
~~ Rapport de projet ~~

IA pour le jeu d'Othello

Semestre 5

Algorithmique Avancée

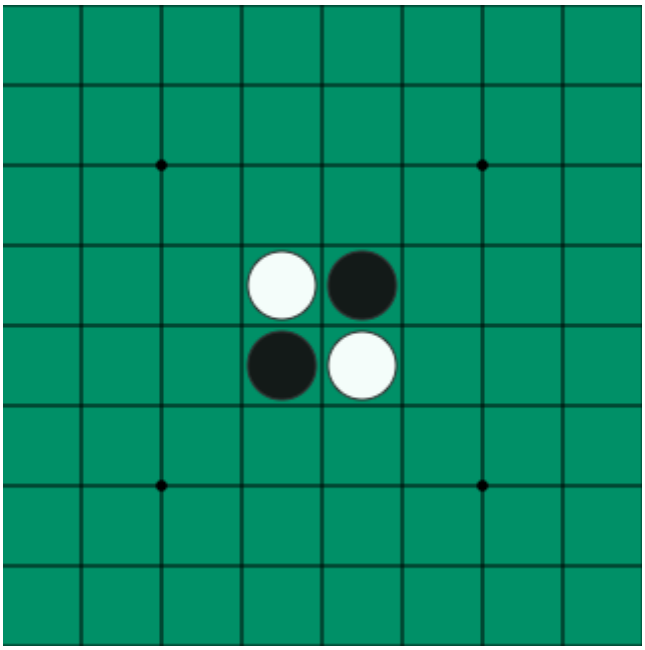
L3-B Informatique

Viet Nguyen – 20006303

Sujet : Implémenter un jeu d’Othello et une intelligence artificielle jount selon un algorithme minimax, puis un élagage alpha-beta. Comparer l’éfficacité de l’IA à différentes profondeurs de jeu.

1) Description au jeu d’Othello

- L'Othello, également connu sous le nom de Reversi, est un jeu de stratégie pour deux joueurs qui se joue sur un plateau de 8x8 cases. Chaque joueur dispose de pions de couleur différente, généralement noir et blanc. Le but du jeu est de posséder plus de pions de sa couleur que son adversaire à la fin de la partie.



- Le jeu commence avec quatre pions placés au centre du plateau en disposition diagonale. Les joueurs placent alternativement un pion sur le plateau de manière à encercler un ou plusieurs pions adverses entre le pion qu'ils viennent de jouer et un autre pion de leur couleur déjà placé sur le plateau. Les pions encerclés sont alors retournés et changent de couleur.

- La partie continue jusqu'à ce que ni l'un ni l'autre joueur ne puisse effectuer un coup valide, généralement lorsque le plateau est plein ou qu'aucun joueur ne peut encercler de pions adverses. À ce moment, le joueur avec le plus de pions de sa couleur sur le plateau est déclaré vainqueur.

- L'Othello est un jeu de réflexion qui exige une bonne anticipation, stratégie et capacité d'adaptation. Chaque coup peut radicalement changer la dynamique du jeu, ce qui le rend captivant et imprévisible.

2) Analyse et Conception

- Introduction:

  Dans cette section, j’aborde l’analyse et la conception du projet de jeu d’Othello intégrant une intelligence artificielle. Le jeu Othello, également connu sous le nom de Reversi, est un jeu de stratégie entre deux joueurs, où l'objectif est de posséder plus de pions que l'adversaire à la fin de la partie.

- Analyse des Besoins:

  Le jeu nécessite une grille de jeu 8x8, deux ensembles de pions (noirs et blancs), et des règles pour placer et retourner les pions. Les joueurs alternent leurs mouvements, et chaque mouvement doit capturer au moins un pion adverse. Le jeu se termine lorsque la grille est complète ou aucun mouvement n'est possible.

