UMONS
Université de Mons

Projet d'informatique (Quoridor)

‣ Jason Fouret et Alan Altruy ◂

BAB1 sciences informatique

Table des Matières

1. Introduction

Dans le cadre du cours "projet d'informatique", nous devions créer nous-même, de toutes pièces, un jeu programmé en java appelé Quoridor. Ceci est le rapport que nous devions rendre en même temps que le jeu en question. Dans ce rapport, vous allez suivre notre parcours et plus précisément nos idées pour la création de notre projet, nos problèmes rencontrés et les solutions apportées à ces mêmes problèmes.

Au début, ce projet nous faisait peur mais nous n'avons pas tardé à le commencer, et au fur à mesure que le temps avançait, l'objectif final nous semblait de plus en plus accessible et nous donnait donc l'envie de continuer. Le résultat nous rendait de plus en plus fiers et nous encourageait à continuer davantage. La principale difficulté que nous avons rencontrée était de continuer à évoluer malgré les problèmes sanitaires récents qui ont engendré un léger manque de confiance, il nous a donc fallu retrouver l'envie de continuer et terminer ce projet.

2. Avancement du projet

2.1 Répartition des tâches

Dès le début, nous nous sommes distribué les tâches du programme ainsi que la rédaction du rapport équitablement. Nous avons bien évidemment fait en sorte que chacun d'entre nous comprenne chaque petit bout de code.

2.2 Commencement (interface terminal)

Au tout début, nous avons pris le temps de réfléchir afin de savoir par où nous allions commencer ce projet, nous avons donc noté toutes nos idées par écrit. Par la suite, nous avons lu attentivement les documents fournis afin de comprendre réellement quelles étaient les attentes vis-à-vis du projet et ainsi, nous aider dans notre tâche. La première chose que nous devions faire était la prise en main du logiciel "Visual Studio", nous avons choisi ce logiciel pour coder car il est complet et assez facile d'utilisation.

Nous avons ensuite réellement commencé à programmer le 17 février 2020. Notre première idée était de créer le plateau[1] sur un terminal, car nous ne savions pas encore comment créer une interface graphique à ce moment-là.

Puis, nous avons commencé à créer la classe nommée "QuoridorLoop" qui, dans notre idée, allait gérer la méthode de jeu dite "tour par tour". Cette classe est donc la classe principale qui gère l'avancement de la partie et permettra de mettre fin au jeu s'il y a un gagnant ou si l'on demande tout simplement de quitter le jeu.

Les classes suivantes que nous avons implémentées sont les classes "QuoridorGame", "QuoridorTray", "QuoridorCell" et "QuoridorDisplay".

[1] Nous voulions directement faire un plateau de 19x19 car nous pensions déjà à l'avenir du projet et nous nous disions que ce serait plus simple si on décidait de bouger les pions sur les cases impaires, et les barrières sur les cases paires

- "QuoridorGame" permet de choisir l'action d'un joueur et de poser une barrière.
- "QuoridorTray" permet d'initialiser et afficher le plateau dans le terminal, et de vérifier si l'on pouvait poser une barrière,.
- "QuoridorCell" permet de conserver l'état d'une cellule du plateau.
- "QuoridorDisplay" permet uniquement d'appeler une méthode de "QuoridorTray".

Le programme, à ce moment là, permettait donc uniquement de placer une barrière et d'afficher le plateau, nous avons donc commencé à implémenter les classes "QuoridorPlayers" et "QuoridorMovePlayer" afin que le jeu soit plus complet et qu'on ait une meilleure vision du potentiel de notre programme.

- "QuoridorPlayers" permet d'initialiser un joueur, accéder à ses informations et de savoir s'il a gagné.
- "QuoridorMovePlayer" permet de savoir quels déplacements un joueur peut effectuer.

Le jeu, à ce moment, fonctionnait dans l'ensemble, les vérifications suffisaient pour permettre de jouer mais cela n'était évidemment pas pratique étant donné l'interface "terminal" très arbitraire, peu polyvalente et peu intuitive.

2.3 Le début des problèmes (interface graphique)

Nous nous sommes dès lors intéressés à une interface graphique juste après avoir terminé la classe "QuoridorMovePlayer", fin février. On peut parler du début des problèmes étant donné que c'était la toute première fois que nous entrions dans ce sujet sans aucune connaissance au préalable. Nous allions tellement vite pour tout créer que nous avons très vite dépassé le cours. Le défi suivant était donc de comprendre par nous-même les interfaces graphiques et d'autres concepts encore non étudiés.

Nous avons commencé par la classe "QuoridorGui" qui faisait fonctionner l'interface graphique sans thème, toute simple. Avec la création de cette classe, le premier gros problème est arrivé. Le programme ne fonctionnait pas. Nous n’avions aucune idée du pourquoi il ne fonctionnait pas. Donc par pur hasard nous lui avons demandé d’afficher un texte en guise de vérification et puis là, surprise, le programme a subitement fonctionné et a décidé d’afficher les panneaux voulus à l'écran. Plus tard, nous avons su résoudre ce problème grâce à une méthode héritée de JPanel.

