ALGORITHME D'ARIANE

SAE 2.1

28 AVRIL 2023

SOUIDI Lyanis & DAOUADI Amir

Sommaire

I. Introduction

Ce projet consiste en l'étude d'un algorithme de guidage pour diriger un objet mobile à travers un labyrinthe d'obstacles jusqu'à une sortie. Dans notre cas, nous utiliserons Thésée, personnage de la mythologie grecque, perdu dans un labyrinthe crétois. Le labyrinthe sera représenté sous forme d'une grille carrée, où chaque cellule peut être bloquée ou libre.

À chaque étape, Thésée aura la possibilité de se déplacer d'une case dans une des quatre directions (haut, bas, droite, ou gauche). L'algorithme dictera le prochain mouvement de Thésée en fonction de sa position actuelle et de ses actions précédentes. Le programme se divise en trois parties : la création de la grille, le choix de l'algorithme et le test de l'algorithme. Le choix de la grille peut être effectué à partir d'un fichier ou en créant une nouvelle grille.

L'utilisateur peut modifier individuellement chaque cellule et choisir la position initiale de Thésée et de la sortie. Une fois la grille terminée, elle peut être sauvegardée. L'utilisateur peut choisir entre deux algorithmes, aléatoire ou déterministe, qui seront utilisés pour diriger Thésée.

Enfin, l'utilisateur peut choisir un mode de visualisation manuel ou automatique pour observer le déroulement de la simulation. L'objectif est de fournir un algorithme déterministe efficace capable de trouver la sortie si un chemin existe, avec des performances supérieures à celles de l'algorithme aléatoire.

II. Fonctionnalité du programme

1) Écran d'accueil

Au démarrage du programme, l'écran d'accueil suivant s'affiche, il permet à l'utilisateur de créer sa propre grille ou d'en importer une existante :

2) Création d'une grille

Si l'utilisateur décide de générer lui-même sa grille, le programme lui demandera la taille souhaitée. Un avertissement est affiché si l'utilisateur tente de créer une grande grille afin de l'informer qu'il pourra éprouver des difficultés lors de l'édition de la grille et que la durée des simulations pourra être impactée.

3) Remplissage de la grille

Une fois que l'utilisateur a choisi la taille de sa grille, il peut choisir entre 2 modes de remplissage.

a) Remplissage aléatoire

Dans ce mode, le programme placera aléatoirement la position de Thésée et de la sortie dans le labyrinthe. Le programme placera également des murs tout en veillant a ce que le labyrinthe soit réalisable (que Thésée puisse aller jusqu'à la sortie).

b) Partir d'une grille vide

Avec ce choix, l'utilisateur est directement redirigé vers l'éditeur de labyrinthe.

