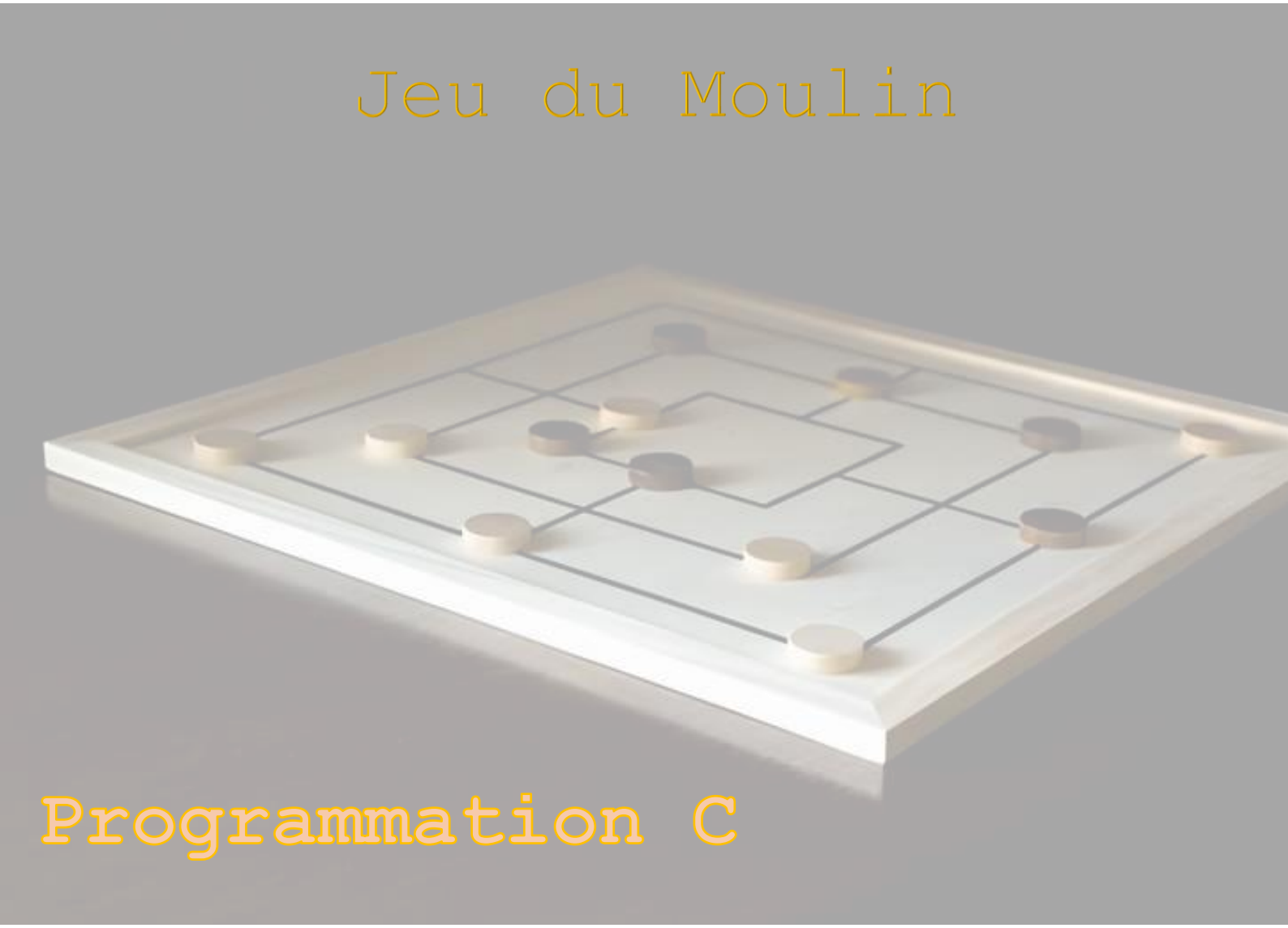
Rapport de projet :

Jeu du Moulin



Programmation C

Réalisé par :

+Omar Amdouni.

+Saoud Amine.

