Rapport

Complément de POO

Bataille Navale

Zineb Azoui 21709992
Maëlys Boudet 21700975
Lucie Lepetit 21203742
Tatiana Yang 21605680

Licence Informatique
Deuxième année
Session : 2019

Table des matières

1 Introduction

Dans le cadre de notre deuxième année en licence informatique, au cours du second semestre, nous devons rendre un devoir pour l'UE **Programmation Avancée** et plus particulièrement en **Complement de POO**, pour Programmation Orienté Objet .
Le Sujet du devoir était la Bataille navale, aussi appelé `touché-coulé`, il s'agit à la base d'un `jeu de société` dans lequel deux joueurs placent des « navires » sur une grille tenue secrète et tentent de toucher les navires du joueur adverse. Le gagnant est celui qui parvient à couler tous les navires de l'adversaire avant que tous les siens n'aient été coulés. Un navire est coulé si chacune des cases qui le compose ont été touchée par l'adversaire.
La première version commerciale du jeu fut publiée en 1931 par la Starex `Novelty Co.` sous le nom de Salvo. Ce jeu est devenu populaire lors de son apparition en 1943 dans les publications américaines de divertissement de la Milton Bradley Company. L'entreprise l'exploita sous la forme papier jusqu'en 1967, où elle sortit une nouvelle version du jeu sous forme de plateau, puis en réalisa une version électronique en 1977. [1]

Nous allons au travers de ce rapport vous présenter la conception de ce projet en quatre points principaux.

Tout d'abord l'objectif du projet, ce qu'il fallait faire.
Nous décrirons ensuite la façon dont nous avons organisé le projet.
Nous parlerons dans un troisième point de l'architecture du Projet : comment nous avons réparti les différentes classes du jeu et comment nous les avons ordonnées dans différents répertoires.
Une partie conclusion permettra de récapituler les objectifs qui ont été remplis et présentera les éventuelles améliorations possibles qui pourraient être ajoutées au jeu.

1. source : `https://fr.wikipedia.org/wiki/Bataille_navale_(jeu)`)

2 Objectif du Projet

Ci-dessous l'intitulé de notre devoir :

Ce devoir a pour objectif de réaliser un jeu de bataille navale en permettant à un utilisateur de jouer contre l'ordinateur qui joue de façon aléatoire.
Dans un premier temps, il s'agit de réaliser un jeu jouable pour un humain.
Bien entendu, une interface graphique devra être réalisée, mais le jeu pourra être joué dans un terminal aussi.

2.1 Qu'est ce qu'une bataille navale ?

Au début du jeu, chaque joueur place à sa guise tous les bateaux sur sa grille de façon stratégique. Le joueur et son adversaire possèdent chacun 5 bateaux. Le but étant de compliquer au maximum la tache de son adversaire, qui est de détruire tous nos navires. Bien entendu, le joueur ne voit pas le contenu de la grille de son adversaire.
Une fois tous les bateaux en jeu, la partie peut commencer. Un à un, les joueurs se tirent dessus pour détruire les navires ennemis.
Si une partie d'un bateau est touchée, on dit que le joueur à touché.
Si la partie du bateau touché était la dernière partie non touchée du bateau, il s'agit alors d'un touché coulé. L'affichage sera différents selon si la case sur laquelle le joueur à tiré était une partie d'un bateau ou non.

2.2 Ce qu'il fallait faire

Le but principal de ce projet était de réaliser un jeu de bataille navale jouable entre un joueur humain et un joueur aléatoire.

Nous avons tout d'abord commencé la conception de notre bataille navale de façon à ce qu'il soit jouable en ligne de commande. La difficulté de ce devoir a été la création de la grille qui représente notre Mer, *zone où nous poserons nos navires*, qui est initialement une grille carré de taille 10. La complexité que nous avons rencontré se trouve dans la différenciation d'une simple case et d'un bateau. En effet, une case peut être une partie d'un bateau mais elle

peut également être nulle (c'est à dire ne pas avoir un bateau posé sur cette case).

Notre seconde difficulté rencontrée fut dans la réalisation de la partie graphique du jeu. Nous avons tout d'abord trouvé un grand obstacle dans l'indépendance du terminal par rapport aux graphismes. Mais le fait d'avoir eu a travailler en parallèle sur un autre projet en TPA (Travaux Pratique Avancée) nous a permis de mieux comprendre l'utilisation d'un MVC et de pouvoir le mettre en place dans notre Bataille Navale.

