Rapport de Projet :

Implémentation du Jeu de Moulin en Langage C

Introduction

*Le **Jeu de Moulin** est un jeu de stratégie à deux joueurs qui se joue sur un plateau de 24 intersections reliées par des lignes. Le but du jeu est de former des **moulins** (alignements de trois pions) pour capturer les pièces adverses et, à terme, réduire le nombre de pièces de l'adversaire à deux ou l'empêcher de jouer.*

*Dans ce projet, nous avons développé une version numérique du **Jeu de Moulin** en langage **C**. Notre implémentation propose plusieurs fonctionnalités, dont :*

- *Un mode **joueur contre joueur.***
- *Un mode **joueur contre une intelligence artificielle qui joue aléatoirement.***
- *Un mode **joueur contre une intelligence artificielle avancée** capable de stratégies offensives et défensives.*

Ce rapport détaillera l'architecture du projet, les choix d'implémentation, les défis rencontrés et les améliorations possibles.

Affichage et Interface Utilisateur

Pour améliorer l'affichage du jeu et offrir une meilleure lisibilité, plusieurs fonctions ont été implémentées afin de manipuler l'affichage de la console :

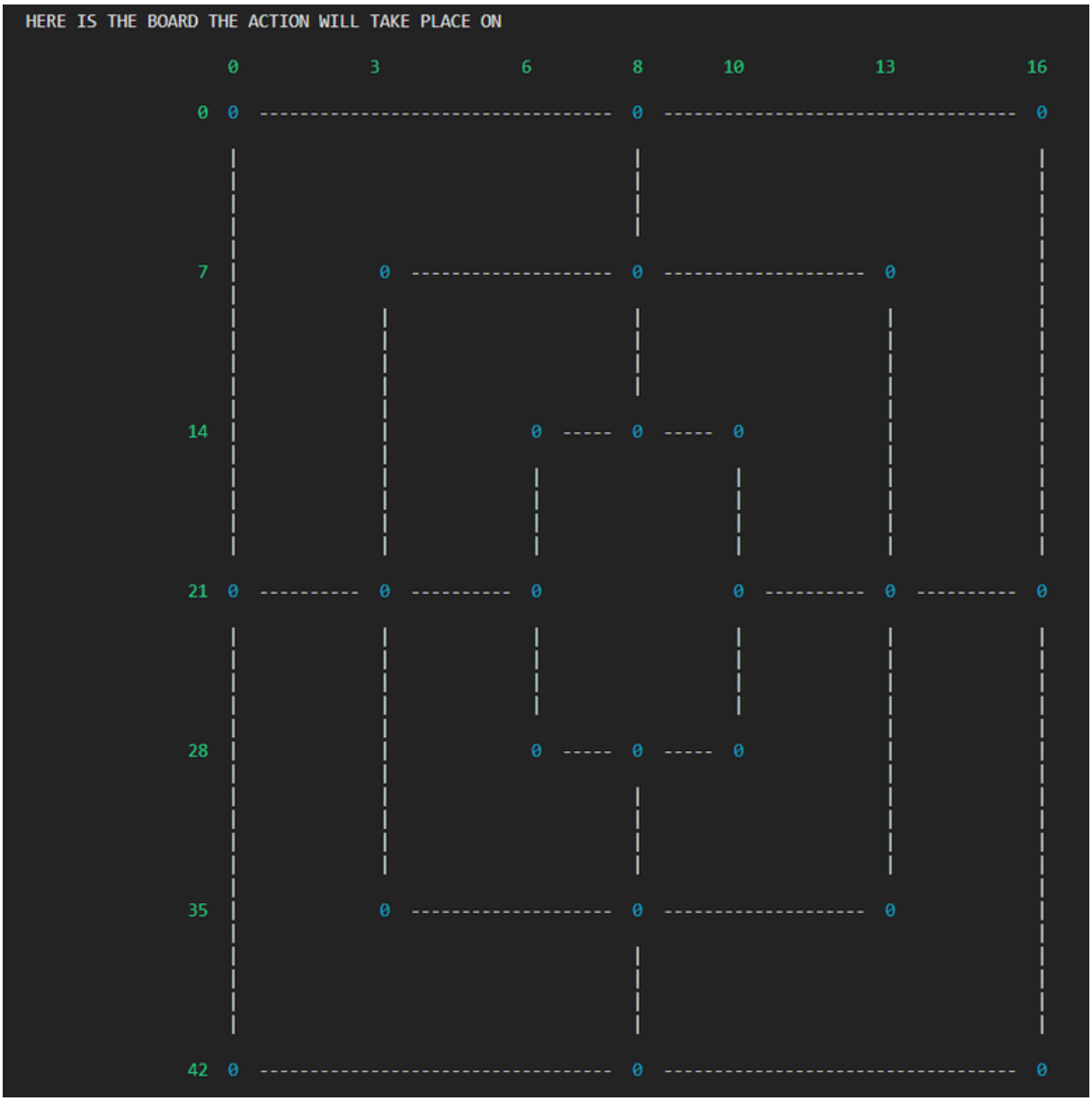
```
void gotoxy(int x, int y) {
    COORD coord;
    coord.X = x;
    coord.Y = y;
    SetConsoleCursorPosition(GetStdHandle(STD_OUTPUT_HANDLE), coord);
}
Tabnine | Edit | Test | Explain | Document
void setColor(int textColor) {
    HANDLE hConsole = GetStdHandle(STD_OUTPUT_HANDLE);
    CONSOLE_SCREEN_BUFFER_INFO consoleInfo;
    WORD newAttributes;

    if (GetConsoleScreenBufferInfo(hConsole, &consoleInfo)) {
        newAttributes = (consoleInfo.wAttributes & 0xF0) | (textColor & 0x0F);
        SetConsoleTextAttribute(hConsole, newAttributes);
    }
}
Tabnine | Edit | Test | Explain | Document
void setColors(int textColor, int backgroundColor) {
    HANDLE hConsole = GetStdHandle(STD_OUTPUT_HANDLE);
    WORD attributes = (backgroundColor << 4) | (textColor & 0x0F);
    SetConsoleTextAttribute(hConsole, attributes);
}
```