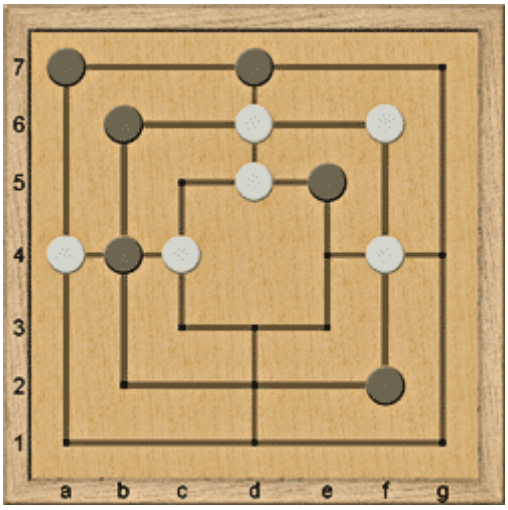
+Hamza Amhidi.

Encadré par :

Mme. Meryem Cherrabi

Présentation du jeu du moulin :

Le jeu de moulin, également connu sous le nom de "jeu du moulin" ou "neuf trous", est un jeu de stratégie traditionnel qui remonte à des millénaires. Ce jeu se joue généralement sur un plateau comprenant trois carrés concentriques reliés par des lignes médianes, formant ainsi un réseau de 24 points d'intersection.



Règles :

Chaque joueur dispose de neuf pièces qu'il doit placer sur les intersections libres du plateau. L'objectif principal est de former des "moulins", c'est-à-dire des alignements de trois pièces sur une même ligne, horizontalement ou verticalement.

Lorsqu'un joueur parvient à former un moulin, il a le droit de retirer une pièce de l'adversaire, affaiblissant ainsi sa position sur le plateau.

Le jeu alterne entre des phases de placement et des phases de déplacement, où les joueurs cherchent à maximiser leurs moulins tout en bloquant ceux de l'adversaire.

But du projet :

Le but de ce projet consiste à la réalisation, à l'aide du langage C, 2 versions du jeu :

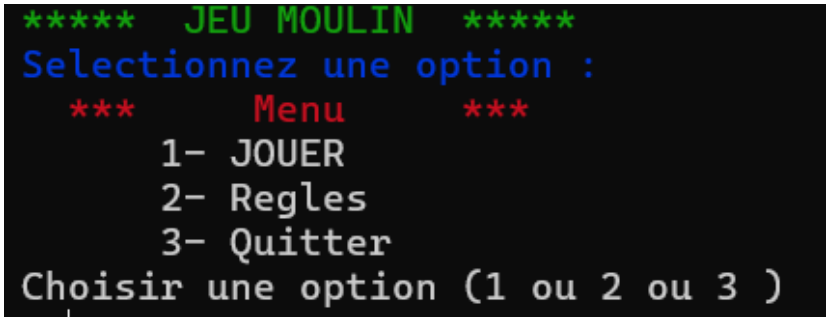
+ joueur contre joueur : où 2 utilisateurs s'affronteront.

+ joueur contre machine : où le joueur humain jouera contre la machine, avec le choix d'une difficulté débutant ou normale.

Simulation du jeu :

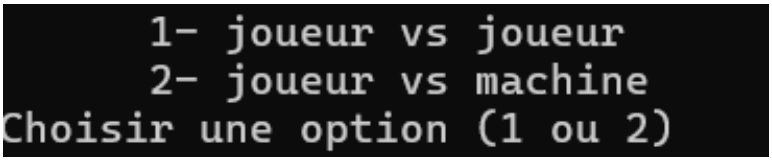
I)Menu :

Le jeu commence d'abord par un menu d'accueil où l'utilisateur aura 3 options : jouer, lire les règles, ou quitter.



+Jouer :

L'utilisateur a le choix entre 2 modes :



-joueur vs joueur : Dans ce mode, l'utilisateur joue contre un joueur humain.

-joueur vs machine : Dans ce mode, le joueur aura la possibilité de jouer contre une machine, à travers deux difficultés :

*difficulté facile.

*difficulté avancée.

+Règles :

Si le joueur souhaite comprendre plus en détail les règles du jeu, il peut toujours consulter cette section avant de commencer le jeu.

