

Projet

Puissance 4

Alexandre AUDA
&
Rémy AUDA

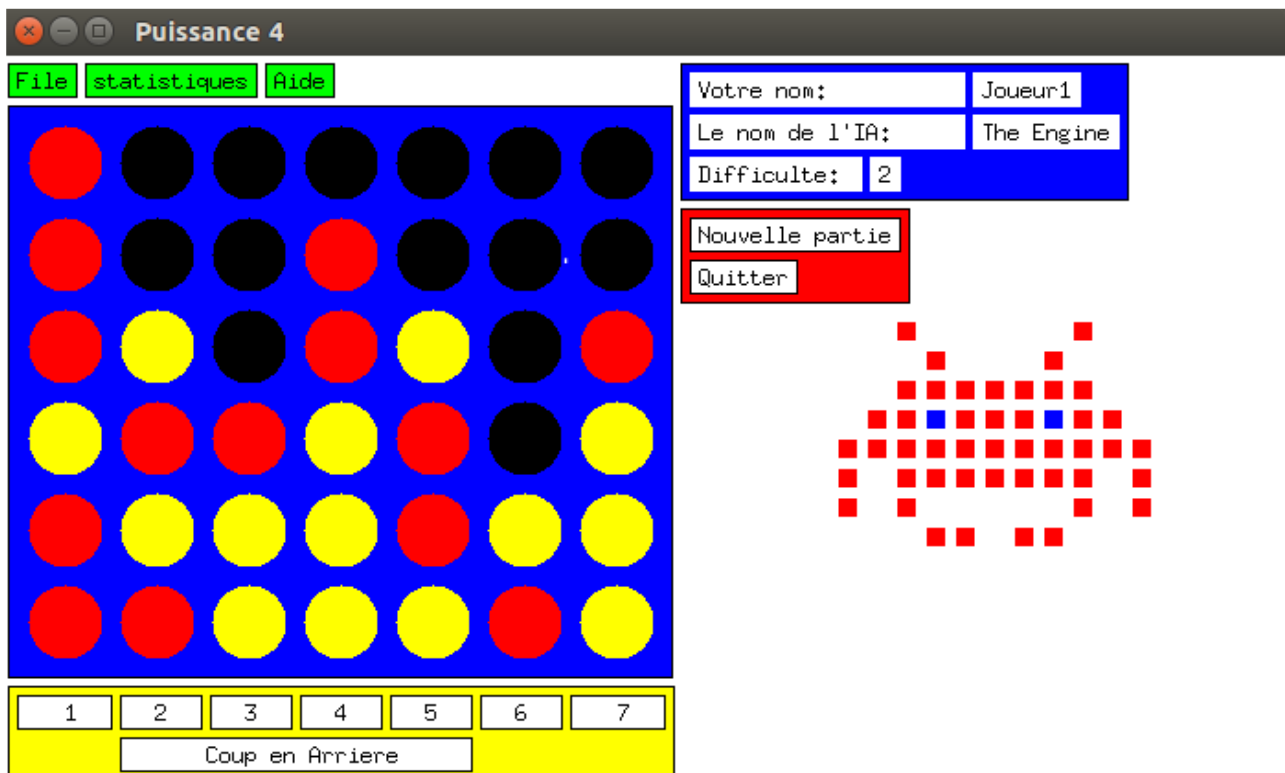


Table des matières

Projet	1
Puissance 4	1
Introduction:	2
1. Compilation et exécution:	2
2. Déroulement d'une partie:	3
2.1-Règles du jeu d'un Puissance 4:.....	3
3. Présentation de l'interface du projet:	4
3.1-Interface du jeu et déroulement d'une partie:.....	4
3.2-Pixels fantômes:	13
4. Intelligence artificielle:.....	13

Introduction:

Dans ce projet, l'objectif est d'implémenter en C le célèbre jeu de Puissance4 à l'aide de la bibliothèque libsx.

La bibliothèque ne distribuant que l'OS Linux, ce projet devra se compiler exclusivement sous une distribution Linux, à moins d'adapter la bibliothèque libsx pour compiler sous une autre distribution.

1. Compilation et exécution:

Tout d'abord, il s'agit d'installer la bibliothèque libsx. Pour cela, nous rappelons qu'il faut être sous une distribution Linux à moins d'adapter la bibliothèque libsx pour compiler sous une autre distribution.

Pour installer la bibliothèque libsx sous une distribution Linux, il suffit de saisir la commande suivante:

```
sudo apt-get install libsx-dev libsx0
```

La compilation se fera à l'aide d'une invite de commande.

- Tout d'abord, se placer à la racine du Projet

```
auda@auda-VirtualBox:~$ cd Documents/C-C++/Puissance4
```

- Puis pour compiler, saisissez la commande:

```
gcc -ansi -pedantic -Wall -o main main.c -lsx
```

```
auda@auda-VirtualBox:~/Documents/C-C++/Puissance4$ gcc -ansi -pedantic -Wall -o  
main main.c -lsx
```

- Enfin, pour exécuter, saisissez la commande:

```
./main
```

```
auda@auda-VirtualBox:~/Documents/C-C++/Puissance4$ ./main
```

2. Déroulement d'une partie:

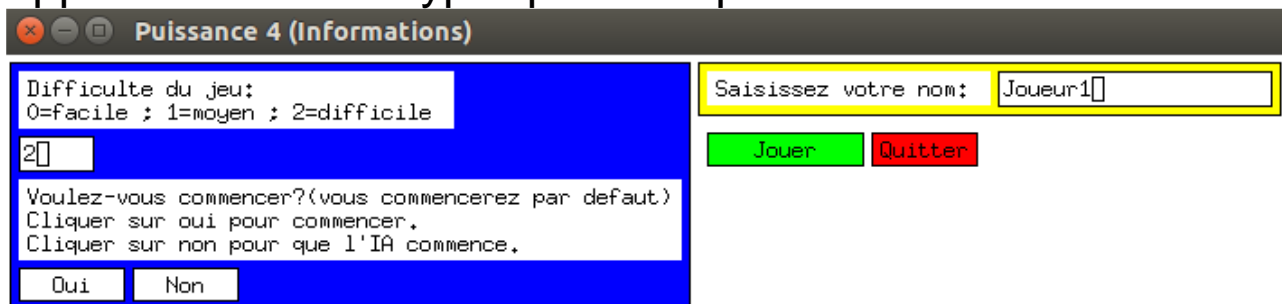
2.1-Règles du jeu d'un Puissance 4:

Le but du jeu est d'aligner 4 pions sur une grille comptant 6 rangées et 7 colonnes. Chaque joueur dispose de 21 pions d'une couleur (par convention, en général jaune ou rouge). Tour à tour les deux joueurs placent un pion dans la colonne de leur choix, le pion coulisse alors jusqu'à la position la plus basse possible dans ladite colonne à la suite de quoi c'est à l'adversaire de jouer. Le vainqueur est le joueur qui réalise le premier un alignement (Horizontal, vertical ou diagonal) d'au moins quatre pions de sa couleur. Si, alors que toutes les cases de la grille de jeu sont remplies, aucun des deux joueurs n'a réalisé un tel alignement, la partie est déclarée nulle.

