

EL AITA Meriem

BABA Salma

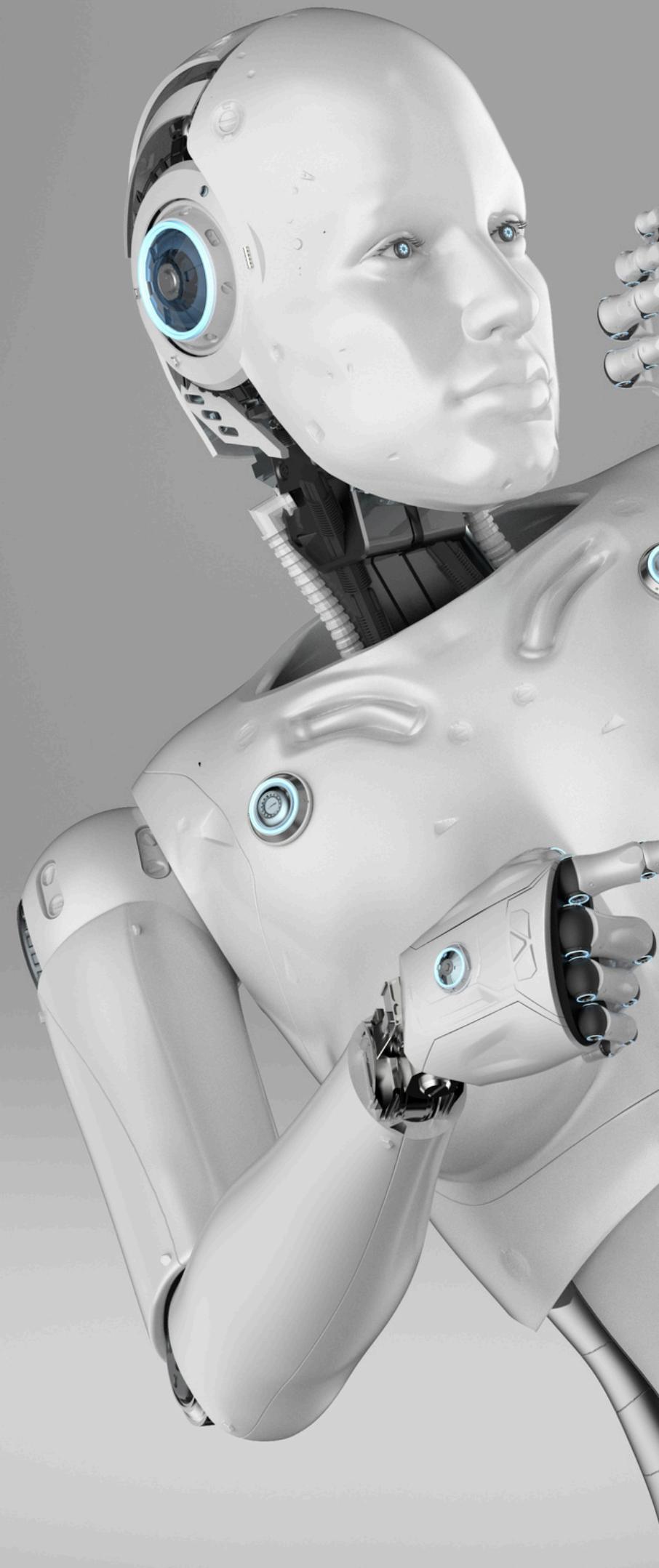
DESAUBLIAUX Arthur

SOUTENANCE DE PROJET :