II)Exploration approfondie des modes de jeu :

1)Joueur vs Joueur :

+Le jeu commence par le choix des symboles/caractères qui représentera chaque mouvement des joueurs. Ensuite, un tirage au sort sera effectué pour désigner l'ordre de jeu. La couleur rouge sera assignée au premier joueur, et la couleur verte, au deuxième joueur.

+Le dernier coup joué sera désigné par la couleur jaune.

```
le joueur droite ,choisissez un symbole (un seul caractere):  o
le joueur gauche ,choisissez un symbole (un seul caractere):  l

****************************************
          COULEURS DES JOUEURS
****************************************
 Joueur 1 : ROUGE (o)
 Joueur 2 : VERT (l)
 Dernier coup joue : JAUNE
****************************************
```

C'est ainsi que la phase de placement commence.

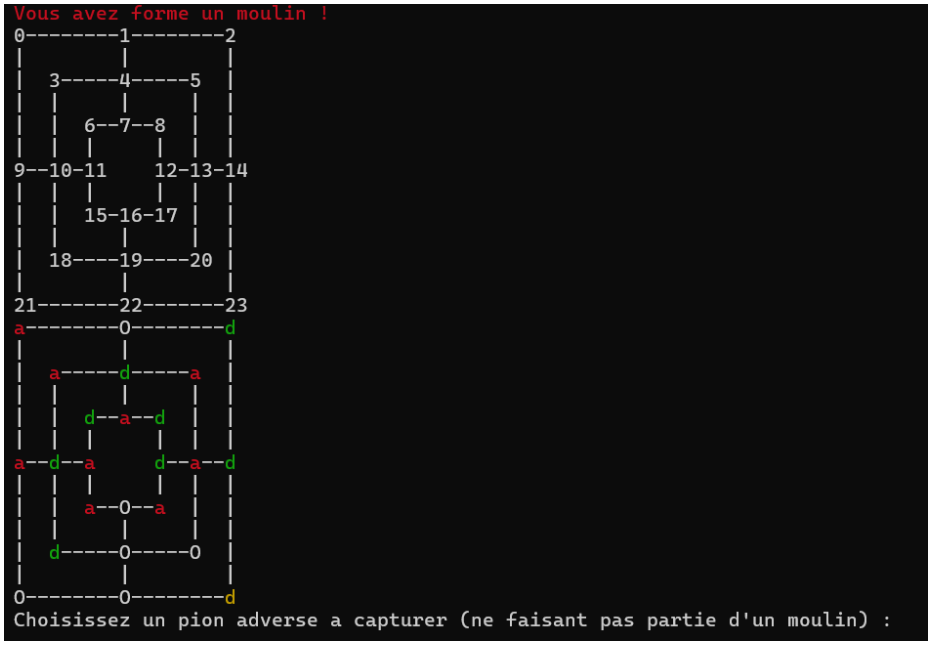
```
C'est la phase de Placement :

0--------1--------2
|        |        |
|  3-----4-----5  |
|  |     |     |  |
|  |  6--7--8  |  |
|  |  |     |  |  |
9--10-11    12-13-14
|  |  |     |  |  |
|  |  15-16-17 |  |
|  |     |     |  |
|  18----19----20 |
|        |        |
21-------22-------23
0--------a--------d
|        |        |
|  a-----d-----a  |
|  |     |     |  |
|  |  d--a--d  |  |
|  |  |     |  |  |
a--d--a     d--a--d
|  |  |     |  |  |
|  |  a--d--a  |  |
|  |     |     |  |
|  0-----0-----0  |
|        |        |
0--------0--------0
Tour du joueur 2 (d)
 Choisissez une position pour placer votre pion (0 a 23) :
```