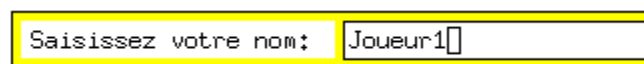
3. Présentation de l'interface du projet:

3.1-Interface du jeu et déroulement d'une partie:

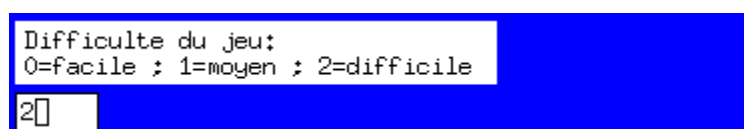
Tout d'abord, à l'ouverture du jeu une première fenêtre apparaît du même type que celle présente ci-dessous.



Dans la première zone d'écriture en haut à gauche de cette fenêtre, le joueur peut saisir son nom qui sera automatiquement pris en compte dans la fenêtre de jeu. Par défaut, c'est le nom «Joueur1» qui sera pris en compte.

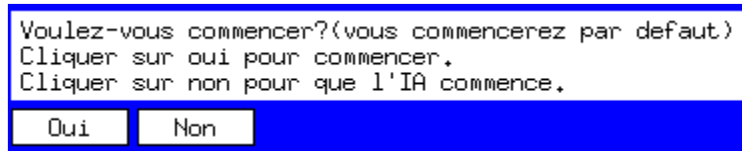


A gauche, l'utilisateur est invité à choisir le niveau de difficulté du jeu. Comme il est indiqué, 0 fera lancer le jeu en mode facile, 1 fera lancer le jeu avec une difficulté moyenne. Enfin, 2 lancera le jeu en mode difficile. Par défaut, le jeu se lancera en mode difficile. En fait, ce paramètre définit en réalité l'IA (intelligence artificielle) que nous détaillerons un peu plus loin.



Enfin, dans la partie en bas à gauche, l'utilisateur a la possibilité de choisir si il veut commencer à jouer ou non. Le processus de ce dernier paramétrage et gère à l'aide de

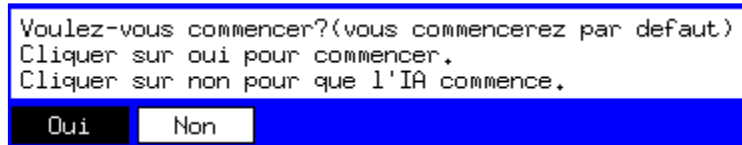
Toggle comme explique dans la documentation de la bibliothèque Libsx.



Voulez-vous commencer?(vous commencerez par default)
Cliquer sur oui pour commencer.
Cliquer sur non pour que l'IA commence.

Oui Non

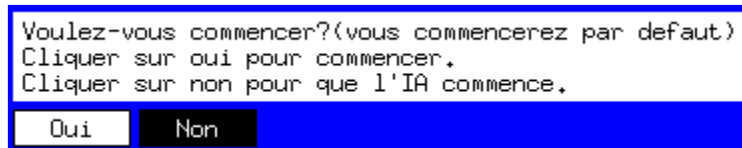
L'utilisateur commence:



Voulez-vous commencer?(vous commencerez par default)
Cliquer sur oui pour commencer.
Cliquer sur non pour que l'IA commence.

Oui Non

L'IA commence:



Voulez-vous commencer?(vous commencerez par default)
Cliquer sur oui pour commencer.
Cliquer sur non pour que l'IA commence.

Oui Non

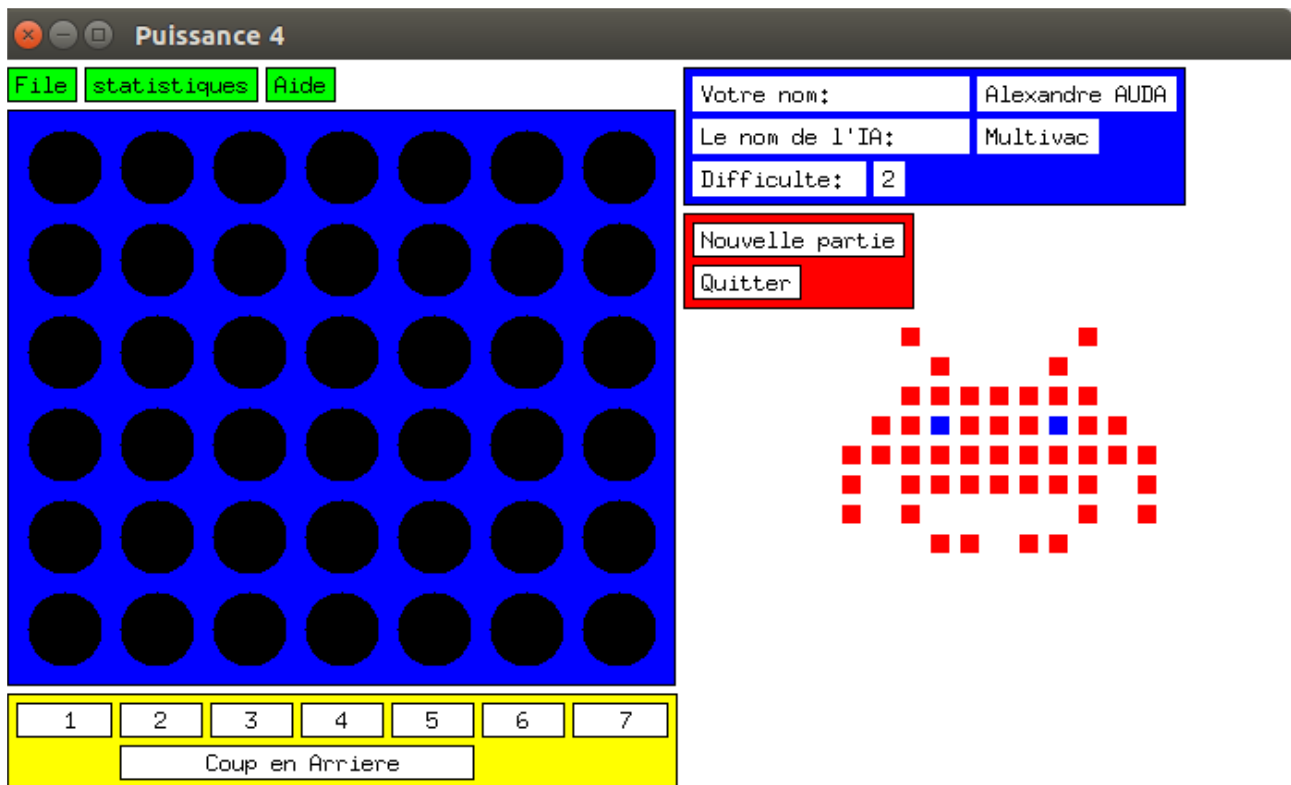
Par défaut, ce sera l'utilisateur qui commencera à jouer.
Enfin, une fois que l'utilisateur a saisi tous les paramètres
comme il l'entendait, l'utilisateur peut commencer à jouer en
cliquant sur le bouton <<Jouer>> en vert.



Jouer Quitter

Si l'utilisateur se ravise et souhaite quitter le jeu, il peut le
faire en appuyant sur le bouton <<Quitter>> en rouge.
Ainsi, si l'utilisateur quitte le jeu, la fenêtre se fermera.

