



UNIVERSITE IBN ZOHR FACULTE DES SCIENCES

Département Informatique
Filière Sciences Mathématiques et Informatique

PFE

Présenté par : ASAOKOUR Ihsane
N'ANYI Loubna

Pour l'obtention de la
Licence en Sciences Mathématiques et Informatique

JEU VEDIO : OTHELLO

Encadré par : Mr. CHARFI Said

Soutenu le : 06 Juillet 2021

Année universitaire : 2020-2021

Cette page a été laissée blanche intentionnellement.

Dédicaces

Les mots ne suffisent guère pour exprimer l'attachement, l'amour et l'affection que je porte pour vous. Je vous dédie ce travail avec tous mes vœux de bonheur, de santé et de réussite.

A l'âme de mon cher père, qui était mon soutien et la source de ma joie, qui m'a toujours aidé sans excuse et m'a donné toute l'énergie positive. Je prie toujours pour que vous soyez parmi les gens de paradis.

A ma chère mère, mon petit frère et ma petite belle sœur qui sont avec moi tout le temps et qui font leur meilleur pour m'avoir heureuse. Que dieu les préserve en bonne santé et leur accorde une longue vie.

Je remercie également mes amis pour leur encouragement.

N'ANYI Loubna

Dédicaces

Aucune dédicace ne saurait exprimer mon respect, mon amour éternel et mon considération pour les sacrifices que vous avez consenti pour mon instruction et mon bien-être.

J'adresse mes profondes gratitude à ma famille, Mon père, ma mère, ma sœur et leur petite famille et mon frère qui est toujours à mon côté, pour son soutien moral et matériel et pour ses sacrifices consentis. Ils ont tout fait pour mon bonheur et ma réussite. Que dieu les préserve en bonne santé et leur accordent une longue vie.

Je remercie également tous mes amis pour leur encouragement.

ASAKOUR Ihsane

Remerciements

Avant d'entamer ce rapport, nous profitons de l'occasion pour remercier tout d'abord notre encadrant Monsieur **CHARFI Said** pour son écoute, sa disponibilité, sa bienveillance, son implication, ses directives précieuses, et surtout pour la qualité de son suivi durant toute la période de notre projet.

Nous tenons à remercier sincèrement chaque membre de jury pour la patience et l'intérêt d'avoir accepté d'évaluer ce projet.

Finalement, Nous présentons notre gratitude à toute personne ayant contribué de près ou de loin à réalisation de ce modeste travail.

Résumé

Le présent rapport a été préparé dans le cadre du projet de fin d'études pour l'obtention de la licence en science mathématique et informatique, parcours de génie logiciel. Il a été réalisé entre Mars et Juin 2021. Ce projet a pour but de développer le jeu vidéo Othello intelligent en utilisant l'algorithme MinMAX et AlphaBeta.

Pour ce faire, le projet a été élaboré en quelques grandes étapes, à savoir, la première étape c'est de faire des recherches sur différents algorithmes et de voir les avantages d'un par rapport aux autres, La deuxième consistait à étudier le projet et le langage de programmation convenable. La troisième étape était consacrée à la préparation de l'environnement de travail et les outils à utiliser. La dernière étape, portait sur l'analyse, conception, le développement et tests.

Pour le développement du projet nous avons utilisé le langage CPP, pour la partie conception nous avons opté pour le langage UML. Puis, nous avons exploité plusieurs Framework et outils tels que : CodeBlocks, Gtkmm, MSYS2 a fin de développer notre application.

A travers ce document, nous allons décrire plus en détails chaque partie de la réalisation de ce projet.

Mots clés : MiniMax, AlphaBeta, Cpp, Uml.

Abstract

This report has been prepared as term end of project to get the degree in mathematical and computer science studies, Software engineering course. It is carried out between March and June 2021. This project intended to develop the intelligent Othello video game using the MinMAX and AlphaBeta algorithms.

To do so, the project was developed in a few major steps, the first step is prepare a literature study on different algorithms and to see the advantage of each one, and the second step was the study of the project and the appropriate programming language. The third step was the preparation of the working environment and the tools to be used. The last step was analysis, design, development and testing of the application.

For the development of the project, we used the CPP programming language, for the conception we used UML language, and for the development we use many framework and tools such as : CodeBlocks, Gtkmm, MSYS2.

Through this document, we will describe, in details, each part of the realization of this work.

