

PROJET INFORMATIQUE BAC 1: PROJET BATAILLE NAVALE



Année scolaire: 2022/2023

1. Parenthèse	2
2. Introduction	3
3. Description App/ mini-guide utilisateur	3
4. Choix persos/Description algorithmes	5
4.1 Choix personnels	5
4.2 Description Algorithmes	5
5. Bugs Connus	6
6. Points positifs	6
7. Points négatifs	6
8. Apports positifs	7
9. Apports négatifs	7
10. Sources/Aides	7
11. Tests unitaires - explication	8

1. Parenthèse

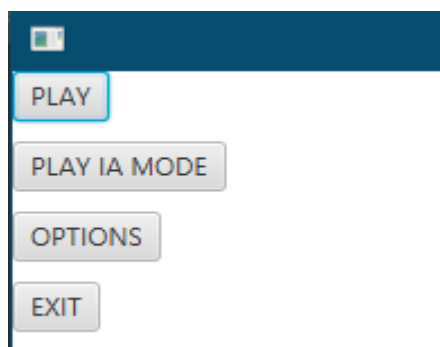
Tout d'abord je tenais à m'excuser auprès de vous et aussi de mes anciens membres de groupe pour le projet de juin que j'ai totalement laissé tombé. Je ne cherche aucune excuse et assume mon choix et ses conséquences mais je tenais tout de même à m'excuser. Bien à vous. Emile.

2. Introduction

Le but de ce projet était de programmer un jeu de bataille navale fonctionnel et interactif à l'aide de nos aptitudes dans le langage Java appris en cours de première année. Le but du jeu est de trouver et de couler l'ensemble de la flotte ennemie en tirant tour à tour sur la grille adverse, à chaque coup, on reçoit 2 informations importantes: la distance avec le bateau le plus proche (en nombres de cases hors diagonale) et la taille de ce dernier. Si plusieurs bateaux sont à la même distance, on reçoit la taille du bateau le plus petit.

Grâce à ces infos, on peut jouer de manière plus ou moins optimisée. Dans cette implémentation de la bataille navale, nous pouvons regarder un IA intelligente jouer à notre place contre une IA basique.

3. Description App/ mini-guide utilisateur



Lors du lancement du programme via gradle, une fenêtre s'ouvre, avec plusieurs boutons.

Le premier mode "play", permet de lancer une partie rapidement contre la fameuse IA aléatoire sur une grille 10x10 avec la flotte de base càd 5

bateaux: 1 de taille 5, 1 de taille 4, 2 de taille 3 et 1 de taille 2. On peut ensuite cliquer sur la grille du haut pour "tirer" sur la grille adverse contenant le même nombre de bateaux de même taille que les nôtres.

Après cela, 3 options: case bleu = tir dans l'eau, case noir = bateau touché, cases vertes = touché coulé. Voilà pour le premier mode.

Le 2e mode, "PLAY IA", c'est ici qu'avec les mêmes paramètres et règles qu'aux dessus, on va pouvoir laisser une IA intelligente jouer à notre place. Bouton "next" -> joue le tour de l'ia intelligente. Bouton "end" -> joue tout les tours de la partie jusqu'aux résultats.

Le 3e mode "OPTIONS", permet avant de lancer une partie de changer la taille de la grille et le nombre de bateaux ainsi que leurs tailles respectives. En bas de cette fenêtre 2 boutons permettant de lancer cette grille personnalisée soit dans le premier mode "joueur vs ia" soit dans le second "ia intelligente vs ia aléatoire".

The screenshot shows a web-based options menu for a game. At the top, there is a dark blue header bar with a small icon on the left and a button labeled "Return to Main Menu" on the right. Below the header, the menu is organized into sections. On the left, under the heading "Taille Grille:", there are two dropdown menus: "Lignes:" set to "10" and "Colonnes:" set to "10". To the right of these, there is a list of six items, each consisting of a label "Nombre de bateaux de taille :X" followed by a dropdown menu. The labels are for sizes 1, 2, 3, 4, 5, and 6. All six dropdown menus currently display the value "0". At the bottom of the menu, there are two buttons: "Start" and "IA START".

