Centro Universitário UniAcademia

DESENVOLVIMENTO DE JOGOS DIGITAIS

Relatório Técnico Jogo 2D no Canvas com LLMs

IGOR GABRIEL RODRIGUES

PEDRO JOSÉ GUIMARÃES COELHO

JUIZ DE FORA

2025

CENTRO UNIVERSITÁRIO UNIACADEMIA

1. INTRODUÇÃO

Este relatório detalha o processo de desenvolvimento de um jogo 2D para navegador, baseado no clássico 'Asteroides', como parte da avaliação da disciplina [Nome da Disciplina]. O projeto foi realizado em dupla e teve como objetivo principal a aplicação dos conceitos de HTML5 Canvas e JavaScript, com o apoio fundamental de Modelos de Linguagem (LLMs) como ferramentas de geração de código e auxílio à programação. O jogo implementa mecânicas de movimento com inércia, disparo, detecção de colisão e um sistema de pontuação, seguindo os requisitos propostos no desafio.

2. Ferramentas e Tecnologias

- **Linguagens:** HTML5, CSS3, JavaScript (ES6+)
- **Ambiente de Desenvolvimento:** Visual Studio Code
- **Controle de Versão:** Git e GitHub
- **Modelos de Linguagem (LLMs):**
    - Google Gemini (utilizado pelo Integrante A)
    - OpenAI ChatGPT-4 (utilizado pelo Integrante B)
- **Assets:** Sprites de explosão obtidos do site OpenGameArt.

3. Metodologia de Desenvolvimento com IA

Nossa estratégia de desenvolvimento foi baseada no método de "programação em par assistida por IA", conforme proposto no desafio. O objetivo não era apenas ter o código gerado, mas analisar criticamente sua qualidade, eficiência e funcionalidade.

3.1 Divisão de Papéis e Dinâmica da Dupla

Seguindo as diretrizes do desafio, adotamos uma metodologia de 'programação em par assistida por IA'. Inicialmente, Igor Gabriel atuou como

'Gerador de Prompt', utilizando o Google Gemini para criar a estrutura base do projeto e a lógica da nave. Enquanto isso, Pedro Coelho atuou como 'Validador', testando, depurando e integrando o código gerado. Após a implementação da base, os papéis foram invertidos para o desenvolvimento dos sistemas de projéteis e asteroides, com [Nome da Dupla] utilizando o ChatGPT-4.

4 Registro de Interações com as LLMs (Prompts)

Nossa metodologia de desenvolvimento envolveu o uso dos prompts sugeridos no desafio para construir o jogo de forma incremental. Para cada etapa, registramos o prompt enviado e analisamos criticamente a resposta da IA, realizando os ajustes manuais necessários.

Abaixo estão dois exemplos representativos dessa interação:

Exemplo 1: Geração da Nave e Controles

- **LLM Utilizada:** Google Gemini
- **Prompt (Resumo):** "Solicitamos a criação da classe Player (nave), com controles de rotação (A/D) e aceleração (W), incluindo física de inércia e a funcionalidade de 'screen wrap' (atravessar a tela)."
- **Análise da Resposta:**
    - **Acerto:** A IA gerou a classe Player completa, os event listeners de teclado e a lógica de rotação e screen wrap de forma correta e funcional.
    - **Ajuste Manual:** A física de inércia e atrito, embora funcional, resultou em uma nave incontrolável (muito "lisa"). Tivemos que ajustar manualmente os valores das constantes de aceleração e atrito no código gerado para alcançar uma jogabilidade agradável.

Exemplo 2: Detecção de Colisão e Quebra dos Asteroides

- **LLM Utilizada:** ChatGPT
- **Prompt (Resumo):** "Solicitamos a implementação da colisão AABB entre projéteis e asteroides, e a lógica para que os asteroides se quebrassem em pedaços menores ao serem atingidos."
- **Análise da Resposta:**
    - **Acerto:** A IA forneceu uma função AABB (checkCollision) matematicamente correta e a lógica de quebra dentro da classe Asteroid (um asteroide grande gerando dois médios).
    - **Ajuste Manual:** O código gerado não "conectava" as duas partes. Tivemos que implementar manualmente o loop dentro da função

update() que de fato chamava a checkCollision para cada projétil contra cada asteroide. Além disso, corrigimos um bug onde os novos asteroides (médios) apareciam na posição (0,0) em vez da posição do asteroide-pai.