# IA POUR LE JEU DE GOMOKU

Développement de l'IA d'un jeu de stratégie

Promo 2024 3A FISA



# Planning

**Présentation et gestion du Projet**

01

**Règles du Jeu du Gomoku**

02

**Implémentation du jeu**

03

**Interface graphique**

04

**Algorithmes Utilisés et Stratégies de l'IA**

05

**Démonstration et analyse des résultats**

06

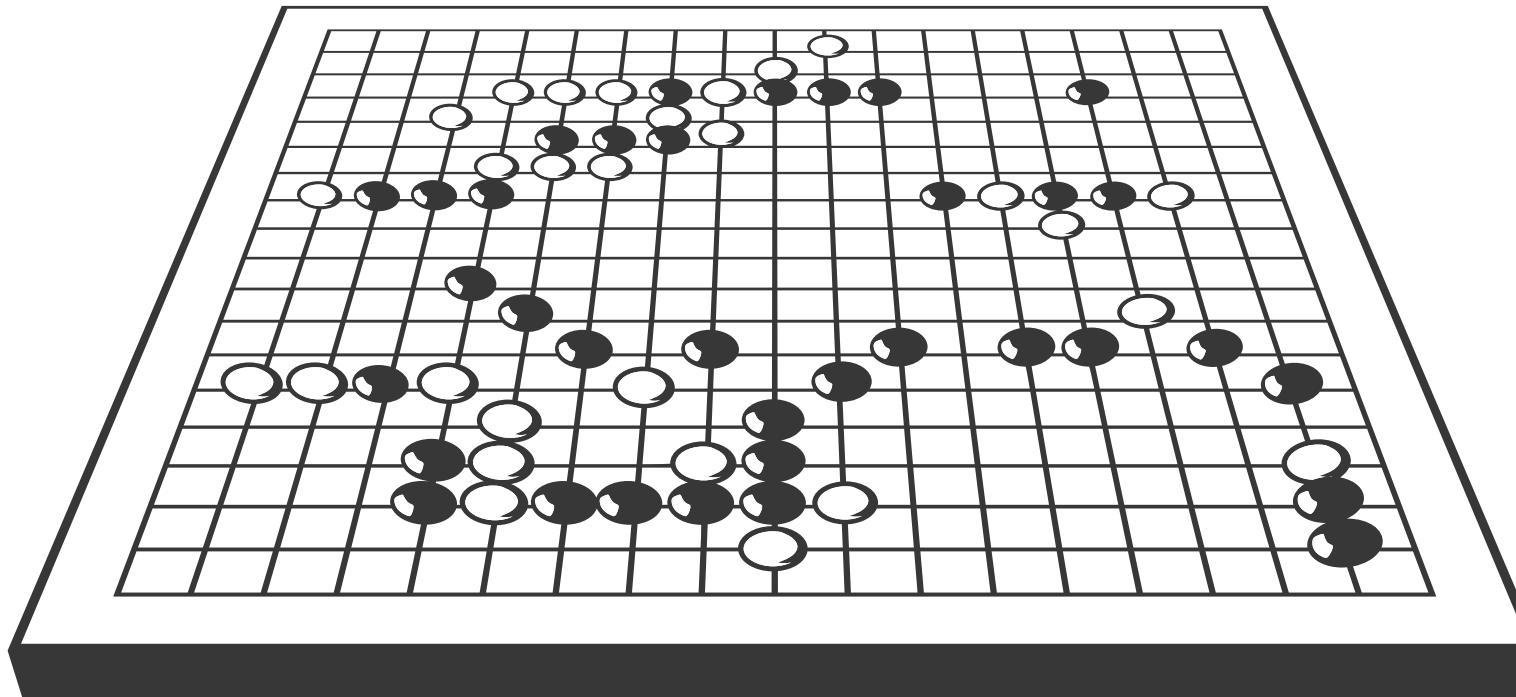
**Conclusion et annexe**

07

# Présentation du Projet

## Introduction au Projet

Le projet consiste à développer l'intelligence artificielle pour le jeu de stratégie Gomoku. Ce jeu à deux joueurs(OrdiVsHumain) nécessite une approche stratégique pour déterminer les coups optimaux.



## Pourquoi avons-nous choisi le Gomoku ?

- Proposition de notre professeur
- Simplicité apparente mais complexité stratégique

**Le Gomoku est un excellent candidat pour l'application des algorithmes d'IA.**

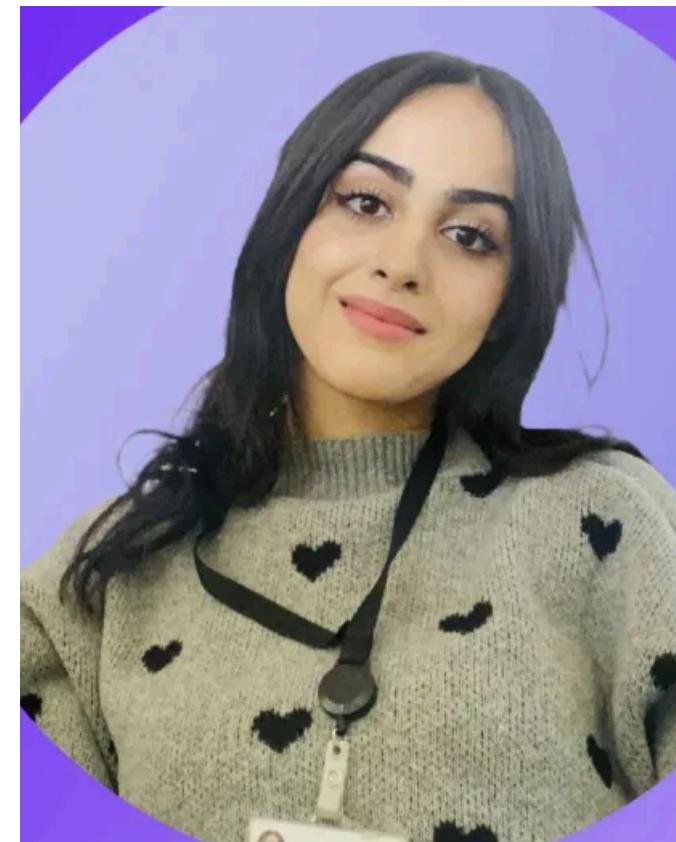
## TECHNOLOGIES UTILISÉES

Le développement se fait en C pour l'algorithme et la gestion des données, et avec SDL pour l'interface graphique.

