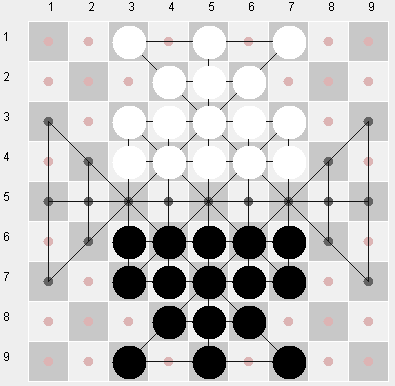
Intelligence Artificielle : 16 Soldats



Sommaire

[Introduction – 16 Soldats 3](#_Toc359584302)

[Présentation du jeu 3](#_Toc359584303)

[Règles du jeu retenues 3](#_Toc359584304)

[Enjeu du projet 4](#_Toc359584305)

[Réalisation du projet 5](#_Toc359584306)

[Mise en œuvre technique 5](#_Toc359584307)

[Mise en œuvre algorithmique 5](#_Toc359584308)

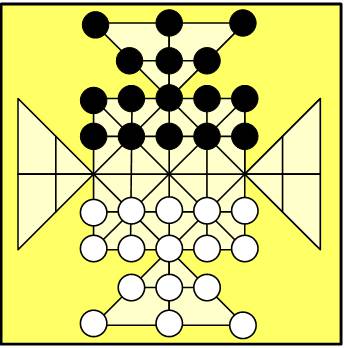
[L’algorithme alphaBeta 5](#_Toc359584309)

[Le calcul de l’heuristique 5](#_Toc359584310)

# Introduction – 16 Soldats

## Présentation du jeu

16 Soldats est un jeu de stratégie sur plateau, à deux joueurs, originaire du Sri Lanka. Il serait aussi partiellement originaire d’Inde, où le jeu est connu sous le nom de « Vaches et Léopards ». *16 Soldats* est similaire à l’Alquerque : les joueurs peuvent utiliser leurs pièces (les soldats) pour sauter par-dessus les pièces adverses et les capturer. Cependant, le jeu se déroule sur une version étendue du plateau de jeu de l’Alquerque : quatre plateaux triangulaires se greffent au plateau principal (carré). Les pièces peuvent de surcroit capturer dans toutes les directions. Chaque joueur dispose de 16 soldats, dotés d’une couleur unique (par joueur).

  
Plateau de jeu « 16 Soldats »

## Règles du jeu retenues

Nous n’avons pas retenu la totalité des règles de 16 soldats pour la réalisation de ce projet. Seules les règles suivantes ont été intégrées :

* Le jeu se joue sur un damier de 9x9 cases, en suivant les lignes de déplacement autorisés (dessinées sur le plateau de jeu).
* La position initiales des pièces est toujours identique (dessinée sur le plateau de jeu).
* Le joueur utilisant la couleur Blanche commence.
* Une pièce est capturée lorsque un pion saute par dessus, mais toujours en suivant la direction initiale du déplacement, en poursuivant la ligne dans la même direction. Le déplacement est donc en ligne droite, de deux segments de la même ligne. On ne peut capturer qu’une seule pièce par tour. Il n’y a pas de prises multiples (comme par exemple on peut en trouver aux dames). Si un pion saute au dessus d’un pion ennemi, ce dernier est obligatoirement capturé.
* La capture de pièce n’est pas obligatoire quand elle est possible.
* Si l’un des deux joueurs ne peut plus jouer, la partie est déclarée nulle.
* La partie est limitée à une durée de 10 minutes.
* La partie est limitée à 200 coups (au total).
* Les colonnes et lignes du plateau de jeu sont numérotées de 1 à 9, (1,1) étant le point supérieur gauche de celui-ci.

## Enjeu du projet

Le but du projet est de développer, en binôme, une intelligence artificielle capable de jouer une partie de 16 Soldats. L’enjeu étant évidemment de développer la meilleure IA possible, afin de remporter le *grand tournoi*. L’IA doit respecter les règles suivantes :

* Ne pas écrire sur le disque dur (lecture autorisée).
* Ne pas effectuer de calculs pendant le tour de l’adversaire.
* Respecter les règles du jeu.

Pour ce faire, nous avons utilisé les notions étudiées en cours, telles que le parcours d’arbre des coups possibles (avec un nombre défini de « coups d’avance »), ainsi que le calcul d’heuristique.

# Réalisation du projet

## Mise en œuvre technique

|  |  |
| --- | --- |
| http://www.digitaltrends.com/wp-content/uploads/2010/11/java-logo.jpg  Java 7 | Conformément aux exigences du projet, nous avons utilisé le langage Java (version 1.7) pour développer notre intelligence artificielle. Le code « squelette » imposé nous proposait une interface IJoueur qui décrivait les méthodes à définir pour développer un joueur virtuel. |

|  |  |
| --- | --- |
| Nous avons créé un dépôt Subversion sur Google Code, que vous pouvez consulter et/ou récupérer à l’adresse suivante :  <http://16-soldats-bro.googlecode.com/svn/trunk/>.  Les principaux avantages d’un gestionnaire de version comme Subversion sont : la facilité à partager du code, à développer en équipe, à créer une sauvegarde du projet à chacune de ses évolutions. | http://www.geckogeek.fr/wp-content/uploads/2009/09/subversion_logo-300x259.png  Subversion |

NP++, JDB. Brasser sur le fait qu’on n’utilisait pas d’IDE et que du coup le debuguage s’est fait en console avec JDB (SUUUUURE :D).