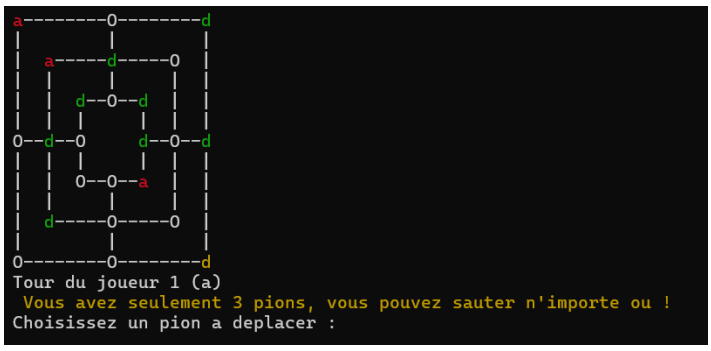
*Phase de placement : Dans cette étape, les joueurs posent à tour de rôle leurs pions (9 pions), en suivant le rôle assigné en début de partie. Cette phase est primordiale et nécessite pas mal de réflexion.

Ensuite, on passe à la phase de mouvement.

*Phase de mouvement : Dans cette étape, les joueurs déplacent leurs pions de sorte à former des moulins et ainsi réduire le nombre de pions de l'adversaire.



L'objectif demeure de réduire le nombre de pions à 2 ou tout simplement bloqué les mouvements adverses. On rappelle qu'un joueur peut déplacer librement un pion, si ce dernier est bloqué et a exactement 3 pions restants.



Cette règle du déplacement libre à 3 pions est assez importante. En effet, en cas de bonne usage, le joueur en position de faiblesse peut retourner le cours de la partie.

Si un des 2 joueurs atteint un nombre de pions < 3, ou est tout simplement bloqué avec un nombre de pions > 3, l'adversaire est déclaré gagnant.

2) Joueur vs Machine :

Commençons d'abord par la première difficulté :

*Facile/débutant :

Le joueur humain jouera contre une IA assez simple. En effet, le but de cette difficulté consiste surtout à aider le joueur à mieux comprendre les principes du jeu, en jouant contre une machine qui ne cherche pas vraiment à être un obstacle.

Choix des symboles :

Le symbole de la machine est par défaut 'm'. Le joueur choisit son symbole et un tirage au sort est effectué pour savoir qui jouera en premier.

```
Pour la machine, on choisit le symbole : 'm'
la machine va commencer ;

****************************************
          COULEURS DES JOUEURS
****************************************
 Joueur 1 : ROUGE (m)
 Joueur 2 : VERT (l)
 Dernier coup joue : JAUNE
****************************************
```

Phase de placement :

Après avoir déterminé l'ordre de passage, les joueurs commencent à placer leurs pions à tour de rôle. Contrairement au joueur humain, la machine pose aléatoirement ses pions, en vérifiant à chaque fois la disponibilité de l'emplacement choisi, ce qui constitue la plus grande différence par rapport au mode joueur contre joueur.

Phase de mouvement :

Cette étape est également basée sur des mouvements aléatoires de la machine, ce qui est censé faciliter la victoire du joueur humain.

*Deuxième difficulté : avancée

Phase de placement :

Dans la phase de placement, la machine utilise une stratégie intelligente pour placer ses pions en tenant compte des meilleures positions disponibles sur le plateau tout en anticipant les menaces potentielles de l'adversaire.

Elle choisit des emplacements qui maximisent ses chances de former des moulins tout en bloquant les opportunités adverses. La machine évalue également les configurations qui pourraient mener à un double moulin, ce qui lui permettrait de capturer plusieurs pions en quelques tours.

À chaque fois qu'elle forme un moulin, elle sélectionne avec précision le pion adverse le plus stratégique à capturer, visant à

réduire la mobilité de l'adversaire ou à briser des configurations importantes.

Phase de mouvement :

Dans la phase de mouvement, la machine choisit ses déplacements pour former des moulins tout en évitant les blocages.

Elle utilise des tactiques avancées comme démonter temporairement un moulin pour le reformer et capturer à nouveau. La machine analyse aussi les risques de blocage pour garder ses pions actifs.

Elle analyse également les situations où ses pions pourraient être bloqués et adapte ses déplacements pour maintenir un contrôle optimal du plateau.