Depuis le menu options, on peut revenir au menu principal grâce au bouton prévu à cet effet en haut de page.

Lors de chaque partie, un bouton "cheat" est présent, il permet de tricher et de voir les bateaux adverses. Il est activable et désactivable. Il se désactive aussi après chaque coup du joueur principal (choix personnel).

A la fin de chaque partie, 3 choix s'offrent à nous en plus de l'affichage du score. "Recommencer" relance la partie avec les mêmes paramètres. "rejouer", nous remet dans le menu où l'on peut modifier les paramètres avant de lancer une partie avec ces paramètres. "Exit" quitte le programme.

Le bouton "exit" du menu principal ferme juste le programme aussi.

4. Choix persos/Description algorithmes

4.1 Choix personnels

A la base je n'avais pas bien divisé le projet et n'avais pas de constructeur de bateaux etc, j'ai vite compris que j'en aurais besoin pour récupérer une liste de cellules qui les composent par exemple. Le choix de faire des joueurs m'a vraiment facilité la vie afin de créer des joueurs humain, des joueurs ia et des joueurs ia intelligente en utilisant juste l'hérédité de la classe joueur. Je pense que le reste des classes sont plutôt logiques (Cellules, Grille, Bateaux, Jeu, Direction, Position) Pour la partie javafx, on a ici le choix d'un stage avec plusieurs scènes s'échangeant la place au premier plan.

Dans la classe jeu, mettre une liste de joueurs était plus simple pour y accéder par index et non pas par nom.

Chaque joueur a sa grille et sa liste de bateaux + sa fonction play
exemple: ia, smartia et joueur sont des Joueur mais ont des méthodes play différentes qui permettent chacune de jouer son tour différemment. Exemple la méthode play de ia tire au hasard, celle de joueur attend un click sur une case en graphique ou attend un x,y en console et Smartia joue ses coups en essayant de les optimiser.

L'idée d'utiliser beaucoup de fonctions qui retournent des booléens m'a été très utile car lors de l'implémentation de l'ia intelligente je les utilise beaucoup dans les conditions.

En général, j'ai essayé de toujours bien séparer le javafx de la logique car je savais que cela était très important.

Pour les menus, j'ai opté pour un stage et différentes scènes: Options, Menu principal, Jeu. Cela permet lorsque l'on clique sur un bouton de changer la scène et non de la superposer ou quoi que ce soit.

4.2 Description Algorithmes

Algorithme de l'ia intelligente:

L'algorithme est plutôt simple et pas très développé mais optimise un tout petit peu mieux le résultat.

Premier tir: L'ia tire au milieu de la grille (plus grande probabilité de toucher un bateau et/ou d'avoir des infos utiles).

Autres tours: 2 cas possibles,

- 1) Bateau touché
- 2) Bateau pas touché

Si l'on touche un bateau, l'ia va tirer à une distance soit au-dessus soit à droite. Si elle ne peut pas le faire car elle a déjà tiré ici ou qu'elle devrait tirer en dehors de la grille, alors elle tire autre part au hasard.

Si l'on ne touche pas de bateau, l'ia va tirer à la distance qu'elle a obtenue lors de son premier tir, soit à gauche soit en bas. Si elle ne peut pas le faire car elle a déjà tiré ici ou qu'elle devrait tirer en dehors de la grille, alors elle tire autre part au hasard.

Algorithme du placement de bateaux sur la grille:

La méthode prend en entrée: un bateau, une nombre de ligne , un nombre de colonnes et une direction