- ***gotoxy(x, y)*** *: Permet de positionner le curseur à une coordonnée spécifique pour un affichage structuré.*
- ***setColor(textColor)*** *: Change la couleur du texte pour distinguer les éléments du jeu.*
- ***setColors(textColor, backgroundColor)*** *: Modifie simultanément la couleur du texte et du fond pour une meilleure lisibilité.*

Ces fonctionnalités améliorent l'expérience utilisateur en rendant l'interface plus claire et interactive.

Résultat :



introdution du jeu

(les fonctions utilisées Introduction, Menu, Instructions, Play, Quit)

*La fonction **Introduction** sert à introduire au jeu en guidant le joueur avant qu'il ne fasse son choix parmi les options disponibles. Elle contribue à une interface utilisateur plus engageante et fluide, et en fin elle fait appelle au fonction **Menu**.*

*La fonction **Menu** structure l'expérience utilisateur en lui offrant des options claires et en empêchant les erreurs de saisie. Elle assure une navigation fluide entre les différentes sections du jeu.*

*La fonction **instructions** donne aux deux joueurs les différentes règles du jeu.*

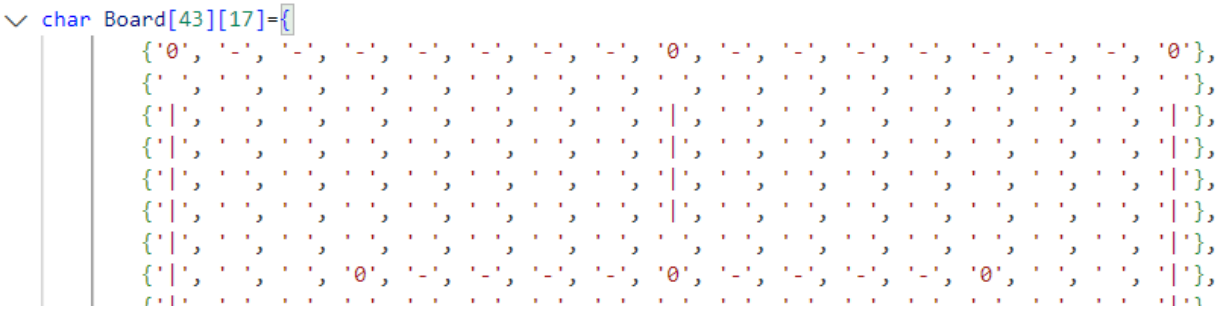
*La fonction **Quit** permet aux joueurs de quitter le jeu.*

*La fonction **Play** assure une navigation fluide entre les différents modes de jeu tout en évitant les erreurs de saisie.*