Key words: MiniMax, AlphaBeta, Cpp, Uml

Table des matières

Dédicaces.....	iii
Dédicaces.....	iv
Remerciements	v
Résumé	vi
Abstract.....	vii
Table des matières	viii
Liste des figures.....	x
Liste des abréviations	xiii
Introduction générale.....	1
Chapitre 1 :Contexte général du projet	2
1. Contexte Du Projet :	3
1.1 Intelligence Artificiel (IA) :.....	3
1.2 Théorie Des Jeux :	4
1.3 L'algorithme MinMax (MiniMax) :	5
1.4 L'algorithme Alpha-Beta :.....	7
2. Description Du Projet :	9
3. Problématique Et Objectifs :	10
4. Conduite Du Projet :	10
5. Planification Du Projet :.....	10
Chapitre 2 : Analyse et conception	12
1. Spécification des Besoins :.....	13
1.1 Les Besoins Fonctionnels :	13
1.2 Les Besoins Non Fonctionnels :	15
2. Conception :.....	15
2.1 Diagramme de cas d'utilisation :	15
2.2 Diagramme de séquence :	17
2.3 Diagramme de classe	17
Chapitre 3 : Etude technique et environnement	21
1. Capture des besoins techniques :.....	22
2. Architecture adoptée :.....	22
3. Choix des langages :	23
4. Framework utilisé :.....	24
Chapitre 4 : Réalisation, Interfaces, Test	26
1 Environnement matériels :.....	27
2 Les logiciels et les outils :	27
3 Interface :	28
4 Tests :.....	35
4.1 Game type => One Palyer :	35
4.2 Game type => Two Palyer :.....	39

4.3 Conclusion :	39
Conclusion générale et perspective	40
Bibliographie	42
Annexes	43

Liste des figures

Figure 1 : Jeu POCKER	3
Figure 2: Jeu RUBIK'S CUBE	3
Figure 3 : Jeu BACKGAMMON	4
Figure 4 : Jeu ECHECS.....	4
Figure 5 : Schema explicatif d'algorithme MinMax.....	5
Figure 6 : pseudo code d'algorithme Minimax.....	6
Figure 7 : Les Positions Possibles Par Application De MinMax.....	7
Figure 8 : pseudo code d'algorithme AlphaBeta	8
Figure 9 : Schéma explicatif d'algorithme Alpha Beta.....	8
Figure 10 : Jeu OTHELLO au début.....	9
Figure 11 : Jeu OTHELLO à la fin	9
Figure 12 : Diagramme de Gantt.....	11
Figure 13 : Plan De Réalisation.....	11
Figure 14 : Le Damier	13
Figure 15 Les pions	13
Figure 16 : Les mouvements possibles pour le noir.....	14
Figure 17 : Fin du Jeu.....	14
Figure 18 : diagramme de cas d'utilisation générale	16
Figure 19 : diagramme de cas d'utilisation détaillé.....	16
Figure 20 : Diagramme de sequence generale	17
Figure 21 : Diagramme de classe	20
Figure 22 : Interface De Jeu	28
Figure 23 : Interface Si Le Joueur Clique Sur Strat.....	29
Figure 24 : Interface S'il choisit One Player ou Two Player.....	30
Figure 25 : Début De Jeu.....	30
Figure 26 : Barre d'outils	31
Figure 27 S'il veut Revenir Au Menu Principale	31
Figure 28 : Message Indique Le Gagnant	32
Figure 29 : Les Paramètre de Jeu à Modifier	32
Figure 30 : Tutoriel de Jeu	33
Figure 31 : la fin de tutoriel.....	34
Figure 32 : Regarder un match (Ordinateur Vs Ordinateur).....	34
Figure 33 : Choisir le niveau 'Easy'	35
Figure 34 : Le jeu au début et au milieu.....	36
Figure 35 : le joueur gagne (score = 35)	36
Figure 36 : Choisir le niveau 'Moyen'	37
Figure 37 : l'ordinateur gagne (score = 36)	37
Figure 38 : Choisir le niveau 'Hard'.....	38

Figure 39 : L'ordinateur gagne (score = 55).....	38
Figure 40 : graphe illustratif.....	39

Liste des tableaux

Tableau 1 : Fonction de la classe Game.....	18
Tableau 2 : Fonction de la classe Board.....	19
Tableau 3 : Fonction de la classe Help_b.....	19
Tableau 4 : Fonction de la classe Start_b.....	19
Tableau 5 : Fonction de la classe Option_b	19
Tableau 6 : Fonction de la classe Fenetre	19
Tableau 7 : comparaison de quelques langages de programmation.....	23
Tableau 8 : Quelque Classe De GTKMM.....	25
Tableau 9 : comparaison de gain par rapport les niveaux.....	39

Liste des abréviations

Acronyme	Signification
UML	Unified Modeling Language
AI	Intelligence Artificielle
POO	Programmation Orientée Objet
MV	Machine Virtuel
API	Application Programming Interface
CSS	Cascading Style Sheets
IDE	Integrated Development Environment

Introduction générale

A notre époque, les jeux vidéo représentent une des industries les plus dynamiques au monde. Ils occupent une place particulière surtout pour les jeunes, ces jeux comptent comme des outils d'amusants et parfois comme un loisir.

Les jeux vidéo sont différents, il existe des jeux hors ligne et d'autres en ligne, ce dernier type nécessite plusieurs joueur c'est pour ça on les appelle multijoueur. Il y a différents jeux de ce type comme le jeu OTHELLO.