- 1) Si le nombre de ligne > nombre de ligne de la grille ou est plus petit que 0 cela renvoie une exception. Même chose pour le nombre de colonnes.
- 2) On regarde si avec la direction du bateau et de sa longueur, il peut être placé à un certain endroit. On fait ceci en comparant la taille du bateau et la taille de la grille en fonction de l'orientation du bateau. Si il dépasse du plateau, on renvoie une exception
- 3) On regarde si le bateau ne chevauche pas un autre bateau en récupérant pour chaque cellule du bateau que l'on veut placer, la valeur boat de la cellule qui nous indique s' il y a un bateau sur la cellule en question.
- 4) On regarde pour chaque cellule du bateau que l'on veut placer si il n'y a pas un bateau dans un rayon de 1, pour cela on créer une liste de cellules correspondant aux 8 cellules possibles autour. On trie évidemment avant de les ajouter les cellules qui sont hors du plateau etc. Une fois ceci fait, on regarde juste dans les 8 cellules si l'attribut boat de ses cellules est vrai ou faux, si il est vrai, un bateau se trouve dans le rayon on renvoie donc une exception.
- 5) Après toutes ces étapes, on peut donc enfin poser notre bateau sur la grille en utilisant sur chaque cellules du bateau la méthode setBoat de cellule.

5. Bugs Connus

-Lorsque l'on essaye de créer une game avec trop de bateaux pour la grille: boucle infinie donc freeze.

- Même chose lorsque l'on essaye de créer une game et que des petits bateaux se mettent en pleins milieux -> plus de place pour le reste des bateaux -> boucle infinie donc freeze.

-Lorsqu'une partie smartla vs la est terminée, si l'on ne clique pas sur 1 des 3 boutons pour rejouer, recommencer ou quitter, et que l'on clique cheatmode après avoir complété la grille -> freeze.

-on peut encore jouer lorsqu'une partie est finie tant que l'on ne clique pas sur un des 3 boutons

-lorsque l'on joue en ia intelligente vs ia de base, la distance de Man et la longueur du bateau le plus proche sont décalées dans le temps. En effet, lors du premier tir la distance est correcte puis lors du deuxième elle nous renvoie celle du premier tir et cela ainsi de suite, Le 3e tir renvoie celui du second... MAIS cela est juste ce que NOUS voyons. De fait, l'ia elle récupère les bonnes valeurs pour faire ses calculs.

6. Points positifs

-Code plutôt clair (pour un premier projet je pense).

-Plutôt rapide entre chaque tour.

-Bonne séparation de javafx et de la logique.

- On peut jouer dans la console avec la grille txt
- L'application en elle même n'est pas trop buggy, tout fonctionne plutôt bien.

7. Points négatifs

- Certains algo pas opti (distance man qui parcourt toutes les cases et pas juste les cases des bateaux donc si grille bcp plus grand ex: $1 \cdot 10^9$ cela prendrait bcp de temps de check ttes les cases de la grille)
- Code pas terrible pour le retour du bateau le plus petit si 2 a la même distance (cf: attribut de classe récupéré avec un guetter).
- 1 point demandé pas implémenté (Mode statistiques). Je pourrais néanmoins vous expliquer comment je l'aurais implémenté.
- Plusieurs erreurs récapitulées au-dessus dans bugs connus.
- Pas de personnalisation du jeu, ni de fonctionnalités en plus si ce n'est de pouvoir vraiment jouer une partie équilibrée en chargeant un fichier texte.(mm nombre bateaux, mm longueur bateaux) uniquement en console par contre.
- Tests unitaires pas disponibles (voir fin du rapport).
- L'ia intelligente n'est pas très bien développée, j'ai beaucoup hésiter à demander à chatgpt une logique qui exploitait les 2 données reçues à chaque tir mais j'ai préféré assumer le fait que je n'arrivais tout simplement pas moi même à trouver une logique et une implémentation convenable.

8. Apports positifs

Je me suis rendu compte que j'aimais vraiment bien l'univers du code. Si j'avais validé le cours en juin avec le groupe sans avoir rien fait je n'aurais rien appris. J'ai beaucoup dû réfléchir en terme de logique (bon pour le cerveau^^)

De plus, je me suis vraiment donné à fond jusqu'à la fin afin d'envoyer un projet répondant au plus de critères possibles.

9. Apports négatifs

Pas vraiment d'apports négatifs si ce n'est le temps passé devant un écran en plein été..

Ainsi que de pas mal de frustration ...