- Conception du Système:

  Interface Graphique Utilisateur (GUI) : L'interface utilisateur est conçue pour être intuitive et visuelle. Elle affiche la grille de jeu, les pions, et indique le tour actuel. Des boutons supplémentaires sont inclus pour démarrer une nouvelle partie ou choisir entre différents modes de jeu.

  Structure de Données et Logique de Jeu : Une matrice 2D est utilisée pour représenter la grille de jeu en mémoire. Chaque cellule de la matrice peut être vide, contenir un pion blanc ou un pion noir. Des fonctions sont développées pour vérifier la validité des mouvements, placer les pions, et compter les scores.

  Intelligence Artificielle (AI) : Deux algorithmes principaux sont implémentés pour l'AI - Minimax et Alpha-Beta Pruning. Ces algorithmes permettent à l'AI de choisir le meilleur mouvement possible en évaluant les différents états futurs du plateau de jeu.

  - Minimax : Un algorithme de recherche récursive pour minimiser la perte potentielle dans un scénario de jeu pire cas.
  - Alpha-Beta Pruning : Une amélioration de Minimax qui élimine la nécessité d'explorer des branches qui ne peuvent pas influencer la décision finale.

  Tests et Validation : Des tests sont effectués pour assurer la robustesse de l'application, notamment la validation des règles de jeu, le fonctionnement de l'AI, et la réponse de l'interface utilisateur.

- Conclusion :

  La conception détaillée du projet Othello vise à fournir une expérience de jeu engageante tout en intégrant des concepts complexes d'IA.

3) Développement et mise en oeuvre

Le projet a commencé avec l'élaboration d'une interface utilisateur graphique pour le jeu Othello, en utilisant la bibliothèque openGL GLUT. L'objectif initial était de créer un plateau de jeu fonctionnel pour deux jouers, avant d'intégrer l'algorithme AI.

- Création de l'interface graphique avec GLUT :

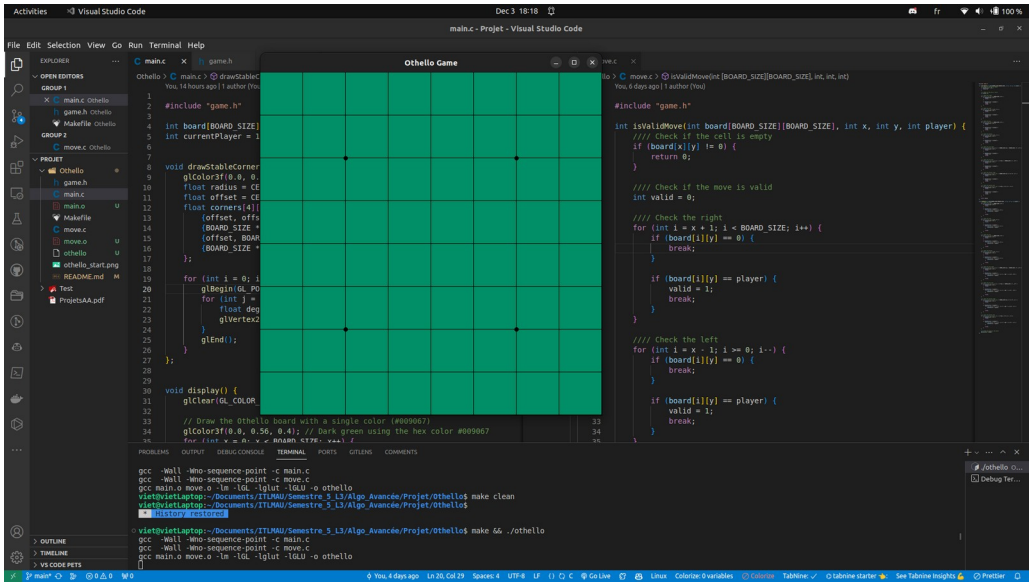
```
#include <GL/gl.h>
#include <GL/glut.h>
#include <GL/freeglut.h>
```