OTHELLO est un jeu de réflexion à deux joueur (ordinateur Vs ordinateur, ordinateur Vs player1 ou player1 Vs player2), le principe de ce jeu c'est d'avoir le grand nombre possible des pions à la fin de jeu.

Notre mission en tant qu'équipe est l'obtention de la licence en science mathématique et informatique, parcours : génie logiciel, est de développer le jeu Othello intelligent pour être pratiquement impossible de battre le programme par un humain.

Pour la réalisation du projet nous suivons une démarche qu'est bien expliquée dans le présent rapport. Ce dernier s'articule essentiellement autour de quatre principaux chapitres :

- ✓ Chapitre 1 : présente le contexte générale du projet.
- ✓ Chapitre 2 : traite les spécifications des besoins et la partie de conception détaillée incluant la modélisation avec UML.
- ✓ Chapitre 3 : consacrée à l'étude technique et l'environnement, de plus les outils nécessaires pour faire la réalisation de ce projet.
- ✓ Chapitre 4 : expose la réalisation du jeu, présentation des interfaces et aussi des tests.

Finalement, on termine cette mémoire par une conclusion générale qui récapitule les différentes parties de notre travail et présente ses principales perspectives.

Chapitre 1 :

Contexte général du projet

Cette partie du mémoire a pour objectif de donner un cadre général du projet de fin d'études. Premièrement nous présenterons l'intelligence artificielle. Ensuite Nous expliquerons les algorithmes utilisés, passant par la problématique et les objectifs à atteindre, avant de finir par le planning et la conduite du projet.

1. Contexte Du Projet :

1.1 Intelligence Artificiel (IA) :

Il n'y a pas longtemps qu'on a écouté ‘L'intelligence Artificiel ‘, cette expression signifie tout simplement, l'ensemble des théories et des techniques mises en œuvre afin d'apporter aux machines une forme d'intelligence.

Intelligence Artificielle est utilisée dans plusieurs domaines, comme l'industrie des jeux vidéo notamment les jeux de stratégie par exemple ECHECS, POCKER, RUBIK'S CUBE, BACKGAMMON



Figure 1 : Jeu POCKER



Figure 2: Jeu RUBIK'S CUBE



Figure 3 : Jeu BACKGAMMON

Le jeu BACKGAMMON est le premier jeu qui a été battu par une intelligence artificielle en 1979, ce match jouer entre machine Vs humain (plus grands champions de backgammon au monde), le score final était sans équivoque 7 à 1 ce qui montre la différence au niveau d'intelligence.



Figure 4 : Jeu ECHECS

Ce jeu est développé par l'utilisation de AI. En, 1997 la première victoire d'une machine Vs Humain (le meilleur joueur d'échecs au monde à l'époque). En Plus, en 2018 une autre intelligence artificielle appelée AlphaZero qui est plus amélioré est apparue.

1.2 Théorie Des Jeux :

La théorie des jeux est presque une théorie de la décision en interaction, Elle a pour objectif d'étudier les situations possibles.

Le modèle d'un jeu est constitué de trois éléments : les joueurs, leurs ensembles de stratégies et les règles du jeu. Après avoir caractérisé chacun de ces éléments on passe à représenter les déroulements possibles d'un jeu par un arbre par exemple, suivant toutes les sorties possibles et leurs probabilités d'être les meilleures. Pour la déterminer, il existe plusieurs algorithmes qui nous permettent de trouver le meilleur choix pour gagner.

1.3 L'algorithme MinMax (MiniMax) :

L'algorithme MinMax est un algorithme qui s'applique à la théorie des jeux à deux joueurs. Cet algorithme consiste à trouver pour le joueur qui commence (joueur Max) la meilleure situation qui lui permet de maximiser ses profits parmi toutes les situations à une position donnée. Et pour son Adversaire, il essaie de trouver la situation qui minimise les profits de l'autre joueur. Généralement *MinMax = minimiser la perte maximum*.

La figure suivante représente un arbre de jeu sur lequel on effectue une recherche MinMax :

