upmf Grenoble Université Pierre-Mendès-France

2012

SHS IMSS

Intelligence Artificielle

Le Pentago

Ce projet porte sur la réalisation d'une intelligence artificielle pour le Pentago. C'est un jeu d'alignement utilisant des billes. Ce document présente les choix techniques et la mise en place de l'intelligence artificielle.

Aouizerate Erik, Faucher Julien
MASTER IC²A – spécialité
02/01/2012

Sommaire

1. Introduction

Le Pentago est un jeu reprenant le concept assez classique des alignements. Il peut s'apparenter à un morpion avec d'avantage de cases. Cependant, le nombre de combinaisons est beaucoup plus important. De plus, il nécessite une certaine habilité à manipuler des objets dans l'espace car la structure de la partie se modifie très rapidement.

Nous avons été attirés par ce jeu car il possède des règles simples tout en offrant de nombreuses stratégies. Il est donc accessible mais peut s'avérer d'une grande complexité.

2. Présentation du jeu

2.1. Les Règles du jeu

Le Pentago ce joue à deux joueurs. Il se constitue d'un plateau (6x6), de 18 billes noires et de 18 billes blanches (cf. figure1$$). Le but du jeu est de faire un alignement de cinq billes de sa couleur. Cet alignement peut être vertical, horizontal ou en diagonal. Quatre petits plateaux (3x3) (appelé mini-plateau) divisent le support de jeu. Le premier joueur pose une bille sur le support et tourne un des mini-plateaux d'un quart de tour dans le sens désiré. Il est obligatoire de faire tourner un des mini-plateaux mais celui-ci n'est pas forcement le plateau ou le joueur vient de poser la bille. C'est alors à l'autre joueur de jouer.

2.2. Objectifs et analyse

Nous avons cherché à faire un programme qui joue rapidement afin d'être ludique et agréable pour l'utilisateur. Le programme doit aussi offrir des niveaux de difficultés et des paramètres afin d'augmenter sa durée de vie et son intérêt.

Ce jeu nous a nécessité un certain nombre de partie afin de bien comprendre ses mécanismes. Cette phase d'analyse à été longue mais nous a permis de mieux appréhender son fonctionnement et d'établir des lignes directrices de travail et de recherche. La synthèse de ces ressources nous ont servies tous au long de la réalisation du programme.

Le but de ce programme est d'exploiter au mieux les capacités de l'ordinateur et d'optimisé ses capacités. Nous voulions aussi pouvoir évaluer notre IA. Pour cela, nous en avons crées plusieurs afin de les comparer et d'évaluer leurs avantages et inconvénients.

3. Elaboration du programme

3.1. Les choix techniques

Pour réaliser ce projet nous avons sélectionné le langage JAVA. Nous avons hésité avec PROLOG qui est très adapté à la recherche d'heuristique mais l'implémentation de l'interface de l'application aurait était plus difficile. De plus, JAVA est un langage familier que nous pratiquons régulièrement.

3.2. Représentation du jeu

La représentation du jeu pour l'ordinateur nous a semblée être un aspect très important car il détermine quel calcul l'ordinateur peut effectuer. Nous avons donc choisi de représenter d'une part le plateau. Celui-ci est un tableau d'entier acceptant 3 valeurs (vide, bille noire, bille blanche). D'autre part, nous avons représenté les nœuds constituant l'arbre de recherche. Cet arbre manipule des plateaux. Nous vous invitons à consulter la javadoc pour de plus amples renseignements à ce sujet.

3.3. Heuristique(s)

3.3.1. Détail de l'heuristique

Nous avons réussi à modéliser le but du jeu sous forme d'heuristique. En effet, celle-ci cherche compléter les alignements encore possibles tout en minimisant ceux de l'adversaire.

L'IA dispose d'un tableau recensant tous les alignements du jeu et compare ce tableau au plateau courant. De cette manière, l'IA ne joue pas à un emplacement n'offrant aucune possibilité de gagner ou de ne plus perdre. L'heuristique retourne donc une valeur proportionnelle à l'intérêt du plateau pour l'ordinateur.

