

Jeu de la Bataille Navale

LAVANDIER Augustin, FAUQUETTE Mael
CAUSEUR Matthieu, TRANSON Alexandre

L2 Informatique Groupe 4B

13 avril 2023



**UNIVERSITÉ
CAEN
NORMANDIE**

Table des matières

1	Introduction au projet	1
1.1	Description générale de la consigne du projet	1
1.2	Description de la problématique du projet	1
1.3	Présentation du plan du rapport	1
2	Objectifs du projet	2
2.1	Présentation des objectifs à atteindre	2
3	Fonctionnalités apportées	2
3.1	Explications des fonctionnalités	2
3.2	Organisation du projet	2
4	Eléments techniques	3
4.1	Description des diverses bibliothèques utilisées	3
4.2	Description des diverses structures de données	3
4.3	Description des principaux algorithmes	4
5	Architecture du projet	5
5.1	Diagramme des divers modules et classes	5
5.2	Explication du traitement des éléments par l’algorithme principal du projet . . .	5
6	Expérimentation et usages	6
6.1	Cas d’utilisation	6
6.1.1	Documentation	6
6.1.2	Lancement du projet	6
6.2	Résultats quantifiables	7
7	Conclusion	7
7.1	Récapitulatif des fonctionnalités principales	7
7.2	Propositions d’améliorations	7
7.2.1	Certains Algorithmes	8
7.2.2	Une interface plus fonctionnelle	8
7.2.3	Un plus grand nombre de fonctionnalités	8

1 Introduction au projet

1.1 Description générale de la consigne du projet

Le thème du projet est le jeu de la bataille navale. C'est un classique des jeux de société familiaux. Le but du jeu est simple, il s'agit de faire couler les navires de son adversaire en devinant leur position sur une grille de coordonnées de taille 10 par 10. Bien que paraissant simple au premier abord, ce jeu offre une grande variété de stratégies et peut être un excellent exercice pour développer la réflexion tactique. Ce jeu peut être joué que par deux joueurs simultanément.

C'est avec ce thème principal que débute notre projet avec pour consignes de développer une version informatisée du jeu de la bataille navale, qui permet à un joueur d'affronter sur son ordinateur une intelligence artificielle qui tentera de manières aléatoires de trouver la position des bateaux du joueur.

1.2 Description de la problématique du projet

Afin de réaliser de la meilleure manière ce projet nous avons dû identifier une problématique principale à laquelle nous nous devons de répondre de façons optimales.

Après un premier temps de recherche nous avons décidé de tenter de répondre à la problématique suivante :

Comment créer un programme du jeu de la bataille navale optimisé et plaisant à jouer ? C'est à partir de cette dernière que nous allons construire le fil conducteur de ce rapport.

1.3 Présentation du plan du rapport

Ce rapport va se construire plusieurs grandes parties...

Dans un premier temps nous verrons et présenterons les différents objectifs qui seront fixés et qui devront être atteints. Par la même occasion nous commencerons à explorer des pistes à suivre afin d'atteindre ces objectifs.

Ensuite nous parlerons des principales fonctionnalités implémentées dans le projet afin d'avoir les premiers éléments de réponses. Nous dévoilerons également comment notre groupe s'est réparti les différentes tâches et comment chacune des parties du groupes se sont organisées.

Suite à cela nous dévoilerons les aspects techniques de notre projet et expliquerons notre manière de programmer ce dernier par le biais des divers algorithmes mais également en présentant les diverses bibliothèques auxquelles nous avons dû avoir recours afin de permettre la bonne exécution de nos programmes. Puis nous détaillerons les structures choisies pour l'exploitation des nombreuses données du jeu de la bataille navale.

Dans la foulée nous présenterons l'architecture de nos programmes par l'intermédiaire d'un diagramme qui expliquera et définira les différents modules et classes. De plus nous verrons la manière dont sont traitées les divers éléments composants notre projet par l'algorithme principal de notre jeu.