Si au contraire l'utilisateur commence à jouer, la première
fenêtre se fermera et la grille de jeu apparaîtra comme si
dessous:



Ici, nous avons lancé le jeu en mode par défaut mise à part la saisi du nom. On peut donc remarquer que tous les paramètres ont été pris en compte.

De plus, juste en dessous du nom du joueur se trouve le nom de son adversaire qui change à chaque partie aléatoirement.

Quelques exemples:

Le nom de l'IA: The Ox

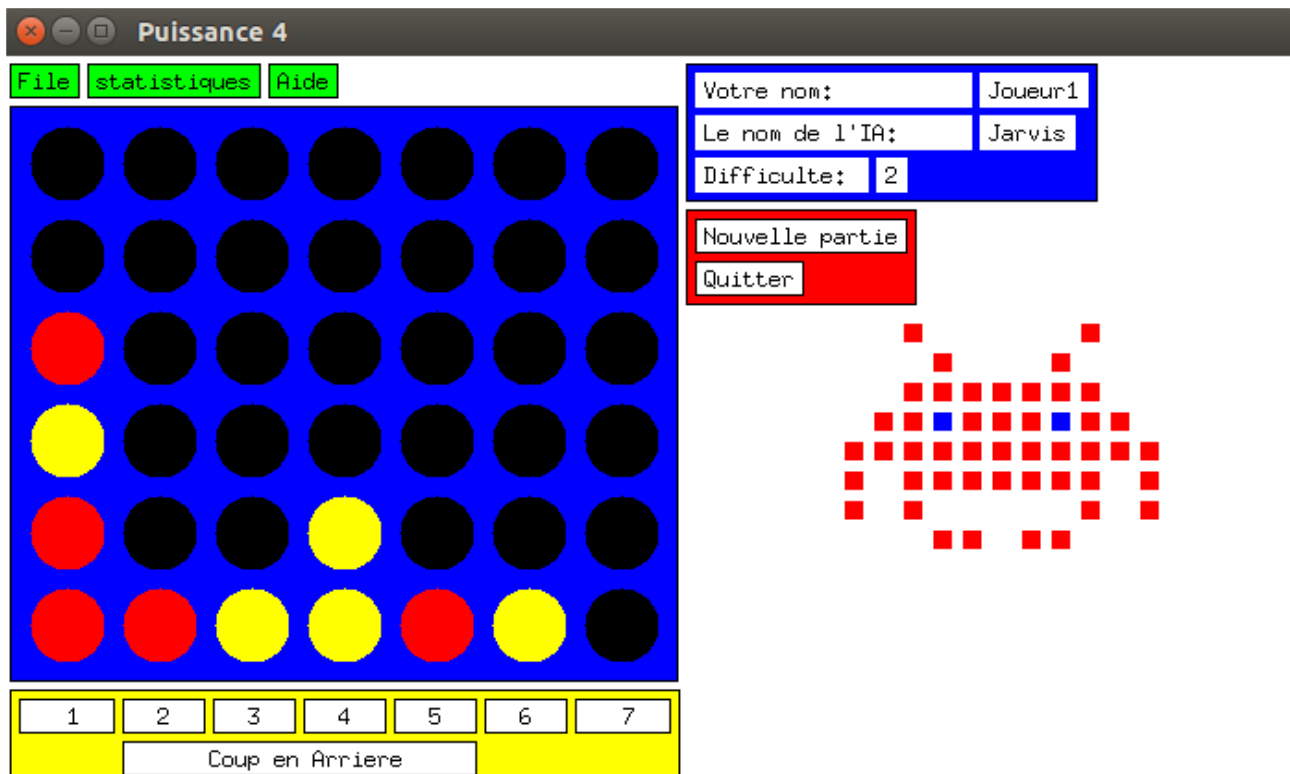
Le nom de l'IA: matrix

Le nom de l'IA: HAL 9000

Début du jeu:

L'utilisateur commence à placer ses jetons dans les colonnes qu'il veut. Après que le joueur est place son jeton, l'IA joue automatiquement après lui.

Malheureusement, le bouton «coups en arrière» est inactif par manque de temps et il aurait dû effectuer un retour en arrière en annulant la dernière action du joueur.

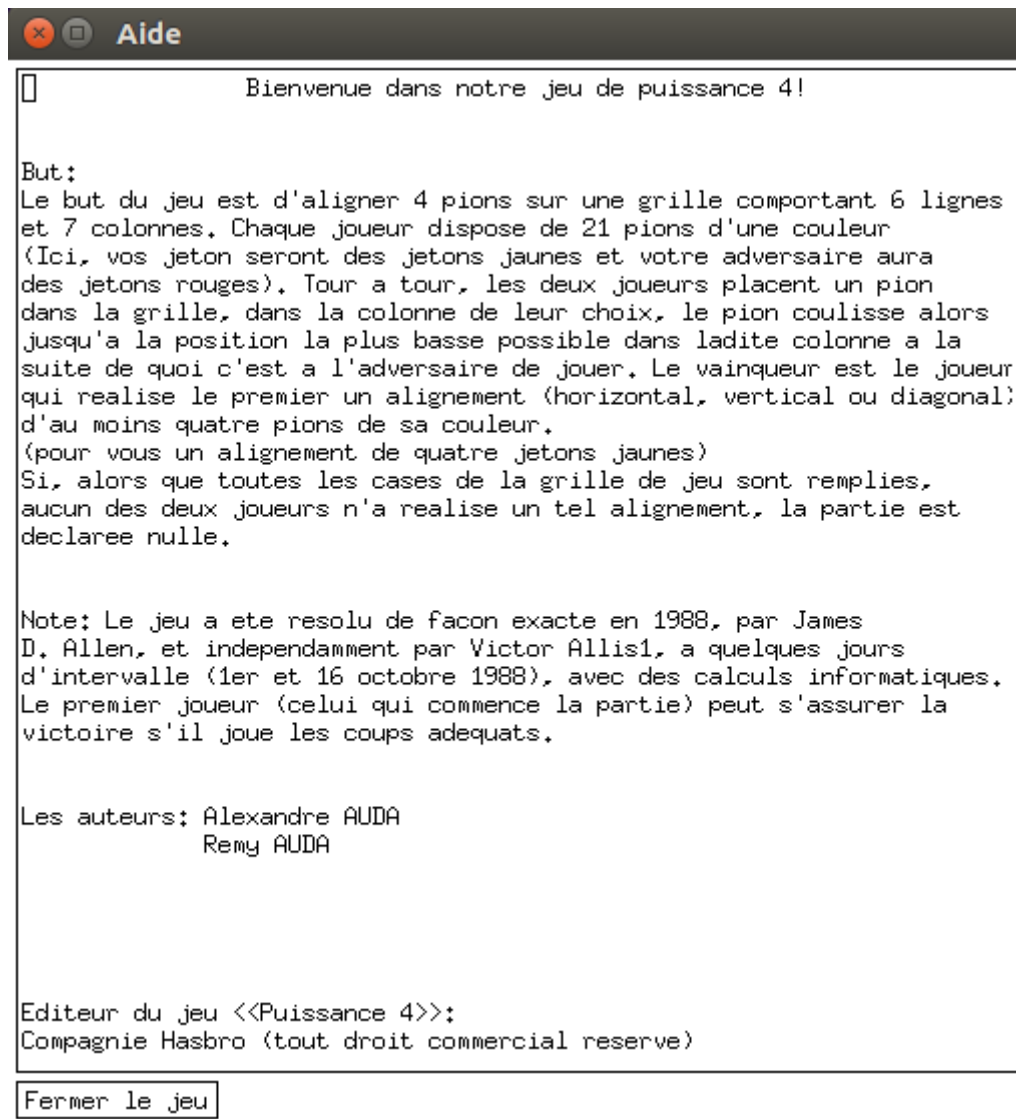


En haut, l'utilisateur peut cliquer sur le déroulant «File» dans lequel se trouvent les onglets: Load, Save et quit. Malheureusement, par manque de temps, les onglets Load et save ne sont pas actif.