*La fonction **Again** affiche un menu permettant de rejouer ou de quitter et redirige en conséquence. Elle boucle tant qu'une entrée invalide est saisie.*

*La fonction **Play_again** propose encore une fois les différentes modes de jeu et appelle la fonction correspondante en fonction du choix du joueur, en vérifiant la validité de l'entrée.*

Table de jeu



*La variable **Board** représente graphiquement le plateau de **jeu du Moulin** sous forme d'un tableau 2D de caractères. Ce tableau permet d'afficher la structure du jeu, incluant les positions des pièces, les lignes de connexion et les emplacements valides pour placer des pièces.*

*La fonction **Display** affiche le plateau du jeu en console avec une mise en forme structurée et des couleurs distinctes pour chaque élément (lignes, cases vides, et pièces des joueurs).*

Les différentes fonctions nécessaires dans les différentes modes

*les fonctions utilisées sont (**showavailibleboxes**, **firstmovemaker**, **TheMoveIsValid**, **PlacePiece**, **Millverifier**, **IsValidRemove**, **takeout**, **Isprotectedline**, **Isprotectedcol**, **validecoordinatesstart**, **validecoordinatesend**, **CanPiecemove**, **IsPieceSurrounded** ,**Jumppieceavailable**, **Jumppiece**, **areNodesconnected**, **gameover**)*

*° la fonction **showavailibleboxes** donne les coordonnées des places valables où le joueur peut placer la piece.*

*° la fonction **Firstmovemaker** détermine aléatoirement le joueur qui place la piece en premier .*

maintenant le premier joueur décide où placer la piece (faire un movement)

*° la fonction **TheMoveIsValid** vérifie cette le mouvement est valide , ça vaut dire vérifier si les coordonnées sont valables, si c'est le cas on vérifie si la place est vide.*

*si le mouvement est valable on place la pièce dans la place désirée par la fonction **PlacePiece.***

après chaque mouvement , on vérifie si le joueur a réussi à former un moulin, si c'estle cas il est sensible tirer une piece du joueur adversaire.

```
void millVerifier(int list[3], char Board[][17], int player, int millformed[][3][2], int possibleMills[16][3][2])
    for (int i = 0; i < 16; i++) {
        int row1 = possibleMills[i][0][0], col1 = possibleMills[i][0][1];
        int row2 = possibleMills[i][1][0], col2 = possibleMills[i][1][1];
        int row3 = possibleMills[i][2][0], col3 = possibleMills[i][2][1];
        if (Board[row1][col1] == '0' + player &&
            Board[row2][col2] == '0' + player &&
            Board[row3][col3] == '0' + player) {
                if        ((
            millformed[i][0][0] != row1 || millformed[i][0][1] != col1 ||
            millformed[i][1][0] != row2 || millformed[i][1][1] != col2 ||
            millformed[i][2][0] != row3 || millformed[i][2][1] != col3)){
                list[player]++;
    }
                }
    }
    for (int i=0;i<16;i++){
        millformed[i][0][0]=-1;millformed[i][0][1]=-1;
        millformed[i][1][0]=-1;millformed[i][1][1]=-1;
        millformed[i][2][0]=-1;millformed[i][2][1]=-1;
    }
    for (int i=0;i<16;i++){
        int row1 = possibleMills[i][0][0], col1 = possibleMills[i][0][1];
        int row2 = possibleMills[i][1][0], col2 = possibleMills[i][1][1];
        int row3 = possibleMills[i][2][0], col3 = possibleMills[i][2][1];
        if (Board[row1][col1]!='0' && Board[row1][col1] == Board[row2][col2] &&
            Board[row1][col1] == Board[row3][col3]){
                millformed[i][0][0] = row1;
                millformed[i][0][1] = col1;
                millformed[i][1][0] = row2;
                millformed[i][1][1] = col2;
                millformed[i][2][0] = row3;
                millformed[i][2][1] = col3;
            }
        }
}
```

(la fonction la plus difficile à mon avis)

si list[player]>0 ça veut dire que le joueur a formé un moulin, il va tirer une piece de l'autre.