Postérieurement nous expliquerons l'utilisation à avoir de notre projet autrement dit la manière dont le jeu s'exécute, mais aussi une explication dans le but de générer la documentation des diverses fonctions utilisées dans les programmes des classes et packages. Nous apporterons à ceci des données importantes sur notre jeu comme par exemple le temps d'exécution de notre projet.

Nous finirons ce rapport en apportant une conclusion globale sur notre jeu de la bataille navale tout en apportant des axes d'améliorations qui seraient à suivre.

2 Objectifs du projet

2.1 Présentation des objectifs à atteindre

Afin de mener à bien ce projet nous nous sommes fixer un certains nombres d'objectifs que sont :

- Concevoir une interface utilisateur intuitive et agréable pour permettre aux joueurs de naviguer facilement dans le jeu.
- Programmer un bot capable de jouer au jeu de la bataille navale tout en respectant les règles.
- Offrir un environnement de jeu agréable et esthétiquement plaisant aux utilisateurs.
- Documenter par le biais de contrat le processus de développement pour faciliter les futurs ajustements et améliorations du jeu.

C'est à partir de ces éléments que nous allons guider notre projet afin de le réaliser de la manière la plus optimale.

3 Fonctionnalités apportées

3.1 Explications des fonctionnalités

Notre jeu de la bataille navale offre plusieurs fonctionnalités avancées pour offrir une expérience de jeu optimal. Voici les principales fonctionnalités de notre application :

Génération de grilles : Concevoir une grille de jeu afin de permettre le positionnement des bateaux du joueur mais aussi ceux de l'ordinateur. Cette grille sera la base du jeu c'est elle qui sera le "modèle" de notre projet notion sur laquelle nous reviendrons un peu plus tard dans ce rapport.

Gestion des coups : Faire en sorte que le jeu soit capable de gérer et de prendre en compte les coups du joueur et de l'ordinateur autrement dit être capable d'identifier si le coup joué à des coordonnées données tombe sur une case où un bateau est présent ou si elle est vide ou bien encore si un coup a déjà été joué.

Interface graphique conviviale : Cette interface doit être claire et pratique pour que l'utilisateur ait du plaisir à jouer et puisse comprendre du premier coup ce que le programme attend de lui afin de pouvoir poursuivre le jeu.

Un bot capable de jouer : Nous avons décidé d'implémenter un bot piloté par le biais d'algorithmes jouant au jeu de manières aléatoires afin de proposer à l'utilisateur de jouer à la bataille navale seul. Ce bot est dépourvu d'intelligence artificielle et se contente de placer ses coups sur cases aléatoires mais qui ne sont pas déjà jouées. /item[Un double affichage] : Nous avons implémenter cette fonctionnalité, un premier affichage de la bataille navale dans le terminal et un second dans une interface graphique cliquable. Le premier de ses affichages nécessite de rentrer les coordonnées à la main alors que le second permet de simplement cliquer aux coordonnées voulues. De ce fait, il est aussi possible de lancer les deux affichages en simultanés, mais seulement le cliques ont détectables par l'interface et non plus les entrées dans le terminal.

3.2 Organisation du projet

La liste des principales fonctionnalités implémentées dans notre projet faite nous allons pouvoir expliquer dans cette partie l'organisation que nous avons pu adopter pour réaliser ce

projet. Nous avons procédé de manière à ce que notre groupe soit divisé en deux parties. La première partie composée de Mael et de Matthieu était chargée de créer le jeu par le biais de la grille mais aussi toutes les méthodes qui vont avec afin de permettre de vérifier bon nombre de paramètres. Le second groupe composé d'Augustin et d'Alexandre avait la tâche de créer les diverses classes qui seront utilisées dans le jeu (par exemple boat, SeaObject...) mais aussi de s'occuper de l'interface qui permettra l'affichage de la grille qui sera le point principal du jeu. Une fois ces deux tâches accomplies ce groupe s'est regroupé avec l'autre afin d'aider dans la conception du jeu qui s'est montrée plus complexe qu'attendue.

