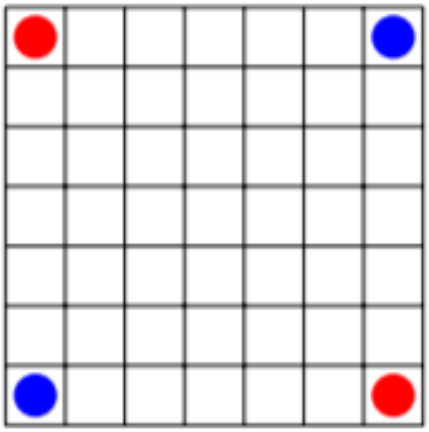
- KERMEZIAN AXEL : 22008897
- KRIMI IBRAHIM : 22011592
- IHANNOUBA NIHAL : 22009099

UNIVERSITÉ CAEN NORMANDIE

Projet Sécurité et aide à la décision

Jeu d'infection



Sommaire

1.Introduction

Le projet consiste à réaliser le jeu de l'infection. Le jeu se joue sur un plateau de 49 cases par deux joueurs de couleurs différentes. Ayant pour l'un des pions bleus et des pions rouges pour le second. Lorsque la possibilité de jouer leur est attribué, chaque joueur peut jouer un coup en effectuant un saut ou clone de ses propres pions.

Les cases qui sont adjacentes à ce coup contenant le joueur adverse sont infectées et se transforme en pion de la couleur du joueur ayant joué.

La finalité est d'obtenir le plus de pions de sa couleur sur le plateau.

2.Le programme réalisé

2.1. Environnement et conception

Le développement de ce projet a été effectué en Java sur le système d'exploitation Linux. Le code est compilé et exécuté via le terminal grâce à un fichier bash. Le projet contient cinq classes différentes qui représente l'ensemble du jeu. Trois joueurs ont été implémentés :

- Un joueur aléatoire (fonction dans la classe State)
- Minimax (Classe)
- Alphabeta (Classe)

2.2 *Fonctionnalités*

A l'exécution, le programme propose aux deux joueurs de choisir une IA : Alphabeta ou Minimax ainsi que sa profondeur de recherche.

Lorsque la saisie est valide, la grille du jeu est initialisée avec les quatre pions situés dans les angles.

La grille du jeu est affichée, le joueur courant exécute un coup valide celui-ci est affiché sur la console suivis de la liste des coups étant valide pour ce joueur.

La grille du jeu actuelle et celle actualisé sont alors affichées avec le score et nom de chaque joueur.

L'affichage de la grille de jeu est une chaine de caractère contenant :

- Un "B" qui représente les pions bleus
- Un "." qui représente les cases vide
- Un "R" qui représente les pions rouges

