UMONS
Université de Mons

BATTLESHIP

Projet d'Informatique

BAC II - MATHÉMATIQUES

Faculté
des Sciences

SWART LAUREN

2018-2019

TABLE DES MATIÈRES

INTRODUCTION

Ce projet informatique porte sur la réalisation d'une application graphique permettant de jouer à la bataille navale.

Dans la suite de ce document seront détaillés les aspects importants de la construction de cette application. Tout d'abord quelques choix importants effectués au niveau de la structure du projet seront listés. Ensuite les complications seront abordées en deux étapes : les problèmes rencontrés et résolus, et les erreurs présentes dans le projet. Par la suite sera entamé le thème de l'intelligence artificielle : son principe de fonctionnement et une description des algorithmes utilisés. Enfin apparaitront les points forts et faibles de ce projet, pour finalement conclure avec les aspects négatifs et positifs de ce cours.

STRUCTURE DU PROJET

Quelques choix ont été effectués au cours de la réalisation du projet. Ci-dessous en seront listés les plus importants.

- *Utilisations des singletons*

La notion de classe singleton a été employée pour 3 classes : Controller, MyScene et Game. Pour ce projet une seule instance de chacune de ces classes était nécessaire, et cette pratique a facilité la manipulation de ces objets, et de ceux qui y sont stockés.

- *Positions des bateaux*

Une première idée était de garder la trace du positionnement des bateaux via les bateaux eux-mêmes. Chaque bateau possèderait une ArrayList contenant les positions (x, y) qu'il occupe. Néanmoins, cette approche parue peu pratique pour la vérification d'un « hit or miss ». En effet, il aurait fallu parcourir la liste des bateaux composants la flotte, et pour chacun de ceux-ci, parcourir une liste de positions (x, y).

Il a alors été décidé de définir un numéro d'identification unique pour chaque bateau de la flotte, et de stocker ces numéros dans une matrice à deux dimensions représentant la flotte sur le plateau de jeu. Si une case (x, y) du plateau est inoccupée, alors la valeur NULL est stockée dans la matrice à l'indice [y][x].

Néanmoins, lorsqu'est arrivé le moment d'implémenter l'importation de carte, il s'est avéré utile que chaque bateau connaisse sa position via l'ArrayList décrite ci-dessus.

C'est pourquoi finalement ces deux méthodes sont utilisées en parallèle, pour garder une trace du positionnement de la flotte.

- *Absence d'interface*

Voulant utiliser les principes orientés objets vus en cours, l'utilisation d'interfaces et classes abstraites a été envisagée. Néanmoins, elle ne parut pas particulièrement utile dans ce projet. En effet, chaque classe a un but et comportement spécifique, totalement différent des autres classes.

COMPLICATIONS

I. Problèmes rencontrés

- *Chargement d'une carte*

Il était impossible de lire le fichier une seule fois, stocker les données lues et créer les bateaux au fur-et-à-mesure de la lecture. Il a d'abord fallu lire le fichier en entier pour avoir les dimensions du tableau à initialiser. Il était donc intéressant d'en profiter pour vérifier que toutes les lignes du fichier étaient de même longueur.

Ensuite, il a été décidé qu'une carte pouvait avoir une hauteur et largeur maximale de 26. Le cas échéant, une exception est lancée. Les caractères sont alors transférés dans une matrice intermédiaire pour faciliter la manipulation des données. Si un caractère autre que « X » ou « . » est rencontré, une exception est lancée. Ce n'est que plus tard, lorsque l'initialiseur de Game tente de créer un jeu à partir de cette matrice intermédiaire, que la vérification sur le positionnement des bateaux se fait. Si deux bateaux se touchent, une exception est lancée.