J'ai d'abord développé une interface graphique en utilisant la bibliothèque GLUT, qui m'a permis de dessiner un plateau de jeu et de gérer les interactions des utilisateurs. La fenêtre du jeu affiche un plateau de 8x8, avec des fonctionnalités permettant aux joueurs de placer leurs pions sur le plateau.

```
#define BOARD_SIZE 8
#define CELL_SIZE 80
```

```
void display()
```

```
void mouseClick(int button, int state, int x, int y)
```

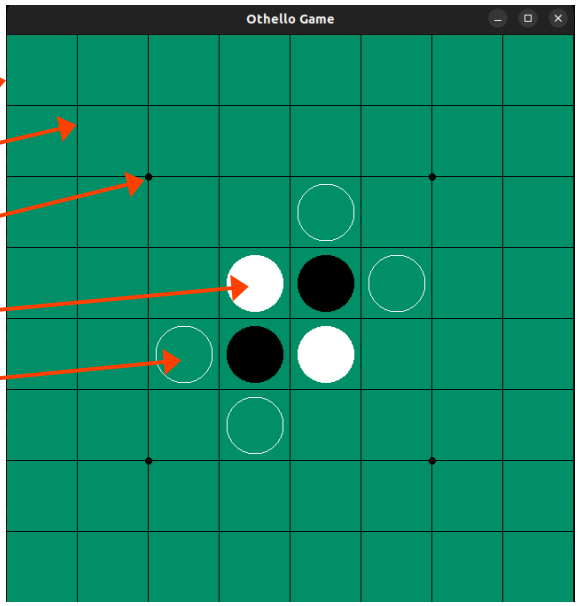


- Ajout des fonctionnalités de base :

Des fonctions telles que

  - **drawBoard()**,
  - **drawGridLines()**,
  - **drawStableCorners()**
  - **drawPiece()**
  - **drawHintCircle()**

dans le fichier "**draw.c**" ont été ajoutées pour afficher interfaces du jeu.



- Implémentation des règles de jeu :

  Les règles de base d'Othello ont été implémentées dans le fichier “**move.c**”, permettant deux joueurs de jouer l'un contre l'autre:

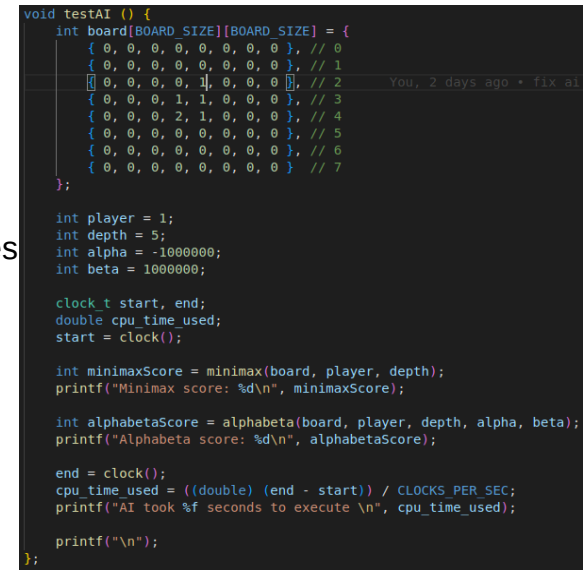
  - Vérification des mouvements légaux : La fonction **isValidMove()** vérifie si un mouvement est légal en se basant sur les règles d'Othello. Elle prend en compte la position actuelle sur le plateau et le joueur actif.
  - Réalisation des mouvements : La fonction **makeMove()** met à jour le plateau de jeu en plaçant un pion à l'endroit indiqué et en retournant les pions adverses conformément aux règles.
  - Comptage des pions : La fonction **countPieces()** compte le nombre de pions pour chaque joueur, ce qui est essentiel pour déterminer le score et la fin du jeu.
  - Fin de jeu : La fonction **isGameOver()** détermine si le jeu est terminé, soit quand le plateau est plein, soit quand aucun joueur ne peut jouer.

