetivo é controlar uma nave, desviar e destruir naves inimigas, e percorrer a maior distância possível.

O projeto explora recursos técnicos obrigatórios da disciplina, utilizando JavaScript puro, sem frameworks externos, e sprites e sons gratuitos devidamente creditados.


3. Objetivos do Jogo


- Desenvolver um jogo funcional com movimentação livre da nave e disparos.
- Implementar paralaxe vertical para criar profundidade no cenário.
- Detectar colisões com inimigos e projéteis usando AABB (Axis-Aligned Bounding Box).
- Registrar distância percorrida e exibir ranking baseado no nome do jogador.
- Criar experiência infinita, aumentando a dificuldade conforme o tempo.


4. Recursos Técnicos Implementados


Recurso | Descrição | Como foi usado


-----|-----|-----


 Loop de Animação | requestAnimationFrame | Loop principal de jogo, atualizando lógica e renderização


 Eventos de Teclado | WASD/Setas + Espaço | Controle do movimento e disparos da nave do jogador

 Paralaxe | Camadas de fundo com velocidades diferentes | LayerFar (mais lento) e LayerNear (mais rápido) simulando profundidade

 Detecção de Colisão | AABB | Colisões entre nave, tiros e inimigos, registrando ações personalizadas

 Disparo / Projéteis | Criação de balas para jogador e inimigos | Arrays de balas atualizados a cada frame, removidos ao sair da tela ou colidir

 Spritesheet | (Opcional) | Preparado para animação de inimigos e player

 Clipping | (Opcional) | Preparado para drawImage com clipping

5. Mecânicas do Jogo

- Movimento: Nave controlada livremente em X e Y.
- Disparo: Pressionar espaço cria projéteis que destroem inimigos.
- Inimigos: Naves inimigas descem de cima para baixo, podendo colidir ou disparar contra o jogador.
- Colisão: AABB determina se o jogador colidiu com tiros ou inimigos.
- Pontuação: Distância percorrida em metros e quantidade de inimigos destruídos.
- Dificuldade crescente: Velocidade e frequência de inimigos aumentam ao longo do tempo.

6. Arquitetura do Código

O código está organizado em módulos funcionais, com separação clara entre:

- Update(): lógica do jogo (movimento, colisões, spawn de inimigos).
- Draw(): renderização das entidades e HUD (distância, velocidade, vida).
- Input: captura de eventos de teclado.
- Colisor: sistema de colisão AABB flexível.
- Entidades: Player, Enemy, Bullet, ParallaxLayer.
- Ranking: armazenamento local usando localStorage, registrando distância e nome do jogador.

7. Evidências do Desenvolvimento

- Prints:

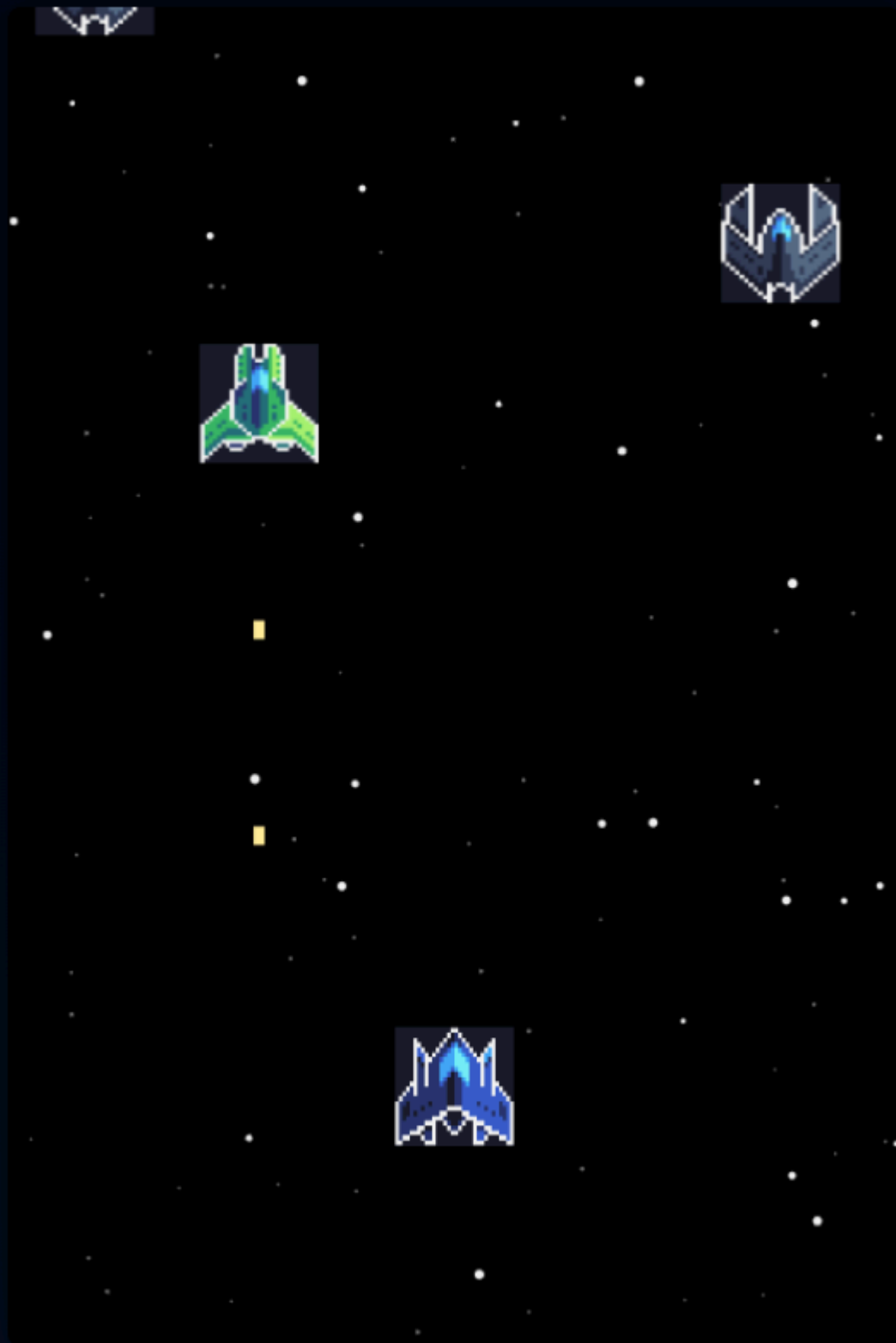
- Tela inicial com campo de nome



- Gameplay com nave e inimigos

Nave Infinita ✨

Use ← → ↑ ↓ ou WASD para mover. Espaço para atirar. Uma batida significa game over.



