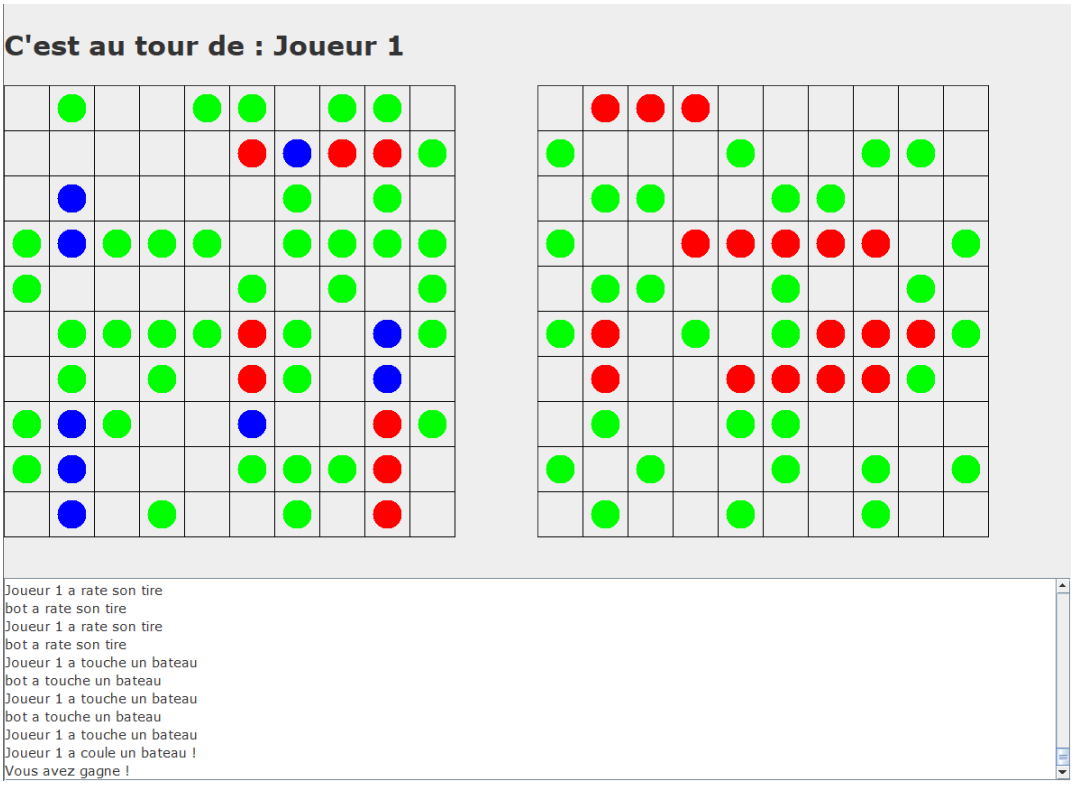
Université de Caen

Rapport complément POO

RAPPORT DU PROJET DE BATAILLE NAVALE



MARTIN MAXENCE - 21807030
MEYER ARTHUR - 21805134

PROFESSEUR : C. Charrier, Y. Mathet
23 Avril 2021

Table des matières

1 Présentation du projet

1.1 La bataille navale

Le projet a réalisé est une bataille navale. Un jeu très connu qui se joue a 2 joueurs ou chacun dispose sur une grille qu'il tiens secrète un flotte de bateaux de différentes tailles. A tour de rôle ils tentent un tir sur une coordonnées de la grille adverse qui lui répond si il a, ou non, touché un bateau et le cas échéant si le bateau est coulé. Un bateau est dit coulé lorsqu'il a été touché sur toutes sa longueurs (2 fois pour un bateau long de 2 cases, 4 fois pour un bateau de 4 etc ...).