**[…]**

## Mise en œuvre algorithmique

### L’algorithme alphaBeta

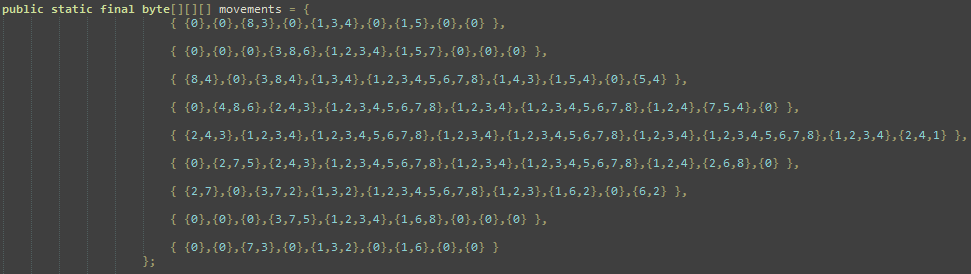
{Récupérer la plus grosse valeur de feuilles… }

### Le calcul de l’heuristique

## Difficultées rencontrées

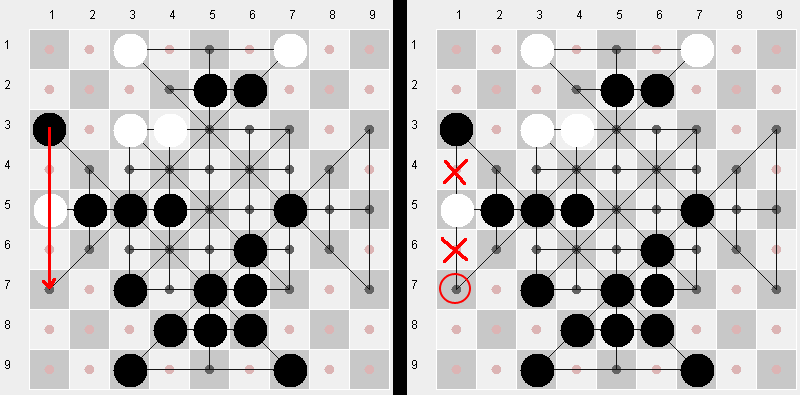
### La légalité des mouvements

Les règles du jeu font que la légalité de certains déplacements des soldats est parfois complexe à établir. C’est notamment le cas des mouvements présentés ci-dessous, qui ont été résolus avec l’écriture de quelques contrôles ; contrôles mis en œuvre à l’aide du tableau à 3 dimensions suivant, décrivant les directions autorisées à partir de chaque case du plateau de jeu :

Tableau 3 dimensions des directions autorisées

#### La capture de pion dans les plateaux triangulaires

Les plateaux triangulaires ont la particularité d’occuper 5 cases du plateau avec leur bord le plus long, pour seulement 3 cases « traversables ». Les soldats ont donc besoin d’avancer de 2 cases pour bouger d’une seule, ou de 4 s’il y a un soldat ennemi à capturer sur le chemin.

**  
Déplacement « doublé » sur les bords des triangles

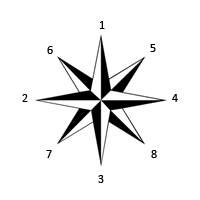
La nécessité de sauter une case supplémentaire (ou deux) est identifiée lors du parcours de chaque direction autorisée par le tableau à trois dimensions :

|  |
| --- |
| nextCol **=** **(**BestSoldier**.**movements**[**i **+** nextCol**][**j **+** nextLine**].**length **==** 1 **?** BestSoldier**.**colMov**[**BestSoldier**.**movements**[**i**][**j**][**k**]-**1**]\***2 **:** nextCol**);**  nextLine **=** **(**BestSoldier**.**movements**[**i **+** nextCol**][**j **+** nextLine**].**length **==** 1 **?** BestSoldier**.**rowMov**[**BestSoldier**.**movements**[**i**][**j**][**k**]-**1**]\***2 **:** nextLine**);** |

Si la case visée (qui est autorisée d’après le tableau 3D) n’autorise aucune direction (elle n’en contient qu’une : « 0 »), alors on sait que l’on doit doubler le déplacement occasionné par cette direction (movements[i][j][k] \* 2). Les déplacements sont stockés en mémoire dans les tableaux colMov et rowMov :

|  |
| --- |
| **// Attention : L'index démarre à 0. colMov[0] correspond au déplacement en colonne du mouvement "1"**  **public** **static** **byte[]** colMov **=** **{** 0**,** **-**1**,** 0**,** 1**,** **+**1**,** **-**1**,** **-**1**,** **+**1**};**  **public** **static** **byte[]** rowMov **=** **{** **-**1**,** 0**,** 1**,** 0**,** **-**1**,** **-**1**,** **+**1**,** **+**1**};** |

Pour rappel, les directions numérotées suivent l’ordre établi par cette rose des vents :



Numérotation des directions dans le programme

Ainsi, colMov[0] rowMov[0] correspond à la direction « 1 », soit : ne pas changer de colonne et remonter d’une ligne. Ce qui explique le déplacement (0, -1).

### L’utilisation de la mémoire vive