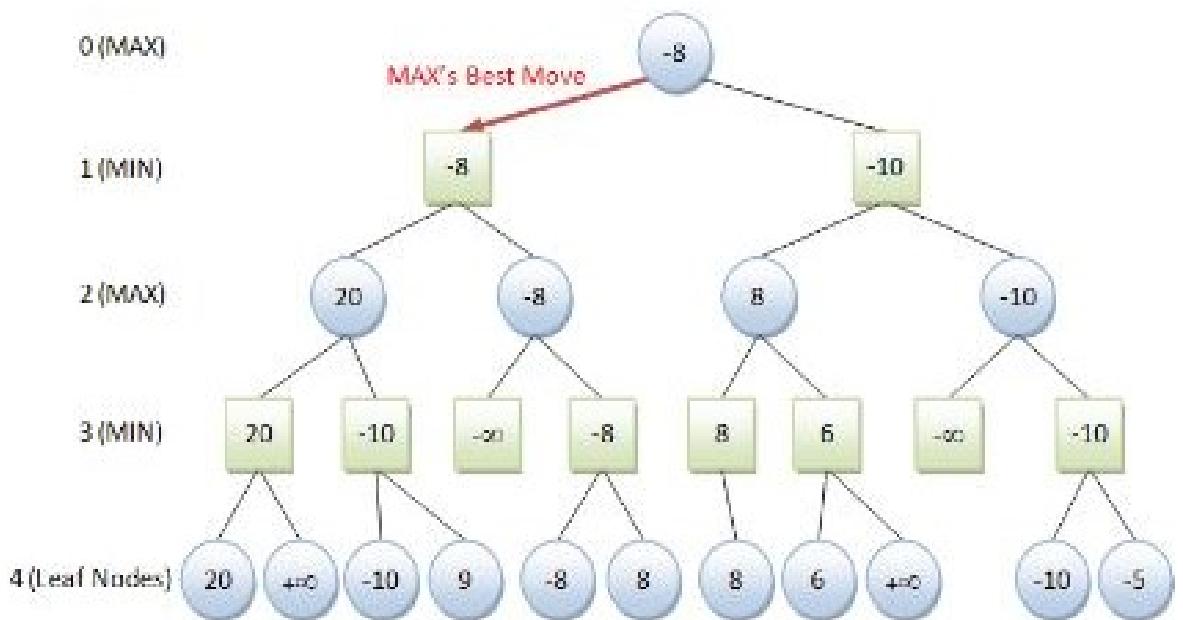


Figure 5 : Schéma explicatif d'algorithme MinMax

- ✓ Chaque branche de l'arbre représente un coup possible.
- ✓ Chaque nœud représente une position du jeu.

Et voilà le pseudo code de cet algorithme :

```

function minimax (board, depth, isMaximizingPlayer) :

    if current board state is a terminal state :

        return value of the board

    if isMaximizingPlayer :
        bestVal = -INFINITY
        for each move in board :
            value = minimax(board, depth+1, false )
            bestVal = max (bestval, value)
        return bestval

    else :
        bestval = +INFINITY
        for each move in board :
            value = minimax (board, depth+1, true)
            bestVal = min (bestVal, value)
        return bestVal

```

Figure 6 : pseudo code d'algorithme Minimax

Pour bien comprendre comment ça marche voici un exemple d'application de MinMax dans le jeu Tic-Tac-Toe pour obtenir le mouvement optimale :

