UNIVERSITÉ
CAEN
NORMANDIE

PESTEL Alexis
COCHARD Olivier
L2 Informatique
Groupe 2B

GÉNIE LOGICIEL
PROJET DE BATAILLE NAVALE

Interfaces Graphiques et Design Patterns

2022-2023

SOMMAIRE

I/ MODE D’EMPLOI

Afin de profiter pleinement de l’expérience de notre jeu, nous allons vous expliquer comment compiler, éxécuter et jouer à notre jeu.

1. Compilation & Éxécution

Tout d’abord, placez vous dans le répertoire /src/ du dossier /livraison/ à partir d’une invite de commande. Ensuite, il vous suffira d’éxécuter la commande suivante afin de compiler le jeu et ses dépendances :

• *javac -d ../build game/*.java*

Ensuite, pour lancer le jeu, vous n’aurez plus qu’à éxécuter l’une des commandes suivantes, toujours au même endroit :

• *java -cp ../build game.ExecutableInterface* : pour lancer le jeu en interface graphique ;
• *java -cp ../build game.ExecutableConsole* : pour lancer le jeu en invite de commande.

2. But du Jeu

a. Invite de Commande

Maintenant que le jeu est lancé, vous n’avez qu’à suivre les instructions affichées dans l’invite de commande afin de profiter pleinement du jeu ! Une fois que la partie sera terminée, il vous faudra relancer le programme pour jouer une nouvelle partie.

b. Interface Graphique

Lorsque les deux grilles sont affichées sur votre écran, il vous suffit de cliquer sur l’une des cases de la grille de droite pour commencer la partie, la grille de gauche étant la vôtre ! Dès qu’un bateau est coulé, qu’il soit à vous ou à votre adversaire, celui-ci s’affichera entièrement en **noir**. Si vous voyez encore une ou des cases **rouges**, c’est qu’un bateau est encore debout ! Une fois que la partie sera terminée, le statut de celle-ci s’affichera en bas de l’interface et il vous faudra relancer celle-ci pour rejouer une partie.

II/ MODÈLE

Les fonctions utilitaires sont l'ADN d'un logiciel et veillent à son bon fonctionnement.

1. Conception

a. Classe Bateau

La classe « *Bateau* » est une classe objet qui représente un bateau individuel, qui sont utilisés dans la classe « *Plateau* » composée de :

- une taille (int [2, 5]) ;
- des coordonnées (2 int [0, 9]) ;
- une orientation (int [0, 1]) ;
- la valeur du numéro du bateau (int [1, 5]) ;
- un booléan qui dit si le bateau est coulé ou non (bool).

b. Classe GrilleDesTirs

Possède :
-une matrice de int de taille 10x10.
Cette classe permet de savoir où le joueur a déjà tiré ou non avec des valeurs allants de 0 à 1. La valeur 0 signifie que le joueur n'a pas encore tiré à cette coordonnées et donc le 1 signifie qu'un tir a déjà été effectué.

c. Classe HashMapBateaux

Possède :
-un HashMap qui signifie combien de coup sont encore nécessaires pour couler les bateaux, la première valeur int est le numéro du bateau et le deuxième int est le nombre de coups restants.
Cette classe est donc utile pour les deux joueurs afin de savoir combien de bateaux il reste à couler, et leur vie restante.

d. Classe Plateau

Possède :
-un nom de joueur (string),
-une matrice de int de taille 10x10,
-une grille de tirs (GrilleDesTirs),

-un arrayList avec les bateaux (ArrayList<Bateau>),
-un HashMap de bateaux (HashMapBateaux),
-un bool qui dit si le plateau est un bot ou non (bool).
Cette classe est très importante elle représente le plateau d'un joueur. Le plateau peut être un bot comme un joueur. Elle rassemble 2 matrices de int en 10x10, une pour les tirs exécutés par le joueur et une autre pour le placement des bateaux. Elle possède aussi une liste avec tous les bateaux en vie et leur nombre de points de vie.

e. Adversaire

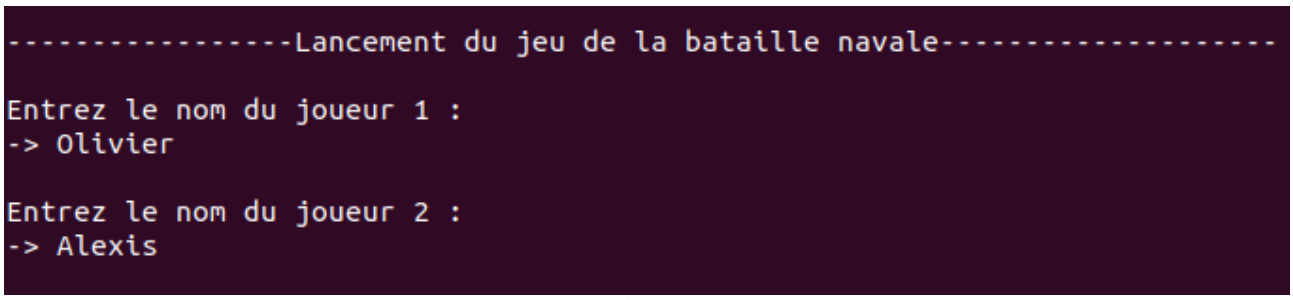
Le bot n'est pas très intelligent, il se contente de tirer aléatoirement sur la grille adverse en prenant la précaution de ne pas tirer deux fois au même endroit. Il est donc très peu probable de le voir remporter une partie à moins qu'il ait énormément de chance. Une amélioration possible aurait été de le faire focus un bateau à partir du moment où il touche un bateau adverse.