En revanche, l'onglet quit permet à l'utilisateur de quitter le jeu.

A cote, se trouve l'onglet «Statistique» dans lequel se trouve l'onglet «Stats des nombres de coups avant fin partie» qui invite l'utilisateur à regarder dans la console sur laquelle est affichée le nombre de coups des parties gagnantes avant que le joueur ne gagne.

A droite se trouve le bouton «Aide» qui lorsque le joueur appui dessus, fait apparaitre une fenêtre expliquant les règles du jeu et les informations complémentaires concernant le jeu.

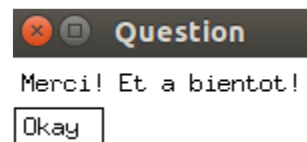
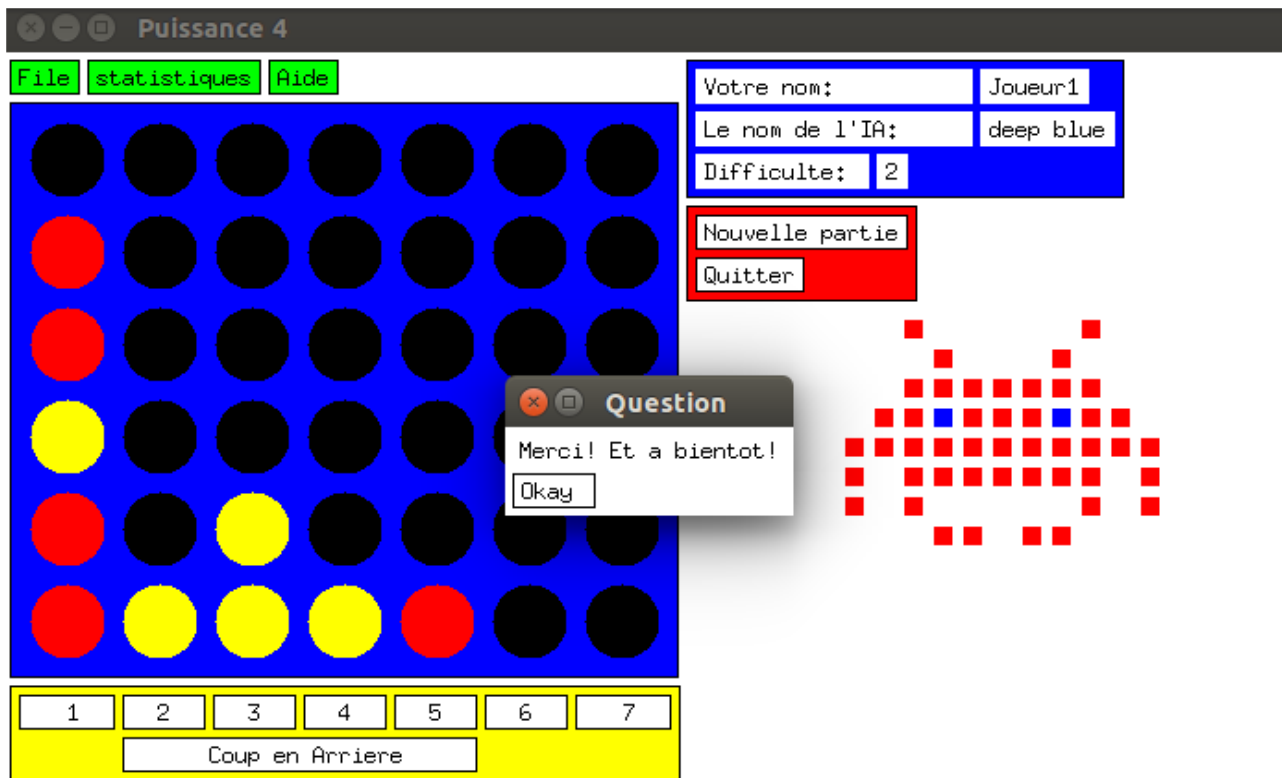


De plus, l'utilisateur peut choisir de recommencer une partie grâce au bouton «Nouvelle partie». Ce bouton ferme la fenêtre courante et re-ouvre une nouvelle fenêtre avec la grille vide.

Nouvelle partie

Juste en dessous, se trouve le bouton «Quitter» qui lorsqu'on l'actionne, fait apparaitre un message de remerciement et quitte le jeu:

Quitter



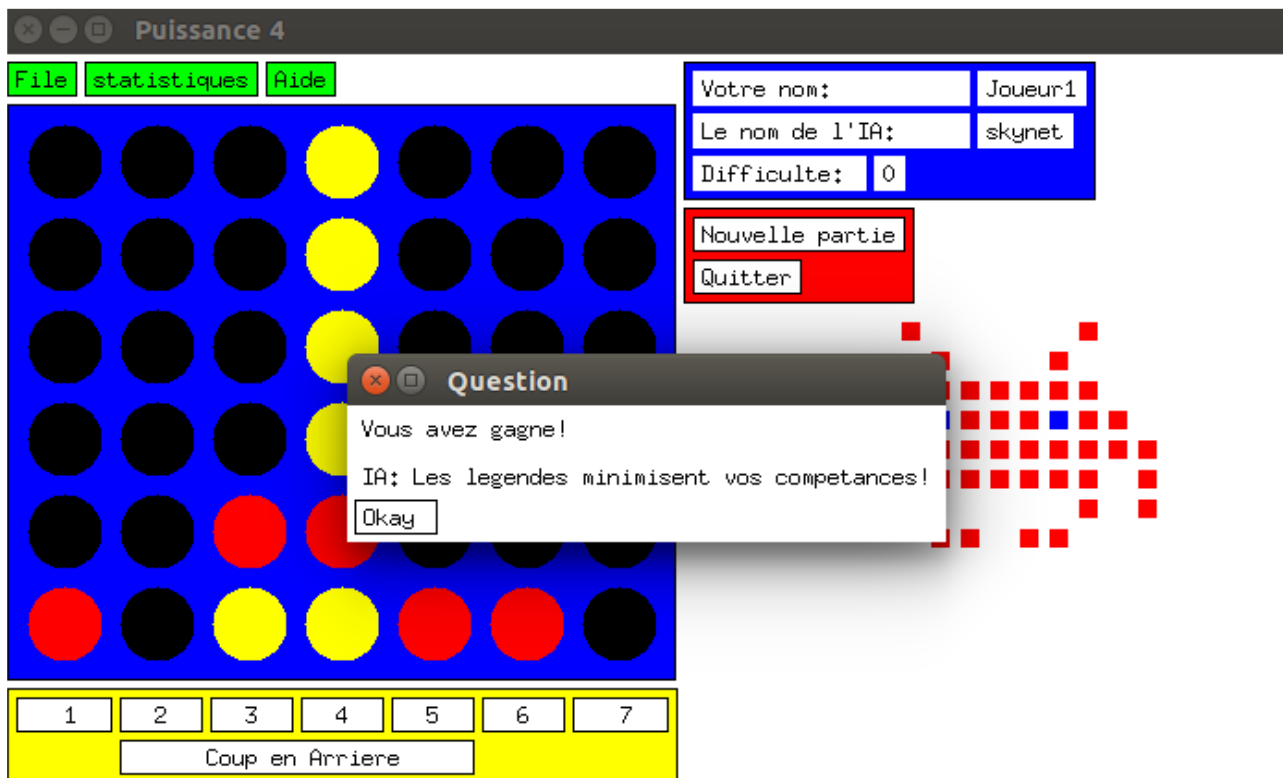
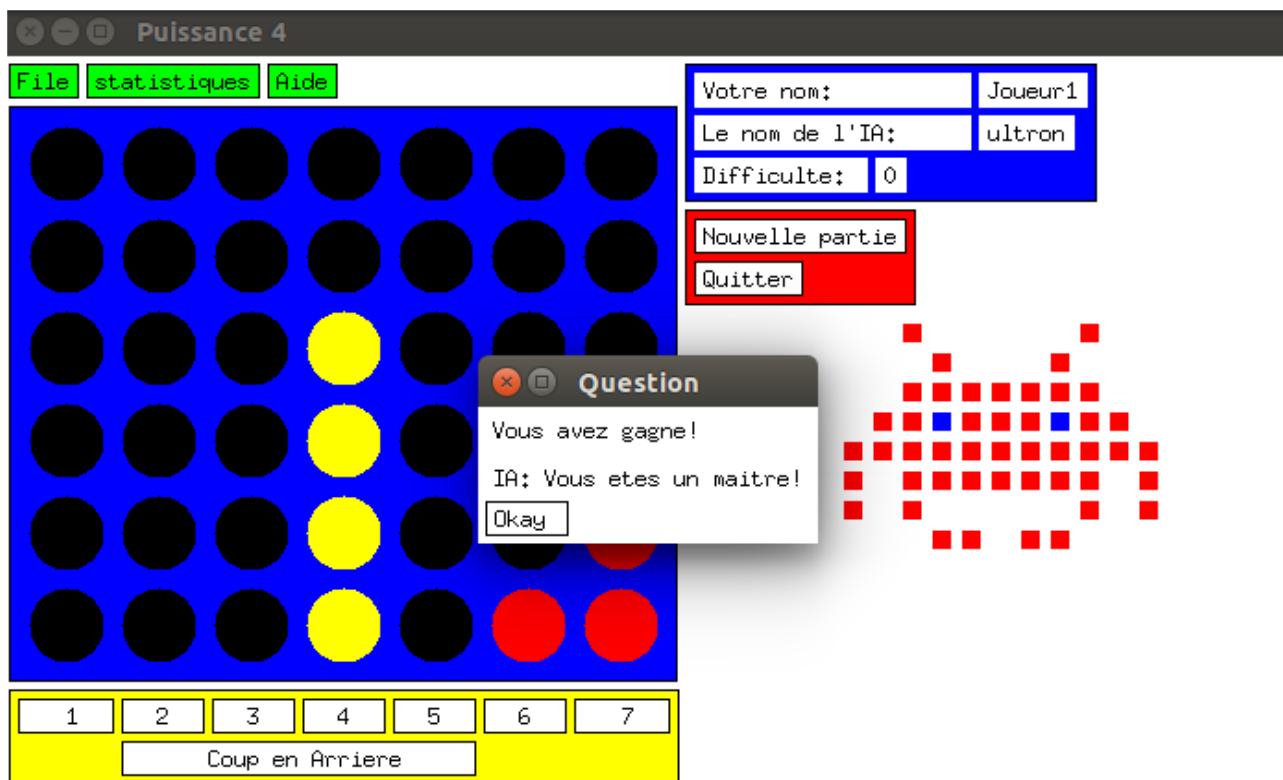
-Fin de partie:

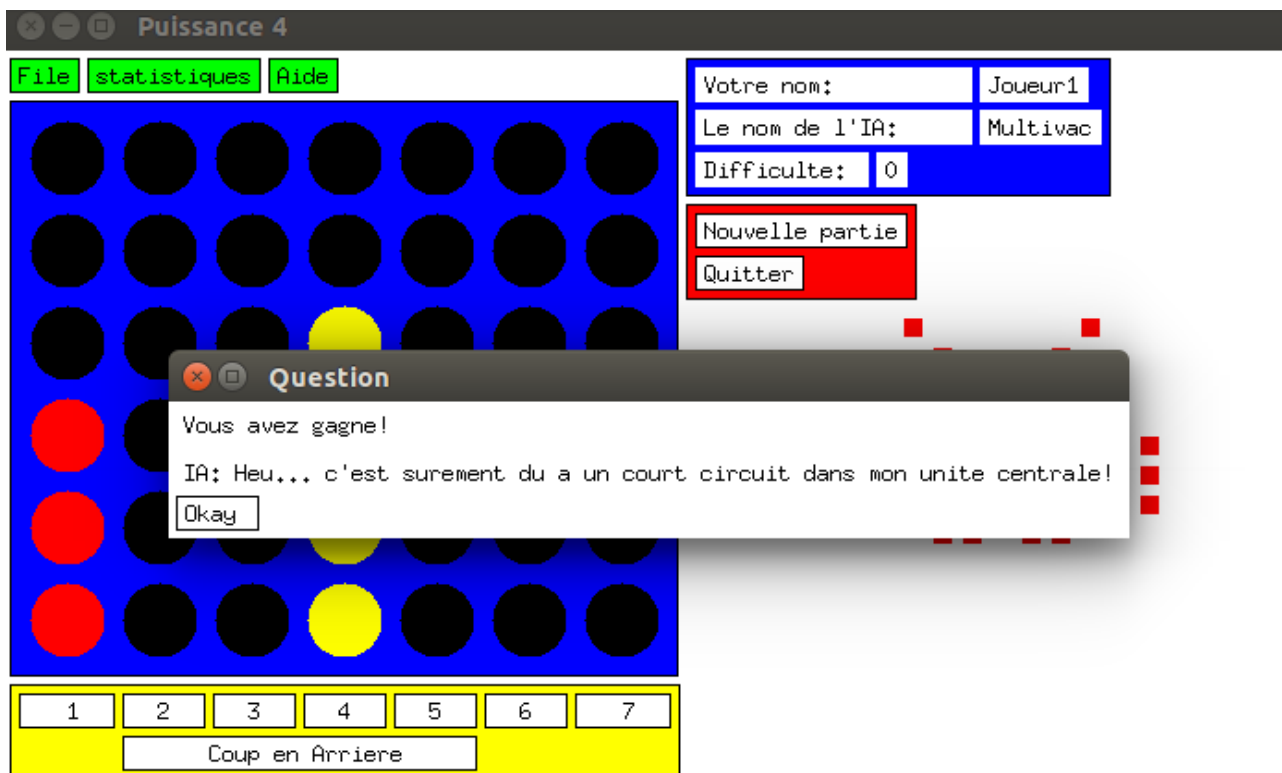
Cas ou le joueur gagne:

Lorsque le joueur gagne, un message apparait en indiquant au joueur qu'il a gagné.