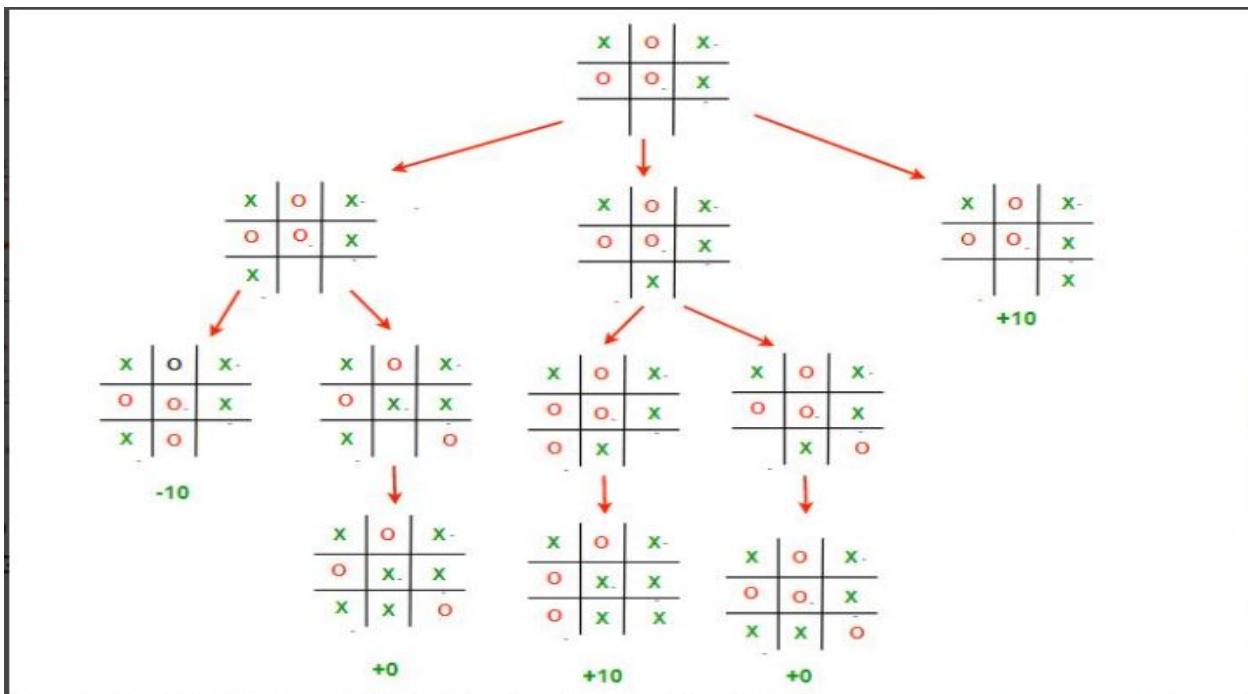


Figure 7 : Les Positions Possibles Par Application De MinMax

1.4 L'algorithme Alpha-Beta :

L'algorithme Alpha Beta ou l'élagage alpha-beta est une version évoluée de l'algorithme MinMax, il est plus économique en termes de temps de calcul et de la mémoire puisque cette technique est capable de couper les sous branches qui n'ont pas besoin d'être visitée (réduire le nombre de nœuds). Cela nous permet de rechercher plus rapidement et d'aller chercher dans des niveaux plus profonds dans l'arborescence. Cet algorithme contient deux paramètres alpha et beta que nous définissons comme suit :

- ✓ Alpha est égale à la valeur sur les feuilles, elle est initialisée au début à -infini, Alpha est la meilleure valeur que le maximiseur peut choisir pour le moment.
- ✓ Beta est égale à la valeur sur les feuilles, elle est initialisée à +infini au début, Bêta est la meilleure valeur que le minimiseur peut choisir pour le moment.

Et voici le pseudo code d'algorithme AlphaBeta :

```

function AlphaBeta (node, depth, isMaximizingPlayer, alpha, beta) :

    if node is a leaf node :
        return value of the node

    if isMaximizingPlayer :
        bestVal = -INFINITY
        for each child node :
            value = AlphaBeta (node, depth+1, false, alpha, beta)
            bestVal = max ( bestVal , value)
            alpha = max ( alpha, bestVal )
            if beta <= alpha;
                break ;
        return bestVal

    else :
        bestVal = +INFINITY
        for each child node :
            value = AlphaBeta (node, depth+1, true, alpha, beta)
            bestVal = min ( bestVal , value)
            beta = min ( beta, bestVal)
            if beta <= alpha;
                break ;
        return bestVal

```

Figure 8 : pseudo code d'algorithme AlphaBeta

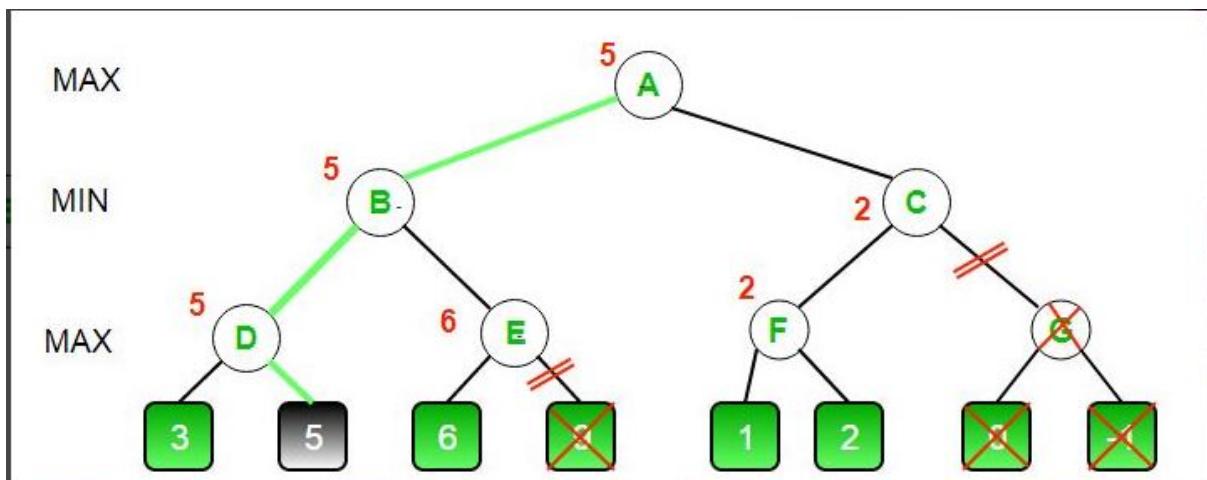


Figure 9 : Schéma explicatif d'algorithme Alpha Beta

2. Description Du Projet :

Othello ou bien Reversi, il a été inventé en 1971 la première fois en Japon. C'est un jeu de réflexion de type multijoueur (nombre maximale de joueur est 2), chacun des joueurs a 32 bicolor pions (les blanches pions d'un côté et les noires de l'autre côté), ils jouent sur une surface unicolore appelée plateau ou othellier qui constituer de 64 cases (8x8/10x10). En début de partie, quatre pions placés au centre comme le montre la figure ci-dessous :