3 Architecture du projet

3.1 Arborescence des packages

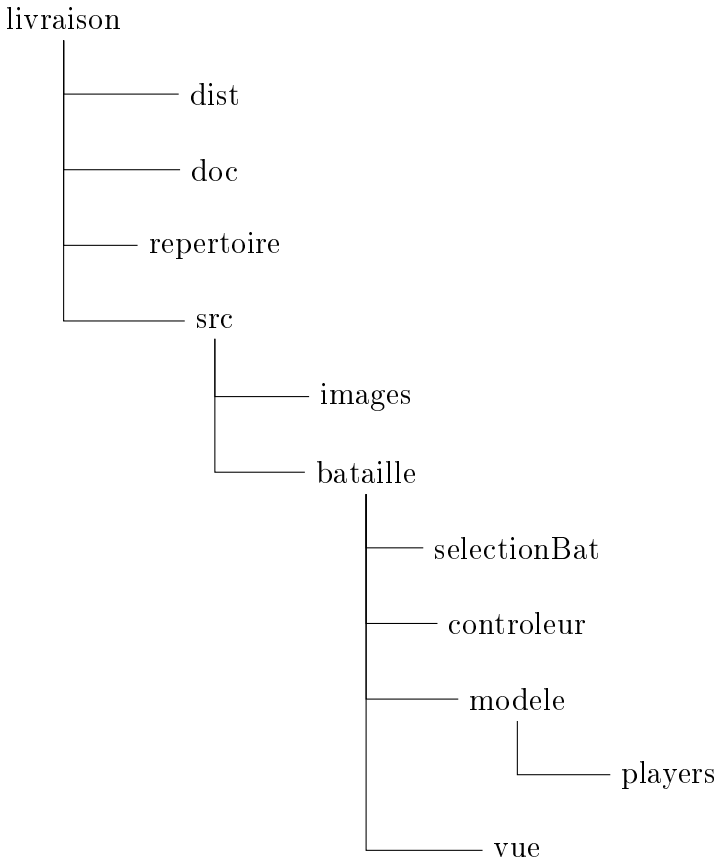

FIGURE 1 – Arborescance des packages

Sur cette figure nous pouvons voir l'arborescence de notre jeu. Nous avons décidé de créer dans un dossier src un répertoire bataille contenant trois répertoires différents : le modèle, la vue et le contrôleur.
Dans ces répertoires, nous avons rangé des fichiers java indispensables au jeu ainsi que d'autres sous-package, notamment pour le répertoire modèle. Ce dernier contient plusieurs fichier java, également indispensables, et un package players.

Le choix de ce rangement a été pour nous évident. Il nous a permis de bien créer une différence entre les différents fichiers java utilisés lors de la conception du jeu.