*° la fonction **IsValidRemove** vérifie si la pièce choisie peut être supprimée, si oui la fonction **takeout** fait la tâche.*

*° les deux fonctions (**isprotectedline** et **isprotectedcol**) vérifient si la pièce à supprimer est protéger par un moulin , formé horizontalement (par ligne) ou verticalement (par colonne). si c'estle cas la pièce ne peut pas être supprimée.*

*° La fonction **allprotected** vérifie si toutes les pièces d'un joueur sont protégées en parcourant le plateau, si c'est le cas aucune pièce ne peut être retirée donc on change le tour.*

après avoir placé 9 pièces par chaque joueur, on termine avec la phase de placement , et on est dans la phase de déplacement des pièces sur la table.

le joueur doit choisir une pièce sur la table qui revient à lui et choisir les coordonnées où il veut la déplacer.

*° les deux fonctions (**validecoordinatesstart** et **validecoordinatescol**) vérifient les coordonnées saisies par le joueur, elles doivent être valables et les dernières doivent être une place vide.*

*° la fonction **Jumppieceavailable** vérifie si le déplacement est valable ou pas.*

*° la fonction **areNodesConnected** vérifie si deux nœuds sont connectés ou pas.*

*° les deux fonctions **(IspieceSurrounded** et **CanPiecemove)** vérifient si une piece peut se déplacer où elle est entourée par d'autres pièces.*

après chaque tour, on calcul le nombre des pieces du joueur , si il a deux pièces, ou bien tous ses pièces sont bloquées, le jeu est terminé et s'il a trois il peut se déplacer librement.

*° la fonction **gameover** vérifie les conditions de fin du jeu.*

Mode joueur contre joueur

*La fonction Player_Player gère le déroulement d'une partie du jeu de Moulin (ou Nine Men's Morris) entre deux joueurs humains. Elle orchestre les deux phases principales du jeu : **la phase de placement** et **la phase de déplacement** (sans oublier le saut libre). Voici un résumé de ses objectifs :*

1. ***Initialisation du jeu** :*
   - *Affiche le plateau de jeu.*
   - *Détermine aléatoirement quel joueur commence (THE_PLAYER_1 ou THE_PLAYER_2).*
2. ***Phase de placement** :*
   - *Les joueurs placent à tour de rôle leurs pièces sur le plateau.*
   - *Vérifie si un joueur forme un "moulin" (ligne de trois pièces) après chaque placement.*
   - *Si un moulin est formé, le joueur peut retirer une pièce de l'adversaire (sous certaines conditions).*
3. ***Phase de déplacement** :*
   - *Une fois toutes les pièces placées, les joueurs déplacent leurs pièces sur le plateau.*
   - *Vérifie si un joueur forme un moulin après chaque déplacement.*
   - *Si un moulin est formé, le joueur peut retirer une pièce de l'adversaire (sous certaines conditions).*
   - *Gère le cas spécial où un joueur n'a plus que 3 pièces : il peut alors "sauter" (déplacer une pièce n'importe où sur le plateau).*
4. ***Fin du jeu** :*
   - *La fonction se termine lorsque la condition de fin de jeu est atteinte (un joueur n'a plus que 2 pièces ou ne peut plus faire de mouvements valides).*
   -

Mode joueur contre intelligence artificielle 0

Ce mode est similaire à celui de joueur contre joueur, il faut juste adapter la situation pour que le deuxième joueur utilise l'aléatoire.

La fonction PlayerVsMachine0 gère une partie du jeu de Moulin (Nine Men's Morris) entre un joueur humain (THE_PLAYER_1) et une machine (THE_PLAYER_2). Elle orchestre les deux phases principales du jeu : ***la phase de placement*** *et* ***la phase de déplacement****, en intégrant des mécanismes pour que la machine joue de manière aléatoire. Voici un résumé de ses objectifs :*

1. ***Initialisation du jeu*** *:*
   - *Affiche le plateau de jeu.*
   - *Informe que le joueur humain est THE_PLAYER_1 et la machine est THE_PLAYER_2.*
   - *Détermine aléatoirement qui commence (THE_PLAYER_1 ou THE_PLAYER_2).*
2. ***Phase de placement*** *:*
   - *Le joueur humain place ses pièces en choisissant des coordonnées valides.*
   - *La machine place ses pièces de manière aléatoire sur des cases disponibles.*
   - *Après chaque placement, vérifie si un moulin (ligne de trois pièces) est formé.*
     - *Si un moulin est formé, le joueur ou la machine retire une pièce de l'adversaire (sous certaines conditions).*
3. ***Phase de déplacement*** *:*
   - *Une fois toutes les pièces placées, les joueurs déplacent leurs pièces.*
     - *Le joueur humain choisit manuellement les déplacements.*
     - *La machine déplace ses pièces de manière aléatoire.*
   - *Vérifie si un moulin est formé après chaque déplacement.*
     - *Si un moulin est formé, le joueur ou la machine retire une pièce de l'adversaire (sous certaines conditions).*
   - *Gère le cas spécial où un joueur n'a plus que 3 pièces : il peut alors "sauter" (déplacer une pièce n'importe où sur le plateau).*
4. ***Fin du jeu*** *:*
   - *La fonction se termine lorsque la condition de fin de jeu est atteinte (un joueur n'a plus que 2 pièces ou ne peut plus faire de mouvements valides).*
   - *Affiche le gagnant (THE_PLAYER_1 ou THE_PLAYER_2).*