Nous n'avons pas tardé à remarquer que la classe "QuoridorGui" ne suffirait pas pour implémenter l'intégralité de l'interface graphique. Nous avons donc créé trois classes supplémentaires pour distribuer les taches: "QuoridorPanel", "QuoridorButton" et "QuoridorPicture".

- "QuoridorPanel" permet d'afficher la bonne fenêtre à l'écran et d'y insérer les bons éléments (boutons, images).
- "QuoridorButton" permet de créer des boutons cliquables et d'y attribuer une action définie.
- "QuoridorPicture" permet de récupérer et de placer correctement une image sur un plateau.

Avec l'arrivée de ces classes, de nombreux problèmes se sont présentés. Nous avions par exemple de la difficulté à placer les boutons au bon endroit, ne pas déformer les images, afficher les panneaux sur la fenêtre et pouvoir rafraîchir une fenêtre dès que la souris passe ou clique sur un bouton. Heureusement, chacun de ses problèmes a vite été résolu.

Une fois que tout fonctionnait, nous avons décidé ensemble d'un thème original pour notre jeu. Nous nous sommes concertés et, étant donné que nous aimions tous les deux Minecraft, nous n'avons pas hésité à réadapter des objets du jeu Minecraft pour notre jeu Quoridor sur Photoshop.

2.4 Gradle et Pathfinding

Nous nous sommes intéressés à Gradle assez tard dans le développement de notre projet car nous ne comprenions pas comment l'implémenter comme il faut, Gradle est donc arrivé après l'interface graphique et la majeure partie de notre programme. Mais une fois que Gradle fonctionnait, nous étions soulagés, nous avons donc enchainé avec les vérifications plus complexes comme le pathfinding et vérifier qu'une barrière ne bloque aucun joueur.

Nous avons ainsi créé la class "QuoridorPathfind", c'est sans doute la partie la plus complexe de notre programme, beaucoup de vérifications s'enchaînent pour ainsi permettre à ces vérifications plus conséquentes de fonctionner parfaitement et avec la meilleure complexité possible. Nous avons donc utilisé des HashMap et des ArrayList pour nous faciliter la tâche.

C'est aussi à ce moment que nous avons décidé de réunir les classes "QuoridorCell", "QuoridorDisplay" et "QuoridorTray" en une seule et unique classe toujours intitulée "QuoridorTray". En effet, "QuoridorDisplay" et "QuoridorCell" ne nécessitaient pas une classe étant donné le code très court qu'il y avait dans ces classes. "QuoridorTray" prend donc en charge le type des cellules de son tableau et permet d'afficher si nous le souhaitons un plateau dans le terminal et quelques vérifications concernant les cellules.

Nous avons également décidé de réunir toutes les vérifications concernant le positionnement d'une barrière sur le plateau dans une classe appelée "QuoridorFence" car ces vérifications étaient initialement dispersées dans "QuoridorTray" et " QuoridorGame", ce qui ne convenait pas du tout.

2.5 L’aboutissement du projet

Comme demandé dans l'énoncé du projet, nous devions faire 2 Intelligences artificielles : une facile et une difficile. Premier problème, comment faire ? Nous avons tout d’abord fait plusieurs schémas sur papier et suite à cela, nous avons imaginé un procédé: l’IA difficile va toujours choisir la meilleure option possible pour elle, c'est-à-dire soit poser une barrière pour ralentir la progression du joueur soit se déplacer. Ce choix est décidé en fonction du nombre de case qu’il reste au joueur et à l’IA pour finir la partie. Donc si le nombre de case de l’IA pour finir est plus grande que pour le joueur, l’IA pose une barrière a un endroit stratégique, sinon elle se déplace par le chemin le plus cours grâce au pathfinding implémenté plus tôt.

Pour l’IA plus facile, nous avions décidé de reprendre le code de l’IA difficile et de le modifier pour qu’elle repose un peu plus sur de l’aléatoire. Le déplacement de l’IA facile est donc le même que pour l’IA difficile, l'IA facile fait ses choix aléatoirement et n'est donc pas très polyvalente. Par ailleurs nous avons laissé cette option pour voir les deux IA s’affronter; il faut cliquer sur le gros bouton "Quoridor" au menu principal.

3. Subtilités et points forts du programme

Après l’implémentation des deux IA, le projet était terminé, il ne nous restait plus que ce rapport à finir et on pouvait rendre le projet, donc nous avons décidé de créer des fonctions supplémentaires
à savoir: la surbrillance des barrières, c'est-à-dire que lorsqu'on passe la souris sur le plateau, on peut voir les endroits où l'on peut poser les barrières. Pour ce faire, nous avons fait en sorte que, lorsqu'on passe la souris sur les endroits où l'on peut poser une barrière, une image de barrière en opacité moindre s’affiche et dès que la souris est enlevée, l’image se retire.

