UNIVERSITÉ
CAEN
NORMANDIE

Complément de POO

Bataille Navale

Guillaume Chauveau
Vincent Rouilhac
Eric Hu
Yassine Kerimi

L2 Informatique
Groupe 4B
Année 2018-2019

Table des matières

1 Introduction

Dans le cadre de notre cours de "Complément de Programmation Orientée Objet", il nous a été demandé de créer un jeu de bataille navale entièrement MVC, avec une vue graphique et une vue terminale.

La bataile navale est un jeu de société au tour par tour dans lequel deux joueurs posent des navires sur une grille 10x10 et essaient de couler tous les navires adverses. Un navire est coulé si toutes les cases qui lui sont associées sont touchées par les tirs adverses.

Lancé en 1978, le modèle model-view-controleur(MVC) est un un motif d'architecture logicielle destiné aux interfaces graphiques. Ce modèle est divisé en trois parties :

Le modèle : il contient le contenu à afficher.

La vue : elle contient l'interface utilisateur.

Le contrôleur : il relie la vue et le modèle en indiquant à la vue ce qu'elle doit afficher et en donnant au modèle les informations captées par la vue qui sont relatives à l'exécution.

2 Eléments techniques

2.1 Le contrôleur

Le contrôleur est initialisé au lancement du programme. Il récupère dans un fichier les données des joueurs, c'est-à-dire leur nom et score. Puis il initialise le modèle et crée les vues. Notre contrôleur instancie deux vues différentes, une vue graphique et une autre vue sur le terminal. Cela nous permet de jouer à la fois sur le terminal et sur l'interface graphique. Afin de relier les vues au modèle, il ajoute aux vues des écouteurs d'évènements qui permettent au modèle de changer ses données. Dans notre bataille navale, 5 écouteurs d'évènements sont ajoutés :

- Trois de ces écouteurs permettent d'actualiser l'état du modèle(MAIN_MENU, CREATING_GAME, PLAYING_GAME).
- Le quatrième écouteur transmet au modèle les tirs des joueurs lors du déroulement d'une partie.
- Le dernier permet de quitter l'application. Pour cela, le contrôleur enregistre les joueurs et leur score dans un fichier avant de quitter.

2.2 Le modèle

2.2.1 Un jeu à n joueurs

Notre modèle est initialisé à l'aide d'un **PlayerManager**. Celui-ci contient toutes les informations sur les joueurs précédents. Il est possible de lancer le jeu avec plus de deux joueurs différents. Si un seul joueur est rentré, un bot servira d'adversaire. Pendant la partie, le joueur sélectionne la case dans la grille de l'adversaire qu'il veut viser. Puis le jeu change automatiquement de joueur courant. Lorsqu'il ne reste plus qu'un seul joueur avec des navires, le modèle passe au statut GAME_OVER. Le gagnant voit son score incrémenté. Puis le modèle revient à l'état MAIN_MENU.

2.2.2 Représentation des bateaux

Nos bateaux sont représentés par 4 attributs :

1. La tête, ou l'origine. C'est le point d'origine du bateau.
2. Leur longueur.
3. Leur orientation, l'un des quatres points cardinaux.
4. La liste des cases touchées du bateau. Si la longueur de cette liste est égale à la longueur du bateau, le bateau est coulé.

Comme vous l'avez deviné, nos bateaux sont représentés de façon calculatoire (ce sont des vecteurs). Cela est certes plus lent mais présente des avantages certains :

— Les bateaux sont indépendants du territoire associé au joueur. Nous n'avons plus besoin de parcourir une grille pour connaître leur emplacement.

— Ils ont leur propre statut que l'on peut obtenir avec la méthode **isSunk**.

2.3 La vue

2.3.1 Fonctionnement général

Nos vues écoutent les changements d'état du modèle. A chaque changement d'état, une interface appropriée est affichée pour l'utilisateur. Selon les choix de l'utilisateurs, des évènements sont envoyés par la vue au contrôleur. Ces évènements peuvent être :

— Des évènements de changements d'état du modèle.