5 Arquitetura do Jogo e Recursos Implementados

O jogo está estruturado em três arquivos: index.html, style.css e main.js. Toda a lógica está contida em main.js, organizada em classes para as entidades (Player, Asteroid, Projectile, Explosion) e funções globais para o loop de jogo (update, draw) e gerenciamento de estado. Dos recursos técnicos propostos, implementamos os seguintes seis:

- **Loop de Animação:** Utilizando requestAnimationFrame.
- **Eventos de Teclado:** Para rotação, aceleração e disparo.
- **Paralaxe:** Com duas camadas de estrelas para dar profundidade.
- **Detecção de Colisão:** Utilizando o método AABB entre todas as entidades relevantes.
- **Spritesheet / Clipping:** Para a animação de explosão dos asteroides.
- **Disparo / Projéteis:** Com um array gerenciando os projéteis disparados.

6 Desafios e Soluções

O maior desafio técnico foi ajustar a detecção de colisão AABB para objetos que rotacionam, como a nave e os asteroides. Como o AABB não rotaciona, a 'caixa de colisão' muitas vezes era maior que o objeto visual, resultando em colisões 'fantasmas'. A IA nos forneceu o código AABB padrão, e a solução foi um ajuste manual, calculando o AABB a cada frame para se ajustar às dimensões máximas do objeto rotacionado. Outro desafio foi a depuração de projéteis que não eram removidos corretamente do array, causando lentidão no

jogo após algum tempo de partida. Isso foi resolvido revisando o loop que percorria o array de trás para frente ao remover itens.

7 Resultados e Demonstração

O resultado final é um jogo de Asteroides funcional e divertido, que cumpre todos os requisitos técnicos obrigatórios. A jogabilidade é fluida e o uso de IA acelerou significativamente o desenvolvimento das funcionalidades base, nos permitindo focar em ajustes e polimento.

