Compte rendu : Projet Othello

DUEZ Valentine - 12107343,
ERAMIL Kadir - 12007772,
LUGINBUHL Valentin - 12214021
Université Grenoble Alpes

PROJET
Intelligence artificielle
M1 MIASHS IC S7.

Année universitaire 2024-2025

Contents

1 Introduction

Dans le cadre de notre première année de Master MIASHS (Mathématiques et Informatique Appliquées aux Sciences Humaines et Sociales), parcours Informatique et Cognition, nous devons mener, durant le premier semestre, un projet dans notre cours "Intelligence Artificielle".
Dans ce projet, nous devons coder un jeu ou une application qui comporte une stratégie. Ce projet vise à nous familiariser avec les algorithmes utilisés en Intelligence Artificielle, à développer nos compétences informatiques en développant une application pour deux joueurs, ainsi qu'en implémentant un ou des algorithmes pour que deux IA puissent s'affronter ou qu'un humain puisse affronter l'IA .
Pour développer ce projet, nous devons chercher des algorithme précis capable de résoudre le plus efficacement notre problème, et développer une interface graphique. Notre groupe a choisit le jeu "Othello" pour ce projet. Nous allons vous présenter dans ce document, le déroulement de notre projet et le développement de celui-ci.

2 Présentation du jeu

L'Othello est un jeu de stratégie ou deux joueurs s'affrontent. L'un a les pions noirs et l'autre les pions blancs. Les joueurs posent leurs pions sur une grille de 8x8. Le but est d'avoir le plus de ses pions sur le plateau. Le jeu se termine lorsque le plateau est plein ou bien lorsque les deux joueurs ne peuvent plus jouer. Le plateau de départ se présente de cette manière :

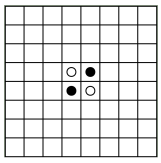

Figure 1: Plateau de départ

Le joueur qui possède les pions noirs commence toujours, puis les deux joueurs jouent chacun leur tour. Si le joueur suivant ne peut plus joueur l'autre joueur rejoue si cela lui est possible. Le joueur ne peut poser son pion qu'à côté ou en diagonale d'un pion déjà posé. Et il faut que le pion qu'il pose soit aligné à un de ses autres pions et qu'il encadre au moins un pion adverse.
Nous avons choisit ce jeu car il est facile à prendre en main et il est facile de développer des stratégies gagnantes. Comme le fait de prendre les angles, de ne pas sortir du carré central 4x4 au risque de ne pas pouvoir poser sur les bords et donc d'encadrer plus facilement les autres pions et bien d'autres.

3 Gestion de projet

Nous avons utilisé l'application Trello pour lister les taches principales, les détailler et les attribuer, ainsi que pour envoyer une notification bi-mensuels à notre professeur pour qu'il constate notre avancement (cf Annexes Trello 2 3). Nous avons aussi utilisé un diagramme de Gantt pour mieux prévoir le temps que les taches allaient nous prendre, qui a été plutôt bien respecté. Cependant, vers la fin du projet, nous avons repoussé certains délais, car le rendu final a été repoussé (Cf Annexes Diagramme de Gantt 4 5).
Pour des raisons que nous expliquerons ci-dessous nous avons préféré commencer par l'interface graphique puis Humain vs Humain, mais il nous était compliqué de coder en même temps c'est pourquoi nous avons fait du pair-programming ce qui explique qu'une seule personne push sur le git.

3.1 Déploiement

Une logique d'intégration et de déploiement continue à été intégré au projet.
La branche "Main" sur GitHub est connectée à un déploiement via le service Vercel grâce à des "jobs" décrit pour les GitHub Actions. A chaque mise à jour de la branche, le projet est déployé sur un serveur et accessible via un lien.
Afin d'éviter tout problème de déploiement, les tests contenus dans le dossier "Tests" sont lancés automatiquement lors d'une modification de la branche "Main" suite à une acceptation de Merge Request par exemple. La partie tests sera décrite plus bas, mais cela permet d'éviter toute régression du code lors de l'implémentation et permet aussi de vérifier une dernière fois que les tests de la fonctionnalités ajoutés sont bien valides. Le déploiement via Vercel est dépendant de la réussites des tests, si un seul tests échoue, le déploiement de la nouvelle version est annulé, cela garanti au mieux une version accessible via le lien stable et fonctionnel.