4 Éléments techniques

4.1 Description des diverses librairies utilisées

Afin de réaliser ce projet nous avons eu recours à un certain nombre de librairies. Les librairies utilisées sont majoritairement celles dans les classes d'affichages et d'interfaces. Nous avons utilisé les librairies swing et event pour l'interface du jeu.

Swing et awt : Swing et awt sont des librairies graphiques en Java qui permettent de créer des interfaces utilisateur (IU). Elles offrent un large panel de composants graphiques, tels que des boutons, des cases à cocher, des champs de texte, des tableaux, des panneaux et des fenêtres. Ces composants personnalisables peuvent être modifiés pour répondre aux besoins spécifiques de l'application. Swing et Awt utilisent le plus souvent un modèle MVC (Model-View-Controller) pour la conception de l'IU dont nous reparlerons plus tard.

Event : La librairie "event" en Java est un ensemble de classes et d'interfaces qui sont utilisées pour implémenter le modèle de conception "Observateur/Observé" (Observer/Observable) dans les programmes Java. Ce concept vise à informer certains objets du changement d'états d'autres objets. Autrement dit dans notre projet l'affichage doit être au courant de l'état de chaque case de la grille. Par exemple si un coup est joué sur une case vide alors l'affichage doit être au courant que cette case est vide et qu'un coup a été joué afin de mettre un point rouge sur cette case et montrer à l'utilisateur qu'elle ne peut plus être jouée. Ce concept est utilisé dans le but de permettre une meilleure gestion des événements dans le programme.

4.2 Description des diverses structures de données

Nous allons exposer dans cette partie la manière dont sont traitées les diverses données nécessaires au bon fonctionnement du jeu.

grille : La class contenant la grille de chaque joueur de notre bataille navale est la class BattleShips. Cette dernière contient notre grille de bateaux sous la forme de grille[[[]], c'est à dire une liste à deux dimensions. Et ce pour un accès beaucoup plus rapide et efficace aux données.

SeaObject : Les SeaObject sont les objets contenus dans la grille, ils peuvent être soit de simples SeaObject, soit des objets qui en découlent, soit des Boat. Ces SeaObject ont un attribut state permettant d'avoir constamment leur état dans la grille et permettre les coups des joueurs ou nan, et surtout l'affichage. C'est à dire qu'un SeaObject peut être touché, flottant, (et les Boat peuvent être sunked).

Boat : Pour reprendre rapidement sur les Boat de notre grille, ces derniers possèdent deux attributs intéressants qui sont deux listes contenant des entiers. Ces entiers représentent

les coordonnées x et y de début et de fin du bateau. Sachant que la position de début se place toujours forcément soit en haut soit à gauche de la fin du bateau.

Player : Les class implémentants l'interface Player sont les joueurs de nos parties, ils servent en général de contrôler à notre modèle et permettent de trier les entrées des utilisateurs avant de les envoyer au modèle. Les trois class implémentants cette interface sont : Bot (le robot aléatoire), Human (qui permet l'interaction depuis le terminal), et HumanGUI (qui permet l'interaction sur l'interface 2D).

4.3 Description des principaux algorithmes

Nous allons voir ici les principaux algorithmes sur lesquels reposent notre projet. Nous expliquerons comment ils fonctionnent et nous expliquerons leur utilité et surtout pourquoi nous avons décidé de les programmer ainsi.

Gestion de la grille : Chacune des fonctions de gestion de la grille se déroulent de la même manière. Nous prenons les coordonnées des objets à modifier, les coordonnées sont toujours exactes car elles sont triées au préalable dans les controllers. Ensuite nous modifions les objets dans le tableau selon l'action (placement d'un bateau via une boucle, mettre un objet à "touché"). Et enfin nous lançons un fireChange() permettant d'envoyer un signal aux vues pour qu'elles se mettent à jour car la grille a été modifiée.