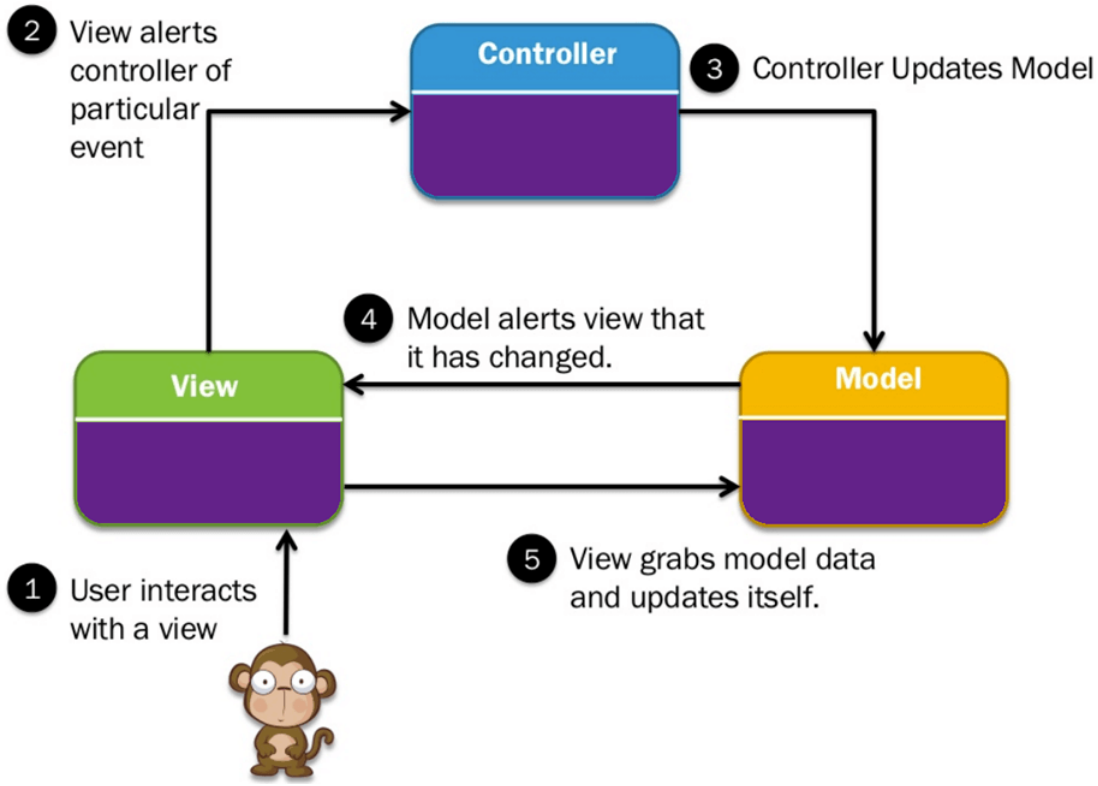
Le joueur coulant en premier toutes la flotte ennemi gagne la partie.

1.2 le MVC

MVC ou modèle-vue-contrôleur est une architecture logicielle pour les interfaces graphiques. Il est composé de trois modules :

- Modèle qui contiens les données
- Vue qui est l'interface graphique
- Contrôleur qui agit sur le modèle

Le contrôleur agit sur le modèle pour par exemple le faire changer d'état et la vue va lire les données du modèle pour l'afficher.



Dans notre cas le contrôleur sera un "listener" qui récupère l'action du joueur pour l'effectuer sur le modèle (un tir). L'interface va ensuite afficher au joueur le résultat a partir des données du modèle (réussi ou raté).