— Des tirs. Ces évènements contiennent le joueur tireur, le joueur visé et les coordonnées d'impact du tir.

— Pour quitter la partie.

Ces évènements correspondent aux écouteurs attachés sur la vue par le contrôleur. Lors d'une partie, les vues récupèrent directement les positions des bateaux à partir du modèle, en accédant au territoire des joueurs du jeu courant.

2.3.2 Spécificités de la vue terminale

L'interface terminal est "non bloquante", en effet la partie bloquante, le scanner des entrées, est placée dans un thread. Ceci nous permet d'exécuter les fonctions sans attendre la réponse du scanner et de jouer sur les deux interfaces à la fois.

A tout moment, l'utilisateur peut quitter la partie en pressant les touches "x" puis "entrée" ou retourner au menu principal en pressant "m" suivi de "entrée".

2.4 Récapitulatif

Lors du lancement de l'application, le controlleur est créé. Il récupère les données des joueurs, si elles existent, et crée alors le modèle. Il crée ensuite les vues. Celle-ci communiquent avec le controlleur à l'aide d'évènements. Ces évènements, créés et lancés par l'utilisateur pendant l'utilisation de l'application, permettent d'actualiser le modèle et de créer une partie puis d'y jouer. Les vues écoutent et changent selon les états du modèle. Pendant une partie, les vues récupèrent directement dans le jeu les données des flottes de navires. De plus, elles captent les coups joués et les envoient au controlleur pour qu'il actualise le modèle. Lorsqu'une partie se termine, le modèle change automatiquement d'état.

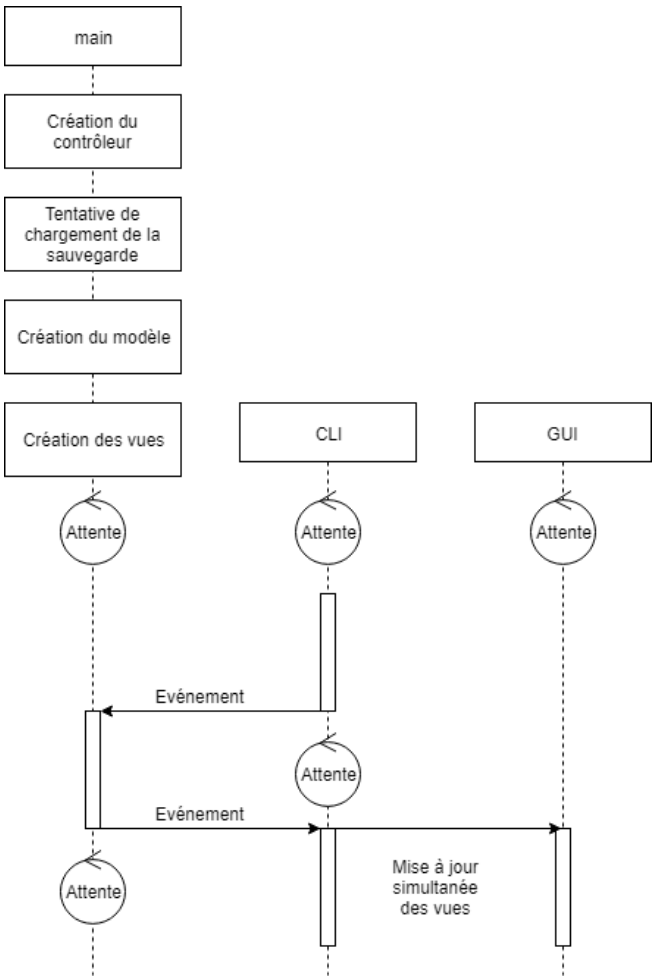

FIGURE 1 – Motif MVC

3 Conclusion

Nous pensons avoir très bien mis en oeuvre et compris le modèle MVC. En effet, nos trois parties communiquent parfaitement entre elles grâce aux divers évènements.

Le sujet a été largement dépassé. En effet, nous pouvons jouer sur la vue graphique et la vue terminale en même temps. De plus, plusieurs joueurs peuvent jouer simultanément au jeu, et une sauvegarde des joueurs a été implémentée.