Space Game

Pharo

POO - 6 décembre 2024

1. Introduction

Ce projet vise à concevoir un jeu de combat spatial dans lequel différentes flottes de vaisseaux s'affrontent dans des régions variées de l'univers. Ce projet exploite pleinement les concepts avancés de la programmation orientée objet (POO), tels que le double dispatch, pour modéliser des interactions complexes entre vaisseaux et régions.

Le système met en œuvre :

- Une architecture modulaire avec des responsabilités claires.
- Des règles influencées par les propriétés des régions spatiales et des types de vaisseaux.
- Un moteur de combat intégrant des ajustements dynamiques basés sur le contexte spatial.

2. Description technique

2.1 Architecture générale

L'architecture du projet repose sur une structure modulaire qui facilite l'ajout de nouvelles fonctionnalités. Voici les principales classes :

- **Battle** : Gère les combats entre deux flottes. Cette classe centralise les règles du jeu et les étapes de résolution des batailles.

- **Fleet** : Représente une flotte composée d'une collection de vaisseaux. Elle inclut des méthodes pour :
    - Ajouter/retirer des vaisseaux.
    - Calculer les dégâts infligés et subis.

- **Ship** : Classe abstraite définissant les propriétés communes des vaisseaux (blindage, coque, dégâts).
    - Sous-classes :
        - **Fighter**
        - **Cruiser**
        - **Destroyer**
        - **BattleCruiser**

- **Region** : Définit les propriétés des différentes régions spatiales. Les modifications appliquées par une région aux vaisseaux sont définies via un double dispatch.

- Sous-classes :
  - InhabitedSolarSystem
  - AsteroidField
  - Nebula
  - DeepSpace

2.2 Double dispatch

Le double dispatch est utilisé dans deux contextes :

1. **Entre vaisseaux** : Pour gérer les interactions spécifiques (e.g., un Destroyer inflige le double de dégâts sur les boucliers d'un BattleCruiser).
2. **Entre vaisseaux et régions** : Les régions modifient les propriétés des vaisseaux selon leur type (e.g., dans l'espace profond, les boucliers des Fighters sont annulés).

3. Règles du jeu

3.1 Caractéristiques des vaisseaux

- **Fighter** :
  - Shields : 100
  - Hull : 400
  - Damage : 50
  - Bonus en nébuleuse : Double dégâts sur les autres vaisseaux.
- **Cruiser** :
  - Shields : 1000
  - Hull : 8000
  - Damage : 400
  - Inaltérable par les radiations de l'espace profond.
- **Destroyer** :
  - Shields : 5000
  - Hull : 10000
  - Damage : 2000
  - Double dégâts sur les boucliers, moitié sur la coque.
- **Battle Cruiser** :
  - Shields : 12000
  - Hull : 6000
  - Damage : 1000
  - Efficace dans toutes les régions.

3.2 Propriétés des régions

- **Inhabited Solar System** : Région neutre.
- **Asteroid Field** : Avantages pour les Fighters, lourdes pénalités pour les autres vaisseaux.
- **Nebula** : Réduit la précision des tirs, mais avantage les Fighters.
- **Deep Space** : Affecte les boucliers (différemment selon les types de vaisseaux).

4. Mécanique des combats

Étapes :

1. Les flottes sont déplacées vers leurs régions respectives.
2. Les modifications régionales sont appliquées.
3. Les combats se déroulent en un maximum de 3 rounds :
    - Chaque flotte attaque en suivant un ordre aléatoire des vaisseaux.
    - Les dégâts sont appliqués aux boucliers puis à la coque.
4. Résultats :
    - Victoire de l'attaquant si tous les vaisseaux défenseurs sont détruits.
    - Victoire du défenseur si tous les vaisseaux attaquants sont détruits.

Exemple :

- **1er round** :
    - L'attaquant inflige 500 de dégâts.
    - Le défenseur inflige 350 de dégâts.
- **2e round** :
    - L'attaquant inflige 400 de dégâts restants après ajustements.
    - Le défenseur inflige 200 de dégâts restants.

5. Tests et scénarios

5.1 Scénarios prévus

1. **1 Fighter** contre **1 Fighter** dans une région neutre.
2. **50 Fighters** contre **5 Destroyers** et **2 Cruisers** dans un champ d'astéroïdes.
3. **10 Cruisers** contre **3 BattleCruisers** entre une nébuleuse et l'espace profond.

5.2 Résultats observés

- Les pénalités/récompenses des régions influencent fortement l'issue des combats.
- Les interactions spécifiques (Destroyer contre BattleCruiser) respectent les règles.

6. Analyse critique

Points forts :

- Utilisation efficace de la POO (double dispatch, polymorphisme).
- Architecture modulaire et extensible.
- Respect des spécifications de l’énoncé.

Améliorations possibles :

- Ajouter une interface utilisateur graphique.
- Étendre les types de vaisseaux et de régions.
- Améliorer l’efficacité pour les combats impliquant de grandes flottes.

7. Conclusion

Ce projet est un exemple concret de l'application de concepts avancés de programmation orientée objet dans un contexte ludique et stratégique. L'intégration du double dispatch, combinée à une architecture modulaire, garantit la flexibilité et la robustesse du système.