Mode joueur contre intelligence artificielle 1

Dans ce mode, on est sensé à implimenter une IA plus avancée, c'est pour cela on aura besoin de d'autres fonctions qui gèrent les décisions de la machine.

- *Les fonctions qui gèrent la* ***PHASE DE PLACEMENT***

les fonctions utilisées sont (***tryCompletMillLine****,* ***tryCompletMillco****l,* ***tryCompletMill****,* ***tryPlacePieceInLine****,* ***tryPlacePieceInCol****,* ***placePieceStratigically****,* ***tryPlacePieceNearSingleInLine****,* ***tryPlacePieceNearSingleInCol***

° les fonctions (***tryCompletMillLine****,* ***tryCompletMillcol****,* ***tryCompletMill*** *) permettent à la machine de compléter des moulins (alignements de trois pièces) dans le jeu de Moulin. Elles vérifient les lignes et colonnes du plateau pour :*

1. *Détecter si deux pièces du joueur sont alignées avec une case vide.*
2. *Placer une pièce dans la case vide pour former un moulin.*
3. *Retourner true si un moulin est complété, sinon false.*

Elles rendent la machine plus intelligente en lui permettant de jouer de manière stratégique pour gagner.

*° les fonctions (**tryPlacePieceInLine**, **tryPlacePieceInCol**) permettent à la machine de **bloquer** la formation de moulins par l'adversaire :*

1. ***tryPlacePieceInLine** :*
   - *Vérifie une ligne horizontale.*
   - *Si l'adversaire a deux pièces alignées et une case vide, la machine place une pièce pour bloquer le moulin.*
   - *Retourne true si un blocage est effectué, sinon false.*
2. ***tryPlacePieceInColumn** :*
   - *Vérifie une colonne verticale.*
   - *Si l'adversaire a deux pièces alignées et une case vide, la machine place une pièce pour bloquer le moulin.*
   - *Retourne true si un blocage est effectué, sinon false.*

***Objectif** : Empêcher l'adversaire de former un moulin en plaçant une pièce stratégique.*

*° la fonction **placePieceStratigically** vise à placer stratégiquement une pièce sur le plateau de jeu pour un joueur donné. Elle fonctionne en deux étapes :*

1. ***Placement Prioritaire** : Elle tente d'abord de placer la pièce sur des positions stratégiques définies dans les tableaux strategicRows et strategicCols.*
2. ***Placement de Secours** : Si aucun emplacement stratégique n'est disponible, elle place la pièce dans la première position libre trouvée sur le plateau.*

Si un placement est effectué, la fonction retourne true, sinon false.

*° les deux fonctions (**tryPlacePieceNearSingleInLine**, **tryPlacePieceNearSingleInCol)** font partie de l'algorithme de placement stratégique de l'IA dans Le Jeu de Moulin.*

- ***Objectif commun***

Optimiser le placement des pièces de l'IA en les positionnant à proximité de ses propres pièces déjà placées, soit en ligne, soit en colonne, afin de faciliter la formation d'un moulin.

Fonctionnement

- ***tryPlaceNearSinglePieceInLine**:*
  - *Vérifie plusieurs lignes du plateau.*
  - *Cherche un emplacement où une seule pièce du joueur est présente avec deux cases vides à côté.*
  - *Si une telle position est trouvée, l'IA y place une pièce et retourne true.*
- ***tryPlaceNearSinglePieceInColumn**:*
  - *Applique le même principe, mais sur les colonnes du plateau.*
  - *Cherche une colonne où une seule pièce de l'IA est entourée de cases vides.*
  - *Place une pièce si une telle configuration est trouvée et retourne true.*

Ces fonctions améliorent la capacité de l'IA à créer des moulins et à bloquer l'adversaire stratégiquement.