Gestion des coordonnées Human : Comme dit précédemment, les coordonnées valables ou non sont examinées dans les controllers, ici nous parlerons du Human (et HumanGUI). Les coordonnées sont donc triées de la manière suivante : nous nous plaçons dans une boucle infinie, la seule sortie est donc possible si nos coordonnées sont valides. Dans cette boucle infinie, nous transformons d'abord les coordonnées sous forme d'entier, elles sont triées pour mettre la première en haut à gauche de la seconde. Ensuite nous vérifions via plusieurs critères si la position du bateau est envisageable, comme par exemple, un bateau qui ait une taille non présente dans la liste des tailles restantes, un bateau posé en diagonales, etc... Si c'est le cas nous créons la liste de tous les bateaux possibles et nous regardons si les coordonnées collent effectivement avec celles d'un bateau possible. Et si c'est le cas, nous renvoyons les coordonnées du bateau. Pour ce qui est de l'attaque, le système est le même, nous vérifions si le coup est contenu dans la liste des coups possibles et nous renvoyons le coup si c'est le cas. De ce fait nous n'avons pas de possibilités de coordonnées impossibles ou de parcours de listes trop grands car nous connaissons déjà les valeurs possibles.

Gestion des coordonnées Bot : Le plus gros problème pour le fonctionnement du bot, c'était qu'il bloque en choisissant des coordonnées déjà utilisées à l'infini. C'est pourquoi ce que nous lui faisons choisir aléatoirement, ce n'est pas les coordonnées des cases ou poser son bateau ou attaquer, mais bel est bien la position dans la liste des choix possibles. Cela signifie que notre algorithme possède une complexité un peu plus élevée que les autres, dûe au parcours de la grille pour connaître tous les coups possibles, mais de ce fait notre programme ne peut pas bloquer indéfiniment sur un coup impossible.

Mise à jour de la vue terminal : Lorsque une bataille Navale est modifiée, cela envoie un signal à la class ViewTerminal. Une fois ce signal reçu elle affiche plusieurs éléments dans le terminal. Etant donné que c'est une vue, elle s'occupe uniquement de prendre les informations du modèle et les afficher, elle n'a pas de lien avec le controller. Donc selon l'étape de la bataille navale (placement des bateaux du premier joueur, du second, ou même coup d'un des deux joueurs, ou victoire), elle affichera ce qui est nécessaire. L'ensemble des affichages n'est pas effectué si un des joueurs est un robot, quel intérêt d'afficher les placements du robot ?

Mise à jour de la vue interface2D : Dans les grandes lignes, notre interface 2D (GameGUI) possède deux Grille qui sont deux pannels affichés dans la Frame. A chaque modification de la bataille navale, le signal est envoyé et le GameGUI repaint() (met à jours) les deux grilles. Cet affichage des grilles se passe en plusieurs étapes. Il parcourt la grille et affiche les coups ratés et touchés, affichent les bateaux coulés, et enfin il affiche les lignes de la grille et en même temps les lettres et nombres sur les côtés.

Boucle du jeu : la boucle du jeu se situe dans la class Launcher et se lance à l'appel de la méthode play. Cette fonction se divise en plusieurs parties : la première, nous plaçons les bateaux d'un premier joueur. Nous obtenons à chaque passage de boucle la liste des tailles de bateau restants à placer. Si ce nombre atteint 0, nous replaçons les bateaux pour le second joueur (nous changeons le joueur courant grâce à la méthode next()). Une fois cela finit, nous rechangeons le joueur courant et débutons les attaques des joueurs. Nous prenons simplement le coup d'un joueur tant que la grille adverse n'est pas pleine, nous l'exécutons sur la grille adverse et nous changeons le joueur courant. Une fois une des grilles finie, nous affichons le gagnant dans le terminal et la partie se termine.

