Rapport de Conception : Bataille Navale

YOCOLI Konan Jean Epiphane - Fotso Kamel - Li Jie

4 avril 2025

Table des matières

Introduction

Ce document présente la conception du jeu "Bataille Navale" développé en Java avec une architecture MVC (Modèle-Vue-Contrôleur). L'objectif de ce document est de fournir une analyse claire des choix techniques, des diagrammes de conception et des explications sur les algorithmes.

1 Architecture Globale

L'application utilise une architecture MVC pour séparer les responsabilités :
— **Modèle** : Gère la logique métier (classes `Game`, `Map`, `Navire`).
— **Vue** : Affiche l'interface graphique (classe `GameView`).
— **Contrôleur** : Médiateur entre la vue et le modèle (classe `GameController`).

2 Diagramme de Classes Simplifié

Les diagrammes sont disponible dans le dossier livraison/rapport/

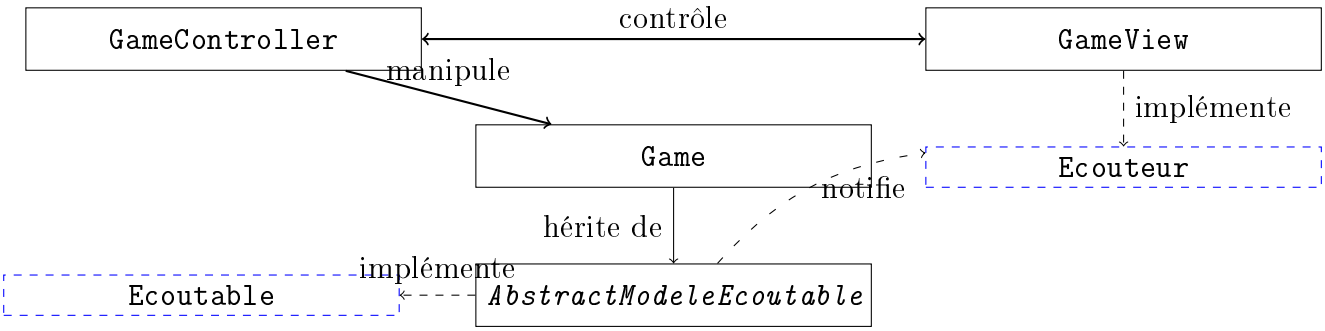

FIGURE 1 – Diagramme Simplifié

3 Pattern Observateur

Le modèle notifie les vues via le pattern Observateur :
— `AbstractModeleEcoutable` implémente `Ecoutable` pour gérer les écouteurs.
— `GameView` implémente `Ecouteur` et se met à jour via `updateGrid()`.

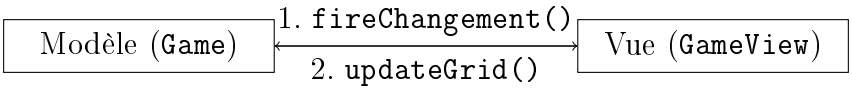

FIGURE 2 – Communication Modèle-Vue via les écouteurs

4 Fonctionnement

4.1 Placement des Navires

— Vérification des collisions et des limites (méthode `Map.placeShip()`)
— Stockage des positions dans une liste de coordonnées
— Transition alternée entre joueurs après placement complet via `Game.switchPlayer()`

4.2 Gestion des Tirs

— Vérification des coordonnées et de l'état de la case (méthode `Map.shoot()`)
— Détection des navires coulés via la réduction de leur vie (`Navire.shooted()`)
— Changement automatique du joueur courant après chaque tir réussi (`Game.switchPlayer()`)
— Notification de la fin de jeu si tous les navires sont coulés (`Map.dead()`)

5 Flux de Jeu

1. Phase de placement :
   — Les joueurs placent alternativement leurs navires.
   — Vérification des positions valides.
2. Phase d'attaque :
   — Les joueurs tirent alternativement.
   — Affichage des résultats (touché, coulé, raté).
3. Fin de jeu lorsqu'un joueur n'a plus de navires.

Conclusion

Cette conception permet une séparation claire des responsabilités et une maintenance facilitée. Les améliorations possibles incluent :

— Ajout d'une intelligence artificielle pour un mode solo.
— Sauvegarde des parties en cours.
— Support réseau pour des parties multijoueurs.