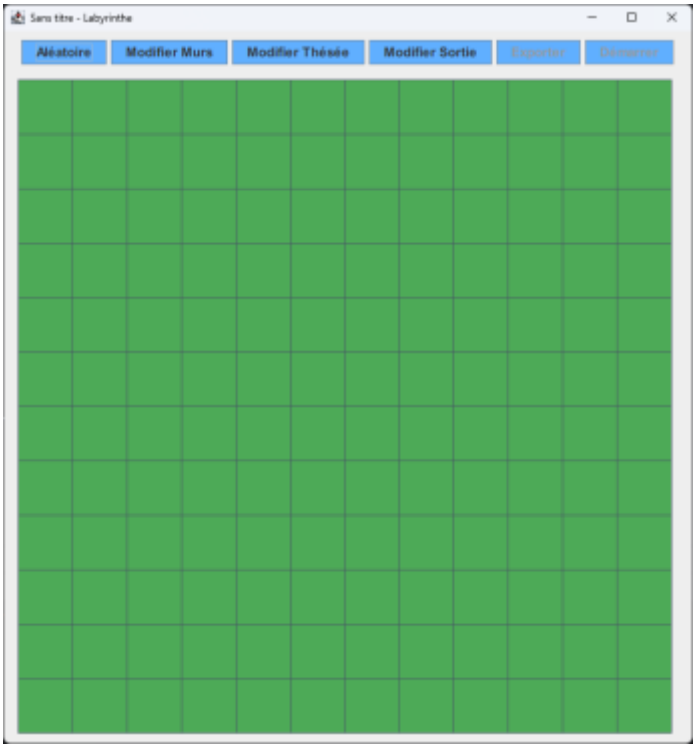
4) Editeur de labyrinthe

L'éditeur de labyrinthe permet à l'utilisateur de modifier un labyrinthe à sa guise. Il peut ainsi modifier la position des murs, placer Thésée et la sortie.

Un bouton **Aléatoire** permet de remplir aléatoirement la grille.

Le bouton **Exporter** permet d'exporter la grille vers un fichier.

Et pour finir, le bouton **Démarrer** permet de lancer un algorithme de résolution du labyrinthe.



5) Utilisation de fichiers *.lab*

Le programme prends en charge l'utilisation de fichiers *.lab*, ces derniers permettent d'importer et d'exporter facilement vos labyrinthes.

a) Importation de fichier

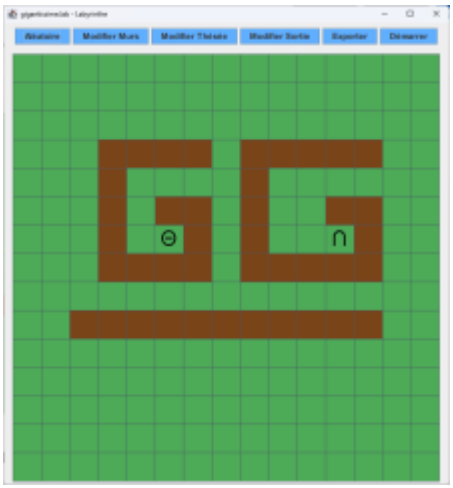
i. Depuis la ligne de commande

L'utilisateur peut ouvrir un fichier *.lab* en ajoutant son chemin d'accès en argument à la ligne de commande lors du lancement du programme. Exemple :

```
$ make run petit.lab
```

ii. Depuis l'écran d'accueil

En cliquant sur le bouton **Importer une grille** sur l'écran d'accueil, l'utilisateur pourra sélectionner le fichier contenant son labyrinthe, la vue est filtré afin d'afficher uniquement les fichiers .lab, cela permet à l'utilisateur de retrouver plus facilement son fichier.



b) Exportation de fichier

L'exportation de labyrinthe s'effectue depuis l'éditeur via le bouton **Exporter**, une interface similaire à celle d'importation permet à l'utilisateur de choisir l'emplacement dans lequel il souhaite enregistrer le fichier. Le bouton **Export** est accessible uniquement si le labyrinthe est valide (contient la position de Thésée, une sortie, et la sortie peut être atteinte depuis la position de Thésée).

6) Algorithmes de résolution

a) Les algorithmes

i. Aléatoire

L'algorithme aléatoire choisi une direction disponible (cela permet seulement de ne pas sortir de la grille, l'algorithme n'a pas connaissance des murs présents, si le cas se présente il pourra donc se diriger vers un mur) sans logique particulière, chaque direction possède des chances équivalentes d'être choisie.

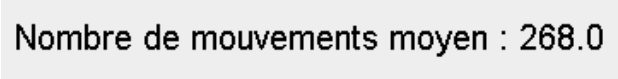
ii. Déterministe

L'algorithme déterministe possède une logique plus poussée lui permettant de réduire considérablement le nombre de mouvements nécessaires pour atteindre la sortie. Reportez-vous à la section "Exposition de l'algorithme déterministe choisi" pour obtenir le fonctionnement en détail de ce dernier.