M&M

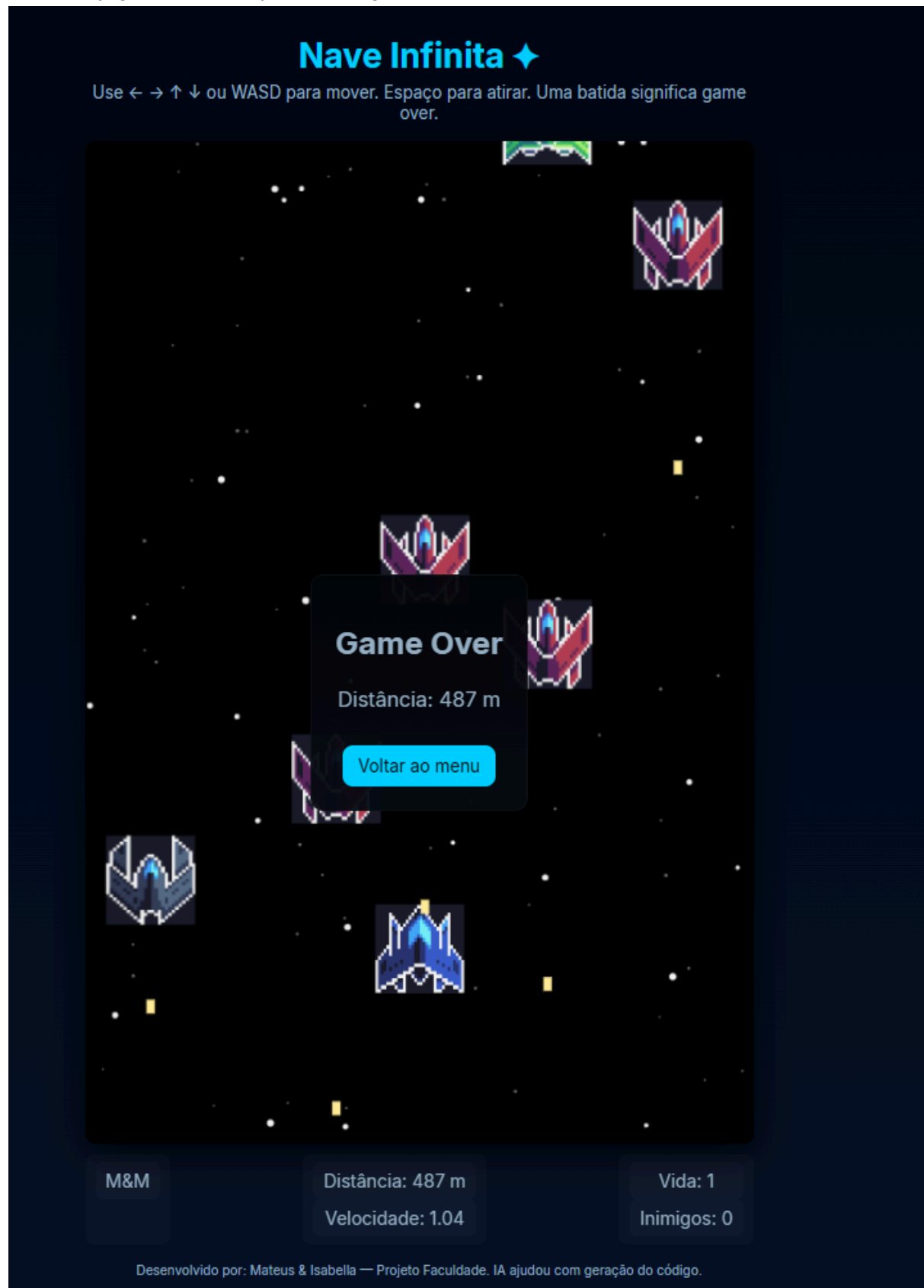
Distância: 197 m

Velocidade: 1.00

Vida: 1

Inimigos: 0

- Morte do jogador e overlay de ranking



- Código-fonte:
 - Arquivo main.js
 - index.html
 - style.cssTodos os arquivos comentados, limpos e organizados.

8. Reflexão sobre o uso de IA

- Geração de código: Utilizamos ChatGPT para estruturar o loop de jogo, gerenciamento de colisão e spawn de inimigos.
 - Sugestões e prompts: Prompts fornecidos pelo professor foram adaptados para gerar funções de update(), draw() e colisão.
 - Benefícios: Agilidade no desenvolvimento, aprendizado sobre boas práticas de JS e Canvas, solução de problemas complexos rapidamente.
 - Limitações: Necessidade de revisar e adaptar o código gerado, testes manuais para garantir estabilidade e performance.
-
- Reflexão sobre o uso de IA (Continuação - Gemini)
 - Experiência com Gemini: O uso do Gemini para correções e ajustes de lógica (especificamente no bug do sprite do jogador) demonstrou uma aparente maior precisão contextual e rapidez na entrega do código completo. Não é possível determinar com certeza se a velocidade de resposta foi devido à superioridade da ferramenta ou se o processo foi acelerado por já possuímos a estrutura de código base.

9. Conclusão

O jogo “Nave no Espaço” cumpre os objetivos propostos:

- Explora cinco ou mais recursos técnicos solicitados.
- Implementa ranking e cálculo de distância percorrida.
- Oferece gameplay infinito com aumento de dificuldade.
- Código limpo, comentado e modular, pronto para expansão com spritesheets e clipping.

10. Referências

- <https://opengameart.org> – Assets gráficos gratuitos.
- ChatGPT (GPT-4/GPT-5-mini) – Auxílio na estruturação de código e lógica de jogo.
- Documentação MDN – Canvas API e requestAnimationFrame.