L'heuristique fait la somme des billes disposées sur chaque alignement encore possible, puis la somme de tous ces alignements encore possible.

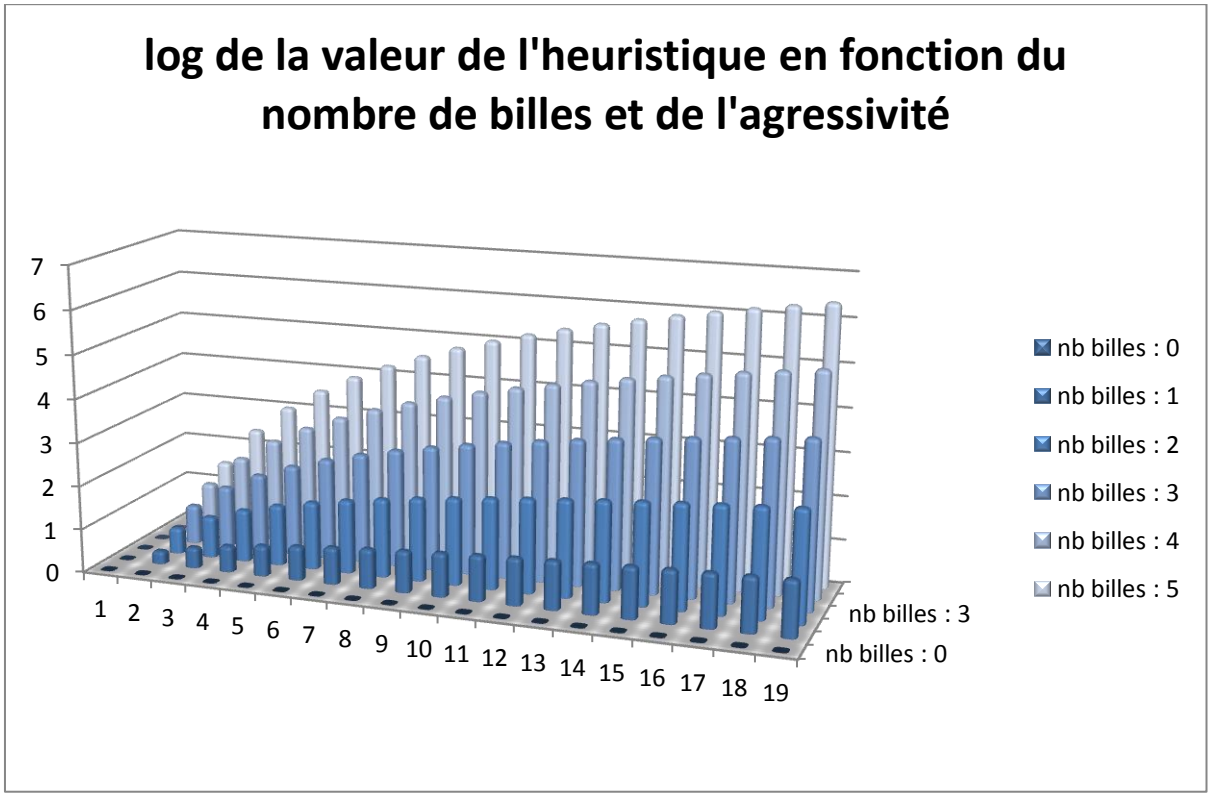
3.3.2. Variantes de l'heuristique

3.3.2.1. Agressivité

La valeur de l'agressivité permet d'élever à la puissance la somme des billes disposées sur un alignement encore possible.

Par exemple un alignement de 3 billes avec une agressivité de 5 donnera une valeur heuristique de 243 (3^5).

Le fait d'élever à la puissance permet de favoriser les alignements déjà commencés.



On peut voir que l'écart type augmente avec l'agressivité.

Ce paramètre permet à l'ordinateur d'effectuer des erreurs et de lui donner ainsi un aspect humain. Avec une agressivité trop faible, l'IA joue « mou » et ne cherche pas réellement à gagner. Avec une agressivité trop forte, l'IA cherche constamment à compléter un alignement et oubli parfois de contrer son adversaire.