b) Les modes d'affichages

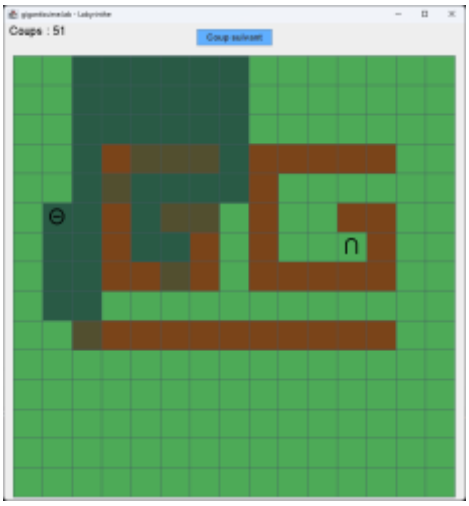
i. Automatique

Ce mode d'affichage effectue 100 simulations avec l'algorithme sélectionné puis affiche la moyenne du nombre de mouvement nécessaires pour résoudre le labyrinthe.



Il est prévu que le résultat des simulations s'affiche en temps réel afin de ne pas avoir d'écran vide dans le cas où les simulations durent longtemps.

ii. Manuel

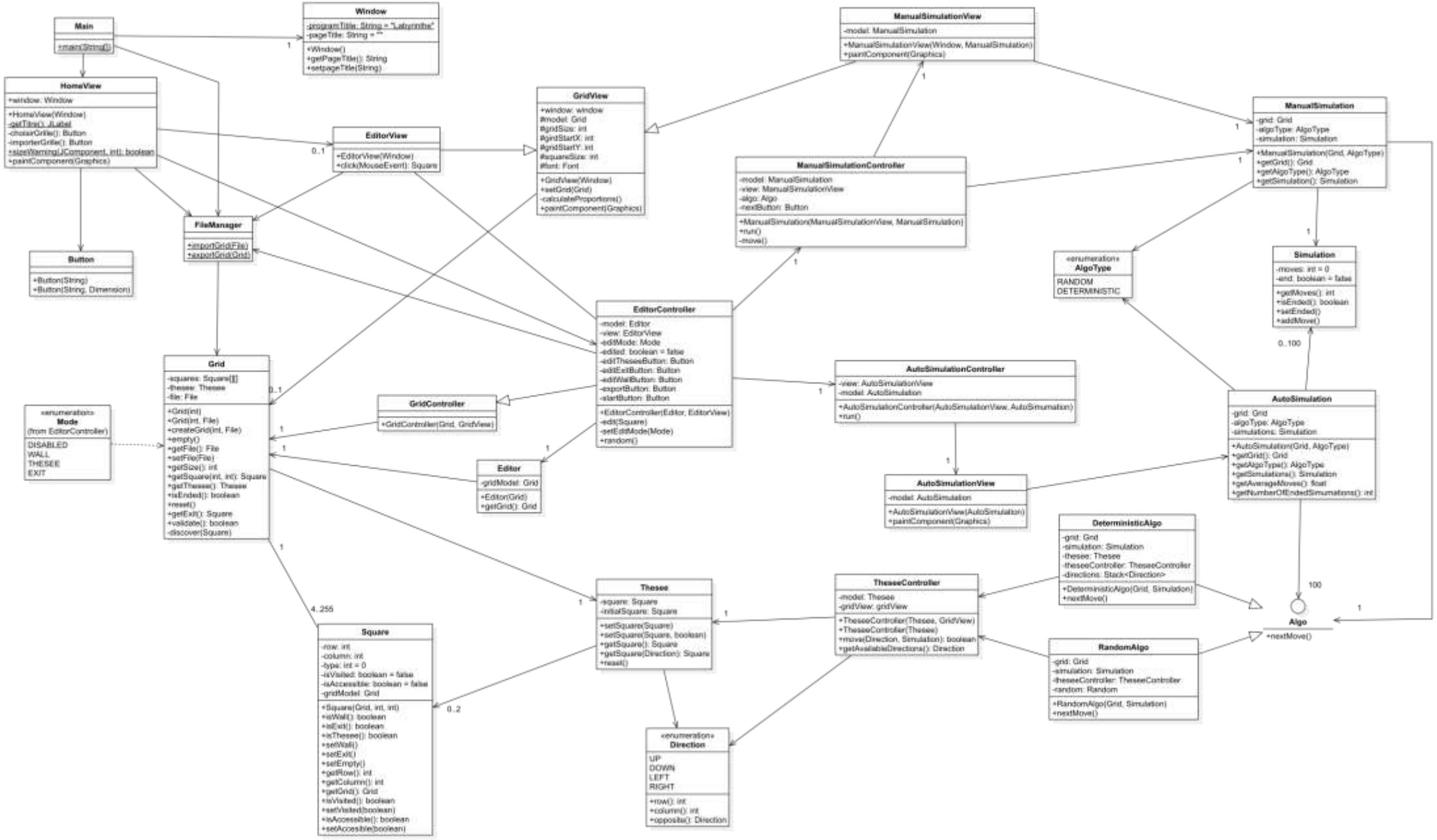