2 Organisation du projet

2.1 arborescence des packages

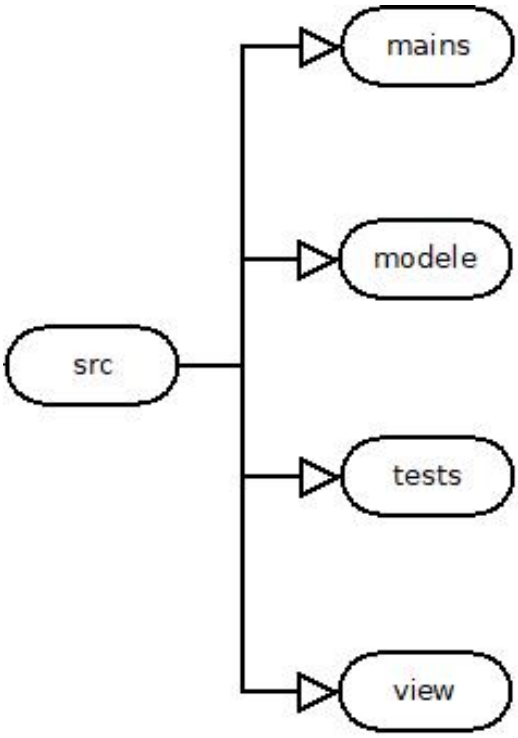


FIGURE 1 – arborescense des packages

Notre projet est constitué d'un dossier principal intitulé "src", ce dossier regroupe quatre autre répertoire nommé "mains", "modele", "tests" et "view". Chacun de ces dossiers contient plusieurs fonctions rangés parmi ces différents répertoire. Le dossier "mains" regroupe différentes classes parmi lesquelles ont peut lancer le jeu. Celui de "modele" possède des classes pour mettre en places les bateaux alliés et ennemies ainsi que bien d'autre. Le répertoire "tests" comme son nom l'indique permet d'exécuter de nombreux tests pour chercher certain défaut et les corriger par la suite. Et enfin le dossier "view" qui contient de nombreuses classes pour l'affichage du jeu.

2.2 diagramme de classe

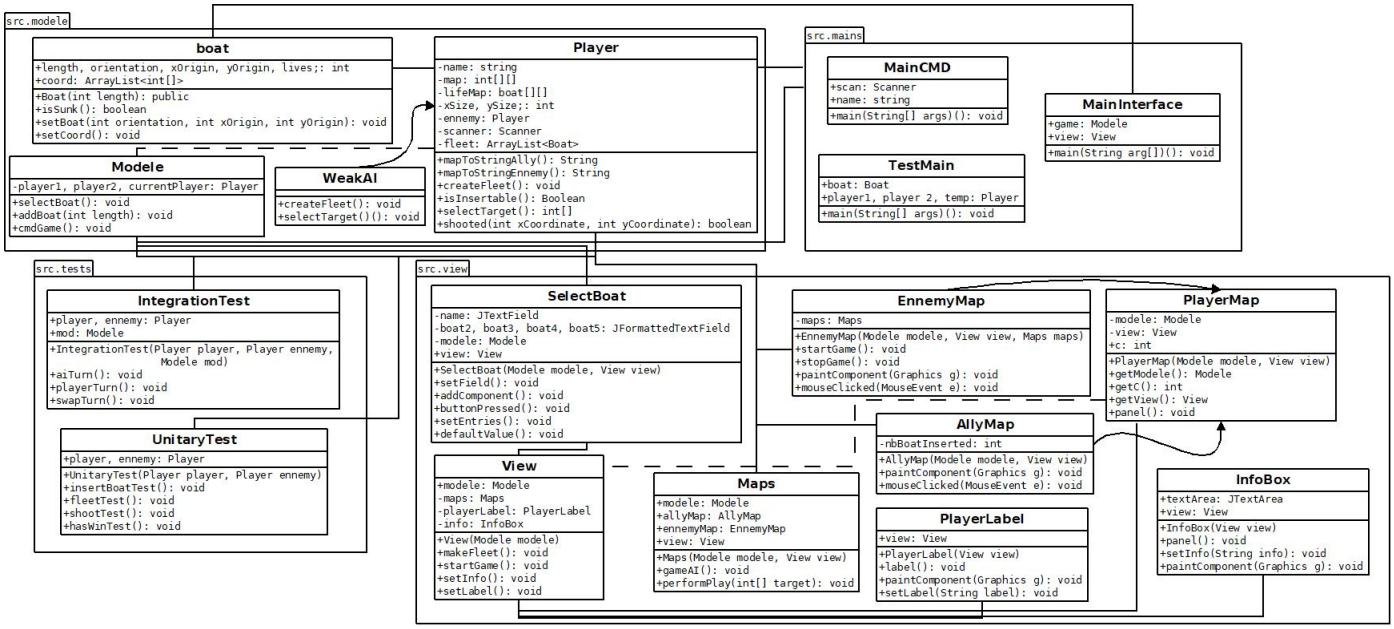


FIGURE 2 – UML

3 Implémentation

3.1 Grille de jeu

Pour réaliser la bataille navale il faut pouvoir afficher l'état du jeu sous forme de grille mais aussi vérifier quand la flotte est coulé. Pour se faire nous avons 2 grilles de jeu, une du type int[][] qui va représenter l'état du jeu :