- *Affichage du plateau*

Au départ, l'affichage du plateau de jeu ainsi que des bateaux se faisait via 3 couches superposées. En fond se trouvait la bordure du plateau, contenant le nom les indices de lignes et colonnes. Ensuite se plaçaient les cases du plateau, changeant de couleur au fur-et-à-mesure du jeu. Par-dessus se trouvait du texte affichant le type de chaque bateau au bon emplacement. Alors que cette approche parait peu adaptée, il s'est avéré difficile de procéder autrement. Inspiré de l'approche de Almas Baimagambetov [1], le plateau est finalement composé de Tiles, qui sont en réalité des StackPanes contenant un fond, un texte pour représenter le bateau, et quelques variables facilitant le jeu.

- *Positionnement aléatoire des bateaux*

Lorsqu'il a fallu écrire une méthode pour positionner les bateaux aléatoirement, la tache ne paraissait pas particulièrement difficile. Il fallait d'abord déterminer si le bateau serait positionné de façon horizontale ou verticale. Ensuite, essayer de placer le bateau quelque part dans la grille. Ceci se faisait pour chaque bateau, du plus petit au plus grand. Pendant tout le développement du jeu, le positionnement aléatoire des bateaux se faisait ainsi, sans poser de problème évident. En revanche lorsque le mode statistique a été créé, un problème

est apparu. Comme un grand nombre de parties étaient jouées, le risque que ce système de placement pose un problème a augmenté. Il arrivait que certains bateaux soient placés de façon à ce que les autres ne puissent être placés par la suite.

L'algorithme de positionnement aléatoire a donc dû être repensé. Premièrement, il est plu raisonnable de placer les bateaux les plus longs en premier. Ensuite, l'algorithme choisit aléatoirement une direction : verticale ou horizontale. Il essaie ensuite de positionner le bateau, une seule fois. Si cette position ne permet pas de placer le bateau, alors il rechoisit aléatoirement une direction et réessaye de placer le bateau, ceci jusqu`à ce que le positionnement est possible.

- *Jeu personnalisé*

Lorsque l'utilisateur entre les données souhaitées pour un jeu personnalisé, il faut vérifier si ces données sont cohérentes pour que le jeu soit réalisable. Il a été décidé que chaque bateau allait avoir besoin d'un certain nombre de cases pour pouvoir être placé : deux fois sa longueur, et 3 cases pour l'avant ainsi que l'arrière du bateau. Il a également fallu vérifier que la grille soit assez large et longue pour accueillir chaque bateau. Une contrainte en plus est qu'une grille ne peut avoir des dimensions supérieures à 26 cases. Avec ces vérifications, la flotte a toujours pu être placée. Néanmoins, elle mène à certaines contradictions. Par exemple, le code ne permet pas de placer un bateau de longueur 1, sur une grille 1x1. Cependant, ce genre de situation n'a pas été considéré comme un problème majeur, car le jeu serait de toute façon trop facile.

- *Ant*

Le plus gros problème s'est avéré être avec l'utilisation de Ant. Les erreurs s'enchainaient, sans raison explicitée. Heureusement, les problèmes ont finalement pu être réglés après une multitude d'essais différents. Néanmoins, il n'a jamais été possible de lancer l'application Battelships par l'intermédiaire de Ant, sur un ordinateur Windows.

II. Erreurs connues

- *Plateau de jeu*

Suivant l'ordinateur sur lequel est lancé le jeu, l'interface graphique apparait faiblement différente. De plus, pour certaines dimensions de carte, le plateau de jeu est légèrement plus grand que l'espace prévu pour.

INTELLIGENCE ARTIFICIELLE

I. Description

L'intelligence artificielle de ce projet travaille avec une matrice, de mêmes dimensions que la grille, représentant en quelque sorte les probabilités que chaque case soit occupée par un bateau. Cette matrice est initialisée avec la valeur 0 pour chaque coordonnée (x, y).

Pour soumettre un essai, l'IA choisit une case (x, y) parmi celles ayant la plus haute probabilité d'être occupées par un bateau. La matrice est ensuite mise à jour suivant l'information reçue en retour.

Ci-dessous sont listés plusieurs concepts qui ont été exploités pour mettre à jour la matrice des probabilités.