3.2 Diagramme des classes

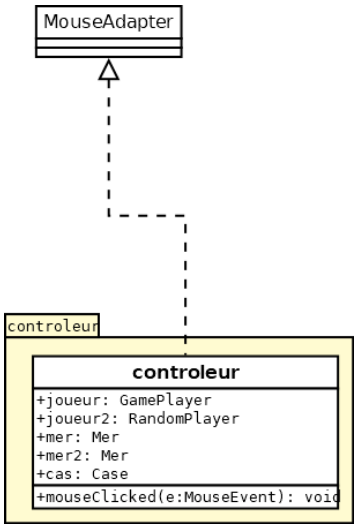

FIGURE 2 – Diagramme de classe dans le package controleur

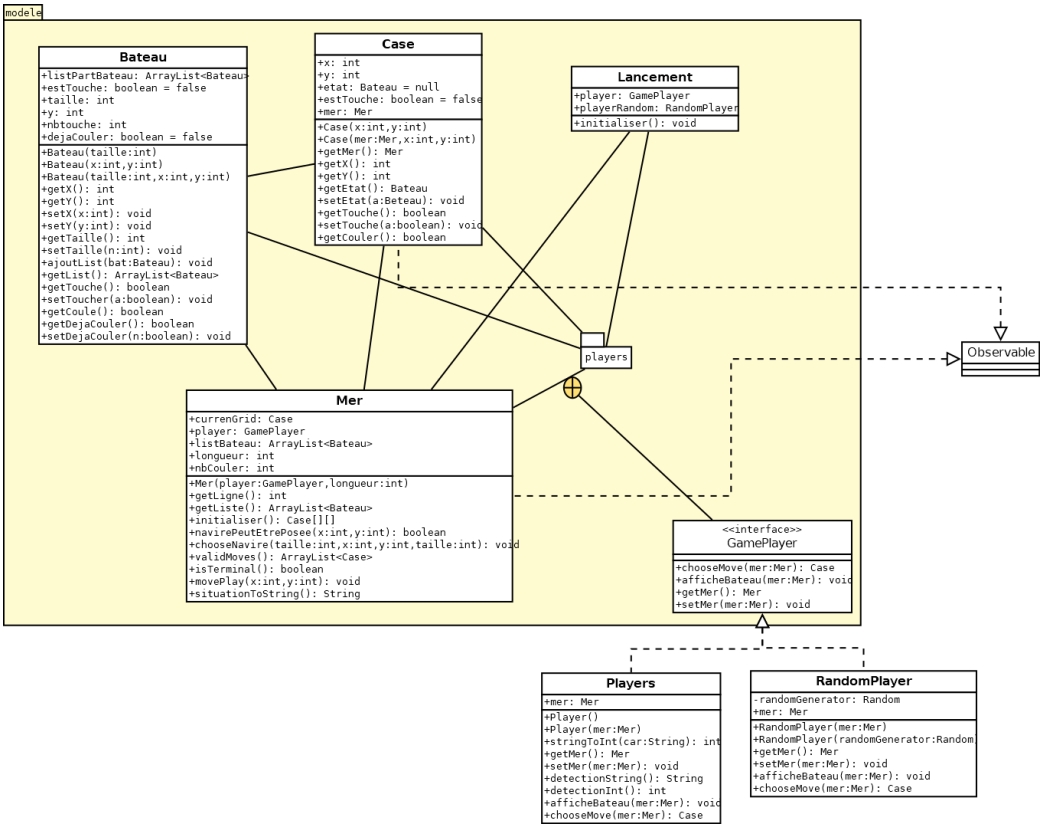

FIGURE 3 – Diagramme de classe dans le package modele

Le lanceur :

`Main`

Le fichier Main permet de lancer notre jeu. Ce fichier créé deux joueurs, un joueur de type `Player` et le second de type `RandomPlayer`. Lors du lancement du jeu, le joueur peut choisir de le lancer dans le terminal en indiquant le chiffre 0, ce qui lance la classe Orchestrateur.
Sinon, le jeu se lance de manière graphique avec la classe de la Fenêtre du menu MenuVue.

Le déroulement du jeu :

`Orchestrateur`

L'orchestrateur possède deux parties qui sont des instances de la classe Mer et deux joueurs : un joueur aléatoire (random) et un humain. L'affichage dans le terminal graphique des deux plateaux des deux joueurs est fait, et

les mouvements de toucher-couler sont faits lors des entrée dans le terminal grâce au scanner. le joueur humain y choisit ses coups.

Modèle :

`Mer`

Le modèle de notre projet est la classe Mer qui est observale, et est observé par la classe MerVue présent dans le package vue. La Mer est un tableau d'objets d'instances de Case, chaque case peut avoir un état, celui d'appartenir à un bateau ou non. On retrouve d'autre part les joueurs qui héritent de la classe GamePlayer et est composé de deux joueurs. Le premier est un joueur humain. Si le joueur choisit de jouer avec la partie graphique, il peut jouer grâce au MouseListener présent dans le package controleur. Ce MouseListener lui permet d'effectuer une action lors d'un clic sur une case. L'autre joueur est un joueur de type RandomPLayer. il joue ses coups de manière aléatoire grâce à une fonction chooseMove, qui reprend une méthode nommée `validMoves` instanciée dans la classe Mer.

Le Contrôleur :

`Contrôleur`

Le contrôleur est ici une classe qui extend un MouseAdapter. En effet, grâce à cet héritage, on redéfinit la façon dont les clics sont utilisés. Le choix d'hériter de MouseAdapter plutôt que d'implémenter MouseListener se fait dans le code. Nous utilisons pour notre jeu seulement, l'évènement d'un clique. Si nous avions fait le choix d'implémenter l'interface MouseListener, nous aurions dû redéfinir toutes les méthodes de la classe. Tandis qu'avec MouseAdapter, nous n'avions qu'à redéfinir la méthode souhaitée. Dans notre cas il s'agissait seulement de la méthode `mouseClicked`.