Ce paramètre est le point fort de notre programme et permet d'ajuster le tempérament de l'ordinateur.

3.3.2.2. *Heuristique dynamique*

Nous avons introduit deux manières de calculé l'heuristique appelé heuristique et heuristique dynamique.

L'heuristique simple calcule une valeur pour un plateau donné. Ce type de calcul à pour avantage de jouer le meilleur plateau possible.

L'heuristique dynamique calcul une valeur pour un coup joué pour toutes les rotations possibles. Elle effectue en faite la moyenne de l'heuristique des 8 plateaux, au maximum, lié à un coup. Le regroupement de ces 8 plateaux est appelé « super-plateau » dans le programme.
Ce calcul à 2 avantages principaux :

- Il divise par 8, au maximum, le nombre de fils d'un plateau et donc de valeur à comparer. Ceci nous fait gagner du temps pour le calcul du coup à joué.
- Il permet d'évaluer un coup de manière plus général. Une bille ne pourra plus être déplacée au cours de la partie alors que les plateaux pourront toujours être tournés. Cette technique permet de jouer des positions qui sont globalement gagnante offrant d'avantage de positions intéressantes pour l'ordinateur.

3.4. Création de l'arbre de recherche

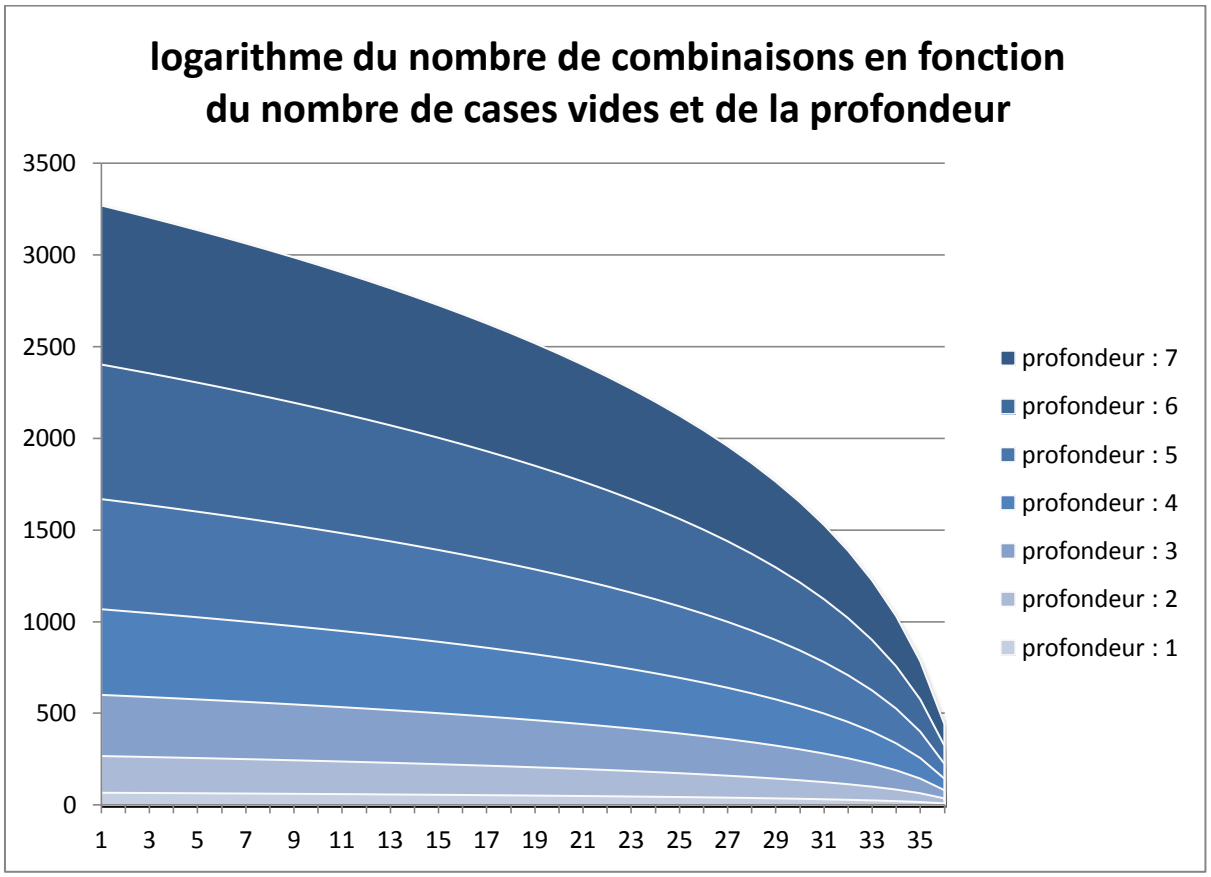
Ce jeu dispose d'un facteur de branchement très élevé. Il y a 8 rotations possibles pour chaque coup que multiplie le nombre de case vide.