- 0 qui représente une case vide
- 1 qui représente un bateau
- 2 qui représente un tire raté
- 3 qui représente un tire réussi (bateau touché)

Il s'agit de la grille principale qui va servir a vérifier si il y a déjà un tire effectué a des coordonnées données.

La deuxième grille de type Boat[][] va servir a vérifier si un bateau est touché, chaque bateau étant un objet de classe Boat, l'instance d'un bateau est donc initialisé a ses coordonnées dans la grille. La longueur du bateau représentant ses "vies" et un tire ne pouvant être effectué 2 fois aux mêmes coordonnées, chaque tire enlevera donc une vie, lorsqu'un bateau n'a plus de vie il est coulé et est donc retiré de la flotte (attribut de la classe joueurs de type Arraylist<Boat>). Lorsque la flotte est vide le joueur a perdu.

Les 2 grilles interagissent entre elles comme ceci :

Algorithm 1: GridInteraction

Input: coordinates : int[][]
Output: shootInformation : String

```
1  grilleInt : int[][]
2  grilleBoat : Boat[][]
3  fleet : ArrayList < Boat >
4  if grilleInt[coordinates.x][coordinates.y] < 2 then
5      if grilleInt[coordinates.x][coordinates.y] == 1 then
6          grilleInt[coordinates.x][coordinates.y] = 3
           grilleBoat[coordinates.x][coordinates.y].lives − 1
7          if grilleBoat[coordinates.x][coordinates.y].lives == 0 then
8              fleet.remove(grilleBoat[coordinates.x][coordinates.y])
9              return "boatSunk"
10         else
11             return "boatHit"
12         end
13     else
14         grilleInt[coordinates.x][coordinates.y] = 2
15         return "missedShoot"
16     end
17 else
18     return "InvalidTarget"
19 end
```

3.2 CMD

Notre jeu de la bataille navale, en accord avec le MVC, est jouable dans l'invite de commande. Pour se faire, un scripts qui lance la classe MainCMD ouvre le jeu en ligne de commande. Pour l'affichage une fonction dans la classe Player.java permet de transformer la grille de jeu en chaine de caractère lisible, cette dernière a 2 variantes :

- MapToStringAlly qui gère la grille allié (celle du joueur)
- MapToStringEnnemy qui gère la grille ennemie (celle de l'IA)

2 versions sont nécessaire car effectuer un code qui ressemblerai a ça poserai un problème :

```
System.out.println(player1.MapToString());
System.out.println(player2.MapToString());
```

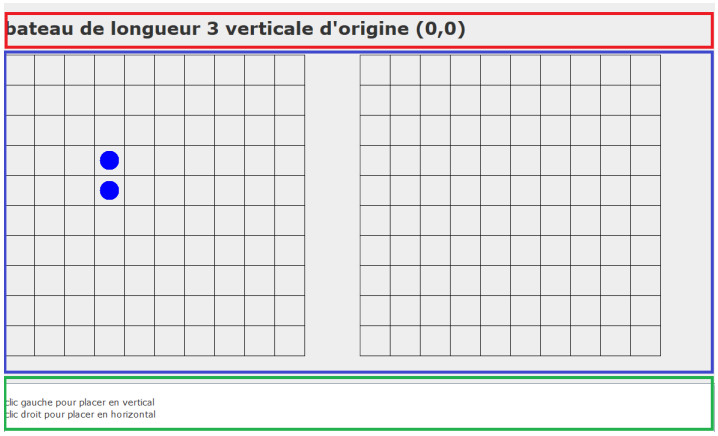
Avec ce genre de code nous aurions le problème de ne pas voir l'emplacement de ses propres bateaux, ou de voir ceux adverse. La fonction MapToStringAlly affiche donc la grille du joueur avec les coup joué de l'IA sur sa grille. La fonction MapToStringEnnemy affiche la grille de l'IA avec les coups joué du joueurs. Tous ceci du points du vue du joueurs : voir ses bateaux mais pas ceux ennemies.