Ce mode d'affichage permet de voir en détail le fonctionnement de l'algorithme choisi, l'utilisateur peut ainsi voir pas à pas quelle cases sont explorées par l'algorithme en cliquant sur le bouton **Coup suivant** ou en pressant n'importe quelle touche de son clavier.

III. Présentation de la structure

Notre programme est structuré de manière à suivre l'architecture MVC, nous avons donc séparé au maximum les classes afin de respecter le principe de responsabilité unique.

Diagramme de classe



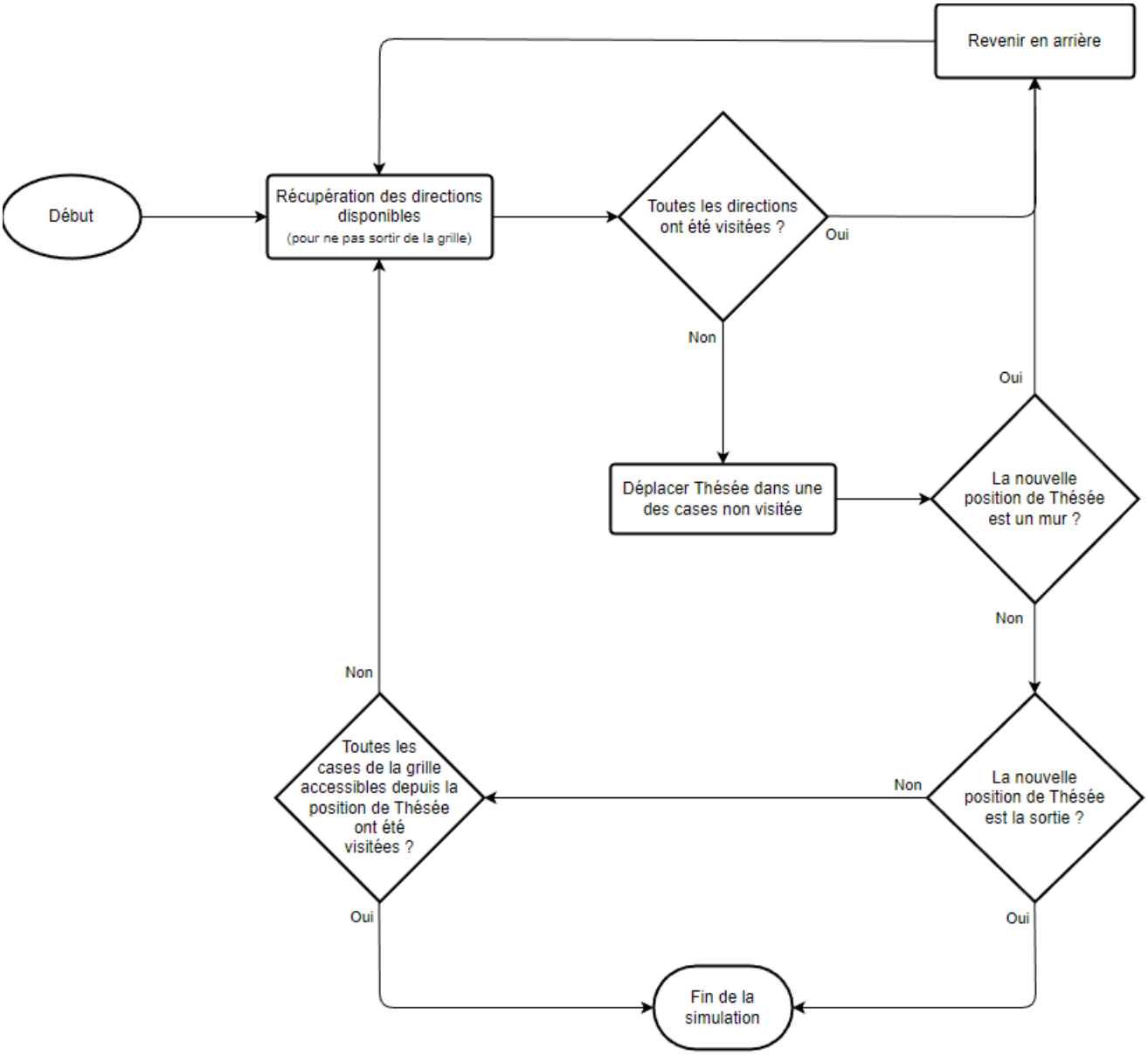
IV. Exposition de l'algorithme déterministe choisi