Au premier coup il y a donc un facteur de branchement de 288. Ce facteur diminue de 8 en 8 à chaque coup joué.

Notre algorithme de recherche calcule à chaque plateau créé s'il est gagnant ou perdant afin de limiter les branches de l'arbre et de ne pas chercher des positions impossibles.

Nous parcourons l'arbre en profondeur en utilisant l'algorithme de recherche min/max couplé à une coupure alpha/beta.

L'arbre de recherche permet de sélectionné la profondeur de calcule. La profondeur de recherche est un autre paramètre permettant d'ajuster le niveau de difficulté de l'IA. Elle correspond au nombre de coup calculé à l'avance. Par exemple, en profondeur 2, l'IA calcule toute les combinaisons pour le prochain coup qu'elle va jouer puis pour un coup de l'adversaire. La profondeur permet à l'ordinateur d'anticiper les pièges de l'adversaire. C'est un moyen de fixer la complexité du coup joué.



Lors de l'ouverture du programme dans éclipse, le nombre de plateau créés et estimés est indiqué dans la console. Cela permet de vérifier l'avancement de la recherche et se rendre compte du nombre de combinaisons.

3.5. Optimisation

3.5.1. Effectuée

Nous avons cherché à optimiser l'algorithme de recherche. La principale contrainte de recherche étant le nombre de plateau créé par coup joué, nous avons donc diminué ce facteur de branchement.

C'est en début de partie que le nombre de combinaison est le plus important. Cependant, un grand nombre de ces combinaisons sont redondantes. En effet, lorsque qu'une seule bille est joué, la rotation des mini-plateaux vides n'apporte pas de combinaisons pertinentes supplémentaires. La méthode rotationPertinente permet de ne pas créer ces plateaux inutiles.
Le plateau initial est sauvegarder en mémoire et chaque rotation du plateau est comparé au plateau initial, s'il est identique alors il n'est pas ajoutée à l'arbre de recherche. Cette optimisation nous permet de jouer en profondeur 3 des le début de la partie avec un temps de calcul quasi-nul.

L'option profondeur dynamique permet de laisser à l'ordinateur le choix de la profondeur de recherche. Le nombre de combinaison diminuant au fur et à mesure de la partie, cette option à pour but de garantir un temps de recherche acceptable tout en maximisant la profondeur. La profondeur dynamique à été très utile lors de l'évaluation de l'IA.

Nous utilisons la coupure alpha/beta afin de limiter encore plus le nombre de combinaisons calculé. La coupure alpha/beta est d'une très grande efficacité. Elle permet de calculé 1 plateau pour environ 5 estimés.

Enfin, nous avons veillé à limiter autant que possible les fuites mémoires. Cela permet de ne pas encombrer la mémoire de l'ordinateur et d'accélérer sensiblement les temps d'écriture et de lecture.

Ces optimisations nous permettent d'évaluer un grand nombre de combinaisons en en calculant seulement une petite partie. Cela permet de jouer à des profondeurs de recherches élevés sans nécessité de grande durée calcule.

3.5.2. Envisagée

Afin d'améliorer encore plus les temps de calculs, nous aurions pu optimiser directement le code de l'application. Il aurait fallut extraire toutes les conditions des boucles. Cette manipulation rend le code peu lisible mais diminue largement le nombre de conditions. Celles-ci sont testées avant la boucle et non à chaque itération.