f. Génération aléatoire des bateaux

Les bateaux sont donc générés de façon complètement aléatoire que ce soit la position ou encore l'orientation. Ils ne peuvent pas se chevaucher ni dépasser de la grille, seul la taille et leur numéro n'est pas aléatoire. Une fois tous les objets Bateau générés, une matrice à l'aide de leur position et orientation les place directement en mettant pour valeur le numéro du bateau. Une amélioration possible ici aurait été de pouvoir plus les customiser par exemple faire en sorte qu'ils puissent avoir 2 de largeurs ou même avoir des formes complètements bizarres (exemple un bateau en T).

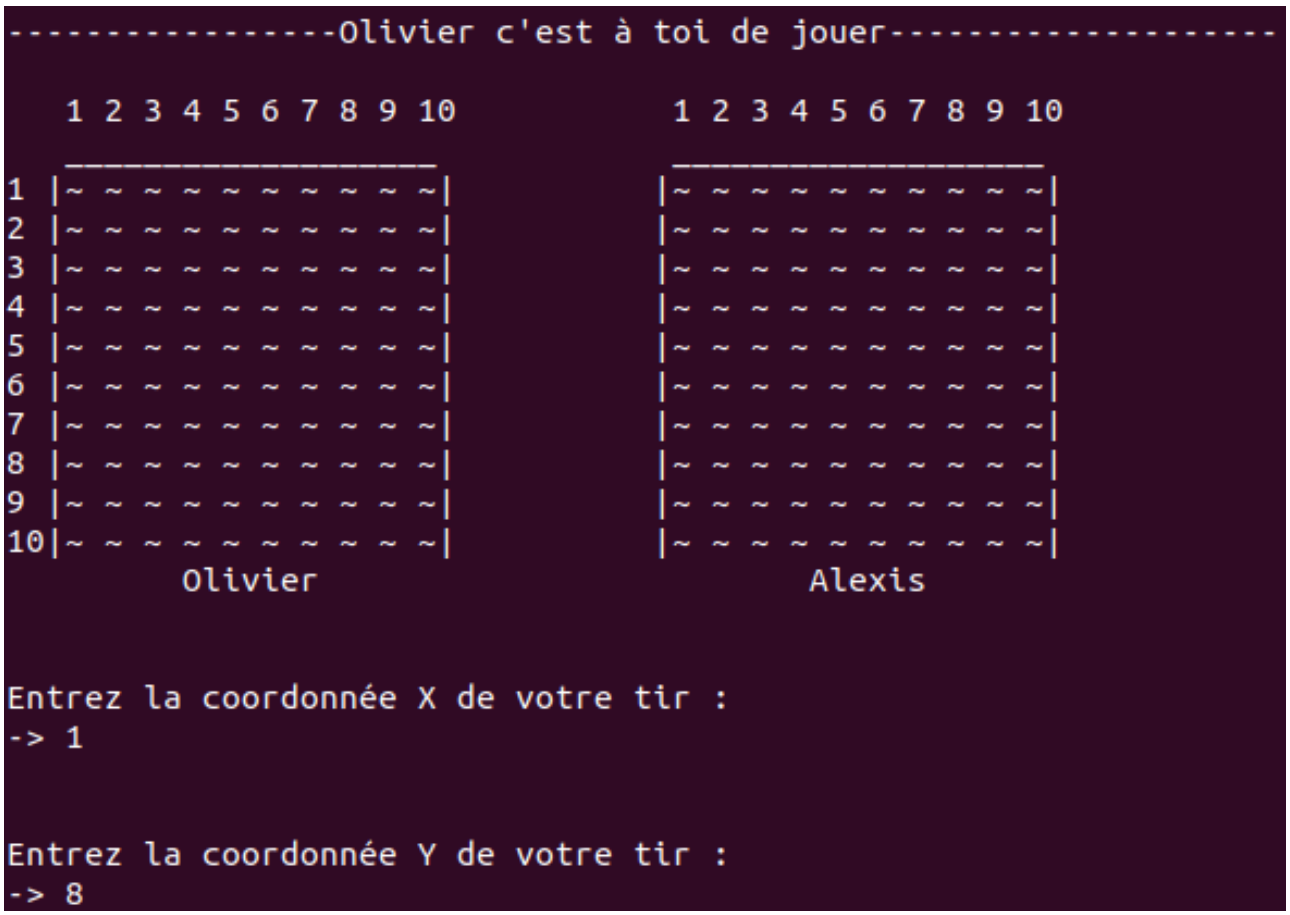
2. Classe ExecutableConsole

Possède :
-deux plateaux de joueurs (Plateau),
-un plateau du joueur courant (Plateau),
-un scanner pour gérer les inputs en console (Scanner),
-une classe Random pour gérer l'aléatoire (Random).
Cette classe créer donc 2 plateaux, ici un bot et un vrai joueur et met le vrai joueur en plateau courant. Les noms des plateaux sont demandés dès le début, et les plateaux se font remplir de bateaux aléatoires, puis un tour de jeu commence.



Exemple de début de lancement

A chaque nouveau tour, si c'est le bot qui joue alors il joue aléatoirement puis s'il n'a pas gagné alors un nouveau tour de lance. Mais si c'est un vrai joueur qui est le plateau courant alors un input de coordonnées de tirs est demandés.



Exemple de début de tour d'un vrai joueur

Si jamais le tir touche une cible alors une méthode regarde le numéro du bateau qui a été touché et réduit de 1 point de vie ce bateau. Et si jamais ce bateau coule alors cette méthode regarde s'il reste au moins un bateau en vie pour déterminer une victoire ou non.