La seconde option supplémentaire est que si nous cliquons sur le pion du joueur nous pouvons voir où les déplacements sont possibles. En effet, nous avons ajouté une image (Ender Pearl, utilisée dans Minecraft pour se téléporter) sur les cellules où le joueur peut se déplacer.

La troisième option est une aide pour un joueur. Il suffit de cliquer sur le panneau avec le nom du joueur. Un système d'aide basée sur l'intelligence artificielle difficile joue le coup le plus approprié pour lui.

Le principal point fort de notre programme est que nous n'avons pris aucune ligne de code venant de quelqu'un d'autre. Le projet a entièrement été écris par nous et uniquement nous.

Le deuxième point fort est que par rapport aux 5 secondes de latence maximale demandées dans l'énoncé du projet, notre projet est très rapide, car tous les coups se jouent en moins de deux secondes et souvent en moins d'une seconde. Par ailleurs, lors de l'exécution du mode IA contre IA, une partie dure en moyenne 15 secondes.

4. Points faibles

Nos principaux problèmes concernent de manière générale Gradle et les tests unitaires. En effet, Gradle a souvent causé des problèmes lors de l'exécution du programme et des tests. Nous avons donc passé beaucoup de temps à tenter de résoudre ces erreurs.

Les tests unitaires ne fonctionnent toujours pas. Nous n'en avons donc pas, nous avons pourtant cherché sur énormément de forum des réponses concernant ces tests mais aucune réponse n'a été une solution pour nous.

5. Fonctionnement

5.1 Fonctionnement du programme ($ gradle run)

Pour lancer notre programme, il suffit de lancer la commande "gradle run", la fenêtre s'ouvrira ensuite. Le menu principal va s'ouvrir, vous pourrez choisir de cliquer parmi les boutons suivants.

- Bouton "Quoridor": lance une partie IA facile contre IA difficile
- Bouton "Solo": lance un menu de sélection de difficulté
  - Bouton "Normal": lance une partie IA facile contre un humain
  - Bouton "Difficile": lance une partie IA difficile contre un humain
  - Bouton "Menu Principal": permet de retourner au Menu Principal
- Bouton "Multijoueur": lance une partie humain contre humain
- Bouton "Charger": charge la dernière sauvegarde si le bouton n'est pas grisé
- Bouton "Quitter le jeu": permet de quitter le jeu

Lors d'une partie:

- Bouton "Quoridor": permet d'ouvrir un menu pause
  - Bouton "Menu Principal": permet de retourner au Menu Principal
  - Bouton "Sauvegarder et Quitter": permet de sauvegarder et retourner au Menu Principal
  - Bouton "Retour": permet de continuer la partie

- Bouton "Steve", "Creeper": permet à un joueur humain de s'aider d'une IA pour un coup
- Pour se déplacer, il faut cliquer sur le pion du joueur afin que le système puisse indiquer les endroits sur lesquels le joueur peut se déplacer

5.2 Fonctionnement de l'utilitaire statistiques ($ gradle runStat)

Pour lancer l'utilitaire de statistiques, il suffit d'entrer la commande:

" gradle runStat --args="**nombre de tests* *difficulté d'IA1* *difficulté d'IA2**"

Exemple: "*gradle runStat --args-"5 facile difficile*"

Si vous souhaitez afficher l'interface terminal, il suffit d'ajouter "oui" à la fin des arguments

Exemple: "*gradle runStat --args-"5 facile difficile oui*"

6. Conclusion

Pour conclure ce rapport, nous pouvons dire que ce projet est rondement mené. Nous avons exécuté pratiquement tous les points demandés et même plus avec la surbrillance, les déplacements possibles et le bouton d'aide. Nous pouvons aussi dire que la création de notre projet nous a procuré une satisfaction certaine d'avoir fait notre premier « jeu » entièrement par nous-même. Nous avons aussi été poussé à mieux comprendre java et la programmation orientée objet pour la suite de nos études.

Cependant, notre projet a une certaine faiblesse qui concerne essentiellement les tests unitaires, cela a été notre gros point faible et, nous ne sommes toujours pas parvenu à les exécuter… Après de nombreuses tentatives, d'échecs et beaucoup de recherches sans aucun résultat concluant, nous avons du nous résoudre à abandonner les tests unitaires.

Finalement, nous pouvons dire que notre jeu fonctionne sans erreurs ou bugs, ce qui nous rassure tout de même.

7. Bibliographie

Toutes nos images ont été tirées du jeu Minecraft, le lien officiel vers le site du jeu se trouve ci-dessous:

https://www.minecraft.net/fr-fr/