Dans le souhait d'une optimisation maximum, nous aurions pu travailler en multithread. Nos ordinateurs actuels étant généralement équipés d'au moins 2 processeurs, nous aurions effectués les traitements en parallèle. Cette démarche modifierait en profondeur l'algorithme de recherche et complexifierait largement notre programme. Cependant, le temps de calcul ne serait pas exactement divisé par le nombre de processeur car en effectuant les traitements en parallèle nous limiterons l'effet de la coupure alpha/beta.

3.6. Evaluation de l'IA

En faisant jouer l'IA contre elle-même nous avons pu trouver sa configuration optimum. Le cas le plus extrême à été de faire joué l'IA en profondeur 4 contre l'IA en profondeur 2. La partie s'est conclue par un match nul. Nous en avons conclu que l'IA ne pouvait pas se piéger elle-même et que même en profondeur 2 elle restait très difficile à battre. Toutes les autres parties avec les différentes configurations (profondeur dynamique, profondeur de recherche et agressivité entre 3 et 9) se sont elles aussi concluent par un match nul.

Nous avons trouvé deux IA du Pentago sur Internet. Notre programme à réussi à les battre en moins de 15 coups. Nous avons nous aussi essayer de battre notre programme mais quasiment toutes les parties se sont soldées par un échec. Sur environ une centaine de parties nous avons gagné deux fois et fait un match nul.

Un des points faibles de l'IA est un coup précis à effectué en début de partie. Il consiste à jouer 3 billes alignées au milieu du plateau de jeu. Si l'ordinateur ne bloque pas cette ligne il a forcément perdu. Ce coup ne peut être décelé qu'en profondeur 4. Malgré toutes les optimisations, le temps de calcul approche des 4 minutes. Cependant, la recherche en profondeur 2 permet de contrer ce coup car il parait trop menaçant pour l'ordinateur. C'est dans ce cas que la profondeur dynamique présente tout son intérêt. Elle va permettre à l'IA de jouer en profondeur 2 durant les 10 premiers coups afin d'écarter toutes combinaisons de ce type avant de jouer de manière plus réfléchit.

3.7. Limite de l'IA

Une des limites de l'IA est de devoir calculé à la fois si un plateau est gagnant puis de calculé son heuristique. Cela est nécessaire mais demande plus de temps de calcul. Cependant, ce calcul étant relativement court, il n'impute pas réellement les performances du jeu.