4 Interface graphique

Nous avons choisit de commencer par l'interface graphique, la logique de la grille étant déjà modélisée grâce au travail pratique n°2 fournit par le professeur. Le framework utilisé pour l'aspect graphique est VueJS. Ce dernier est relativement léger et apporte un minimum de fonctionnalités comme les composants HTML ou le routing, ce qui est suffisant pour créer les deux pages dont nous avons besoin. L'application est découpée en plusieures parties, une vue "Menu" qui représente la page d'accueil avec les règles du jeu et les diffèrents modes de jeux, "Player" (Human vs IA), "MultiPlayer" (Human vs Human) et "Spectator" (IA vs IA). avec, pour chacun, une icône associée pour améliorer l'affordance du menu et pour que l'utilisateur sélectionne plus facilement le mode de jeu souhaité (Cf Annexe Menu 6).
Chaque mode ramène sur une seconde vue "Board" qui est composé d'un composant "Grid" qui comporte toute la logique de la grille de jeu de l'othello.

Nous avons choisit de reprendre les codes couleurs de l'othello avec le vert, les pions noirs et les pions blancs. Nous n'avons pas fait le plateau de la couleur verte habituelle pour une meilleur lisibilité de la grille. Nous avons ajouté un chronomètre pour le confort de l'utilisateur (Cf Annexe Plateau de jeu 7).
Enfin, lorsque la partie est terminée, le nombre de chaque pions est compté et le gagnant est annoncé à l'aide d'un message type (Cf Annexe Fin de partie 8).

5 Humain vs Humain

La version de l'othello que nous avons développé permet à 2 joueurs de s'affronter. Chacun sont tour, et à l'aide du clic de la souris, les joueurs peuvent jouer les coups autorisés par les règles du jeu. Les coups possibles sont affichés en gris à l'aide d'algorithme de recherches.
Pour ce faire, sur la case libre, c'est à dire une case sans pion, on cherche sur chaque direction (côtés et diagonales) si un pion de la même couleur est aligné, si c'est le cas on regarde s'il y a des pions capturables entre les deux pions du joueur. Si ce n'est pas le cas, on cherche sur la case suivante jusqu'à la limite du plateau de jeu. On vérifie de cette manière que le coup du joueur est valide. Si aucun coup n'est valide pour le joueur actif, il passe son tour (Cf Annexe Plateau de jeu 7).
Lorsque le joueur pose un pion, on vérifie si son coup est valide et on change la couleur des pions de l'adversaire pris entre les pions dans toutes les directions à l'aide du même algorithme de recherche que précédemment. On recalcule les coups possibles pour le prochain joueur pour le tour suivant. A chaque fois qu'un pion est posé, une nouvelle grille est créée et redessinée dans le DOM.
La partie s'arrête quand les deux joueurs n'ont plus de coups valide. On compte le nombre total de pions de chaque joueur, le joueur qui possède le plus de pions sur le plateau est le gagnant annoncé (Cf Annexe Fin de partie 8).

6 IA

Explication mode de jeu.
explication choix algorithme de l'IA
exemple certain choix de l'algo.