# ROLE DE L'EQUIPE



SALMA BABA



MERIEM EL AITA

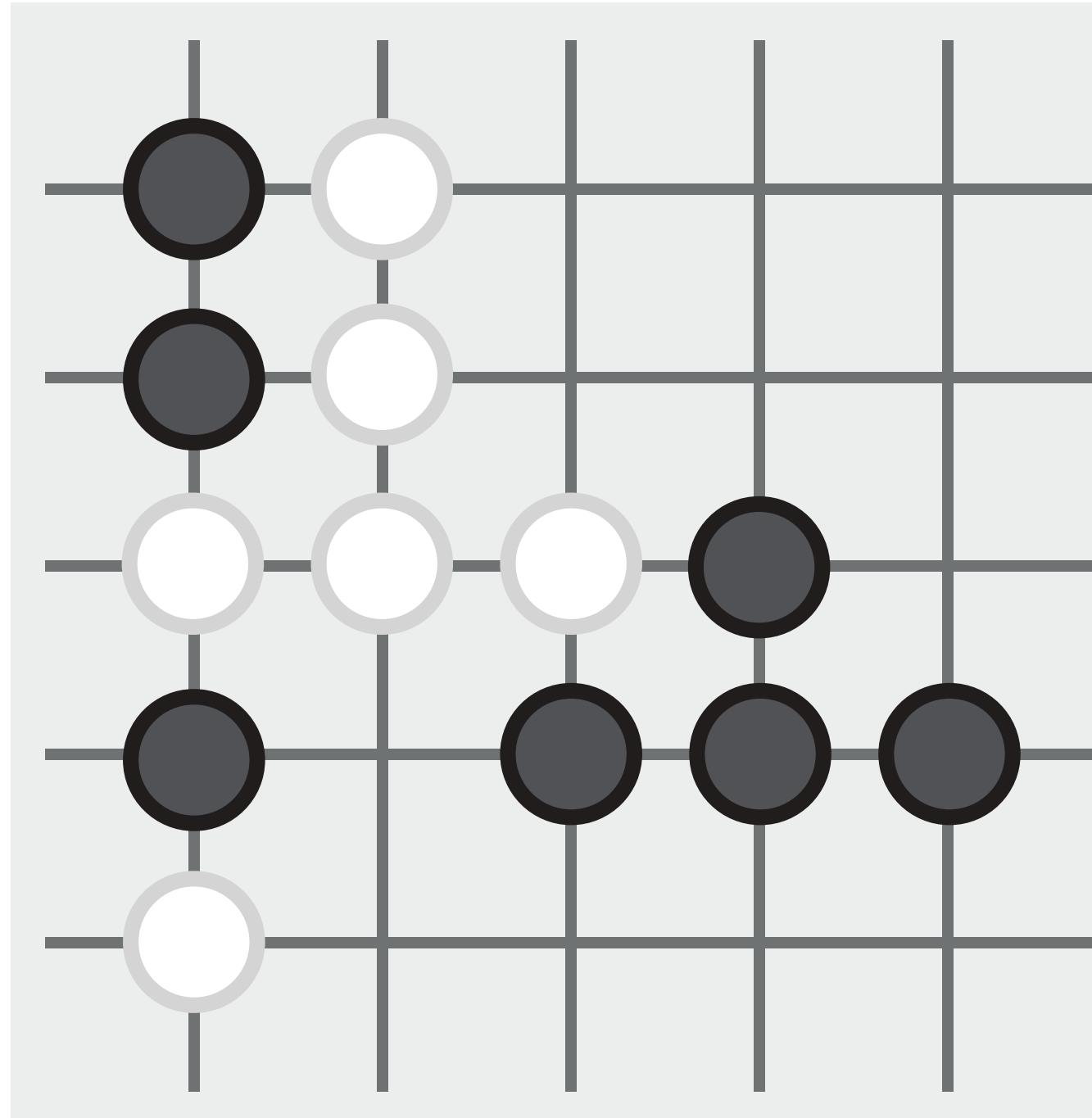


ARTHUR DESAUBLIAUX

Chaque membre de notre équipe a contribué au développement des algorithmes d'IA, à la création de l'interface graphique. Cette approche a permis une répartition équitable des responsabilités et a renforcé notre capacité à surmonter les défis techniques ensemble.

**Méthode de Travail :** réunions hebdomadaires, collaboration et partage des tâches, communication continue (réunions Discord)

# Règles du Jeu



## Présentation du Jeu

Le jeGomoku se joue sur une grille de 15x15 cases. Les joueurs placent tour à tour des pions noirs et blancs pour former une ligne continue de cinq pions de leur couleur.

