Stellar Clash: Um Jogo com Estruturas de Dados

Desenvolver um jogo de batalha espacial integrando **estruturas de dados avançadas** e algoritmos eficientes.

- Andreia Elias Corrêa
- João Vitor de Pina Adorno de Paiva
- Maria Clara Dutra Costa

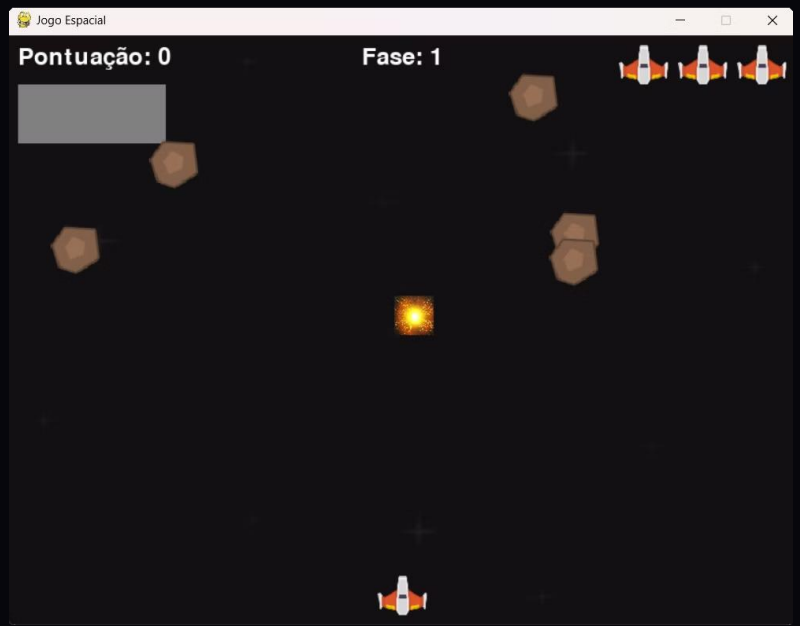
Introdução ao Jogo

Objetivo:

O jogo coloca o jogador no controle de uma nave espacial, com a missão de sobreviver a ataques de meteoros e obstáculos, enquanto coleta pontos e habilidades especiais. O objetivo é alcançar a maior pontuação possível e vencer os desafios de cada fase.

Problema:

Como estruturar um sistema interativo e dinâmico para maximizar desempenho e experiência do jogador?



Interface do jogo

Estruturas de Dados Utilizadas

No desenvolvimento de _Stellar Clash_, exploramos as estruturas de dados que permitiram a construção de uma experiência de jogo desafiadora e fluida.

1. Lista (list)
2. Árvore (Nohabilidades)
3. Grafo

Lista

```
# Configuração dos projéteis
projeteis = []
```

```
# Atualizar posição dos projéteis
for projetil in projeteis[:]:
    projetil[1] -= velocidade_projetil
    if projetil[1] < 0:
        projeteis.remove(projetil)
```

Listas são usadas principalmente para representar coleções dinâmicas de elementos que mudam durante o jogo, como:

1. **Projetéis disparados pela nave.**
2. **Meteoros que caem na tela.**
3. **Bolas de fogo que causam dano ao jogador.**

Essas listas permitem gerenciar facilmente a criação, movimentação e remoção dos elementos do jogo conforme a lógica se desenvolve.

Árvore

- A função `criar_menu()` cria a estrutura hierárquica de opções
- representado pela classe `NoHabilidade`
- Um nome (`nome`), que corresponde ao rótulo do nó (Fácil,Médio , Difícil)
- Aqui, a árvore é usada apenas como uma lista hierárquica de opções.

O sistema poderia ser expandido para exibir submenus ao selecionar uma opção com filhos

```
def criar_menu():
    # Criar a árvore de opções
    raiz = NoHabilidade("Iniciar Jogo", [
        NoHabilidade("Fácil"),
        NoHabilidade("Médio"),
        NoHabilidade("Difícil"),
        NoHabilidade("Sair")
    ])
    return raiz
```

```python
# Criar um grafo direcionado para representar a progressão
grafo_fases = nx.DiGraph()

# Adicionando nós
grafo_fases.add_node("1")
grafo_fases.add_node("2")
grafo_fases.add_node("3")

# Adicionar arestas (conexões entre fases)
grafo_fases.add_edge("1", "2")
grafo_fases.add_edge("1", "3")
grafo_fases.add_edge("2", "3")
```

Crafo

- **Nós**: representam as fases do jogo (`"1"`, `"2"`, `"3"`).
- **Arestas**: representam as conexões entre as fases. Por exemplo, `"1"` está conectada a `"2"` e `"3"`, indicando que a partir da fase 1, o jogador pode progredir para a fase 2 ou 3.

Criou a função `avancar_fase` para determinar qual será a próxima fase com base no grafo.

Implementação do Jogo

Linguagem

Python

Blibioteca

Pygame

JOCAR

Resultados Alcançados

1

Jogabilidade:

Sistema funcional com habilidades desbloqueáveis e inimigos dinâmicos.

2

Desempenho

- Taxas de quadros consistentes.
- Resposta rápida aos comandos.

3

Progressão

Grafo parcialmente implementado, permitindo expansão futura.

Dúvidas?