III. Phase d'analyse et conception

1. Modélisation du plateau

Le plateau est représenté par un tableau « board [24] » où chaque case peut être :

- `'O'` : Vide
- `'X'` : Pion du Joueur 1
- `'Y'` : Pion du Joueur 2

2. Détection des moulins

La fonction moulin (int i) vérifie si un pion placé à l'index « i » forme un moulin avec ses voisins en utilisant un tableau de connexions.

3. Gestion des déplacements

La fonction move (int i1, int i2, char x) s'assure que :

- Le pion déplacé appartient au joueur.
- La case destination est vide.
- Les règles de mouvement sont respectées.

IV. Développement du jeu

Le projet est divisé en plusieurs fichiers :

- start.c : Initialisation du jeu et affichage du menu.
- joueurJoueur.c : Gestion du mode Joueur vs Joueur.
- fonctions_de_jeu.c : Contient les fonctions principales du jeu.

Principales fonctions implémentées

- choixSymboles(&sj1, &sj2, mode) : Choix des symboles des joueurs.
- TourDePlacement() : Gestion de la phase de placement des pions.
- TourDeMvt() : Déplacement des pions sur le plateau.
- checkAndHandleMoulin() : Vérification et gestion des moulins.
- is_win() : Vérifie si un joueur a gagné.

V. Améliorations et qualité du jeu

1. Affichage amélioré

- Utilisation des **couleurs en console :
  - Rouge pour le Joueur 1 (`\033[31m`)
  - Vert pour le Joueur 2 (`\033[32m`)
  - Jaune pour le dernier coup joué (`\033[33m`)

2. Vérification des entrées utilisateur

- Ajout d'un contrôle des entrées utilisateur pour éviter les bugs lorsque l'utilisateur entre un caractère au lieu d'un nombre.
- Utilisation de « while (getchar() != '\n') », pour vider le buffer d'entrée et éviter des erreurs avec `scanf()`.

VI. Version console

1. Commandes utilisées

- `system("cls")` pour effacer l'écran sur Windows.
- `printf()` pour afficher les éléments du plateau.
- `scanf()` pour lire les entrées utilisateur.

2. Gestion des interactions

- L'utilisateur entre **le numéro d'une case** pour placer ou déplacer un pion.
- Vérification des entrées pour éviter les erreurs.

VII. Problèmes rencontrés et solutions

1. Problème des 3 pions restants

Problème : Lorsqu'un joueur n'a que 3 pions, il peut se déplacer où il veut, mais la fonction isValidToMove() limitait les mouvements.

Solution : Modification de isValidToMove() pour permettre **le déplacement libre** si un joueur a 3 pions.

2. Gestion des entrées utilisateur

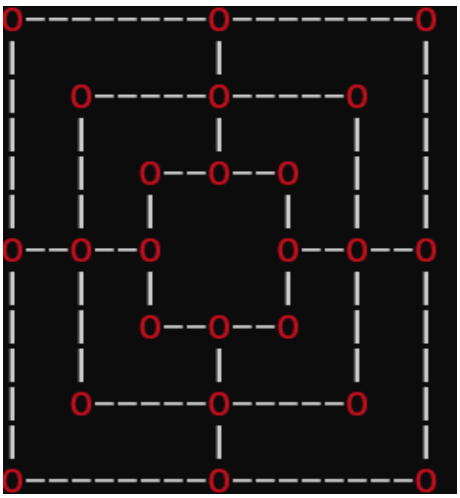
Problème : Si l'utilisateur entre une lettre au lieu d'un nombre, le jeu plante.

Solution : Utilisation de scanf() avec validation et `while (getchar() != '\n');` pour éviter les erreurs.

3. Symboles choisis :

Problème :

-Vu que par défaut, chaque case vide du plateau est représentée par un tableau par un ‘O’, choisir un O comme symbole du joueur constitue un problème.



-Le même problème se pose si le joueur choisit le même symbole de la machine, c’est-à-dire ‘m’, que l’on a considéré comme le symbole par défaut de la machine.

Solution :

On définit la fonction IsDiffMachineAndBoard(), qui rejette le symbole choisi si ce dernier est ‘O’ ou ‘m’.

4) Problème au niveau de la modélisation du jeu :

Difficulté à organiser les fonctions et structurer le code, notamment la *modélisation du tableau de jeu* et des *éléments comme les pions et les connexions*.