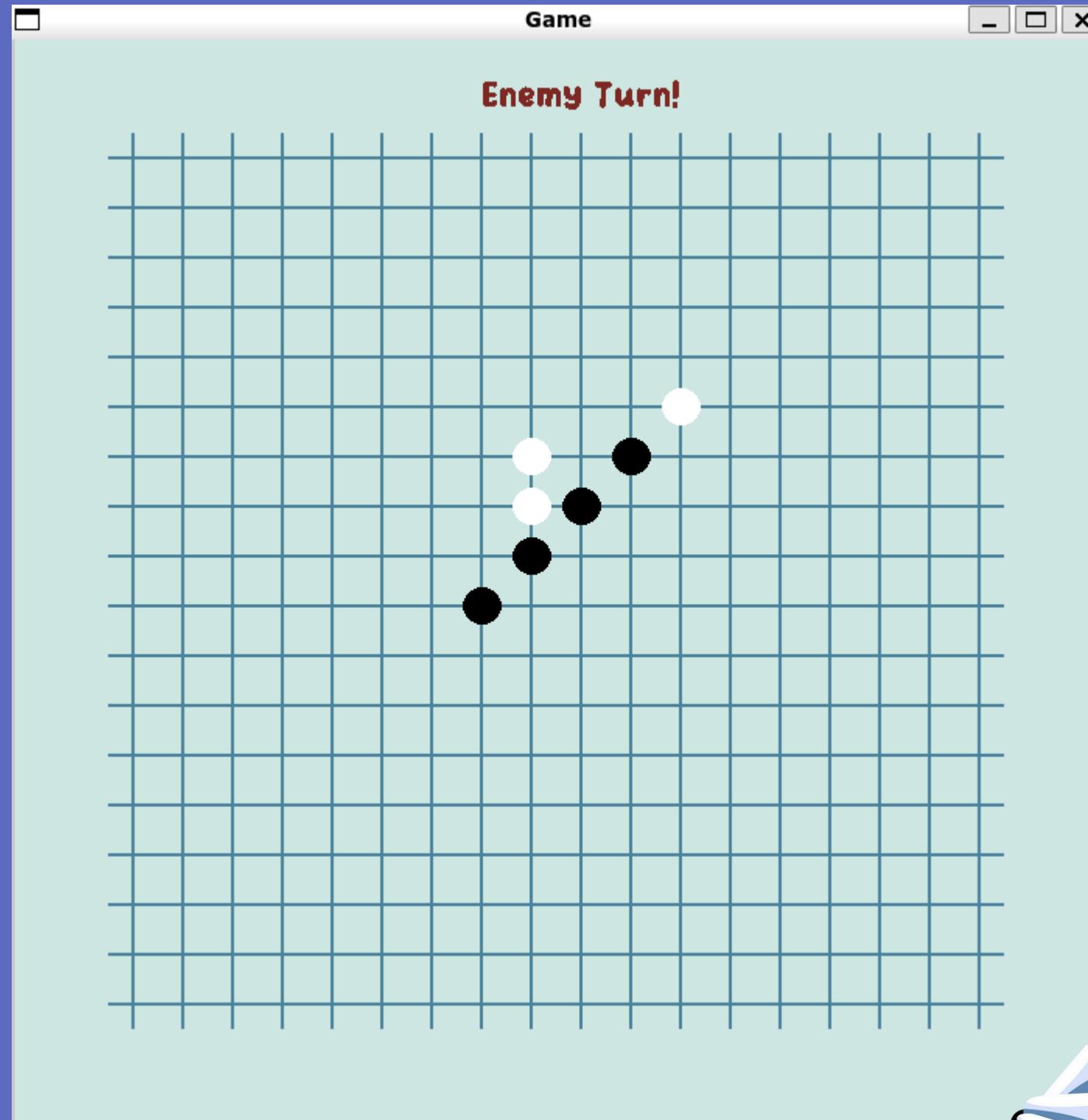
## Règles du Jeu

Les règles sont simples : le premier joueur à aligner cinq pions de sa couleur gagne la partie. Les alignements peuvent être horizontaux, verticaux ou diagonaux.

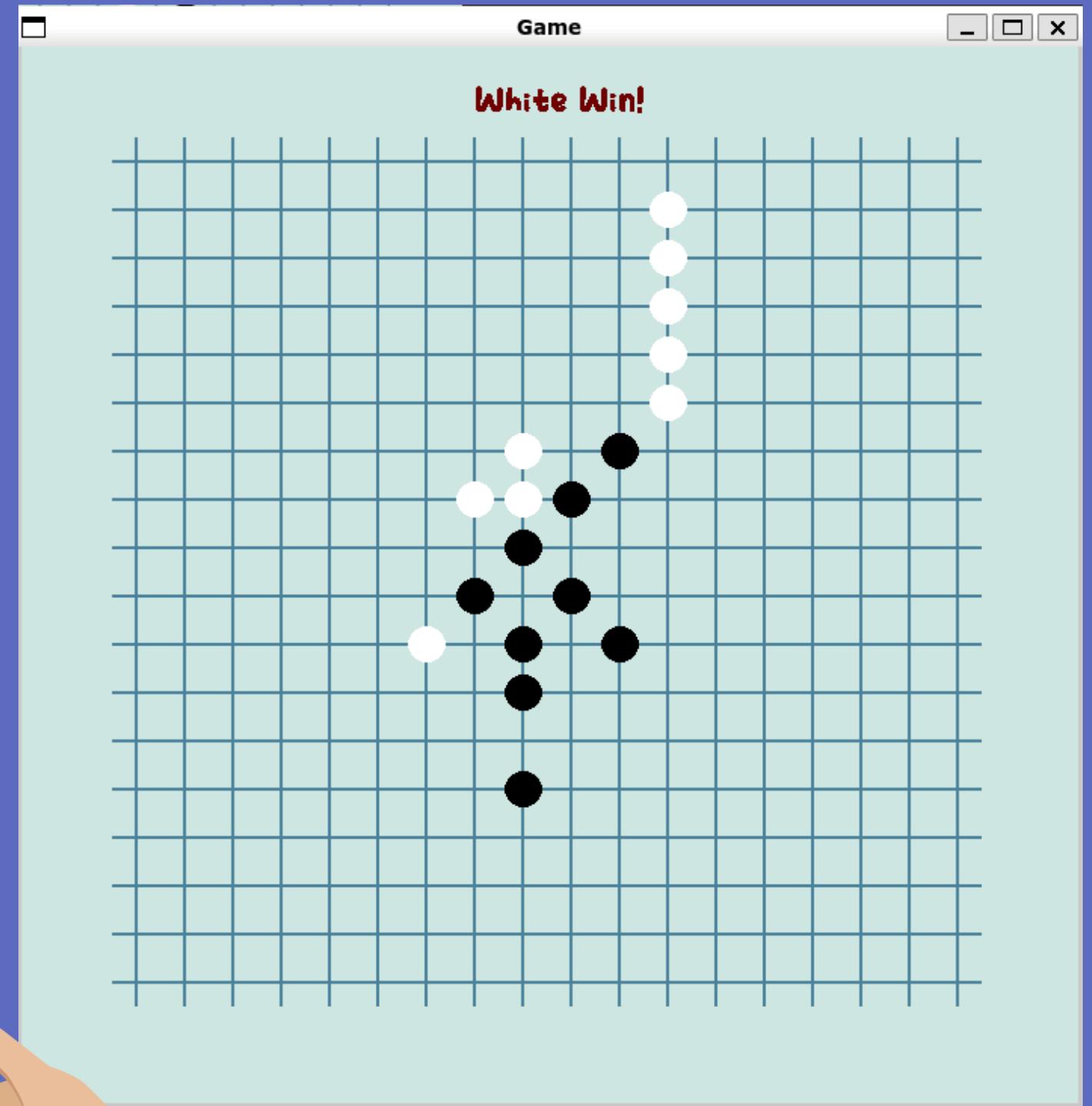
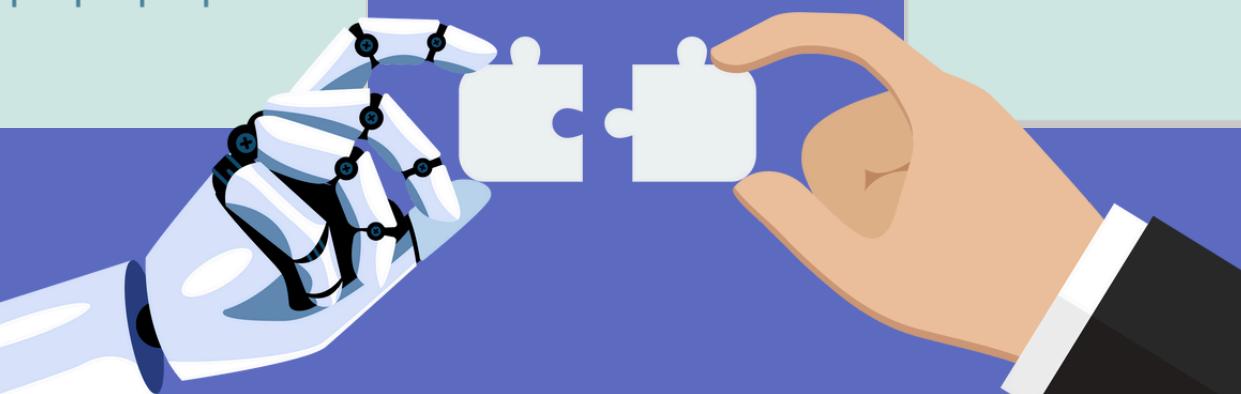
## Démonstration