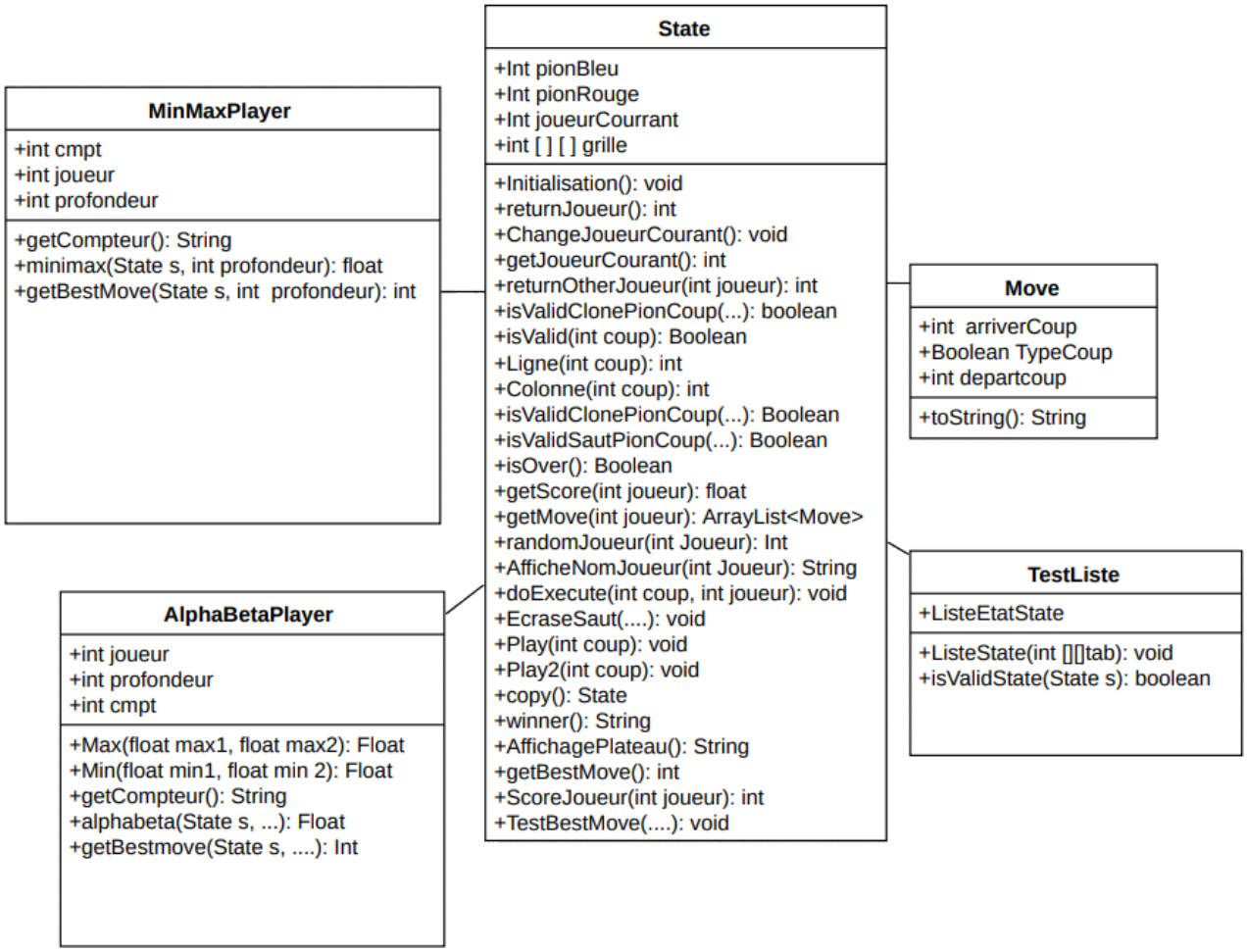
En fin de jeu le score est également affiché suivis du nom du gagnant, du nombre de nœud parcouru pour chaque joueur ainsi que le nom de l'IA utilisée.

3.Code

Le programme est un unique package contenant six classes :

- La classe State contenant des méthodes propres à l'exécution du jeu ainsi que la grille du jeu et ses joueurs.
- La classe Move représente un coup sous forme d'entier de départ, d'arrivé et de type de coup.
- La classe TestListe permet de stocker les états précédents afin de tester si l'état actuelle n'est pas un état précédemment joué (normalement condition de isOver).
- La classe MiniMaxPlayer est le joueur MiniMax qui permet de retourner le meilleur coup selon sa profondeur en paramètre.
- La classe AlphaBeta est le joueur qui permet d'utiliser un élagage pour retourner le meilleur coup.
- La classe Main

4.Documentation UML (Digramme de classe)