- Intégration de l'intelligence artificielle :

  Après avoir établi un jeu fonctionnel pour deux joueurs, j’ai commencé à travailler sur l'intégration de l'algorithme AI. Et l’algorithme Minimax et l’élagage Alpha-Beta sont les plus adaptés pour résoudre Othello.

- Développement des Algorithmes AI :

  Les fichiers “**ai.c**”, “**minimax.c**”, “**alphabeta.c**”, ainsi que leurs en-têtes correspondants “**minimax.h**” et “**alphabeta.h**” ont été créer pour cette tâche. Ces algorithmes calculent les meilleurs mouvement possibles pour l’IA en fonction de l’état actuel du plateau de jeu.



- Test et Optimisation d’algo AI:

  J’ai effectué des tests continus pour assurer le bon fonctionnement des algorithmes AI. Les tests ont aidé à identifier et à résoudre les problèmes, et à optimiser la performance de l’AI.

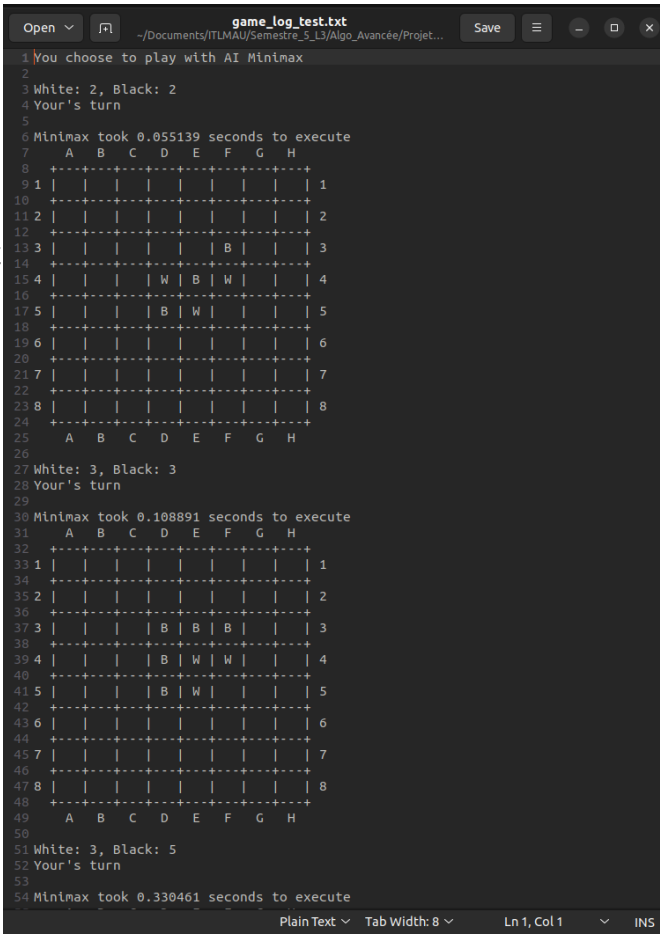
- Implémentation de la Logique de jeu:

  J’ai mis en oeuvre la logiquede jeu dans “**move.c**” et “**draw.c**” qui gèrent les mouvement des pièces, la vérification des mouvements légaux, et l’affifichage des éléments du jeu.

- Enregistrement des Événements de Jeu :