Une démonstration en direct montrera l'IA en action, prenant des décisions en temps réel pour bloquer l'adversaire et aligner ses pions.

# GUI DE JEU



IA joue



IA gagne

# La partie développement

---

## Langage C

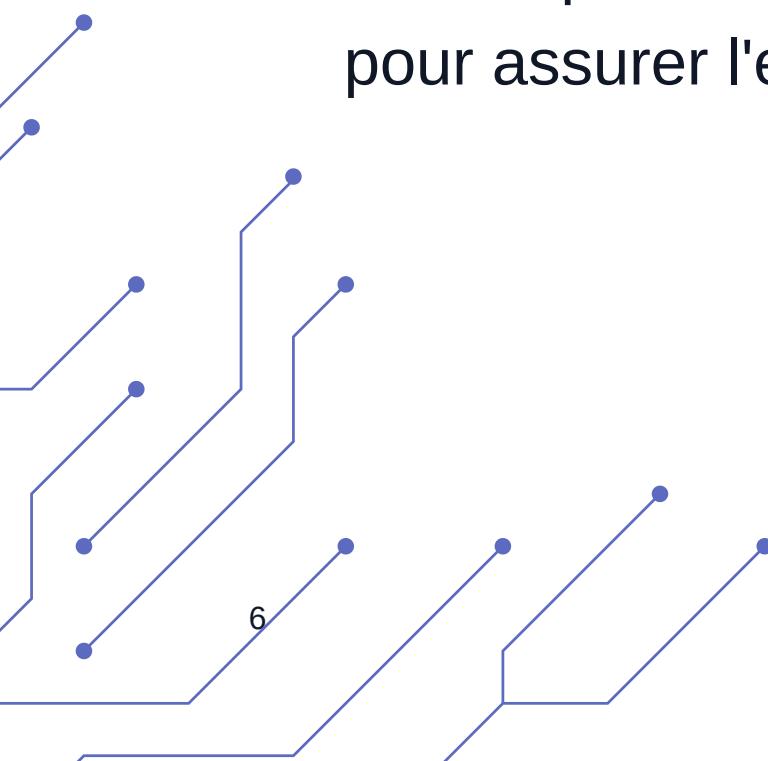
Utilisé pour développer les algorithmes d'IA et gérer les structures de données nécessaires au jeu.

## Structures de Données

Utilisation de listes et de structures pour représenter le plateau de jeu et les mouvements possibles.

## Gestion de la Mémoire

Techniques avancées de gestion de la mémoire pour assurer l'efficacité et éviter les fuites.