Pour la récupération des entrée du joueurs tel que le placement des ses bateaux ou les coordonnées de sa cible, un "Scanner" est utilisé. Il permet au joueurs d'écrire dans le terminal qui sera ensuite traité pour effectuer l'action adéquate. Jouer dans le terminal requiers des fonction spécifique pour l'affichage dans le terminal mais aussi la saisie et le traitement de celles-ci, c'est pourquoi des fonctions utile uniquement pour jouer dans le terminal sont présentes dans les classes Player.java et Modele.java.

3.3 Interface graphique

Le jeu est aussi jouable avec une interface graphique, le principe du MVC est d'avoir trois modules mais pour l'implémentation graphique le contrôleur (Listener) est directement attaché a l'interface pour récupéré ou le joueur clique sur l'interface. Toutefois il reste facile a désactiver, mais il n'est alors plus possible d'agir sur le modèle et donc le jeu. Notre interface graphique est composé de trois parties principales :

1. Un "JLabel" qui permet l'affichage du type de bateau a placer ou du nom du joueur en train de jouer (en rouge)
2. Un "JPanel" qui contient les grilles des deux joueurs du points de vue du joueur 1 (joueur humain, le joueur 2 étant l'IA) lui même divisé en deux JPanel contenant chacun une grille (en bleu)
3. Un "JTextArea" non-éditable qui affiche des informations utile pour le jeu, tel que le résultat d'un tir (touché ou non), si le bateau touché est coulé ... (en vert)



La division des grilles en deux "JPanel" distinct permet d'avoir deux "Listener" différents, un pour chaque grille. Le premier sur la grille dite allié permet le placement des bateaux, le second sur celle dite ennemie permet la sélection de la case sur laquelle effectuer un tire. De plus la fonction pour l'affichage n'est pas la même car il faut cacher les bateaux sur la map dite ennemie.

Tous comme l'implémentation du CMD, l'implémentation de l'interface graphique nécessite des fonctions spécifique a cette dernière qui se situe dans ses différents composant (JFrame, JPanel ...) qui s'occupe principalement de lancer les fonctions du Modèle en fonctions des actions du joueur sur l'interface. Le restes des fonctions sert a l'initialisation et a l'affichage des composants de l'interface. Vue et Contrôleur sont donc étroitement liés.

4 Conclusion

4.1 Amélioration possible

4.1.1 Fonctionnalitées

Actuellement certaines fonctionnalités totalement optionnel pourrait être ajouté. Une fonction rejouer qui permettrait, comme son nom l'indique, de rejouer a la fin d'une partie sans avoir besoin de relancer l'application. Une fonction de score pourrait aussi être ajouter pour comptabiliser le nombre de victoire.

4.1.2 Intelligence Artificielle

L'intelligence artificielle implémenté n'est pas très efficace. Une amélioration pourrait être de la rendre plus "intelligente", car actuellement elle tire au hasard, en la faisant couler totalement un bateau une fois qu'elle en a trouvé un (comme le ferai un vrai joueur). Une version "ultime" serait de faire une IA qui apprend elle même au fil des parties, et donc il y aurait une difficulté croissante (quoi que plutôt lente) en passant par le deeplearning.

4.2 État du projet

L'objectif du projet était de réaliser une bataille navale dotée d'un interface graphique. Au début de la partie l'utilisateur doit simplement rentrer son nom et le nombre de bateaux voulu. Le code a était fait de façon à modifier la grille facilement, et avoir une implémentation facile à comprendre et à utiliser pour afficher la taille de grille voulu. Pour cela de nombreuses classes sont disponibles pour permettre au jeu de fonctionner. La bataille navale est aussi bien fonctionnel en utilisant seulement le terminal. Lors de la partie on peux voir le score actuel, les cibles touchées ainsi que les cibles coulées.

5 Sources

Source utilisée pour la réalisation de ce rapport :

— Bataille navale - Wikipédia