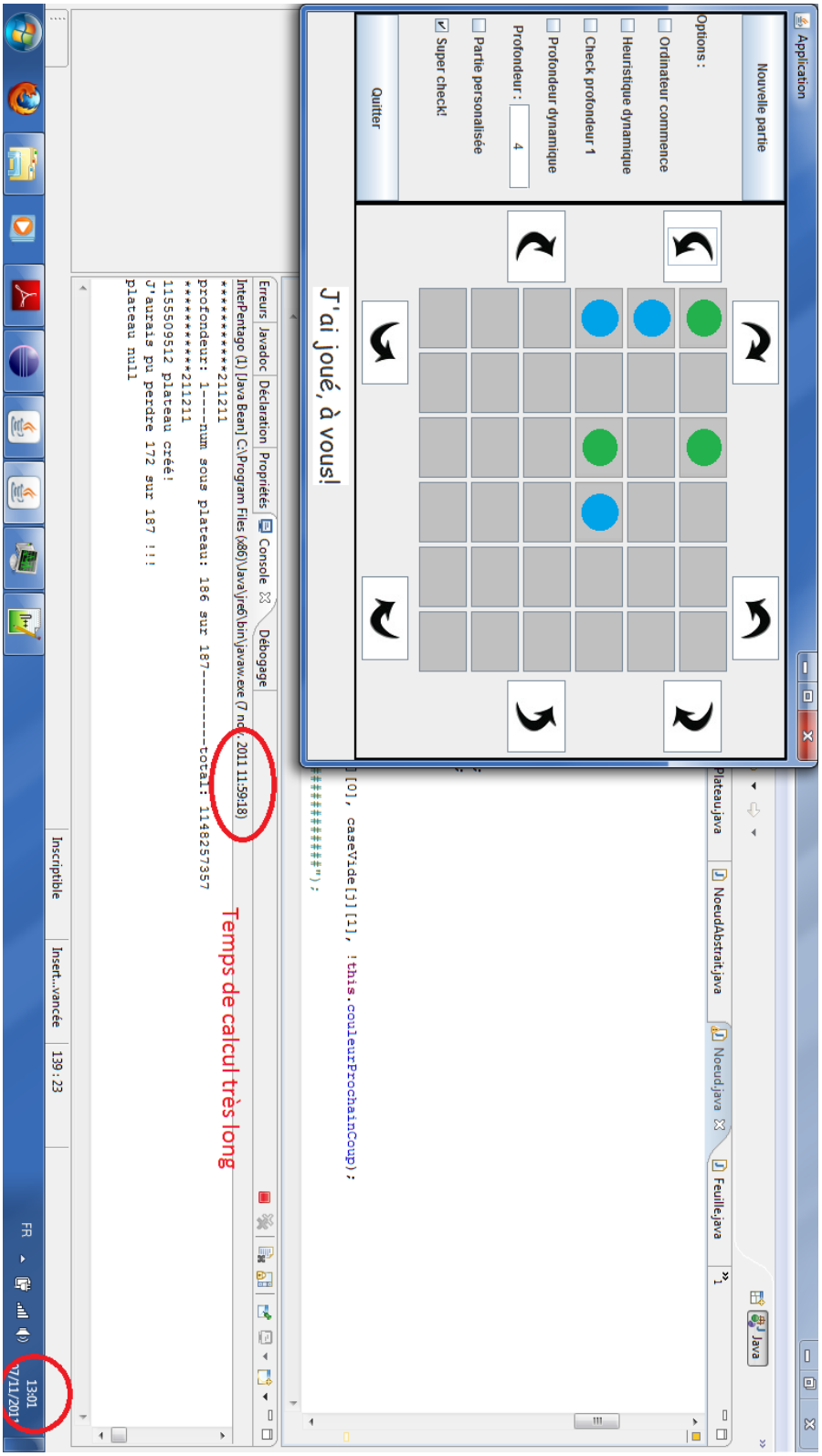
Un problème d'actualisation de la vue du plateau rend les parties difficiles. L'humain n'a pas le temps de voir ce qu'il à joué et ce que l'ordinateur à joué. Les deux modifications du plateau étant dans un listenner, elles sont effectuées à la fin de l'événement. Nous n'avons pas réussi à trouvé une solution simple pour résoudre ce problème.