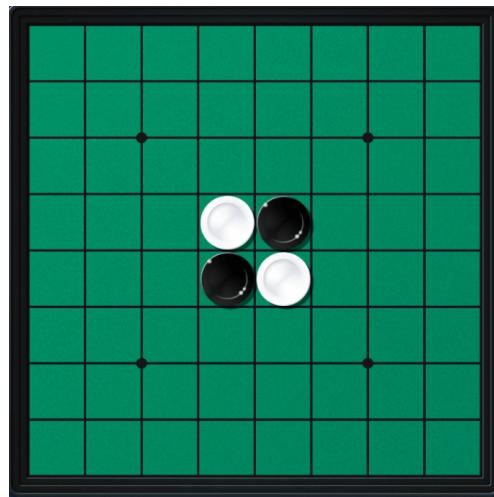


Figure 10 : Jeu OTHELLO au début

Après, chaque joueur pose un seul pion de sa couleur selon des règles précises, puis l'autre joueur pose aussi un seul pion jusqu'à que les deux joueurs ne peuvent plus poser des pions, et c'est la fin du jeu.

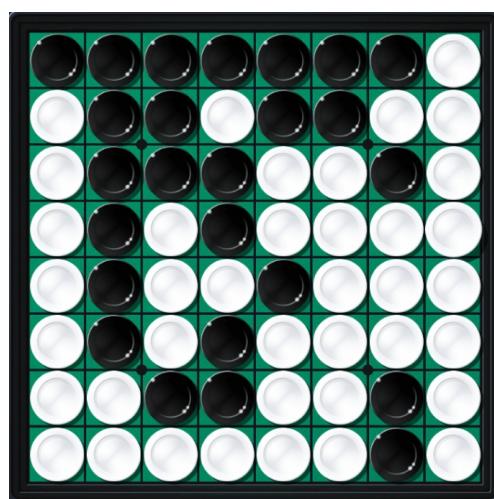


Figure 11 : Jeu OTHELLO à la fin

En fin on compte pour chaque joueur les pions de sa couleur dans le plateau, qui a le plus grand nombre gagne.

3. Problématique Et Objectifs :

L'objectif principal de ce projet est de développer le jeu Othello pour qu'il soit difficile à battre par l'utilisation de différents algorithmes et par l'inclusion de l'Intelligence artificielle. A l'aide de cette dernière technique la machine devient toujours le gagnant devant un homme soit un pourcentage de 100%.

4. Conduite Du Projet :

Pour la création de ce projet nous avons touché plusieurs axes, comme suit :

- a. La recherche de l'historique de jeu Othello, ses règles... .
- b. Etude de la méthode de création des interfaces avec GTKMM et leur sous bibliothèques.
- c. Une recherche sur les algorithmes.
- d. La création de quelques diagrammes pour simplifier la réalisation.
- e. La réalisation du projet par l'utilisation de quelques outils.
- f. Finalement, des tests pour assurer le fonctionnement.

5. Planification Du Projet :

L'une des étapes fondamentales à la mise en œuvre d'un projet est de faire une planification. Cette phase permet de déterminer et d'organiser les tâches du projet pendant une durée précise. Nous avons utilisé le diagramme de GANTT comme outil de planification pour simplifier le suivi d'avancement de projet.