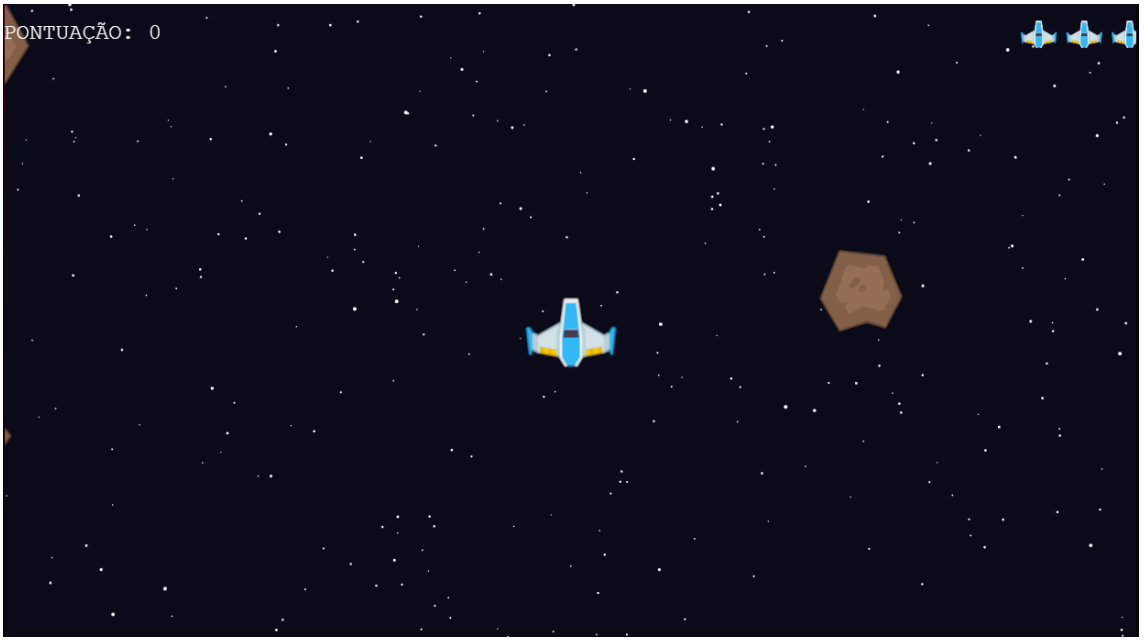

Figura 1: Tela principal do jogo, mostrando a nave, asteroides e o HUD.

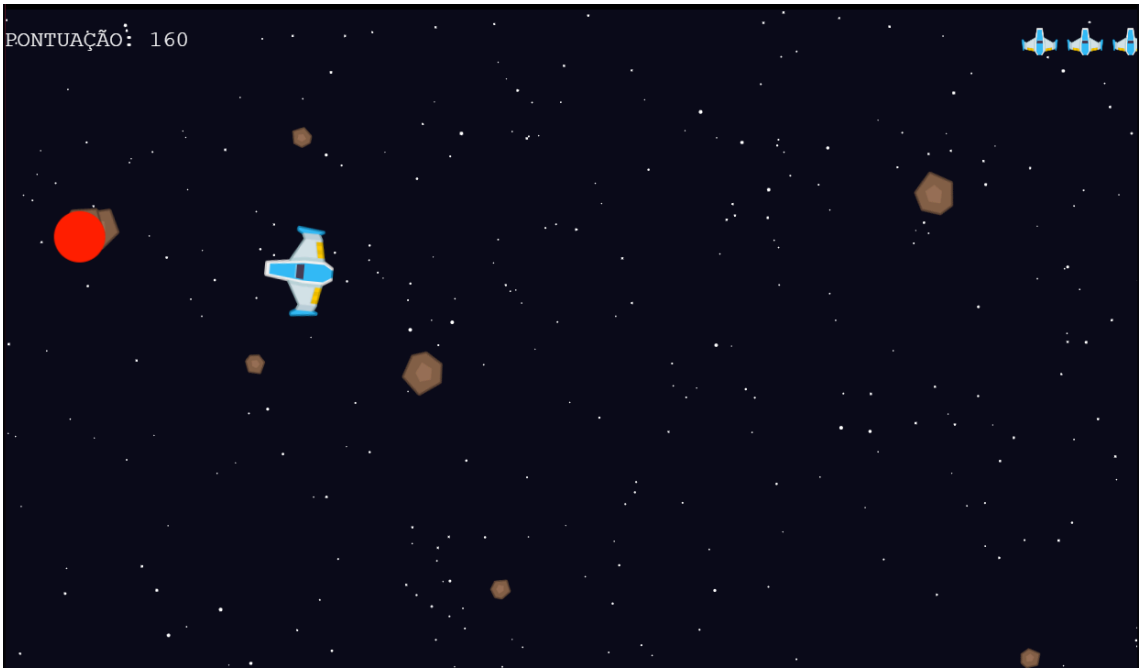

Figura 2: Momento de colisão com a animação de explosão (spritesheet) em efeito.

8 Conclusão

Este projeto foi uma experiência de aprendizado valiosa tanto nos aspectos técnicos do desenvolvimento de jogos com Canvas quanto na utilização de LLMs como ferramentas de programação. Concluímos que as IAs são extremamente eficazes para gerar 'boilerplate' (código repetitivo) e implementar algoritmos conhecidos (como AABB), economizando horas de trabalho. Contudo, a supervisão humana, a depuração e o ajuste fino (como a 'sensação' do controle da nave) continuam sendo indispensáveis. A dinâmica de dupla, alternando entre gerar e validar, provou ser uma excelente forma de garantir a qualidade do código e compartilhar o conhecimento.