Solution : Représentation du plateau avec un tableau board [SIZE].

5) Fonctions critiques

a) Moulin : Vérification correcte d’un moulin :

+Problème : La fonction doit détecter si trois pions alignés forment un moulin en prenant en compte les différentes configurations possibles.

Solution : Vérification des alignements horizontaux et verticaux avec board[i].

b) checkAndHandleMoulin : Gérer la capture après un moulin**

+Problème :

Lorsqu'un joueur forme un moulin, il doit pouvoir retirer un pion adverse, mais *pas compter deux fois le même moulin*.

Solutions mises en place : Vérification de la formation d'un moulin avant d'autoriser la capture d'un pion.

c) isValidToMove : Vérification des déplacements possibles**

Problème :

- Vérifier que le pion peut bouger uniquement vers les cases adjacentes vides.

- Si le joueur n'a plus que 3 pions, il doit pouvoir sauter n'importe où.

Solutions mises en place :

- Vérification des déplacements avec le tableau adjacences[SIZE][4].

-Gestion du cas où un joueur n'a plus que 3 pions (déplacement libre).

6) Exécution et liaison des fichiers :

Problème :

-Le jeu est réparti en plusieurs fichiers .c, mais il faut les exécuter ensemble et bien les lier avec un fichier .h.

Solutions mises en place*

-Création d'un fichier .h pour contenir les prototypes des fonctions et les variables globales.

7) Gestion des variables globales et locales :

Problème :

- Modifier une *variable globale* dans plusieurs fichiers.

- Utiliser des *pointeurs* pour transmettre des variables sans conflits.

Solutions mises en place :

-Définition de certaines variables en extern dans le fichier.h.

- Passage de certaines variables par *pointeurs* dans les fonctions.

8) Gestion des entrées utilisateur :

Problème :

- Un joueur peut entrer une valeur invalide (ex: une lettre au lieu d'un nombre), ce qui fait planter le programme.

Solutions mises en place :

-Vérification des entrées numériques avec scanf().

-Vérification que l'entrée correspond bien à une case vide.

9) Logique du jeu :

a) TourDePlacement et TourDeMvt : Gérer le déroulement du jeu

Problème :

- Gérer correctement les *deux phases* du jeu :

1. *Phase de placement* (poser les pions).
2. *Phase de mouvement* (déplacer les pions).

Solutions mises en place :

Séparation des phases avec TourDePlacement et TourDeMvt.

b) isBlocked : Vérification des pions bloqués :

Problème :

-Si un joueur ne peut plus bouger *aucun* de ses pions, il doit perdre la partie.

Solution mise en place :

- Vérification des mouvements possibles avec adjacences[SIZE][4].

Conclusion :

Ce projet a été une excellente opportunité pour développer une version complète du jeu de moulin en langage C, avec deux modes de jeu distincts : le mode joueur contre joueur et le mode joueur contre machine. Le mode joueur contre machine offre deux niveaux de difficulté, avec une IA qui, en mode avancé, utilise une stratégie réfléchie pour maximiser ses chances de victoire, rendant l'expérience de jeu plus réaliste et stimulante.

Le cœur du projet repose sur une modélisation efficace du plateau de jeu et la gestion rigoureuse des règles, assurant que les phases de placement et de mouvement des pions se déroulent sans accroc.

L'ajout de contrôles d'entrées, de validations et de vérifications pour les mouvements permet de garantir une expérience utilisateur fluide et sans erreurs. Le jeu permet également une gestion adéquate des cas particuliers, comme les situations où un joueur a moins de trois pions, et assure une logique de victoire ou de blocage des adversaires.

Ce projet a non seulement permis de renforcer les compétences en programmation C, mais il a également été une excellente occasion d'explorer la conception et la structure des jeux de société dans un environnement virtuel. En offrant à la fois un challenge pour le joueur humain et une interface conviviale, ce jeu de moulin constitue un accomplissement enrichissant, tout en étant un outil didactique pour ceux souhaitant apprendre les bases de la programmation et de l'intelligence artificielle dans le cadre de jeux interactifs.