# La partie Graphique

---

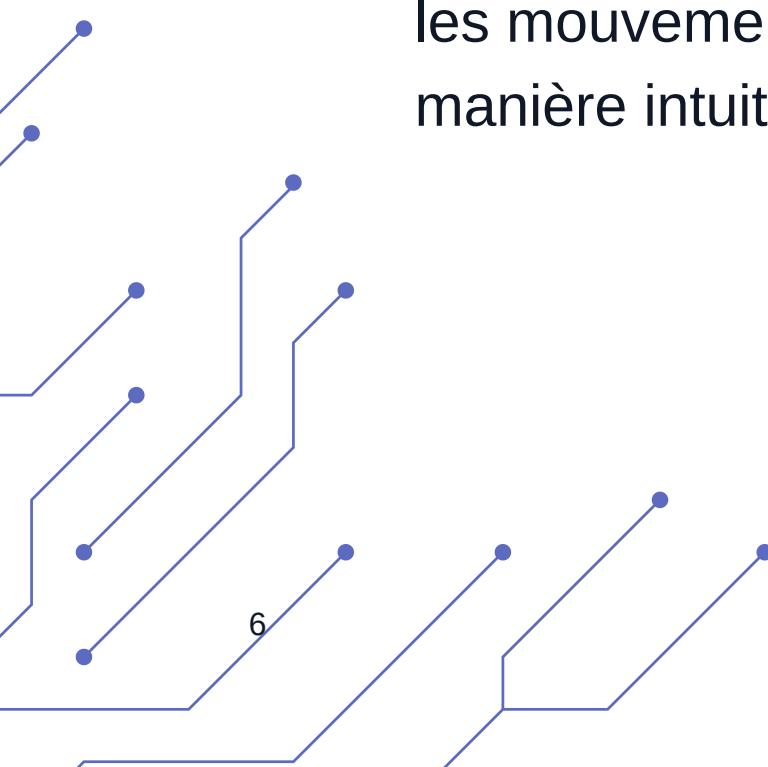
## Bibliothèque SDL2

Développement de l'interface utilisateur en C en utilisant la bibliothèque SDL pour rendre le jeu interactif et visuellement agréable.



## Fonctionnalités

L'interface permet aux utilisateurs de jouer contre l'IA, de voir les mouvements en temps réel et d'interagir avec le jeu de manière intuitive.



# Algorithmes Utilisés



## Algo Minimax

Utilisé pour déterminer le meilleur coup en simulant les mouvements possibles des deux joueurs et en évaluant les positions résultantes.

## Algo Alpha-Beta

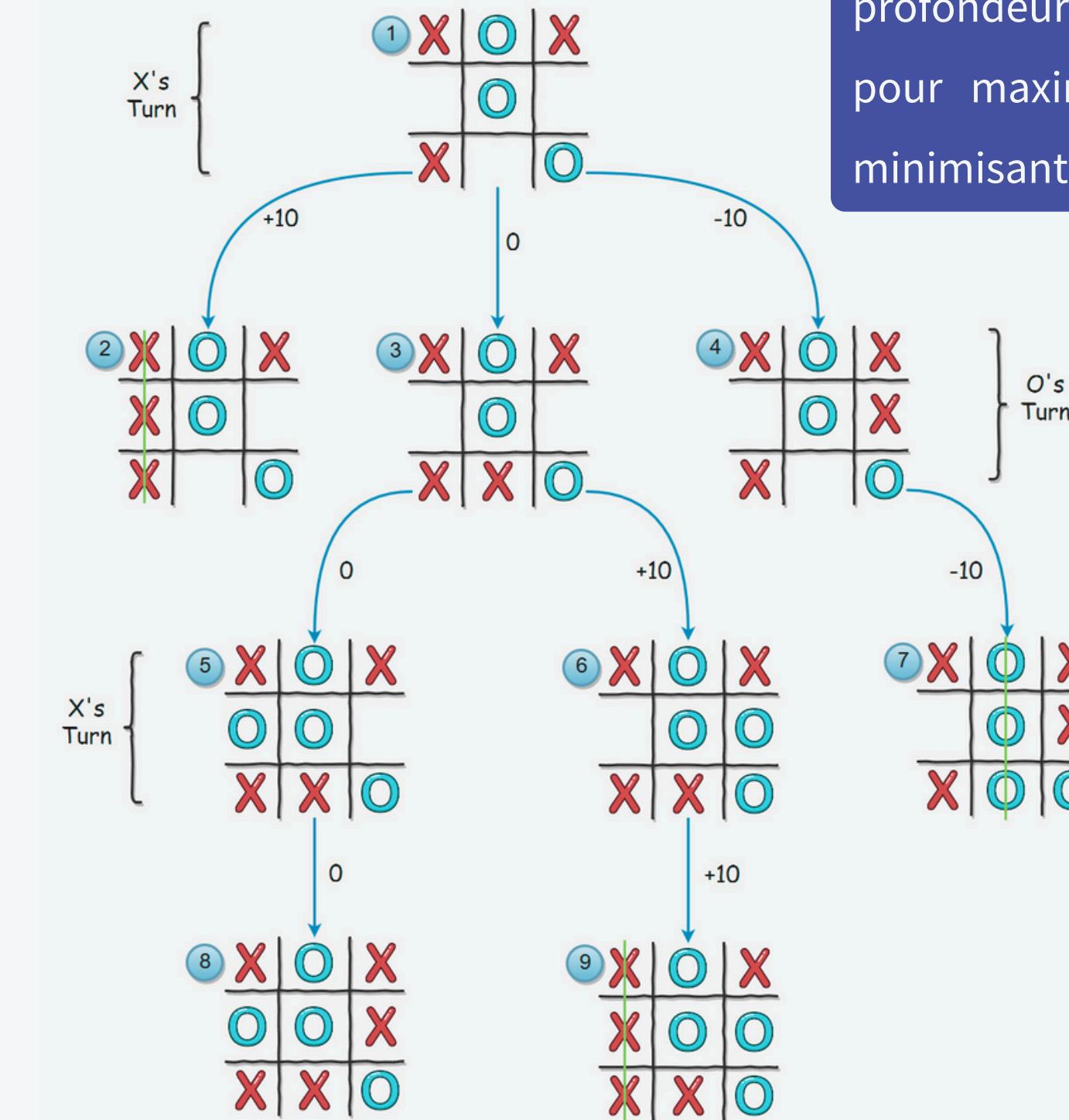
Optimisation du Minimax en réduisant le nombre de nœuds à évaluer dans l'arbre de décision, améliorant ainsi l'efficacité.