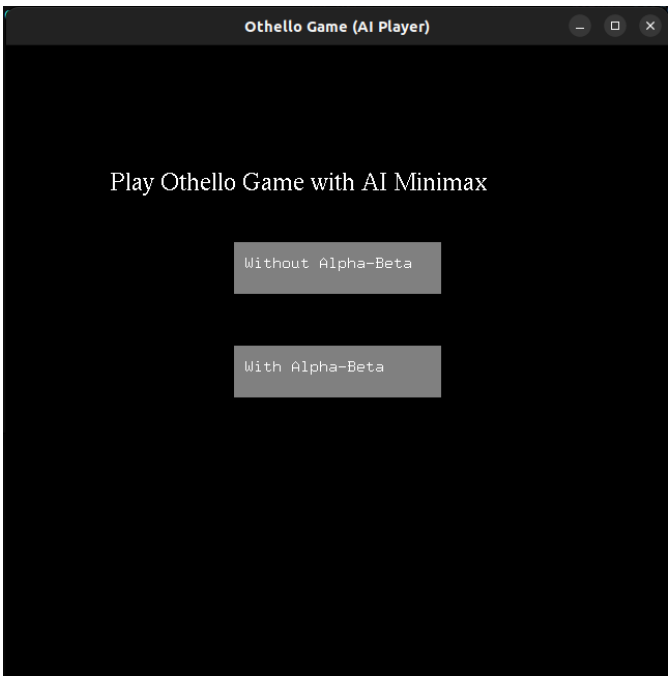
  Pour suivre les actions et l'évolution du jeu, nous avons implémenté un système de journalisation. Ce système écrit l'état actuel du plateau et les mouvements des joueurs dans un fichier log “**game_log.txt**”, facilitant l'analyse postérieure des parties.

```c
void logBoardState(FILE *logFile, int board[BOARD_SIZE][BOARD_SIZE]) {
    fprintf(logFile, "    A   B   C   D   E   F   G   H \n");
    fprintf(logFile, "  +---+---+---+---+---+---+---+---+\n");
    for (int y = 0; y < BOARD_SIZE; y++) {
        fprintf(logFile, "%d ", y + 1);
        for (int x = 0; x < BOARD_SIZE; x++) {
            char piece = ' ';
            if (board[x][y] == 1) piece = 'W'; // white
            else if (board[x][y] == 2) piece = 'B'; // black
            fprintf(logFile, "| %c ", piece);
        }
        fprintf(logFile, "| %d\n", y + 1);
        fprintf(logFile, "  +---+---+---+---+---+---+---+---+\n");
    }
    fprintf(logFile, "    A   B   C   D   E   F   G   H \n\n");
};
```



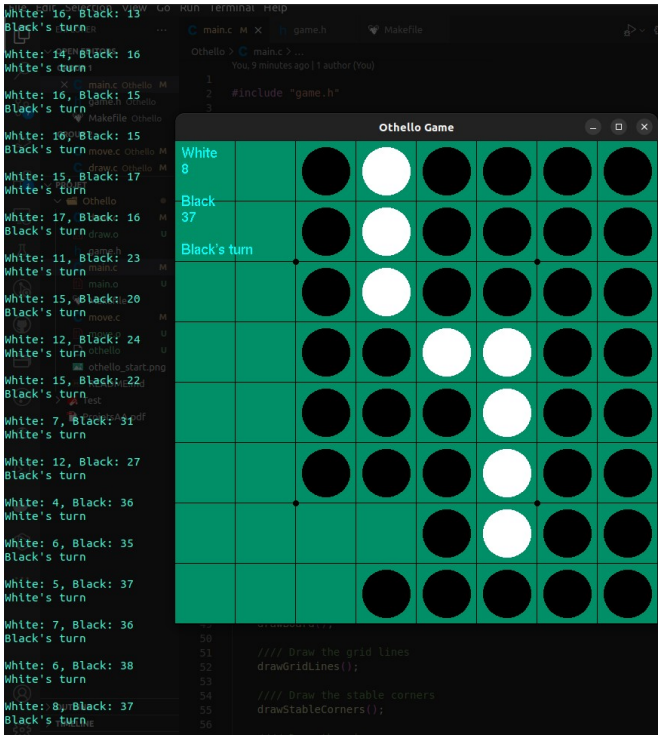
- Interface de Menu et Sélection de l'AI :

  Un menu de sélection a été créé pour permettre aux utilisateurs de choisir entre jouer avec l'AI utilisant Minimax ou avec Alpha-Beta Pruning. Cette fonctionnalité offre une flexibilité et permet aux joueurs d'expérimenter avec différentes stratégies AI.