*° La fonction **intelligenceplacing** permet à l'IA de placer une pièce en suivant une logique de priorité. Elle commence par tenter de compléter un moulin pour gagner immédiatement (tryCompleteMill). Si cela n'est pas possible, elle cherche à bloquer l'adversaire en l'empêchant de former un moulin en ligne (tryPlacePieceInLine) ou en colonne (tryPlacePieceInColumn). Ensuite, elle essaye de se rapprocher de la formation d'un moulin en plaçant une pièce à côté d'une autre déjà posée (tryPlaceNearSinglePieceInLine et tryPlaceNearSinglePieceInColumn). Enfin, si aucune de ces stratégies n'est applicable, elle opte pour un placement par défaut sur une position stratégique (placePieceStrategically). Cette approche permet à l'IA de maximiser ses chances de victoire tout en défendant contre les actions de l'adversaire.*

- *Les fonctions de la **PHASE DE DÉPLACEMENT***

*les fonction utilisées (**completmillline1**, **completmillcolumn1**, **completmill1**, **movepieceline1**, **movepiececolumn1**, **movePieceStrategically1**, **movenearpieceline1**, **movepieceneararcolumn1**, **movepiecemachine1**)*

*° La fonction **completemill1** permet à l'IA de compléter un moulin en déplaçant une de ses pièces lorsqu'elle a déjà deux pièces alignées dans une ligne ou une colonne et qu'une case reste vide. Pour cela, elle utilise completmillline1 et completmillcolumn1, qui vérifient si deux pièces du joueur sont alignées et s'il est possible de déplacer une autre pièce vers la case vide pour compléter le moulin. La fonction parcourt toutes les lignes et colonnes possibles où un moulin peut être formé et, si une position éligible est trouvée, elle déplace une pièce disponible en utilisant Jumppiece. Cette approche optimise le placement de l'IA en maximisant ses chances de verrouiller des moulins et de prendre un avantage stratégique sur l'adversaire.*

*° Les fonctions **movepieceline1** et **movepiececolumn1** permettent à l'IA de bloquer un moulin adverse en déplaçant une de ses pièces vers une case stratégique. Elles vérifient si l'adversaire a déjà deux pièces alignées dans une ligne ou une colonne et s'il reste une case vide qui permettrait de compléter un moulin. Si une telle configuration est trouvée, elles parcourent la liste des pièces du joueur et, si l'une d'elles peut être déplacée vers cette position, elles effectuent le mouvement à l'aide de Jumppiece. Cette approche défensive permet à l'IA d'empêcher l'adversaire de former des moulins et de renforcer sa position stratégique sur le plateau.*

*° La fonction **movePieceStrategically1** permet à l'IA de déplacer stratégiquement une de ses pièces vers une position clé sur le plateau. Elle commence par vérifier si l'une des cases stratégiques définies (aux intersections importantes du plateau) est libre. Si c'est le cas, elle tente d'y déplacer une pièce du joueur en parcourant la liste des pièces disponibles. Si aucune case stratégique n'est libre, elle effectue un balayage complet du plateau pour trouver la première case vide où une pièce peut être déplacée. Cette approche assure que l'IA privilégie d'abord les positions avantageuses avant d'envisager un déplacement plus générique.*

*° Les fonctions **movepiecenearcolumn1** et **movenearpieceline1** cherchent à déplacer stratégiquement une pièce du joueur vers une position voisine vide dans une colonne ou une ligne spécifique, en vérifiant si une telle opportunité existe. Elles parcourent des positions prédéfinies sur le plateau et, si une case est disponible, elles tentent d'y déplacer une pièce du joueur en utilisant la fonction Jumppiece. Ces fonctions permettent ainsi d'optimiser le positionnement des pièces en fonction des configurations existantes.*

*° La fonction **movepiecemachine1** tente de déplacer stratégiquement une pièce pour l'IA en suivant plusieurs étapes prioritaires. Elle commence par vérifier si elle peut compléter un moulin avec completemill1. Ensuite, elle essaie de déplacer une pièce sur des lignes spécifiques avec movepieceline1 et dans des colonnes spécifiques avec movepiececolumn1. Si aucune de ces options n'est possible, elle tente un déplacement proche avec movenearpieceline1 ou movepiecenearcolumn1. Enfin, si aucune autre stratégie ne fonctionne, elle applique une approche plus large avec movePieceStrategically1. Si aucun mouvement valide n'est trouvé, la fonction retourne false.*