III/ VUE & INTERFACE GRAPHIQUE

Un jeu de bataille navale n'est rien sans l'affichage de celui-ci. C'est donc pourquoi cette partie est tout aussi importante que le **Modèle**, qui lui veille au bon fonctionnement du jeu.

1. Fonctionnement

La classe « *VueBatailleNavale* » hérite de la classe « *JFrame* » et implémente l'interface « *ActionListener* ». La classe contient plusieurs variables d'instance, telles que :

• Deux objets « *Plateau* », pour le joueur et l'adversaire ;
• Deux objets « *JPanel* », pour les plateaux de jeu ;
• Deux tableaux à deux dimensions de « *JButton* », pour les cases des plateaux de jeu ;
• Un « *JLabel* » servant à afficher les messages en jeu (tels que « *Touché* », « *Coulé* » ou « *Gagné* » ;
• Un booléen « *tourJoueur* » afin de savoir qui doit jouer ;
• Une constante « *DIMENSIONS* » afin de déterminer la taille des plateaux de jeu ;
• Et enfin un objet « *Bateau* » pour la manipulation des tableaux.

Le constructeur de la classe prend en entrée deux objets « *Plateau* » et crée la fenêtre de jeu en initialisant les plateaux de jeu, en créant les boutons pour chaque case de chaque plateau, en affichant les bateaux du joueur et en initialisant la grille de tirs du joueur. La méthode « *actionPerformed* » est appelée à chaque clic sur un bouton et est utilisée pour gérer les actions du joueur en fonction de la case cliquée.

Une classe utilitaire supplémentaire, « *afficheCoule* », permet de recolorer l'ensemble d'un bateau coulé au dernier clic en noir.

2. Affichage

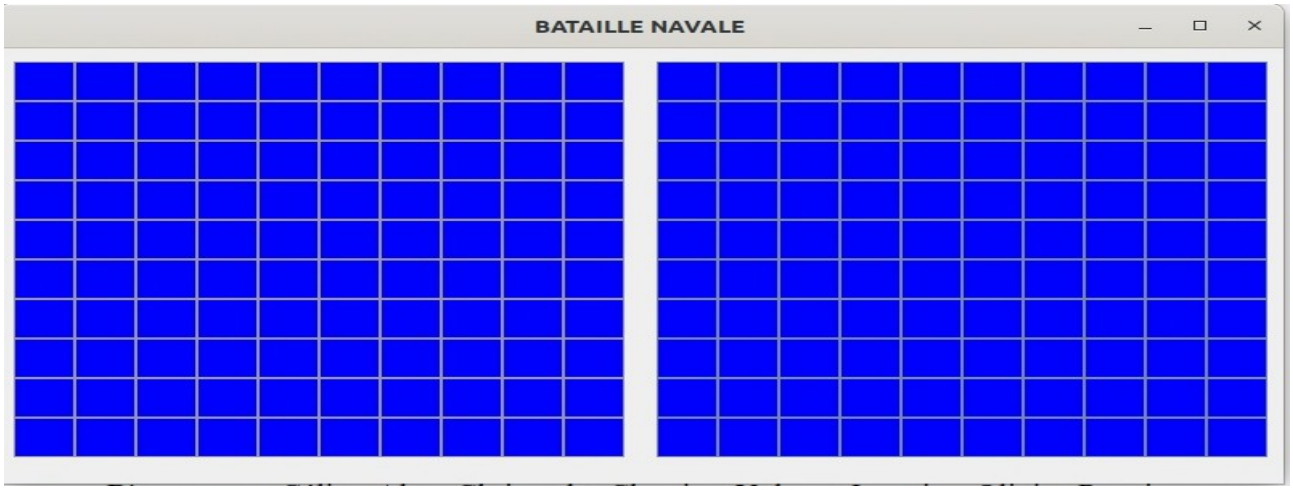


Illustration de l'interface lorsqu'on éxécute celle-ci.

