Elliker Clement
Jeromin Tristan
Sangare Amadou

Rapport Projet Vallée des Rois

Représentation du plateau de jeu:

Nous avons décidé de représenter le plateau de jeu par un tableau statique de taille 6*6. En effet, puisque la taille du plateau ne change pas durant une partie, il n'est pas nécessaire d'utiliser un tableau dynamique. Chaque case contient un type enum KVPiece qui peut prendre les valeurs suivantes: EMPTY, BLUEKING, WHITEKING, BLUESOLDIER et WHITESOLDIER.
Ainsi lorsque notre joueur réceptionne un mouvement, il faut reformater ce message afin de pouvoir l'appliquer dans notre tableau statique. En effet, le plateau de jeu contient des cases numéroter de 1 a 7. Or, notre tableau possède des indices allant de 0 a 6. De plus, il faut pouvoir transformer les lettres de A a G en numéros de 0 a 6. Par exemple, le mouvement « B7-B2 » correspond à un mouvement de la case (6,1) a la case (1,1)

Algorithme utilisé:

Nous utilisons l'algorithme AlphaBeta pour notre joueur. En effet, celui-ci permet d'obtenir les mêmes coups que MiniMax mais avec un temps de calcul moins élevé.

Fonction heuristique:

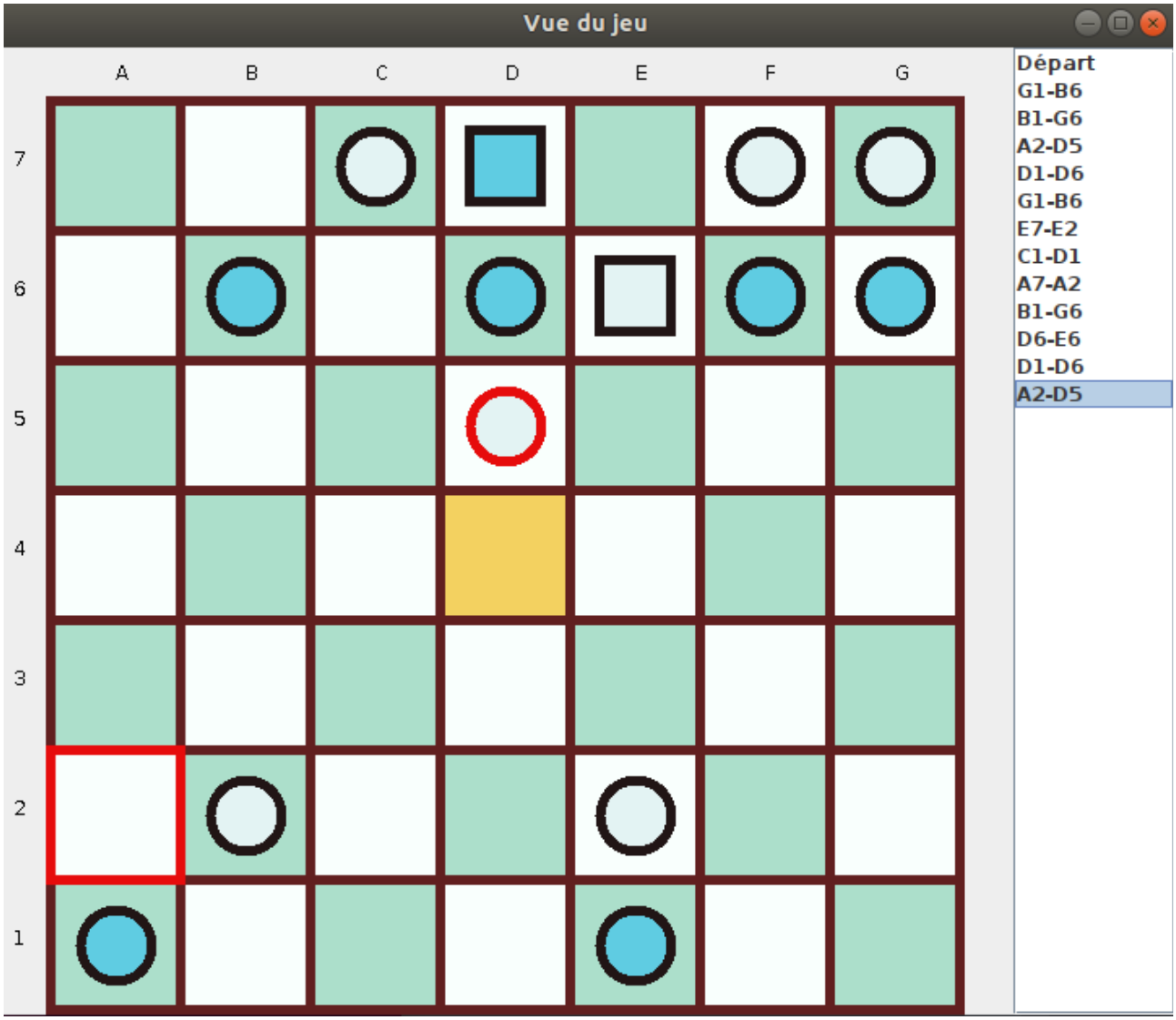
Nous avons au départ implémenté une fonction heuristique qui compte la différence de coup possibles entre notre joueur et son adversaire. Cette heuristique était efficace fasse à un joueur choisissant ces coup aléatoirement. Cependant, cette stratégie perdait systématiquement face à un adversaire plus intelligent car elle ne possédait pas de stratégie sur le moyen et long terme. Nous avons donc décider d'ajouter d'autres critères dans notre heuristique :
Nous allons mesurer à quel point chaque pièce est proche du centre du plateau, et en particulier notre rois. Nous utilisons la ligne de calcul suivante:
6 – Math.abs(3-i) – Math.abs(3-j) qui renvoie une valeur entre 0 et 6 pour chaque case du tableau.
Ces valeurs sont stockés par les variables centreSoldierI pour les soldats et centreI pour notre roi.
A l'inverse, ces valeurs pour sont adversaires sont stockés dans centreSoldierOther et centreOther.