De plus, l'IA émet une phrase aléatoire congratulant le joueur:

Quelques exemples:



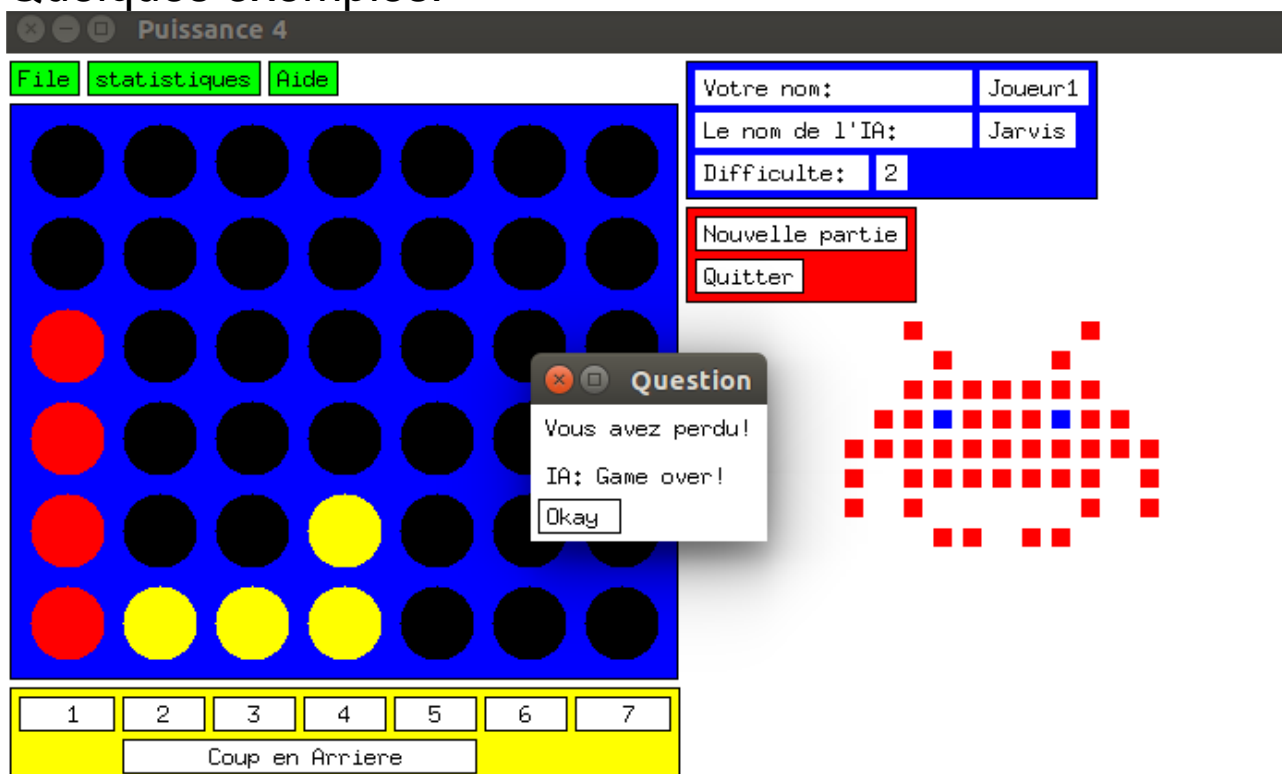


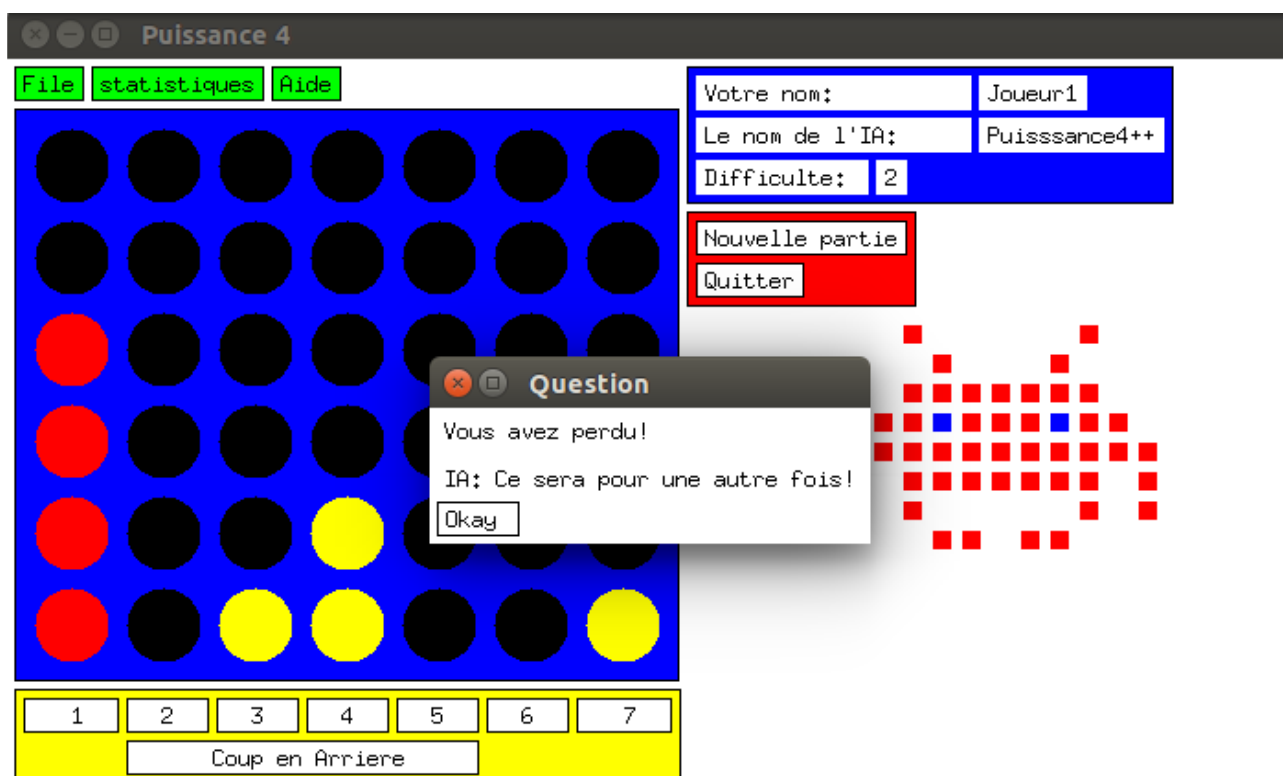
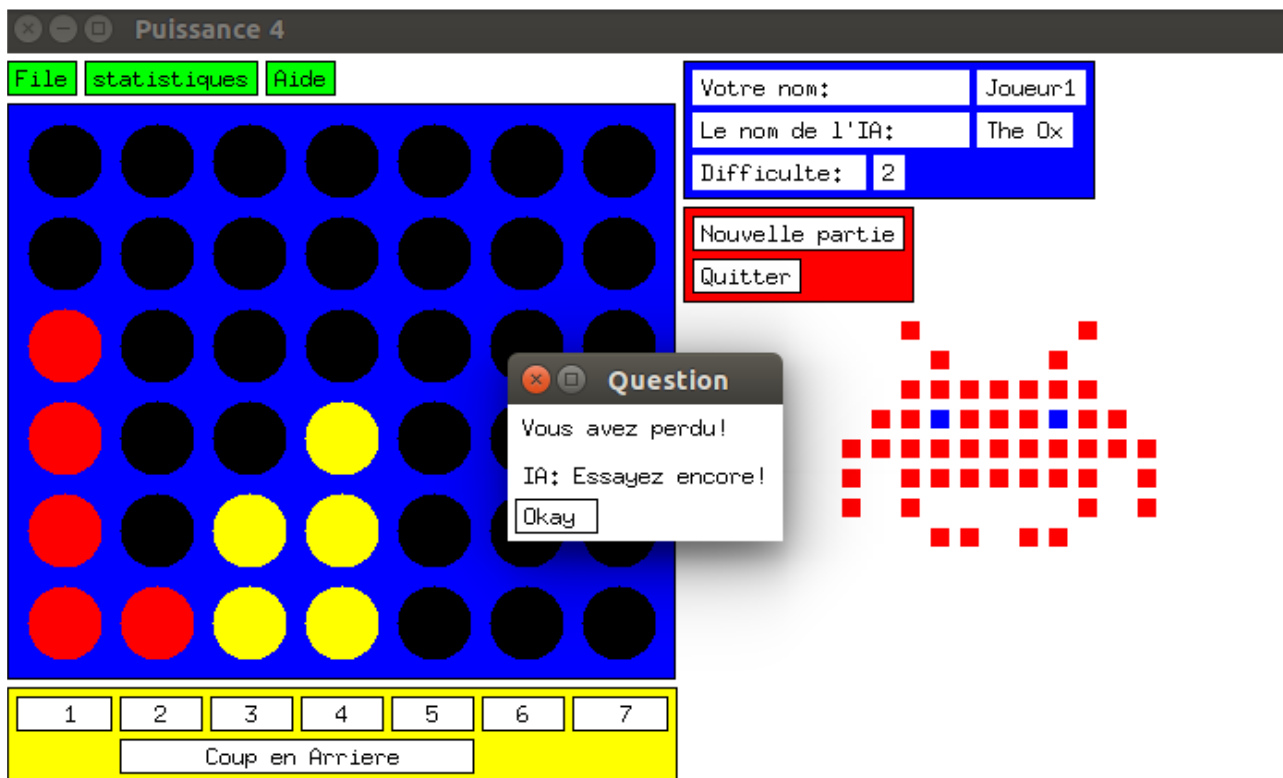
Cas ou le joueur perd:

Lorsque le joueur perd, un message apparait en indiquant au joueur qu'il a perdu.

De plus, l'IA émet une phrase aléatoire raillant ou remerciant le joueur de sa participation:

Quelques exemples:



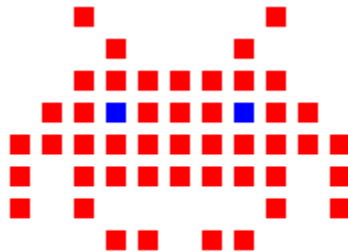


Cas de Match nul:

Lorsque qu'il y a un match nul un message apparait en indiquant au joueur qu'il y a match nul.

De plus, l'IA émet une phrase aléatoire invitant le joueur à refaire une partie ou remercie le joueur de sa participation.

3.2-Pixels fantômes:



Crée à des fins purement décoratives, il permet d'égayer le côté droit de la fenêtre de jeu qui était un peu vide.

A terme, l'objectif final aurait été de l'animer et de faire apparaître d'autres motifs.

4. Intelligence artificielle:

Lors du déroulement de la partie le joueur joue contre l'ordinateur.

Or, cette opération est réalisée à l'aide de l'implémentation préalable d'une intelligence artificielle.

En fait, lors du choix de la difficulté sur la première fenêtre, le joueur choisit d'utiliser une IA (intelligence artificielle) différente.

Ainsi, les niveaux correspondent à des IA définies comme suit:

- Niveau 0: L'IA joue totalement aléatoirement.
- Niveau 1: L'IA joue des coups aléatoires et d'autre réfléchit comme ils le seront au niveau 2. L'IA choisit

donc aléatoirement s'il joue un coup aléatoire ou réfléchit.

NOTE: L'utilisateur peut voir si l'IA a joué de manière aléatoire ou bien réfléchit en regardant en direct sur la console.

- Niveau 2: L'IA joue de manière réfléchit en utilisant un algorithme Min-Max avec élagage alpha-beta (gain en complexité temporelle).

L'Intelligence Artificielle:

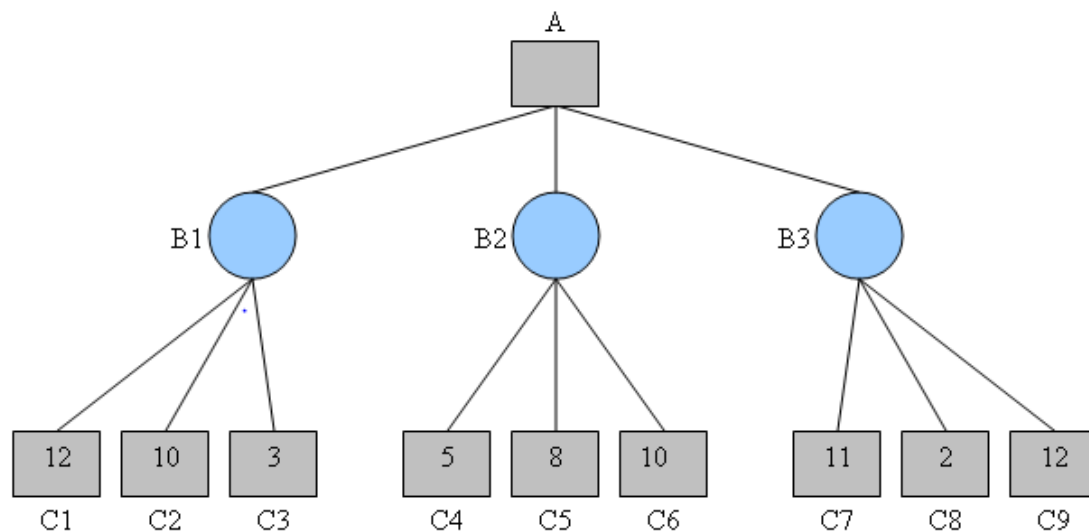
Lors de la programmation de notre puissance 4, nous avons été amenés à créer une intelligence artificielle. En effet, nous voulions que l'utilisateur de notre programme puisse jouer contre l'ordinateur, mais que cet ordinateur joue de façon assez intelligente pour offrir une résistance suffisante pour mettre l'utilisateur de notre programme en difficulté. Pour mettre en œuvre cette intelligence artificielle, nous avons implémenté l'algorithme **du minimax avec l'élagage alpha-béta**.

L'algorithme minimax est un algorithme qui s'applique à la théorie des jeux pour les jeux à deux joueurs à sommes nulles: (un jeu de somme nulle est un jeu où la somme des gains de tous les joueurs est égale à 0. Cela signifie donc que le gain de l'un, constitue obligatoirement une perte pour l'autre). L'algorithme minimax doit s'effectuer également sur un jeu à informations complètes, c'est à dire que chaque joueur connaît l'intégralité du jeu de son adversaire. Ces deux conditions sont remplies pour le jeu de puissance 4.

