Relatório de Desenvolvimento: Jogo Batalha Naval em C

Projeto: Desenvolvimento do jogo Batalha Naval

Linguagem: C

Autores: Euzebio Rocha, Juan Da Silva, Lucas Augusto, Marcos Vinicius e Rodrigo Miranda.

1. Visão Geral

O objetivo deste projeto foi desenvolver uma versão digital do jogo de tabuleiro Batalha Naval para dois jogadores. O sistema permite o posicionamento estratégico de navios em uma matriz 5X5, a alternância de turnos para ataques e a persistência de dados através de um sistema de salvamento e carregamento de arquivos.

2. Decisões de Projeto e Estrutura de Dados

Para organizar as informações do jogo, foi utilizada uma **struct** chamada ***Batalha***. Esta decisão foi fundamental para manter todos os dados pertinentes (tabuleiros dos dois jogadores, matrizes de visão e o turno atual) agrupados em uma única variável.

- **Matrizes:** Foram utilizadas quatro matrizes 5X5: duas representando o posicionamento real dos navios (tabuleiro1, tabuleiro2) e duas representando o que cada jogador vê (visao1, visao2), preenchidas inicialmente com 'água' (~).

3. Implementação das Funcionalidades e Desafios

Durante o desenvolvimento, duas áreas apresentaram maior complexidade técnica:

3.1. Sistema de Salvamento e Carregamento (Persistência de Dados)

A implementação da persistência de dados foi o maior desafio do projeto. O requisito era salvar não apenas os caracteres das matrizes, mas também cabeçalhos textuais (ex: "Tabuleiro 1", "Visao jogador 1") para tornar o arquivo legível.

- **Dificuldade:** O desafio principal foi manipular o cursor de leitura do arquivo. Ao ler o arquivo (*fscanf*), o programa precisava ignorar os textos dos cabeçalhos para capturar apenas os dados do jogo.
- **Solução:** Foi implementada uma variável "lixo" (buffer) para consumir e descartar as palavras dos cabeçalhos, permitindo que a leitura das matrizes ocorresse sem erros de formatação. O arquivo ***jogo_salvo.txt*** armazena o estado completo de ambos os tabuleiros e o turno atual.

3.2. Mecânica de Turnos e Ataques

A lógica de alternância entre os jogadores exigiu atenção especial para garantir que as ações fossem registradas nas matrizes corretas.

- **Implementação:** O jogo utiliza um loop principal (*while*) que verifica a variável ***jogo.turno***.
- **Lógica de Ataque:**
    - Se for o turno do Jogador 1, o sistema lê as coordenadas de ataque e verifica o tabuleiro2 (onde estão os navios inimigos).
    - O resultado (Acerto 'X' ou Erro 'O') é atualizado tanto no tabuleiro do inimigo quanto na matriz de visao1, permitindo que o jogador saiba onde já atacou.
    - A validação impede ataques repetidos na mesma coordenada ou fora dos limites da matriz.

4. Dificuldades Comuns

Como programador iniciante desenvolvendo este sistema, identifiquei obstáculos técnicos que vão além da lógica do jogo:

1. **Manipulação de Arquivos (File I/O):** Entender que o computador lê o arquivo caractere por caractere (sequencialmente) é complexo. Se o arquivo tiver um espaço ou uma quebra de linha inesperada, o *fscanf* pode ler o dado errado e quebrar todo o carregamento.
2. **Índices de Matriz:** A confusão entre linhas e colunas (índice [i][j] vs [x][y]) e o fato de que em C os índices começam em 0 (e não em 1) causaram erros iniciais de posicionamento.
3. **Buffer do Teclado:** O uso do *scanf* para ler números e depois ler caracteres muitas vezes gera erros, pois a tecla "Enter" fica presa no buffer de entrada, sendo lida como um caractere válido na próxima instrução.
4. **Interface de Usuário (Console):** Criar uma interface limpa no terminal, usando comandos para limpar a tela (system("cls")), foi necessário para que o jogo não se tornasse uma "parede de texto" confusa, garantindo que um jogador não visse os navios do outro durante a troca de turnos.

5. Conclusão

O projeto atendeu a todos os requisitos funcionais, entregando um jogo operacional com validação de regras, condição de vitória e sistema robusto de arquivos. A experiência reforçou o entendimento sobre matrizes, estruturas (structs) e manipulação de arquivos em C.