Ici, lorsque le joueur humain clique sur une case dans la Mer, dans la méthode précédemment appelé `mouseClicked` nous reprenons les positions X et Y de la case cliquée. Nous faisons jouer le joueur avec ces coordonnées, ainsi que le joueur aléatoire. Ainsi une modification dans la Mer est faite et elle est relayée à ses Observateurs qui automatiquement paint le composant de la grille, en fonction de s'il y a un bateau adverse ou non, c'est à dire qu'un bateau soit touché ou non, et si ce bateau est coulé.

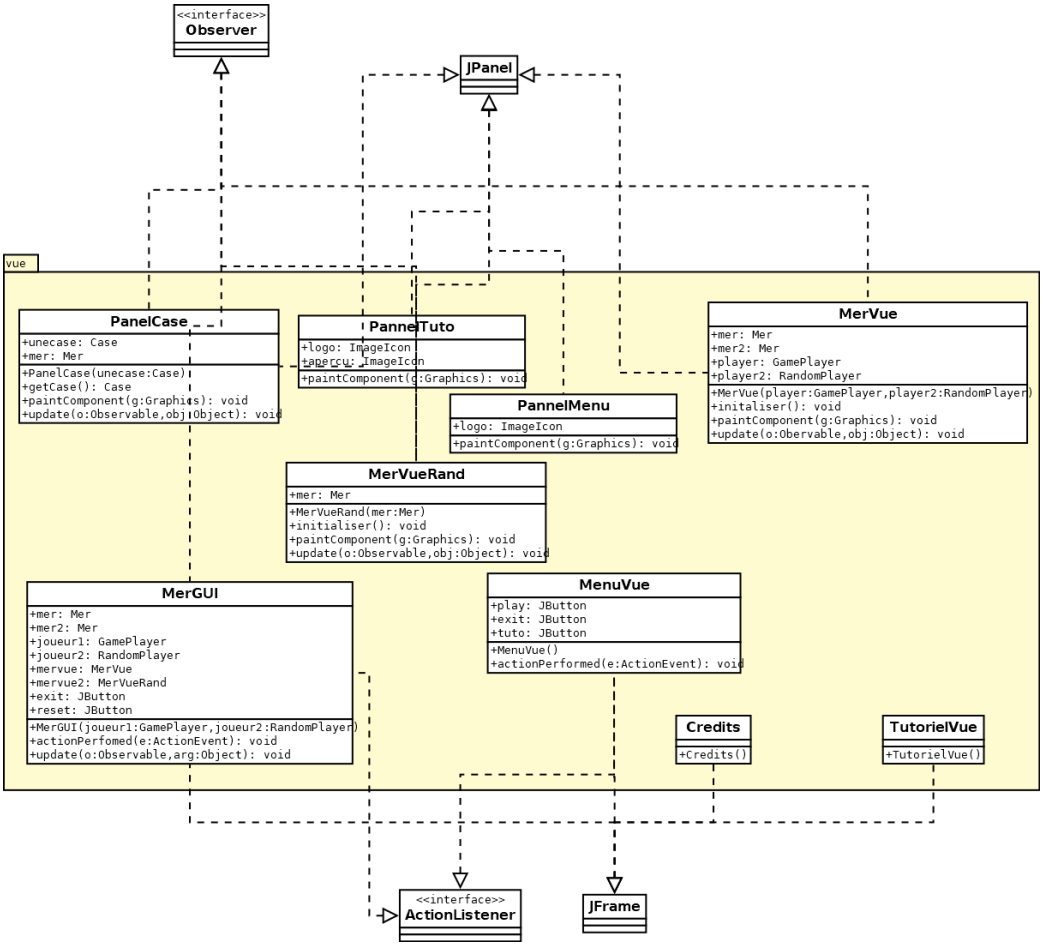

FIGURE 4 – Diagramme de classe dans le package vue

La Vue :

`MerVue`

La vue est l'affichage graphique de la grille de jeu, dans notre cas elle est représentée par la classe Mer. Il s'agit d'un Observateur qui observe les changements et les affiche grâce à la fonction update.