- *les fonctions utilisées pour **retirer les pièces***

*les fonctions utilisées sont (**lookforpiecestoremoveinLine**, **lookforpiecestoremoveinCol**, **lookforpiecestoremoveinLine1**, **lookforpiecestoremoveinColumn1**, **removeone**, **lookforpiecestoremove**)*

*° les fonctions **lookforpiecestoremoveinLine** et **lookforpiecestoremoveinCol** sont capables de détecter si on a deux pièces de l'adversaire qui sont alignées (selon une ligne ou une colonne) donc on le retire une pour qu'il ne forme pas de moulin.*

*° les fonctions **lookforpiecestoremoveinLine1** et **lookforpiecestoremoveinColumn1** sont capables de détecter si une pièce de l'adversaire empêche le joueur actuel de compléter un moulin donc elles la retirent.*

*° si aucunes deux piéces sont alignées, on parcours la table et retire une pièce (**removeone**).*

- *les fonctions utilisées dans **la phase du saut***

*les fonctions utilisées sont (**completmillline**, **completmillcolumn**, **completmill**, **movepieceline**, **movepiececolumn**, **movePieceStrategically**, **movenearpieceline**, **movepiecenearcolumn**, **movepiecemachine**)*

° Ces fonctions sont les mêmes que celles de la phase de déplacement, mais cette fois-ci on ne vérifie pas la connections entre les deux nodes.

---

*° La fonction **PlayerVsMachine1** gère une partie du jeu de Moulin (Nine Men's Morris) entre un joueur humain (THE_PLAYER_1) et une machine intelligente (THE_PLAYER_2). Contrairement à*

PlayerVsMachine0, cette version utilise une intelligence artificielle plus avancée pour les décisions de la machine. Voici un résumé de ses objectifs :

1. ***Initialisation du jeu** :*
   - *Affiche le plateau de jeu.*
   - *Informe que le joueur humain est THE_PLAYER_1 et la machine est THE_PLAYER_2.*
   - *Détermine aléatoirement qui commence (THE_PLAYER_1 ou THE_PLAYER_2).*
2. ***Phase de placement** :*
   - *Le joueur humain place ses pièces en choisissant des coordonnées valides.*
   - *La machine utilise une fonction d'intelligence (intelligenceplacing) pour placer ses pièces de manière stratégique.*
   - *Après chaque placement, vérifie si un moulin (ligne de trois pièces) est formé.*
     - *Si un moulin est formé, le joueur ou la machine retire une pièce de l'adversaire (sous certaines conditions).*
3. ***Phase de déplacement** :*
   - *Une fois toutes les pièces placées, les joueurs déplacent leurs pièces.*
     - *Le joueur humain choisit manuellement les déplacements.*
     - *La machine utilise une fonction d'intelligence (movepiecemachine) pour déplacer ses pièces de manière stratégique.*
   - *Vérifie si un moulin est formé après chaque déplacement.*
     - *Si un moulin est formé, le joueur ou la machine retire une pièce de l'adversaire (sous certaines conditions).*
   - *Gère le cas spécial où un joueur n'a plus que 3 pièces : il peut alors "sauter" (déplacer une pièce n'importe où sur le plateau).*
4. ***Fin du jeu** :*
   - *La fonction se termine lorsque la condition de fin de jeu est atteinte (un joueur n'a plus que 2 pièces ou ne peut plus faire de mouvements valides).*
   - *Affiche le gagnant (THE_PLAYER_1 ou THE_PLAYER_2).*

Points clés de la fonction :

- ***Intelligence artificielle** : La machine utilise des fonctions comme intelligenceplacing, movepiecemachine, et lookforpiecetoremove pour prendre des décisions stratégiques, ce qui la rend plus compétente que dans PlayerVsMachine0.*
- ***Gestion des moulins** : La fonction vérifie systématiquement si un moulin est formé après chaque placement ou déplacement, et applique les règles associées (retrait de pièces adverses).*
- ***Phase de saut** : Si un joueur n'a plus que 3 pièces, il peut déplacer n'importe laquelle de ses pièces vers n'importe quelle case vide.*
- ***Fin de partie** : La fonction détecte quand un joueur ne peut plus jouer ou n'a plus assez de pièces, et déclare le gagnant.*