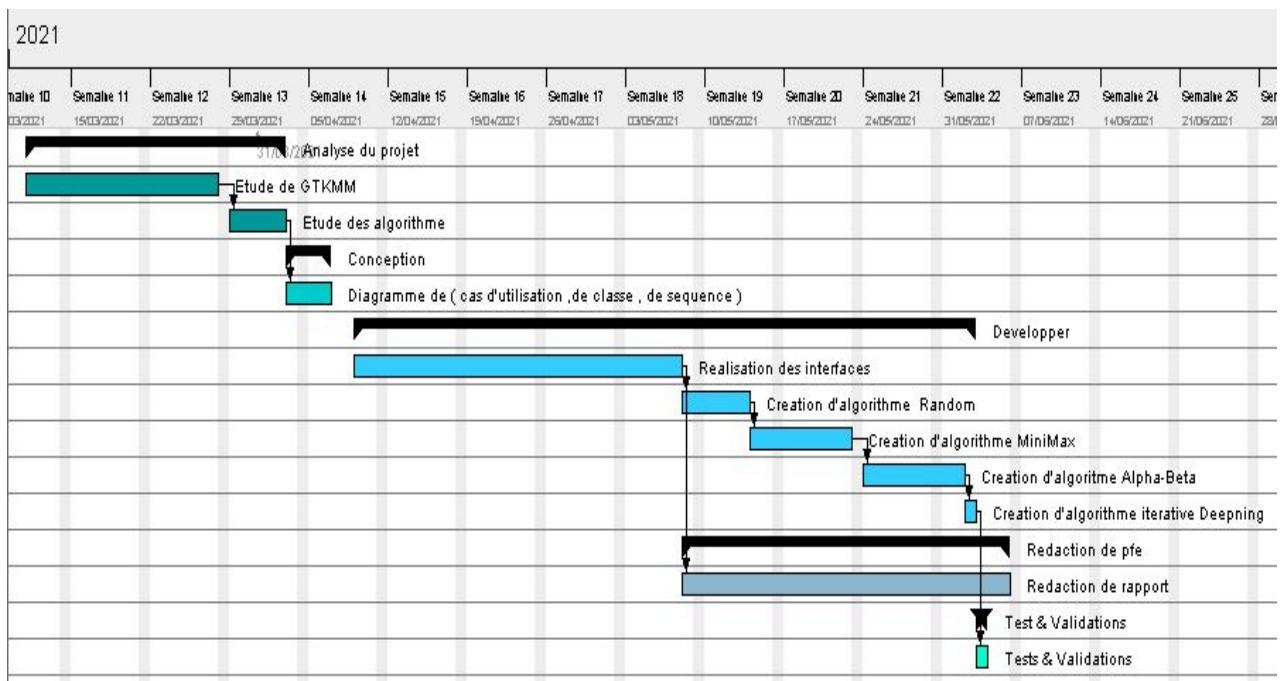


Figure 12 : Diagramme de Gantt

Nom	Date de début	Date de fin
Analyse du projet	11/03/2021	02/04/2021
Etude de GTKMM	11/03/2021	27/03/2021
Etude des algorithmes	29/03/2021	02/04/2021
Conception	03/04/2021	06/04/2021
Diagramme de (cas d'utilis...	03/04/2021	06/04/2021
Développer	09/04/2021	02/06/2021
Réalisation des interfaces	09/04/2021	07/05/2021
Creation d'algorithme Ran...	08/05/2021	13/05/2021
Creation d'algorithme Mini...	14/05/2021	22/05/2021
Creation d'algorithme Alpha...	24/05/2021	01/06/2021
Creation d'algorithme itera...	02/06/2021	02/06/2021
Redaction de pfe	08/05/2021	05/06/2021
Redaction de rapport	08/05/2021	05/06/2021
Test & Validations	03/06/2021	03/06/2021
Tests & Validations	03/06/2021	03/06/2021

Figure 13 : Plan De Réalisation

Chapitre 2 :

Analyse et conception

Au niveau de ce chapitre nous allons représenter l'ensemble des besoins fonctionnels et non fonctionnels. Puis on va mettre l'accent sur la partie conception par l'exposition des différents diagrammes.

1. Spécification des Besoins :

1.1 Les Besoins Fonctionnels :

Pour définir les besoins fonctionnels concernant ce projet, il faut tout d'abord définir tous les termes et les règles de ce jeu.

Le damier : est un tableau unicolore de taille 8x8/10x10 carrées, avec deux pions blanches et deux autres noires positionnés au centre sous forme de lettre comme l'illustre la figure suivante :

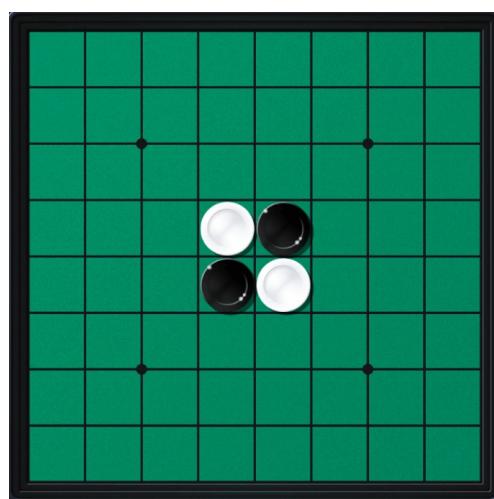


Figure 14 : Le Damier

Les pièces ou Les pions : Ils sont de deux couleurs différentes, les plus populaires sont le noir et le blanc.



Figure 15 Les pions

Chaque joueur a le droit de poser un seul pion à la fois. Par défaut, le pion noir commence, en plaçant le premier pion à un endroit qui enferme un pion de l'adversaire, c'est-à-dire entourer un ou plusieurs pions de l'adversaire avec deux de vos propres pions, on peut enfermer les autres pions en diagonale, horizontale ou verticale.

Par exemple pour le pion noir on peut la poser où il y a le cercle :