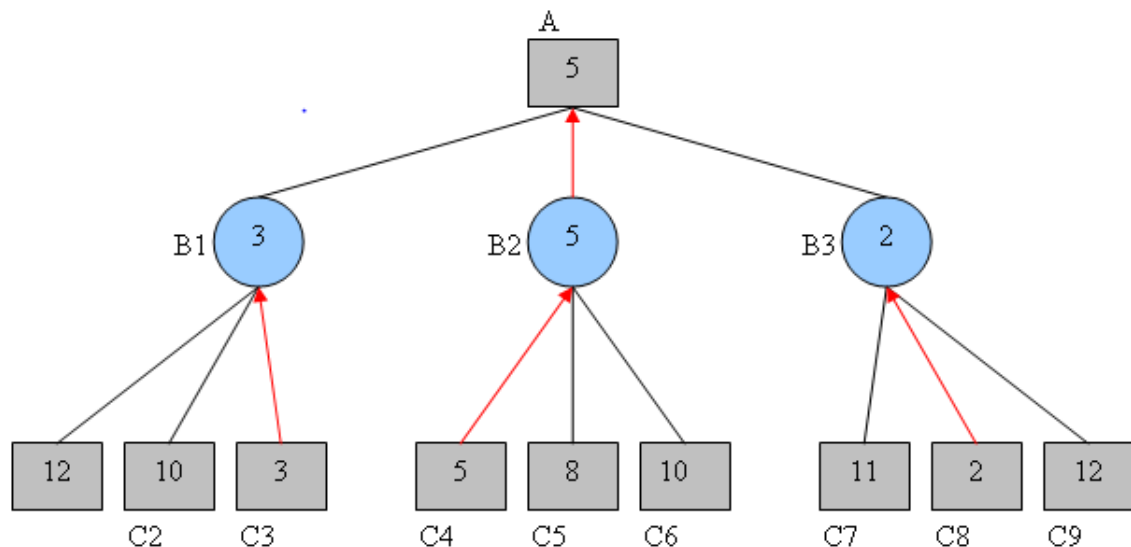
Il existe différents algorithmes basés sur Minimax permettant d'optimiser la recherche du meilleur coup en

limitant le nombre de nœuds visités dans l'arbre de jeu, le plus connu est l'élagage alpha-beta.

En effet, en pratique, l'arbre des coups possibles est souvent trop vaste pour pouvoir être intégralement exploré. Seule une fraction de l'arbre est alors explorée ce qui permet d'accélérer les calculs du meilleur coup possible.



Dans le schéma ci-dessus, les nœuds gris représentent les nœuds joueurs (c'est à dire les coups possibles pour le joueur) et les bleus les nœuds opposants (c'est à dire les coups possibles de l'adversaire). Pour déterminer la valeur du nœud A, on choisit la valeur maximum de l'ensemble des nœuds B (A est un nœud joueur). Il faut donc déterminer les valeurs des nœuds B qui reçoivent chacun la valeur minimum stockée dans leurs fils (nœuds B sont opposants). Les nœuds C sont des feuilles, leur valeur peut donc être calculée par la fonction d'évaluation. Cette fonction d'évaluation a pour but d'analyser une situation et de déterminer s'il s'agit d'une situation favorable ou non. On peut alors remonter l'arbre afin d'assigner une valeur à chaque nœud:



Ainsi, sur l'exemple ci-dessus, on voit que l'algorithme va jouer en B2 car il s'agit du coup qui possède la plus grande valeur assignée: c'est donc le coup le plus pertinent à jouer. En observant l'arbre, on comprend bien que l'algorithme considère que l'opposant va jouer de manière optimale: il prend le minimum. Sans ce prédicat, on choisirait le nœud C1 qui propose le plus grand gain et le prochain coup sélectionnerait amènerait en B1. Mais alors on prend le risque que l'opposant joue C3 qui propose seulement un gain de 3. Cependant, on vérifie bien que l'intelligence artificielle joue des coups cohérents avec cet algorithme.

Cependant, l'algorithme présenté ci-dessus (l'algorithme du Minimax) suppose que l'on parcourt la totalité de l'arbre de jeu, ce qui est très long, et ceci peut ralentir le rythme du jeu.

Ainsi, nous avons codé l'algorithme du minimax que nous avons présenté ci-dessus pour que l'ordinateur joue des coups cohérents.

Mais nous avons de plus apporté une optimisation à cet algorithme: l'élagage alpha-beta, qui permet accélérer la recherche du meilleur coup.

L'algorithme minimax effectue en effet une exploration complète de l'arbre de recherche jusqu'à un niveau donné,

alors qu'une exploration partielle de l'arbre est généralement suffisante: lors de l'exploration, il n'est pas nécessaire d'examiner les sous arbres qui conduisent à des configurations dont la valeur ne contribuera pas au calcul du gain à la racine de l'arbre. L'élagage α - β nous permet de réaliser ceci. Plus simplement, l'élagage α - β évite d'évaluer des nœuds dont on est sûr que leur qualité sera inférieure à un nœud déjà évalué, il permet donc d'optimiser grandement l'algorithme minimax en temps de recherche sans en modifier le résultat.

Nous avons donc implémenté cet algorithme du minimax avec l'optimisation alpha-beta pour coder notre intelligence artificielle ce qui nous permet d'offrir une résistance satisfaisante aux utilisateurs de notre jeu.

Remarque: Il existe d'autres améliorations que l'on peut implémenter pour améliorer la qualité de jeu de notre intelligence artificielle. Nous avons cependant choisi de ne pas les implémenter en raison du temps trop court dont nous disposons pour réaliser de telles optimisations. Les principales optimisations sont l'algorithme du négascout et l'algorithme du MTD-F.

Ce dernier est l'algorithme le plus performant des algorithmes basés sur le principe du minimax. Il reprend en effet les idées du minimax mais possède une mémoire ce qui permet à l'algorithme de réduire au maximum les branches de l'arbre de jeu à calculer et donc il réduit au maximum les temps de calculs.

De plus, même si ces algorithmes de recherches dans un graphe sont performants, il faut remarquer qu'il existe une solution optimale pour le cas particulier du puissance 4. Le jeu a été résolu de façon exacte en 1988, par James D. Allen, et indépendamment par Victor Allis, à quelques jours d'intervalle avec des calculs informatiques. Le premier

joueur (celui qui commence la partie) peut s'assurer la victoire s'il joue les coups adéquats.

Le seul premier coup gagnant est celui dans la colonne centrale.

Un premier coup dans les colonnes adjacentes permet au second joueur d'obtenir une partie nulle, et un premier coup dans l'une des quatre autres colonnes extérieures permet au second joueur de remporter la victoire (à condition qu'il joue parfaitement).

Cependant, l'algorithme du minimax avec élagage alpha-beta constitue un algorithme suffisamment performant pour offrir une résistance satisfaisante au joueur. L'algorithme que nous avons implémenté effectue une recherche du meilleur coup sur un arbre de profondeur égale à 8 (l'intelligence artificielle calcule donc 8 coups à l'avance).