Voici la valeur de ce critère sur les différentes cases d'un plateau:

0	1	2	3	2	1	0
1	2	3	4	3	2	1
2	3	4	5	4	3	2
3	4	5	6	5	4	3
2	3	4	5	4	3	2
1	2	3	4	3	2	1
0	1	2	3	2	1	0

Nous avons choisis de remplacer le comptage du nombres de coups disponibles par la mesure du nombre de cases libres adjacentes a chaque pièces: En effet, cette méthode permet de séparer le nombres de coups disponibles pour le soldat et pour le roi et ainsi vérifier si notre roi est sur le point de se faire encercler.
Pour ce faire, nous avons implémenté la méthode nbFreeSpace qui renvoie le nombre de case libres adjacentes de la case prise en paramètre.
Nous stockons cette valeur dans nbFreeSpaceI pour notre roi et nbFreeSpaceSdI pour les soldats.
Enfin, nous avons pu remarquer que les cases se trouvant sur les deux diagonales du plateau
(c'est a dire les cases A1, A7, B2, B6, C3, C5, D4, E3, E5, F2, F6, G1, G7) sont un point stratégique car elle permettent a notre joueur de placer ces pièces proches du centre rapidement. Nous stockons cette valeur dans la variable diag, cependant nous avons par la suite décidé de ne pas prendre en compte les 4 coins du plateau car ceux-ci ont peu d'intérêt vers la fin d'une partie.
Afin de décider quel critère devrait être le plus pris en compte, nous avons ajouté un coefficient à chaque critère afin de pouvoir tester rapidement un grand nombre d'heuristiques différentes. Pour ce faire, nous avons fait jouer notre challenger contre lui-même a de nombreuses reprises, mais avec a chaque fois des coefficients légèrement différents pour chaque critères.

Nous avons finalement choisis les coefficient suivants pour nos différents critères:
coeffCentreKingI = 5
coeffFreeSpaceKingI = 3
coeffCentreKingOther = 5
ceffFreeSpaceOther = 3
coeffCentreSoldierI = 10/18
coeffFreeSpaceSoldierI = 1/3
coeffCentreSoldierOther = 5/18
coeffFreeSpaceSoldierOther = 1/6
coeffDiag = 3



Ici, dans cet exemple, notre joueur joue les pièces bleus.
On a d'après les positions des rois:
centreI = 3
centreOther = 3

FreeSpaceI = 2
FreeSpaceOther = 3

D'apres les positions des soldats:
centreSoldierI = 2 + 4 + 2 + 1 + 0 + 2 = 11
centreSoldierOther = 2 + 1 + 0 + 5 + 2 +3 = 13

FreeSpaceSdI = 7 + 4 + 4 + 2 + 2 + 4 = 23
FreeSpaceSdOther = 2 + 1 + 0 + 6 + 7 + 7 = 23

Ainsi, à partir de ces critères nous pouvons calculer la valeur finale de notre heuristique qui dépend également de la durée de la partie.

Coup de Départ:

Nous avons implémenté les méthodes isItFirstTurn et isItBlueFirstTurn afin de savoir si nous somme bien en début de partie. Si notre joueur joue les Blancs, il choisira le coup B7-B2. En effet, nous avons déduit après de nombreux tests de parties qu'il s'agit du meilleurs coup de départ car il est ensuite possible de jouer le coup G7-C3 et ainsi placer l'un de ces soldats près de la case centrale.
A l'inverse, si notre joueur joue les bleus, alors il choisira le coup F1-F6 si possible. Sinon, il jouera le coup B1-B6. En effet il y aura forcement l'un de ces deux coup qui sera valide.

Coup de fin de partie:

A chaque fois que notre joueur était sur le point de gagner une partie, celui-ci ne jouais jamais le dernier coup lui permettant de gagner. Nous avons donc implémenté une méthode isWinningMove qui vérifie si un coup fait gagner notre joueur. Nous testons cette méthode sur tous les coups possibles afin que si un coup est gagnant, on puisse immédiatement terminer la partie. Sinon, on applique notre algorithme AlphaBeta normalement.

Gestion du temps réel:
A l'aide d'un compteur, le plateau de jeu enregistre le nombre de coup jouée. Ainsi, pour les trois premier coup, la profondeur d'alphaBeta vaut 4 car il n'y a pas d'intérêt d'avoir une plus grande profondeur en début de partie. De même, au bout de 30 tours, la profondeur est également de 4 car dans ce cas il y a des chances que notre ia soit bloque dans une boucle avec l'adversaire. De plus, il y a un risque de dépasser les 240 secondes de réflexions autorises. Si le nombre de tour se situe entre 3 et 30, alors la profondeur d'AlphaBeta est de 5.

De plus, nous souhaitons que notre joueur joue de manière plus agressive en fin de partie. C'est pourquoi nous augmentons la valeur des coefficients des paramètres de l'heuristique de notre joueur par rapport a ceux correspondant a l'adversaire.

Conclusion:

Nous avons eu des difficultés à faire fonctionner notre joueur avec l'interface du tournois car celui-ci avait souvent tendance a essayer de jouer avec les pièces adverses.
Nous avons également eu du mal à trouver une heuristique qui puisse avoir une stratégie sur le long terme en terme de placements de soldats sur le plateau.
Une amélioration de notre projet pourrait être d'implémenter l'algorithme IDAlphaBeta afin de mieux gérer le temps de réflexion de notre joueur.