À gauche, la grille du joueur, avec ses bateaux placés aléatoirement sous les cases bleues. À droite la grille de l'adversaire, incarné par un ordinateur, avec le même procédé pour ses bateaux. Lorsque le joueur clique sur l'une des cases de la grille de son adversaire, celle-ci devient sois grise, c'est-à-dire qu'aucun bateau n'était sous celle-ci, ou rouge, ce qui veut donc dire qu'un bateau se cache ici.

L'adversaire joue à chaque fois juste après le coup du joueur, rendant le jeu plus fluide pour ce dernier. De plus, pour ne pas se perdre dans les cases sur lesquelles on clique, ces dernières se mettent en évidence lorsqu'on les survole. Aussi, en bas de l'interface se trouve un « *JLabel* », indiquant les actions effectuées par le joueur.

Une fois la partie finie, tous les boutons se désactivent et le « *JLabel* » placé au dessous affiche le gagnant de la partie.

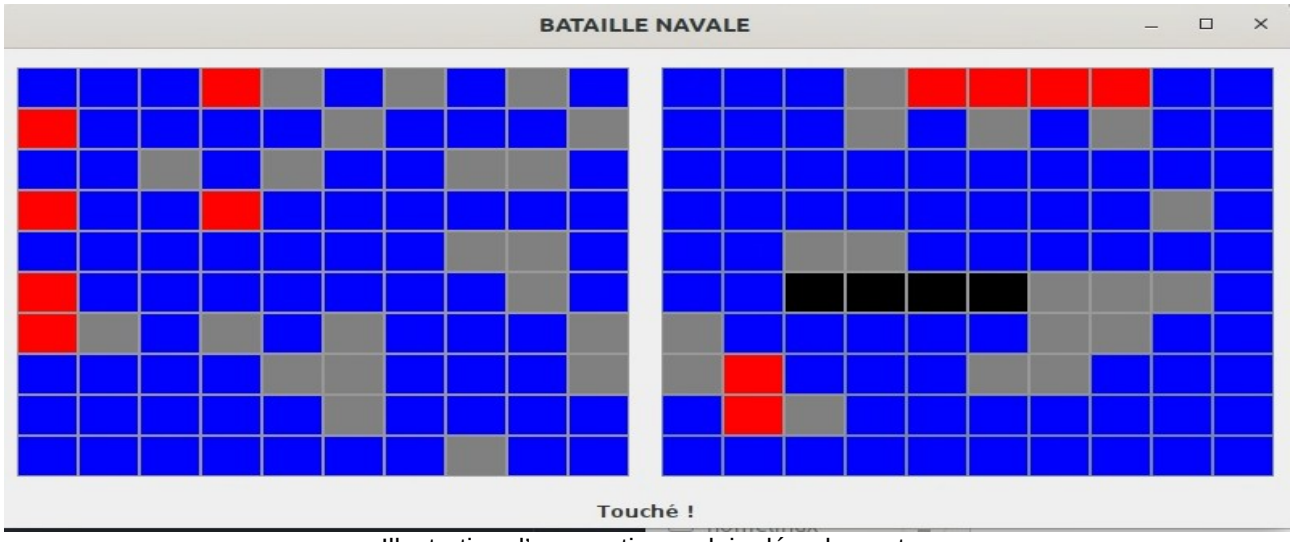


Illustration d'une partie en plein déroulement.

IV/ AMBITIONS & CONCLUSION

C’était un long projet à coder et malgré le fait que nous n’étions que deux on s’en est plutôt bien sorti. Nous avons plusieurs idée d’améliorations qui auraient été intéressantes que ce soit sur le modèle et l’interface. Comme le fait de pouvoir choisir son nom, ajouter du son, ajouter plus d’effets (explosion, bateau qui coule…), pouvoir choisir son nom sur l’interface, le fait de choisir ses bateaux parmi une liste, choisir le nombre de bateaux ou encore choisir la taille de la grille… Mais l’essentiel est déjà là que ce soit sur la version console ou en interface.

Ce projet nous a permis de bien appliquer les concepts de java Swing et d’expérimenter de nouveau avec les interfaces en générales.