6.1 Recherche de l'IA

Algorithme	**Principe**	**Points positifs / négatifs**	**Compatibilité avec Othello**
Minimax (Alpha-Beta Pruning)	Explore les coups en maximisant les gains pour un joueur et en minimisant ceux de l'adversaire. Optimisé par l'élagage Alpha-Beta pour réduire les branches inutiles.	+ Décisions optimales si suffisamment profond. - Gourmand en ressources, dépend de la qualité de l'évaluation.	Adapté
A* (A-star)	Trouve le chemin optimal dans un graphe en combinant coût actuel et estimation restante.	+ Optimal avec heuristique admissible. - Inefficace pour jeux adversariaux.	Incompatible
BFS (Breadth-First Search)	Explore tous les coups couche par couche jusqu'à atteindre une certaine profondeur.	+ Solution garantie pour petit espace. - Inefficace pour jeux complexes.	Peu compatible
DFS (Depth-First Search)	Explore les coups en profondeur avant de revenir en arrière.	+ Peu gourmand en mémoire. - Se perd dans branches non optimales.	Peu compatible
AC-3 (Arc-Consistency)	Résout les problèmes de satisfaction de contraintes (CSP) en rendant les relations entre variables cohérentes.	+ Simplifie les problèmes CSP. - Inefficace pour jeux dynamiques.	Incompatible
IA Aléatoire	Choisit un coup au hasard parmi les options possibles.	+ Simple et rapide. - Aucun aspect stratégique.	Compatible mais non compétitive

Table 1: Comparaison des algorithmes d'IA pour Othello.

7 A FAIRE

1) se documenter en amont 2) moteur de l'appli 3) série de tests 5) rapport decriré l'appli les tecs ia les résultats et coclsio git mettre à jour automatiser les test