Les Fenêtres :

Menu Principal :

Lors du démarrage de notre jeu, l'invité de commande demande à l'utilisateur de choisir son type de jeu. (Soit en ligne de commande soit graphiquement) S'il entre le nombre 0, alors, notre jeu se lance dans notre terminal. Sinon, il

ouvre un Menu Principal `MenuVue`.
Dans ce menu, on trouve trois boutons :

- Exit, qui permet de quitter le jeu. Ce bouton lance à son tour une fenêtre de Crédits. L'utilisateur doit fermer la fenêtre Pour tout quitter correctement.
- Tutoriel, qui permet de connaitre le fonctionnement du jeu. On retrouve ici des composants graphiques placés grâce à la méthode qui vient de JPanel, appelée paintComponent. On trouve un logo en haut de la fenêtre, ainsi qu'un rectangle et du texte que nous avons pu placer aussi avec cette méthode. On y trouve également un aperçu du jeu, et une image de fond.
- Play, ce bouton lance la fenêtre de sélection des bateaux, puis la fenêtre de jeu.

`Fenetre de Selection des bateaux:`
Cette fenêtre nous permet de sélectionner l'emplacement de nos cinq bateaux qui font respectivement cinq, quatre et trois cases de longueur, les deux derniers bateaux font deux cases de longueurs.
On retrouve dans cette fenêtre deux champs d'écriture et un champs de sélection demandant respectivement de choisir :

- une colonne entre A et J.
- une ligne entre 1 et 10.
- une direction `Horizontale` ou `Verticale`.

Pour pouvoir placer un bateau comme vous le souhaitez, et si jamais vous saisissez des information pour un bateau qui ne peut être placer car l'emplacement que vous avez choisis ne convient pas. Un message d'erreur s'affiche dans une fenêtre JPop et vous invite à essayer de nouveau.
Afin de pouvoir créer un champ de formulaire nous nous sommes inspiré d'un code retrouvé sur internet. openclassrooms

`Fenetre du jeu:`
La fenêtre du jeu est principalement composée de deux vues : deux MerVue présentent grâce au modèle MVC.
Les deux joueurs combattent dans cette fenêtre.
Le joueur humain joue sur le Panel de gauche qui est cliquable.
Comme dans le jeu initial, les deux joueurs jouent chacun leur tour jusqu'à ce que l'un des deux remporte la partie. Nous pouvons le savoir grâce à une méthode "isTerminal" qui vérifie si la partie est finie. On retrouve aussi deux boutons, un bouton exit, pour pouvoir quitter le jeu, et un bouton reset, qui

permet de recommencer la partie.

Ainsi, l'utilisation de MVC nous a permis de concevoir un jeu qui peut se jouer de manière indépendante. Dans le terminal le joueur doit entrer des données, tandis que dans la partie graphique, le joueur a simplement besoin de cliquer sur les `PanelCase` que nous avons créé.

`JDialog`:
Dans la fenêtre MerGUI, à la fin de la partie, grâce à la methode isTerminal de la mer, on affiche un Jdialog.

Ce JDialog est là pour afficher si le joueur a gagné la partie ou non.
On y affiche une petite fenêtre avec une image d'un trésor si le joueur a gagné, ou un pirate s'il a perdu.
Deux boutons JButton y sont present : un pour rejouer, qui relance la classe de la sélection de bateau, et un autre bouton quitter, qui sort du jeu et affiche la fenêtre de crédit.

4 Conclusion

Durant la conception du projet, nous nous sommes rendu compte que la programmation orienté objet est très intéressante et permet de faire plusieurs choses que nous ne pourrons pas faire avec d'autres langages de programmation. Elle permet par exemple la création d'une instance en fonction d'une classe créée préalablement. De plus, le Pattern MVC(Modèle, Vue, Controleur), nous a permis de créer notre jeu en utilisant deux parties indépendantes : un partie en ligne de commande et une partie graphique.

5 Sources

Images : libre de droits sans but commercial
Trésor : pngimg.com
Pirate : wikipédia
Crédit : Flickr Charles W. Bailey Jr.
Tutoriel : pexels.com
Menu et fond du jeu : Public Domain Pictures