5 Architecture du projet

5.1 Diagramme des divers modules et classes

Nous allons d'abord parlé de l'arborescence de nos packages. Nous avons organisés notre projet avec l'arborescence suivante :



Nous avons décidé de diviser nos packages ainsi afin de faciliter la compréhension de tous. En effet nous pouvons comprendre directement quel package englobe quelle partie de notre projet. Car pour rappel notre jeu utilise le concept de MVC... Ainsi avec le package "controller" nous savons que les classes contenues remplissent la fonction de contrôle du modèle. Le package "model" quant à lui est le plus important il prend en compte tous les éléments indispensables au bon fonctionnement du jeu. Et le package "View" est celui qui concerne l'affichage du jeu par le biais de l'interface. Le fichier "Main" est celui qui permet le lancement de tout le jeu complet il regroupe et gère l'entièreté des autres packages dans le but d'assurer un déroulement correct de la partie du jeu de la bataille navale.

Maintenant que nous avons vu l'arborescence de notre projet nous allons nous attarder sur ce que contient chacun des packages par le biais de ce diagramme : insérer un diagramme qui présente tout ça

5.2 Explication du traitement des éléments par l'algorithme principal du projet

Nous avons basé notre projet sur le modèle de programmation dit MVC (Model-View-Controller). Ce concept est un modèle de conception largement utilisé dans le développement de logiciels. Il permet de séparer les responsabilités et les tâches d'une application en trois types de modules :

Le modèle : qui représente les données de l'application. Il s'agit de la partie de l'application qui gère les interactions avec les données, effectue des calculs et assure le bon fonctionnement de l'application.

La vue : qui représente l'interface utilisateur de l'application. Elle affiche les données du modèle à l'utilisateur et lui permet d'interagir avec l'application. La vue est responsable de la présentation des données à l'utilisateur.

le contrôleur : qui est responsable de la gestion des interactions entre le modèle et la vue. Il reçoit les actions de l'utilisateur à partir de la vue et utilise le modèle pour répondre à ces actions. Il met également à jour la vue en conséquence.

C'est sur ce modèle de programmation que nous nous sommes basés. Nous pouvons expliquer point par point les différents éléments de notre projet.

Le modèle : correspond à la grille du jeu contenant les divers objets du jeu. Cela peut être un bateau, un coup sur une case vide ou bien un coup sur un bateau. Si aucun de ces objets ne sont dans ces cases alors elle est considérée comme vide.

La vue : est l'interface que l'utilisateur voit et avec laquelle il est en interaction. Il peut cliquer sur la grille afin de lancer un coup celui-ci devra être sur une case vide. Ainsi l'interface lui retournera si son coup a touché ou s'il était à côté (vert si cela a touché, rouge sinon).

le contrôleur : est la partie qui fait le lien entre la grille de jeu qui est le modèle et l'interface avec laquelle le joueur interagit les contrôleurs vont se contenter de prendre en compte les coups envoyés par l'utilisateur les transmettre à la grille.

Ainsi dans notre contexte pour le jeu de la bataille navale, le modèle MVC offre une clarté de l'architecture ce qui nous a simplifié les différentes tâches en tant que développeur de la même manière la séparation des responsabilités offerte par ce modèle a été particulièrement utile pour gérer les interactions entre l'utilisateur, l'application et puis la vue.

6 Expérimentation et usages

6.1 Cas d'utilisation

Voici quelques petits conseils pour faciliter l'utilisation et la compréhension de notre projet.

6.1.1 Documentation

Afin de permettre une meilleure analyse de notre projet nous avons écrit un certain nombre de contrats pour chaque fonction utile au jeu. Tout cela dans le but de permettre de générer une documentation complète permettant de savoir l'utilité de chaque fonction, ce qu'elle prend en argument et ce qu'elle renvoie. Pour générer cette documentation, il suffit de se placer dans le dossier "lavandier-fauquette-causeur-trancon" puis de rentrer la commande "ant javadoc" dans le terminal. Cela générera la documentation automatiquement à partir des contrats précédemment évoqués et sera consultable directement sur un navigateur web.