4) Défis et Solutions

Défi: Reconnaissance de la Fin de Jeu



- Problème Rencontré :

  Au cours du projet, un défi majeur que j'ai rencontré était la reconnaissance de la fin du jeu dans Othello. Le problème spécifique était que la fonction **isGameOver()** ne fonctionnait pas correctement dans la majorité des cas où le plateau n'était pas complètement rempli, mais aucun des deux joueurs n'avait de mouvements valides restants. Cela a posé un problème significatif car cela empêchait le jeu de se terminer naturellement et de manière automatisée.

- Analyse du Problème :

  Après analyse, j'ai constaté que la fonction **isGameOver()** n'arrivait pas à détecter toutes les conditions de fin de jeu. Plus précisément, elle ne pouvait pas identifier les situations où, bien que le plateau ne soit pas entièrement rempli, il n'y avait plus de coups valides possibles pour les deux joueurs.

- Solution Adoptée :

  En réponse à ce problème, j'ai implémenté une solution provisoire qui consistait à ajouter une condition supplémentaire de clic de souris dans **mouseClick()**. Cette solution permettait de détecter la fin du jeu manuellement : lorsque le joueur cliquait et qu'il n'y avait plus de mouvements valides pour les deux parties, le jeu s'annonçait alors terminé.

- Limitations de la Solution :

  Bien que cette solution ait partiellement résolu le problème, elle n'était pas entièrement automatisée et dépendait de l'action de l'utilisateur pour terminer le jeu. Cette approche n'est pas idéale car elle ne capture pas pleinement l'esprit d'un jeu automatisé et intuitif.

- Perspectives d'Amélioration:

Pour une solution plus robuste et automatisée, il serait nécessaire de revoir et d'améliorer la logique de la fonction **isGameOver()**. Cela impliquerait de développer un algorithme plus avancé qui peut reconnaître toutes les conditions de fin de jeu, y compris les situations où il reste des cases libres sur le plateau, mais aucun mouvement valide n'est possible. Cela garantirait que le jeu se termine de manière cohérente et fiable, quelles que soient les circonstances.

5) Résultats et Évaluation

- Résultats Observés :

  Dans mon projet, j'ai mis en œuvre deux algorithmes d'intelligence artificielle pour le jeu d'Othello : le Minimax et le Minimax avec élagage Alpha-Beta. Lors des tests, j'ai observé que, même aux profondeurs de recherche élevées, la différence de performance entre le Minimax standard et le Minimax avec Alpha-Beta était minime, se limitant à quelques millisecondes sur ma machine.

- Évaluation des Performances :

  Cette faible différence de temps était surprenante, étant donné que théoriquement, l'élagage Alpha-Beta devrait réduire de manière significative le nombre de nœuds explorés et, par conséquent, améliorer les performances. Cependant, dans mon cas, les améliorations temporelles étaient marginales. Cette observation soulève des questions intéressantes sur l'efficacité de l'élagage Alpha-Beta dans des scénarios spécifiques, tels que la configuration de notre jeu d'Othello et la capacité de traitement de ma machine.

- Implications et Réflexions :

  Ces résultats suggèrent que pour des jeux avec un espace d'état relativement limité, comme dans mon cas d'Othello, l'impact de l'élagage Alpha-Beta pourrait ne pas être aussi prononcé que dans des jeux avec un plus grand espace d'état. De plus, cela indique également que le choix de l'algorithme et de ses optimisations doit être adapté spécifiquement au contexte et aux exigences de performance du système en question.

- Conclusions :

  Bien que la différence de performance entre les deux algorithmes n'ait pas été significative dans mon environnement de test, l'exercice a été extrêmement précieux pour comprendre les nuances et les applications pratiques des algorithmes d'IA.