Guillaume ROTH
Vincent FAGNON
Paul DECANTER

Conception Architectures Distribuées

Master 1

Bataille Navale

Dossier d'implémentation

30 Avril 2018

Dépot Git : https://github.com/Frodonche/CAD20172018_SemisCroustillants

Gestion du projet :

Nous avons divisé le projet en deux parties : une partie front end pour l'interface et une back end pour le noyau du jeu. Ainsi nous pouvions développer en parallèle ces deux parties de l'application sans difficulté.

Classes implémentées :

L'implémentation est plutôt fidèle à la conception proposée.

Le pattern Data Access Object change légèrement : nous avons abandonné la classe AbstractGame qui n'avait pas d'utilité dans notre cas, et avons implémenté l'interface GameService et la classe qui en hérite GameDAO. GameDAO comporte deux fonctions :

- serialize(Game g) qui sert à convertir une classe en String afin de créer un document XML dans lequel sont stockées les données de la partie en cours.

- create(String s) qui va lire un document XML pour en extraire les données et ainsi créer un objet Game et instancier les différentes valeurs de jeu.

Les classes Strategy, StrategieA et StrategieB n'ont pas été implémentées non plus mais peuvent l'être aisément. Il faut créer des stratégies de jeu pour l'IA dans les classes StrategieA et Strategie B. La stratégie choisie en début de partie pour l'IA sera stockée dans la classe Joueur. Ensuite il faudra modifier la fonction tir dans la classe Modele afin que l'IA utilise la stratégie choisie.

Nous avons également mis en place le pattern Factory pour les sprites. N'ayant pas eu le temps de mettre en place la gestion des images pour les bateaux, nous avons opté pour une coloration des cases bateaux, tandis que les cases brouillard et eau sont correctement affichées.

En effet, l'idée initiale pour l'affichage des bateaux était de récupérer une image (verticale ou horizontale) d'un bateau, puis de la découper via Java en X cases pour enfin donner ces morceaux d'images aux boutons de la grille correspondant à l'emplacement des bateaux.

Spécificités de fonctionnement :

Notre code couleur sur les grilles est donc le suivant :

Pour la grille du joueur (en bas), le remplissage vert correspond aux bateaux placés. Une case jaune correspond à une case de bateau touchée par l'adversaire (l'IA). Enfin, les contours de case jaunes correspondent aux tirs de l'IA.

Pour la grille de l'IA (en haut), le remplissage rouge correspond aux bateaux touchés. Si l'on tire à côté, une case d'eau se révèlera dans le brouillard.

On ajoutera qu'une ligne est commentée dans la vueGrilleTir, permettant, à des fins de test, de révéler la position des bateaux adverses sous la forme de contours rouges.

Notre programme possède une fonction de chargement, ainsi qu'une fonction de sauvegarde. Ces fonctions ne sont disponibles que sous certaines conditions.

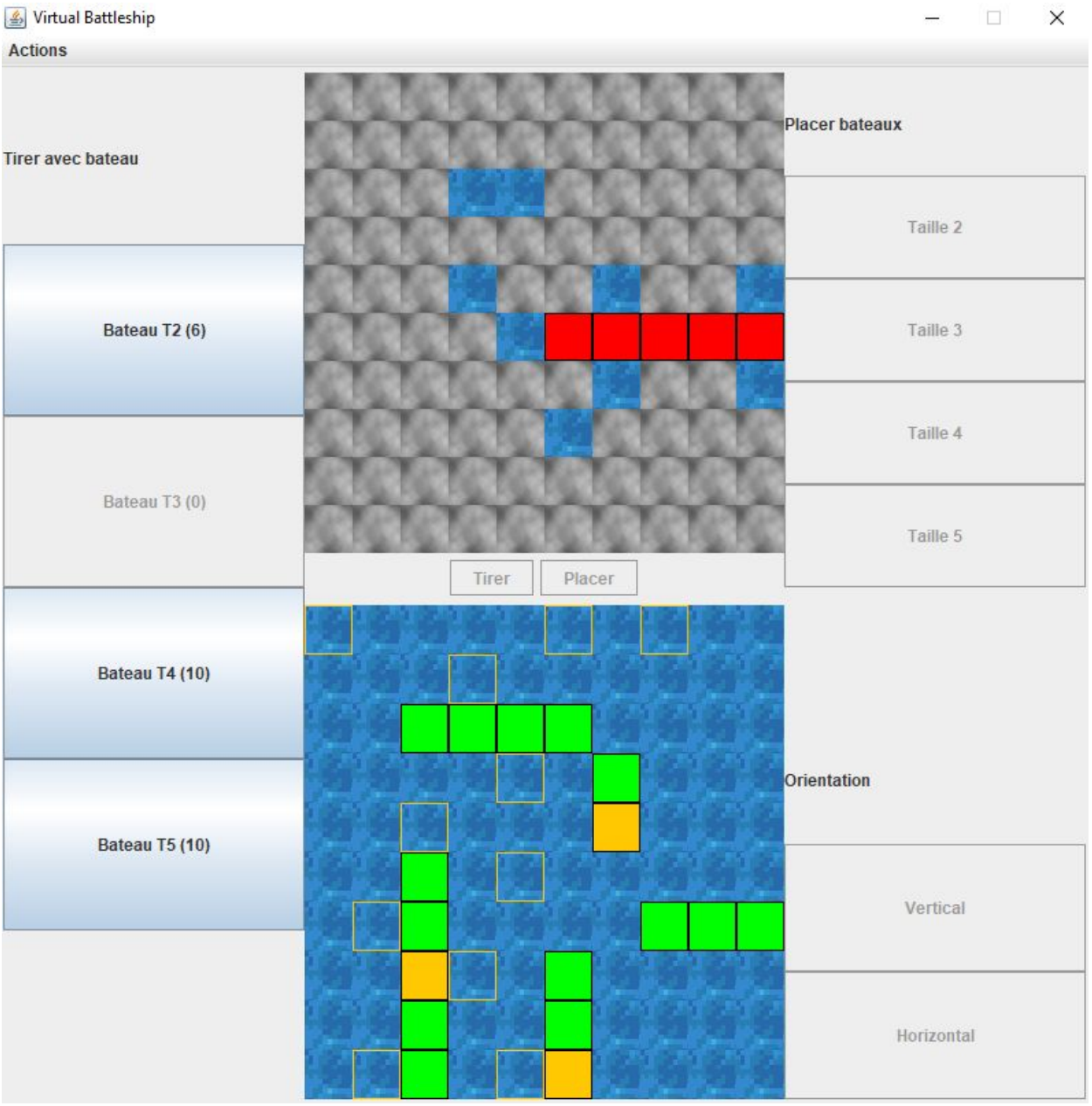
En effet, Save n'est disponible que si l'on a lancé une partie et posé ses bateaux ; cela n'a aucun sens de sauvegarder une partie où des bateaux sont manquants.

De plus, pour des questions de simplicité et par choix d'implémentation, la fonction Load n'est disponible que lorsqu'une partie est lancée (que l'on a passé le menu donc), et qu'on n'a pas encore placé de bateau.

Enfin, on ajoutera que les boutons réagissent de façon dynamique.

En effet, le bouton Placer servant à placer les bateaux ne sera disponible que si une taille de bateau, ainsi qu'une case sont sélectionnées.

Il en va de même pour les boutons relatifs au tir, qui fonctionne comme les boutons de placement, à ceci près qu'ils ne sont disponibles qu'une fois tous les bateaux placés.



Screenshot de l’interface du programme