# Algorithme MiniMax

```
int minimax(int depth, int isMaximizing, int alpha, int beta) {
    int score = evaluate();
    if (score == 10 || score == -10 || depth == MAX_DEPTH) return score;

    ArrayList moveList;
    getmoveList(&moveList);

    if (isMaximizing) {
        int best = -1000;
        for(int i = 0 ; i < moveList.total;i++){
            Vec2i* rowCol = (Vec2i*) moveList.items[i];
            int row = rowCol->y;
            int col = rowCol->x;
            if(grid[row][col] == 0){
                grid[row][col] = 1;
                best = fmax(best, minimax(depth + 1, 0, alpha, beta));
                alpha = fmax(alpha, best);
                grid[row][col] = 0;
                if (beta <= alpha) break; // Beta cut-off
            }
        }
        return best;
    } else {
        int best = 1000;
        for(int i = 0 ; i < moveList.total;i++){
            Vec2i* rowCol = (Vec2i*) moveList.items[i];
            int row = rowCol->y;
            int col = rowCol->x;
            if(grid[row][col] == 0){
                grid[row][col] = -1;
                best = fmin(best, minimax(depth + 1, 1, alpha, beta));
                beta = fmin(beta, best);
                grid[row][col] = 0;
                if (beta <= alpha) break; // Alpha cut-off
            }
        }
        return best;
    }
}
```



Evalue récursivement les coups possibles jusqu'à une certaine profondeur dans l'arbre des décisions pour maximiser son score tout en minimisant celui de l'adversaire.

# Alpha-Beta : Optimisation de MiniMax

```
void findBestMove() {
    int bestVal = -1000;
    int bestMoveRow = -1;
    int bestMoveCol = -1;
    ArrayList moveList;
    getmoveList(&moveList);
    int alpha = -1000; // Initialize alpha for maximizer
    int beta = 1000;   // Initialize beta for minimizer

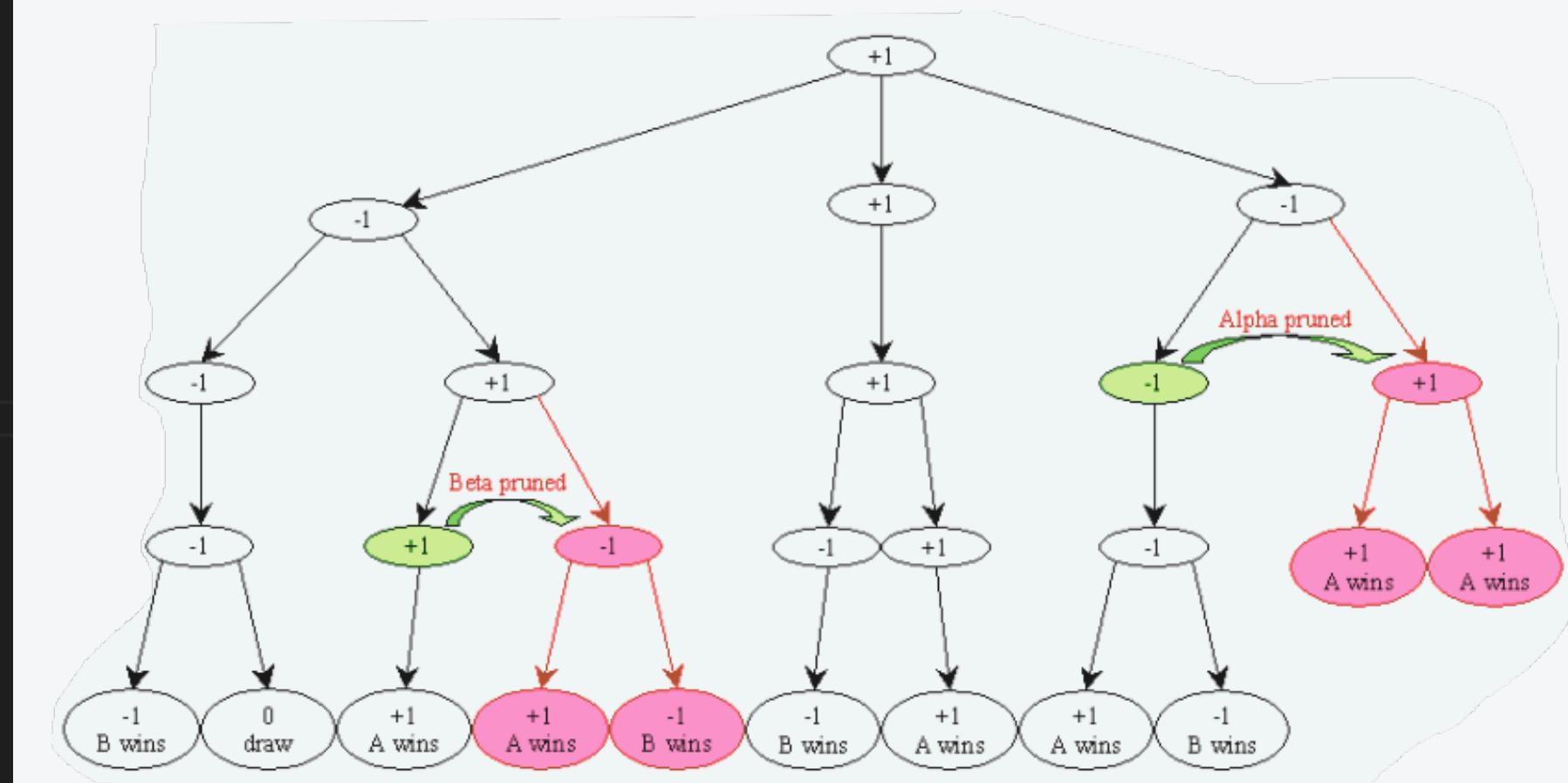
    // Mesure du temps de début
    clock_t start, end;
    double cpu_time_used;

    start = clock();
    for (int i = 0; i < moveList.total; i++) {
        Vec2i* rowCol = (Vec2i*) moveList.items[i];
        int row = rowCol->y;
        int col = rowCol->x;
        if (grid[row][col] == 0) {
            grid[row][col] = 1;
            int moveVal = minimax(0, 0, alpha, beta); // Call minimax with appropriate parameters
            grid[row][col] = 0;
            if (moveVal > bestVal) {
                bestMoveRow = row;
                bestMoveCol = col;
                bestVal = moveVal;
            }
        }
    }
    end = clock();
    cpu_time_used = ((double) (end - start)) / CLOCKS_PER_SEC;

    printf("place-----!\n");
    printf("%d %d\n", bestMoveRow, bestMoveCol);
    printf("-----best: %d\n", bestVal);
    printf("Temps de calcul: %f secondes\n", cpu_time_used);

    if (bestMoveRow != -1 && bestMoveCol != -1) {
        grid[bestMoveRow][bestMoveCol] = -1;
        ArrayList_add(&nodes, newVec2P(bestMoveCol, bestMoveRow));
    }
}
```