Notre algorithme prend la première direction disponible (cela permet seulement de ne pas sortir de la grille, l'algorithme n'a pas connaissance des murs présents s'il ne les a pas déjà visités, si le cas se présente il pourra donc se diriger vers un mur) qui n'a pas été visitée et ajoute la direction utilisée dans une stack. Si toutes les directions disponibles ont été visité, il dépilera le stack pour obtenir le dernier mouvement, et il fera le mouvement opposé pour revenir en arrière.

Cette connaissance des mouvements passées et des cases déjà visitées permet à l'algorithme de limiter un maximum le nombre de passage sur une case.

L'algorithme ne s'arrête qu'une fois la sortie trouvée ou que toutes les cases accessibles depuis la position de Thésée ont été découvertes, de cette manière on est sûr que l'algorithme réussira à trouver la sortie si elle est bien accessible.

Algorigramme



V. Conclusion

Amir Daouadi

En conclusion, le projet que nous avons réalisé en utilisant le modèle MVC en Java a été une expérience extrêmement enrichissante. Malgré le peu de temps dont nous avons disposé pour le réaliser, nous avons pu mettre en pratique nos connaissances en programmation orientée objet, en architecture logicielle et en développement web.

Le modèle MVC s'est avéré être un choix judicieux pour ce projet, car il nous a permis de séparer efficacement la logique métier de la présentation et de la gestion des événements. Cela a rendu le code plus facile à comprendre, à maintenir et à étendre, et a également permis une collaboration plus efficace avec les autres membres de l'équipe de développement.

En résumé, grâce à ce projet, nous avons pu acquérir de nouvelles compétences en programmation et en conception de logiciels, ainsi qu'en travail en équipe. Nous sommes fiers du travail accompli et convaincus que les compétences que nous avons acquises seront utiles dans nos projets futurs.

Lyanis Souidi

Ce projet à été très enrichissant et m'a permis encore une fois de d'approfondir mon usage d'outil de collaboration et de versionnage tel que Git. Le développement de ce programme m'a permis d'en savoir plus sur le fonctionnement d'une application Java, notamment la manière dont les instructions sont traitées et comment faire en sorte que certaines instructions soient effectuées en simultané à l'aide de Thread. Je suis également très content du travail réalisé et heureux d'avoir pu approfondir mes connaissances.