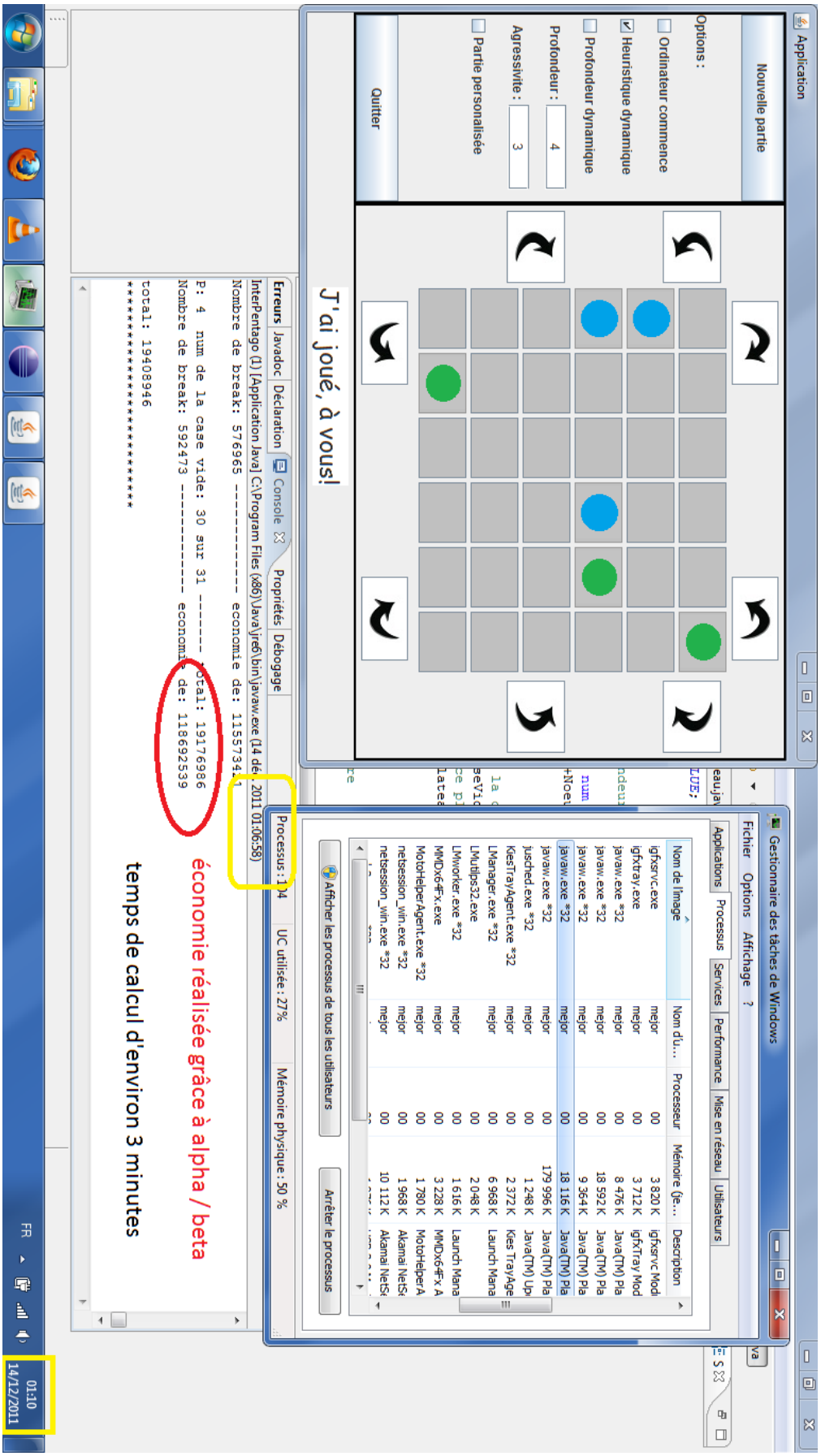
4. Conclusion

Au niveau de l'intelligence artificielle, ce jeu présentait un bon compromis. En effet, face au facteur de branchement, l'ordinateur doit chercher à éliminer le maximum de combinaisons. Devant la modification rapide de la structure du jeu, l'humain doit faire de nombreux effort de visualisation.
Au cours de notre courte vie de programmeur, ce programme est le plus intéressant que nous avons eu à réaliser. Il à fait appel à de nombreuses connaissances et concepts vu au cours de notre scolarité.

Nous avons été réellement séduits par la création d'une intelligence artificielle. La puissance de calcul allié à une prise de décision "intelligente" est passionnant.

Annexe - Images du jeu




Application
Nouvelle partie
Options :
Ordinateur commence
Heuristique dynamique
Profondeur dynamique
Profondeur :
Agressivité :
Partie personalisée
Quitter
J'ai joué, à vous!
Gestionnaire des tâches de Windows
Fichier Options Affichage ?
Applications Processus Services Performance Mise en réseau Utilisateurs
Nom de l'image Nom d'u... Processeur Mémoire (je... Description
Afficher les processus de tous les utilisateurs
Arrêter le processus
Processus : 114 UC utilisée : 27% Mémoire physique : 50 %
Erreurs Javadoc Déclaration Console Propriétés Débogage
InterPentago (1) [Application Java] C:\Program Files (x86)\Java\jre6\bin\javaw.exe (14 déc. 2011 01:06:58)
Nombre de break: 576965 ---------- economie de: 1155734
P: 4 num de la case vide: 30 sur 31 ---------- total: 19176986
Nombre de break: 592473 ---------- economie de: 118692539
total: 19408946
économie réalisée grâce à alpha / beta
temps de calcul d'environ 3 minutes
FR
01:10
14/12/2011