Evalue les nœuds tout en maintenant deux valeurs, alpha et bêta, qui représentent les valeurs minimales garanties pour le joueur maximisant et maximales pour le joueur minimisant respectivement.



# Resultats et performances de l'IA

```
[100%] Built target Gomoku
click: 8 8
place-----!
7 8
-----best: 0
Temps de calcul: 0.374238 secondes
click: 4 12
place-----!
3 12
-----best: 0
Temps de calcul: 6.677667 secondes
click: 9 8
place-----!
2 12
-----best: 0
Temps de calcul: 14.855668 secondes
click: 6 8
place-----!
1 12
-----best: 0
Temps de calcul: 30.231325 secondes
click: 10 8
place-----!
0 12
-----best: 0
Temps de calcul: 53.646454 secondes
click: 5 12
```

**Minimax seul**

L'augmentation exponentielle du temps de calcul montre les limitations de l'algorithme

Minimax non optimisé pour des profondeurs de recherche élevées.

L'algorithme optimisé réduit significativement le temps de calcul en coupant les branches inutiles.

```
place-----!
5 7
-----best: 0
Temps de calcul: 0.112662 secondes
click: 12 8
place-----!
4 7
-----best: 0
Temps de calcul: 0.269913 secondes
click: 3 7
place-----!
2 7
-----best: 0
Temps de calcul: 0.473743 secondes
click: 10 8
place-----!
8 6
-----best: 10
Temps de calcul: 2.357551 secondes
click: 8 5
place-----!
5 8
-----best: 10
Temps de calcul: 4.541421 secondes
click: 13 8
place-----!
1 7
-----best: 10
Temps de calcul: 6.735113 secondes
click: 6 7
```

**Minimax avec Alpha-Beta Pruning**

# Conclusion et Annexe

- Le développement d'une IA pour Gomoku, en combinant Minimax et Alpha-Beta Pruning, offre une solution équilibrée entre compréhension de l'algorithme et réactivité.
- L'intégration d'Alpha-Beta réduit significativement le temps de calcul, améliorant ainsi l'expérience de jeu.
- Bien que des défis subsistent, tels que la complexité croissante du jeu, l'utilisation de techniques avancées comme l'optimisation Alpha-Beta permet de les surmonter efficacement, garantissant une IA performante et réactive.

[https://github.com/MrsMeriem/GomokuGame\\_MinMax\\_AlphaBeta](https://github.com/MrsMeriem/GomokuGame_MinMax_AlphaBeta)

[https://github.com/MrsMeriem/GomokuGame\\_MinMax](https://github.com/MrsMeriem/GomokuGame_MinMax)

**MERCI  
DE VOTRE ATTENTION**