10. Sources/Aides

-ChatGPT pour l'une et l'autre chose. (En commentaires dans le code).

-Certains tuteurs m'ont aidé pour implémenter ou améliorer certains algorithmes dont j'avais la logique.(Dont l'algo distance Man).

-Rémy Plazza, qui m'a beaucoup aidé dans la résolution d'erreurs + conseils.

-Forums comme StackOverFlow. <https://stackoverflow.com/>

-Vidéos youtube. Dont la chaîne de ce monsieur:
(<https://www.youtube.com/@AlmasB0>) qui avait fait une vidéo d'implementation en java du jeu de la bataille navale qui m'a inspiré pour la taille des cases de la grille et un peu de code en général au début, puis tout rechangé pour une meilleur division de javafx et de la logique car lui mélangeait les deux.

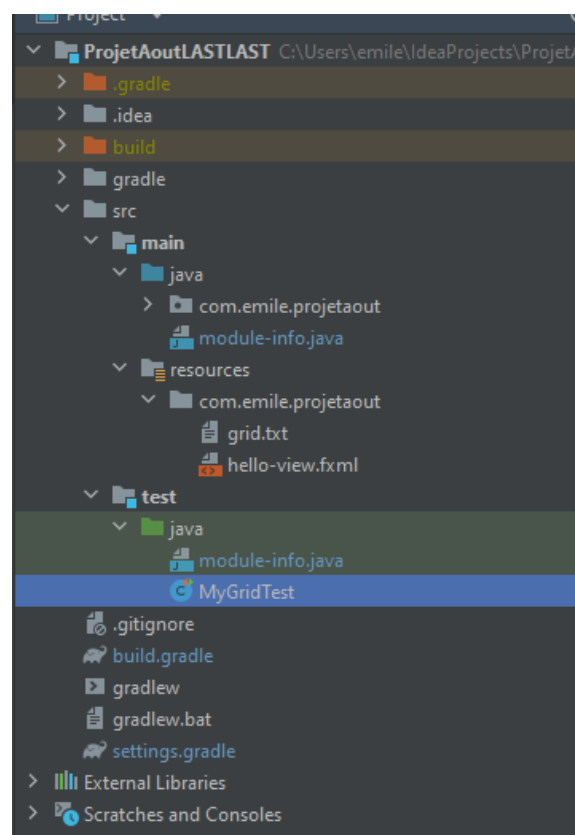
-Google images pour le rapport.

11. Tests unitaires - explications

J'ai essayé plusieurs heures durant, de faire fonctionner Junit avec gradle mais en vain. J'ai même demandé de l'aide aux élèves des années supérieures mais non plus, rien n'y fait. Cependant bien qu'il ne soient pas exécutables, je souhaiterais vous montrer comment je m'y serais pris en donnant en exemple le premier test unitaire demandé qui était de charger un fichier contenant une grille de combat naval prédéfinie.

Ceci aurait été mon **arbre de classes**

Ci-dessous, vous pouvez retrouver la définition de la classe de test qui je pense est correcte mais je n'ai jamais pu essayer donc difficile de savoir si je suis dans le bon ou non



Code de la classe de test:

```
public class MyGridTest {
    @Test
    public void loadGridTest() {
        Game game = new Game();
        try {
            game.loadGame( filename: "src\\main\\resources\\com\\emile\\projetaout\\grid.txt");
            Grid grid = new Grid( columns: 10, rows: 10);
            Boat boat = new Boat( size: 2);
            Boat boat3 = new Boat( size: 3);
            grid.placeBoat(boat, row: 2, col: 0, Direction.VERTICAL);
            grid.placeBoat(boat3, row: 0, col: 2, Direction.HORIZONTAL);
            assertEquals(game.getPlayersList().get(0).getGrid(), grid);
        }
        catch (Exception exception){

        }
    }
}
```

Pour conclure:

Le projet est correct mais n'est pas formidable, manque de personnalisation etc. J'aurais aimé y passer plus de temps.

J'espère que mon projet vous plaira tout de même.

Bien à vous et merci de m'avoir lu.