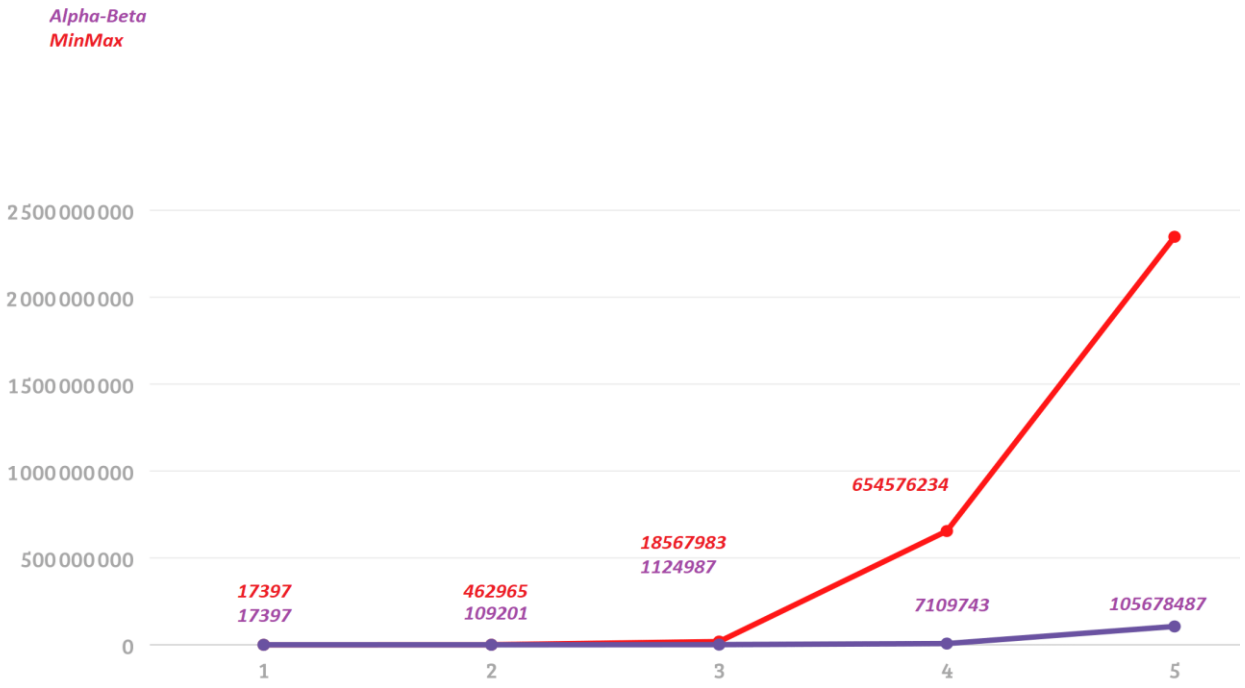
5.Organisation

Ayant évolué sur ce projet à trois, nous avons opté en début de projet pour une écriture du code en cohésion et non via un dépôt de code tel que GitHub. Dans un premier temps il nous a fallu définir clairement les règles du jeu. Par la suite le code a été répartis uniquement pour la classe State. Pour le reste des classes implémentant une intelligence artificielle nous avons écrit le code en ce concertant.

Chaque soir, durant environ trois heures nous travaillons sur le code afin de mettre en place les principes du jeu de l'infection ainsi que les différentes IA.

6.Expérimentation

6.1 *Graphique des expérimentations*



6.2 *Commentaire*

Durant nos expérimentations nous avons constaté certaines différences concernant les deux IA.

Dans un premier temps la profondeur étant de 1 pour les deux IA nous renvoi un nombre de nœuds égaux parcourus. Cela est tout à fait normal car Alphabeta ne fait aucun élagage.

Pour l'utilisation d'une même profondeur pour les deux IA le score final en nombre de pion est le même pour chaque joueur (score différent pour les deux joueurs qui s'affronte).

Nous nous sommes ensuite vite aperçus que lorsque la profondeur de l'IA était supérieure à trois le temps de calcul augmentait considérablement (plus de 1h).

Tout de même la conclusion étant :

Selon la profondeur qui engendre un temps de calcul supérieur , Alphabeta est nettement plus rapide que celui Minimax qui lui augmente de façon considérable.

7.Conclusion

Ce projet nous a permis d'utiliser la connaissance apportée par l'université sur le langage Java pour implémenter un jeu et ses règles en autonomie. L'étude et l'écriture d'une intelligence artificielle nous a permis d'aborder ce vaste domaine très intéressant et d'expérimenter différentes situations de jeu.

Selon le jeu, l'IA sera implémentée différemment et son temps de parcours sera différent également.

Certain jeu y compris le jeu de l'infection pour l'utilisation d'une IA tel que Minimax ou Alphabeta avec une profondeur supérieure à trois prend un temps de parcours beaucoup trop important. Cela n'est donc pas optimale pour un jeu commercialisé.

Alphabeta reste tout de même plus rapide que MiniMax dû à son élagage.