7.1 Implémentation de l'IA

8 Tests

9 Annexes

9.1 Dépôt GITHUB

https://github.com/lelinguine/othello-project

9.2 Trello

https://trello.com/b/BVCk8acy/projet-othello

9.3 Trello : Organisation et exemple de tâche plus détaillée

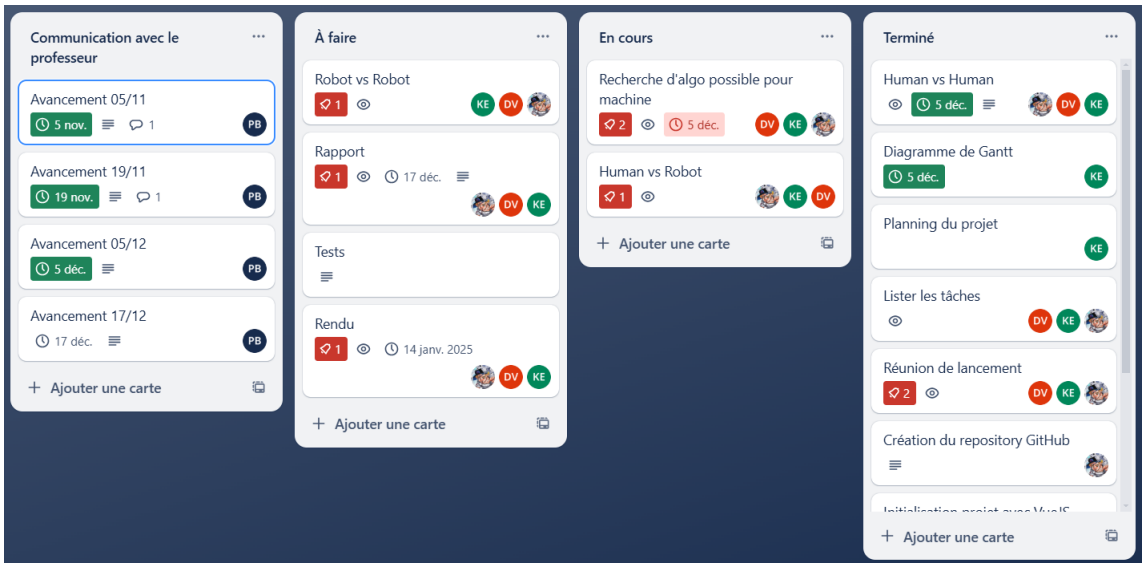

Figure 2: Capture d'écran du trello

Figure 3: Exemple de tâche détaillée

9.4 Diagramme de Gantt

Figure 4: Diagramme de Gantt

9.5 Diagramme de Gantt corrigé

Figure 5: Digramme de Gantt corrigé

9.6 Menu

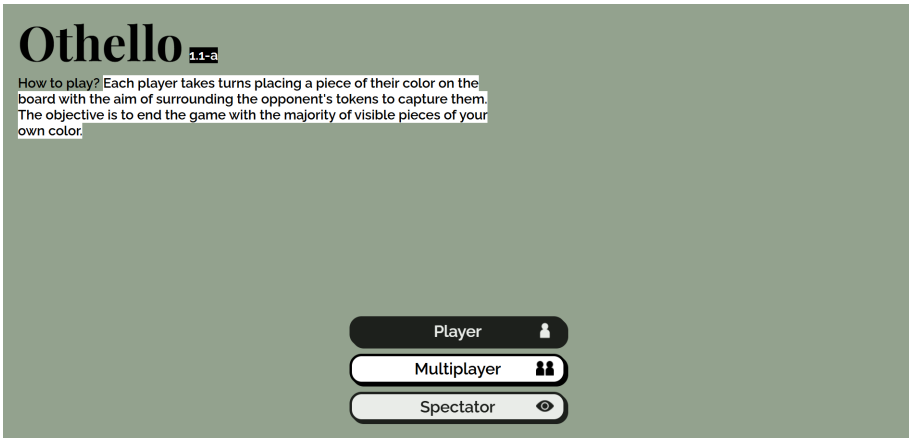

Figure 6: Menu

9.7 Plateau de jeu

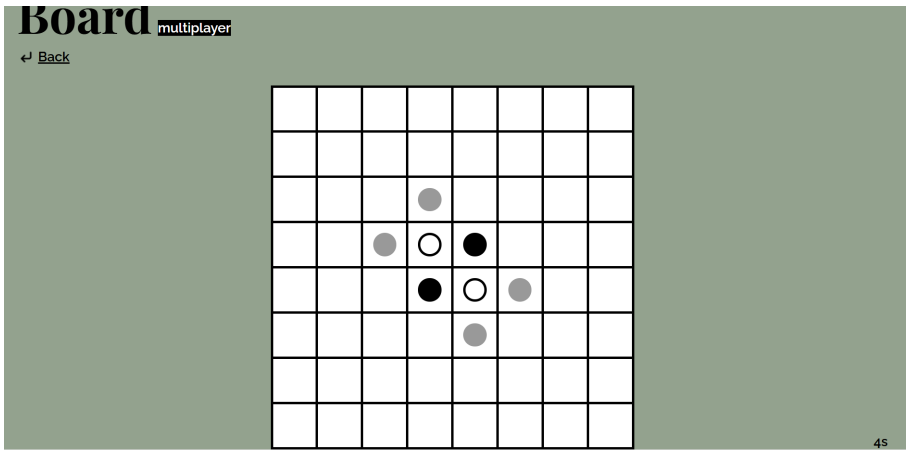

Figure 7: Plateau de jeu

9.8 Fin de partie

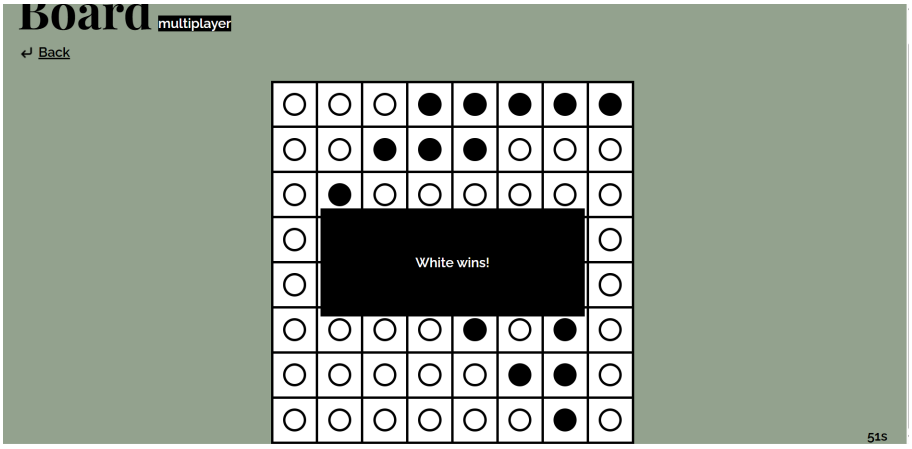

Figure 8: Fin de partie