6.1.2 Lancement du projet

Pour lancer le jeu il faut dans un premier temps le compiler dans un dossier. Afin de faire ceci vous pouvez utiliser la commande suivante dans un terminale à la racine du projet :

```
javac -d dist src/controller/*.java src/controller/player/* src/model/*.java src/-  
model/seaObjects/* src/view/* src/Menu.java
```


Une fois compilé le programme du jeu de la bataille navale est prêt à être lancé pour cela il vous faut rentrer cette commande toujours à la racine du projet :

java -cp bin Menu

6.2 Résultats quantifiables

Nous pouvons prendre comme test celui du temps d'exécution du projet avant de faire débiter la partie. Autrement dit la création de la grille ainsi que la mise en place de l'ordinateurs ainsi que des bateaux. Afin de mesure le temps nous avons mis en place deux variables de mesure de temps par le biais de la méthode "System.currentTimeMillis()"

Lors de l'exécution du programme nous avons cette ligne de code :

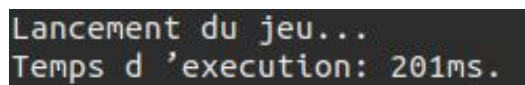
```
1 private long startTime = System.currentTimeMillis();
```

Cette ligne permet de mesurer le temps au moment où est exécuté le programme.

Puis une fois que la partie est prête nous avons cette ligne de code :

```
1 long endTime = System.currentTimeMillis();
2 long totalTime = endTime - startTime;
3 System.out.println("Temps d'execution: " + totalTime
4 + "ms.");
```

Ici nous avons une variable qui mesure le temps à la fin de la configuration de la partie cette valeur est ensuite soustraite à celle qui est déclarée au lancement du programme afin d'avoir le temps pris par l'ordinateur pour générer la partie avec un bot puis l'interface graphique. Une fois exécutée nous avons obtenus ce résultat :



```
Lancement du jeu...
Temps d'execution: 201ms.
```

Ce qui est un temps assez court et satisfaisant afin de garantir les meilleures performances possibles au chargement de la partie.

7 Conclusion

7.1 Récapitulatif des fonctionnalités principales

Dans notre rendu final nous avons réussi à implémenter les principales fonctionnalités souhaitées que sont :

- La grille avec les différents états de chaque case.
- Le bot contrôlé par l'ordinateur dans le but de créer un adversaire contre qui jouer.
- La possibilité de faire du Joueur contre Joueur, Bot contre Joueur ou Bot contre Bot
- Une interface propre et agréable à parcourir.
- L'utilisation du modèle MVC par l'entièreté du projet

7.2 Propositions d'améliorations

Nous avons tout de même certains points sur lesquels nous pourrions améliorer notre projet.

7.2.1 Certains Algorithmes

En effet si la totalité des algorithmes sont fonctionnelles et utiles à notre projet nous aurions un axe sur lequel nous pourrions améliorer nos programme notamment sur certains algorithmes qui présentent une forte complexité que ce soit en espace comme en temps. Mais aussi sur la lisibilité du code de certains autres programmes.

7.2.2 Une interface plus fonctionnelle

Nous avons certes une interface convenable mais qui pourrait être améliorée grâce à l'apport de nouvelles fonctionnalités ou bien par un esthétisme un peu plus travaillé. C'est à dire afficher plus clairement le joueur courant, le nombre de bateaux à placer et leur taille, ou même le nom du joueur en cas de victoire.

7.2.3 Un plus grand nombre de fonctionnalités

Nous pourrions à l'avenir implémenter un plus grand nombres de fonctionnalités à l'image d'un bot proposant plusieurs niveaux de difficulté plutôt que d'utiliser ses coups aléatoirement. Ou bien encore l'implémentation d'un mode de jeu joueur contre joueur visuellement acceptable avec l'interface 2D. Mais nous aurions aussi aimé pouvoir faire fonctionner en même temps le choix des coordonnées des bateaux via le terminal mais aussi via l'interface 2D. Car pour l'instant nous pouvons afficher les deux, mais seulement le choix des coordonnées par l'interface est possible.