Projet Othello

Kevin Hivert Ronan Abhamon

1 Fonctionnalités

Notre programme permet de choisir entre 2 types de parties, joueur humain contre IA et IA contre IA via un serveur suivant les normes officielles définies en début de semestre. Le premier type de jeu n'étant pas le but du projet il ne permet de faire qu'une seule partie puis quitte le programme. Durant les phases de développement ce mode à été modifié afin de faire jouer deux versions d'IA différentes l'une contre l'autre (Simplement en appelant la fonction d'une autre version de l'IA plutôt que la fonction permettant au joueur de donner une position ou placer un pion).

Le second type de partie, permet de ce connecter à un serveur d'othello et de commencer une partie contre une autre IA. Étant donné que les deux groupes qui étaient chargés des serveurs n'ont pas pu nous fournir quelque-chose de fonctionnel (la connexion ne fonctionne pas toujours correctement et le tableau reçu n'est pas toujours correct) et qu'ils ne respectaient pas les normes fixées au début du projet (entre autre en imposant l'envoi de "\n"), les tests ont été effectués sur un serveur fonctionnel et respectant les normes, fait par M. Alexis Braine, sur lequel nous avons pu tester notre programme et affronter de nombreuses autres IA.

2 Évolutions du programme

Durant le développement nous avons pu construire cinq version d'IA plus ou moins différentes.

La première version de l'IA est basique, utilise un algorithme min-max et évalue un plateau grâce à un tableau donnant un poids à chaque cellule. La qualité d'un plateau est donc simplement donnée par la somme des poids des cellules détenus par le joueur.

La seconde version utilise un algorithme min-max avec un élagage alpha bêta nous faisant ainsi gagner en profondeur de calcul. La fonction d'évaluation est moins naïve et gère les cases avant les coins au cas par cas. En effet dans la première version ces positions étaient simplement considérées comme mauvaises, ici on étudie plus en détail ce qui ce passe autour. On regarde ce qu'il y a dans le coin et ce qu'il y a au bout de la ligne, la colonne ou la diagonale ou se trouve la positon testée, suivant les cas on décide si la position est plus ou moins bonne. Nous avons fait cette fonction en observant le fait que jouer avant le coin, si le coin est vide est effectivement un mauvais choix mais jouer avant le coin si on possède le coin est un bon choix puisque le pion ne peut être perdu.

Cette IA à également été codé avec l'algorithme Negamax ce qui permet de réduire la quantité de code à écrire mais ne change en rien la profondeur de calcul ou la qualité de l'évaluation.

La troisième version de l'IA affine un peu plus l'évaluation en se préoccupant en plus du moment de la partie. Il faut en effet adapter une stratégie différentes suivant le moment de la partie, en début de partie il est préférable de n'avoir que peu de pions afin que l'adversaire prenne les positions délicates, puis en milieu de partie on doit maximiser notre nombre de pions et en toutes fin de partie on tente de limiter les choix de l'adversaire jusqu'à éventuellement ne lui laisser aucune possibilité de jeu et alors jouer deux fois consécutivement. Nous avons donc appliqué ce principe et différencié les moments de la partie simplement grâce à un compteur nous donnant le nombre de coups restant à jouer et nous permettant ainsi de nous situer dans la partie. Le milieu de la partie est évalué grâce à la même fonction que dans la version précédente.

La quatrième version utilise le même principe que la précédente mais en début de partie utilise les degré de liberté de chaque pion. Ce qui affine un peu plus le jeu en ne comptant donc pas les pions qui sont totalement encerclés et qui ne constituent donc pas des possibilités de jeu pour l'adversaire. Le milieu de la partie est un peu améliorer également puisqu'en plus d'utiliser la fonction d'évaluation utilisée dans la version 1 de l'IA, on prend en compte le nombre de pions de l'adversaire et on tente donc en plus de le limiter.

3 Bonus

Nous avons aussi implémenté une fonction (POSIX mais non ANSI...) permettant de déterminer le temps mis en ms pour que notre IA joue un coup. Au maximum, elle met 3000ms pour jouer un coup avec une profondeur de 7.

Nous avons pris le soin de rendre notre code portable Windows/UNIX/Mac. Les fuites mémoires ont aussi été traitées par le biais du soft Valgrind.