II. Algorithmes

- *Répétition des tentatives*

Lors d'une tentative (x, y), la valeur de la matrice à l'index [y][x] est mise à -1 afin que l'IA ne considère plus (x, y) comme un choix potentiel.

- *Bateau coulé*

Les règles du jeu citent que deux bateaux ne peuvent se toucher (même en diagonale). Lorsqu'un bateau a été coulé, on sait alors qu'aucune case autour de ce bateau ne sera occupée. La matrice est donc mise à jour en mettant à -1 la valeur en [y][x], pour toute position (x, y) touchant le bateau coulé.

- *Touché*

Comme deux bateaux ne peuvent se toucher, un hit nous informe qu'aucune case touchant en diagonale le hit ne sera occupée. Les valeurs de ces cases dans la matrice des probabilités sont alors mises à -1. De plus, pour tenter de couler le bateau touché, les valeurs des cases à une distance 1 du hit sont incrémentées de 3, si leur valeur actuelle est positive.

- *Bateau le plus proche*

Lors de la soumission d'un essai, l'IA reçoit entre autres la distance pour atteindre le bateau le plus proche. La matrice des probabilités est mise à jour en incrémentant de 1 chaque case se situant à cette distance de l'essai préalablement soumis, pourvu que la case ne possède pas la valeur -1. Les cases se situant à une distance strictement inférieure à la distance pour atteindre le bateau le plus proche sont ensuite mises à -1.

MON PROJET

I. Points forts

- *Vitesse*

En mode graphique, le jeu réagit instantanément aux actions de l'utilisateur. De plus, en mode statistique, il est possible d'exécuter 1.000.000 de parties en 52 secondes.

- *Intelligence artificielle*

Une partie par défaut comporte 17 cases occupées. La moyenne du nombre de coups nécessaires pour que l'IA gagne la partie est de 28. L'IA réussi donc environs 3 coups sur 5.

- *Fonctionnalités supplémentaires*

L'utilisateur est averti qu'il a touché un bateau de plusieurs manières : le changement de couleur de la case touchée, l'apparition du type du bateau touché et l'affichage un message en bas de fenêtre.

II. Points faibles

- *Complexité*

Certains algorithmes ne sont pas forcément optimisés au niveau de la complexité. En particulier, l'usage de structures « if/ else if / else » et de boucles « for » imbriquées est assez fréquent.

- *Orienté objet*

Il s'est avéré particulièrement difficile de déterminer une bonne manière de structurer le projet. N'ayant aucune expérience préalable en création d'un projet de si grande taille, il est plus que probable que la structure des classes ne soit pas optimisée.

CRITIQUES

I. Aspects positifs

- *Un projet de taille importante*

Les séances pratiques du cours de Programmation et Algorithmique II permettent de mettre en pratique, individuellement, de nombreux aspects du cours théorique. Cependant, elles n'offrent pas la possibilité de travailler sur un projet de plus grande ampleur, qui nécessite un plus grand nombre de classes et une profonde réflexion sur la structure et la cohérence du projet. C'est pourquoi le cours de Projet d'informatique est particulièrement intéressant.

- *Une compréhension approfondie*

Ce projet permet également d'apporter une meilleure compréhension de certains aspects. Par exemple, la nécessité d'exploiter les exceptions m'a particulièrement servi. La mise en place de tests unitaires était également intéressant.

II. Aspects négatifs

- *Aide*

N'ayant aucune expérience de plus que les séances pratiques du cours de Programmation et algorithmique II, ce projet s'est avéré assez compliqué. Il aurait été intéressant de mettre en place quelques séances de présentations, en plus de celles qui sont déjà prévues. Par exemple pour expliquer comment fonctionne un jeu, ou quelques bonnes pratiques pour la hiérarchie des classes dans un projet d'une telle taille.

- *Mauvaises surprises du travail en groupe*

Le projet à réaliser au cours de l'année est un projet par 2. Nous avons été mis en garde quant au choix de notre compagnon de projet. Hélas, comment trouver avec certitude une personne sur qui on peut compter, pour nous accompagner dans la réalisation du projet ?

Personnellement, mon expérience à ce niveau n'a pas été particulièrement agréable. Lorsqu'il est devenu de plus en plus évident que je ne pouvais pas compter sur l'aide de mon binôme, je n'ai pas réellement eu d'autre choix que de passer le projet en seconde session. C'est pourquoi pour un travail d'une aussi grande ampleur, je trouve personnellement qu'il aurait été préférable de le réaliser individuellement, ou d'avoir le choix dès le départ de le réaliser seul ou à deux.

GUIDE UTILISATEUR

I. Comment jouer

- *Configuration de la partie*

Au lancement du jeu, l'utilisateur est d'abord invité à choisir un type de jeu parmi les 4 suivants :

→ Default game : initialise le jeu suivant les conventions requises

→ Custom game : permet à l'utilisateur de choisir la taille de la grille, et le nombre de bateaux de chaque longueur. La hauteur et largeur de la grille doivent chacun être compris entre 1 et 26.

→ Load map : transcrit les dimensions de plateau et positionnement des bateaux d'une flotte, représentés dans un fichier texte.

→ Load game : charge une partie précédemment sauvegardée.

L'utilisateur doit ensuite choisir s'il veut lui-même jouer la partie, ou si ce sera l'IA.

Tout au long de la partie, le bouton cheatMode est disponible pour rendre visibles les bateaux non coulés. Il est également possible de sauvegarder la partie, ou en démarrer une nouvelle.

- *Déroulement de la partie*

Suivant le type de joueur (humain ou IA), différentes fonctionnalités sont disponibles.

→ Humain : l'utilisateur peut entrer des coordonnées x et y pour tenter de toucher un bateau. Il reçoit alors des informations en retour, au bas de l'écran.

→ IA : l'utilisateur peut visionner tour par tour les choix effectués par l'IA, ou éventuellement passer directement au résultat en fin de partie.

Lorsque tous les bateaux ont été coulés, l'utilisateur en est averti et une fenêtre de fin de jeu apparaît. Il peut alors voir son score, décider de relancer la même partie, ou de quitter le jeu.

II. Mode console

Le mode statistique se lance via Ant. Il faut pour cela écrire la commande suivante en ligne de commande :

```
ant stat -Dargs=30
```

où le paramètre 30 peut être modifié, selon le nombre de jeux souhaités.

BIBLIOGRAPHIE

[1] ALMAS BAIMAGAMBETOV. MemoryPuzzleApp. 13 mars 2016. Consulté le 25/08/2019 sur : https://github.com/AlmasB/FXTutorials/blob/master/src/com/almasb/mp/MemoryPuzzleApp.java

Regular Expressions. Consulté le 25/08/2019 sur :
https://www.regular-expressions.info/quickstart.html
https://www.regular-expressions.info/numericranges.html

Find letter's position in Alphabet using Bit operation. Consulté le 25/08/2019 sur :
https://www.geeksforgeeks.org/find-letters-position-in-alphabet-using-bit-operation/

Learning about Electronics: How to create a Pop Up Window in JavaFX. Consulté le 25/08/2019 sur : http://www.learningaboutelectronics.com/Articles/How-to-create-a-pop-up-window-in-JavaFX.php

MOH-AW. How to close a java window with a button click – JavaFX Project. 30 juillet 2014. Consulté le 25/08/2019 sur : https://stackoverflow.com/questions/25037724/how-to-close-a-java-window-with-a-button-click-javafx-project

BYRON KIOURTZOGLOU. Java Code Geeks: How to write an Object to File in Java. 24 février 2013. Consulté le 25/08/2019 sur :
https://examples.javacodegeeks.com/corejava/io/fileoutputstream/how-to-write-an-object-to-file-in-java/

ANIRUDH BHATNAGAR. Java Code Geeks: Java ObjectInputStream and ObjectOutputStream Example. 6 octobre 2014. Consulté le 25/08/2019 sur :
https://examples.javacodegeeks.com/core-java/io/objectinputstream/java-objectinputstream-and-objectoutputstream-example/