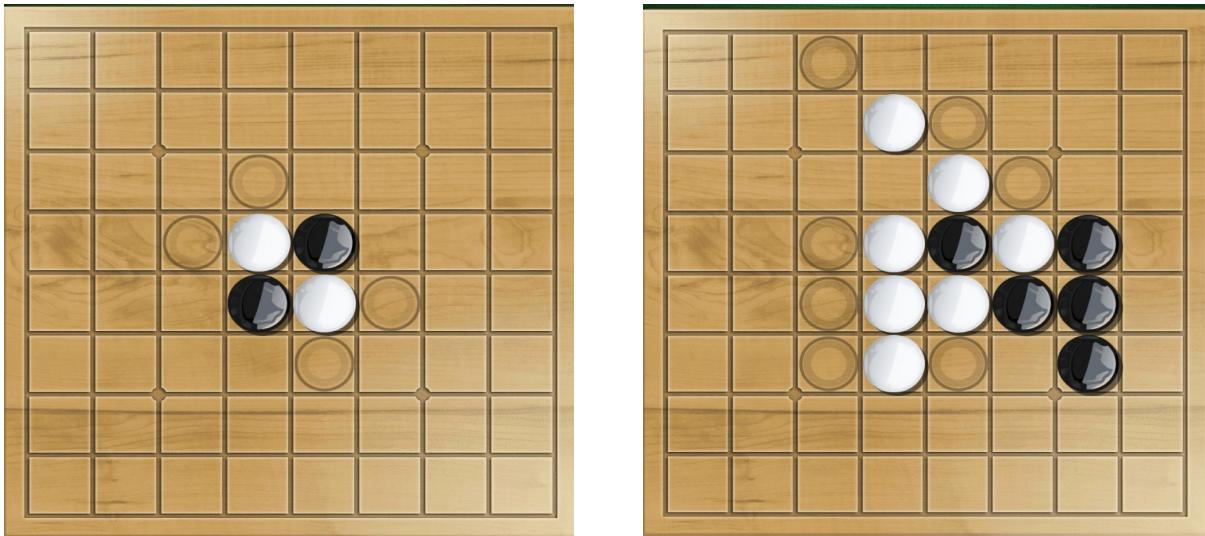


Figure 16 : Les mouvements possibles pour le noir

Si l'un des joueurs ne peut pas enfermer au moins un pion, il est forcé de passer son tour. Par la suite si aucun des joueurs ne peut déplacer son pion, donc la partie s'arrête et le gagnant est celui qu'il a le plus grand nombre de pions sur le plateau.

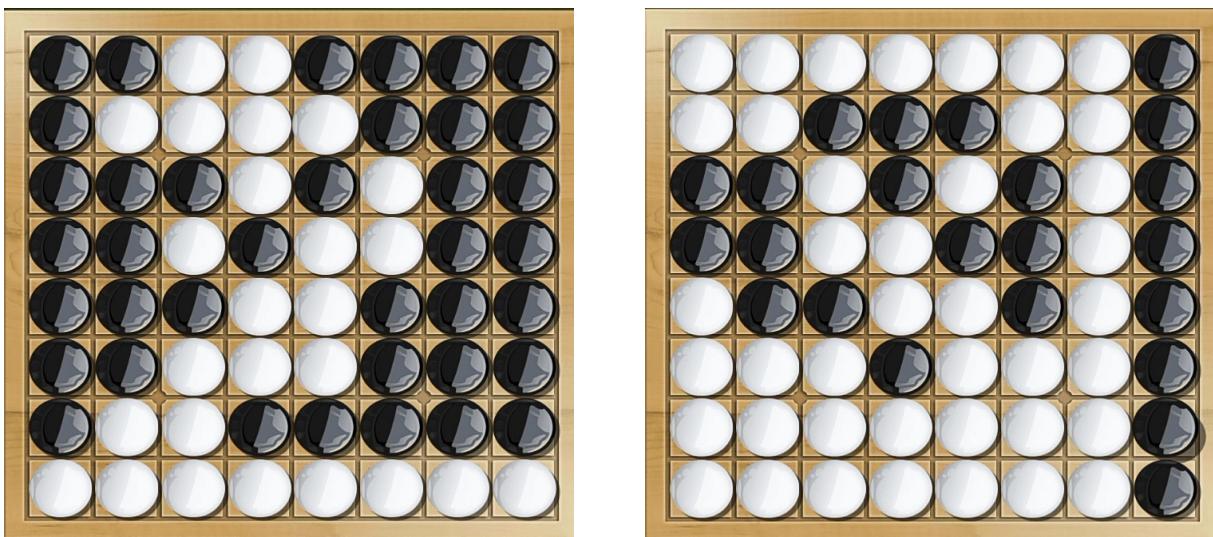


Figure 17 : Fin du Jeu

Apres la fin du jeu, si le joueur veut rejouer on fait un redémarrage du plateau et on le restaure comme au début avec les quatre pions au milieu sous forme d'un X.

1.2 Les Besoins Non Fonctionnels :

Les besoins non fonctionnels sont des besoins qui caractérisent le système en matière de performance, de type de matériel ou le type de conception.

- Ergonomie : le jeu doit être utilisable par toute personne, il faut qu'il soit simple et claire pour adresser tous les âges surtout les personnes qui ne sont pas familiers avec la technologie.
- Performance : le temps de calcul d'un mouvement ne doit pas être trop long pour éviter l'ennui du joueur.
- Portabilité : le jeu doit être multiplateformes est bien fonctionne dans les différents systèmes d'exploitation Windows, Unix et Linux...

2. Conception :

2.1 Diagramme de cas d'utilisation :

Quand le joueur clique sur Start il aura les choix suivant :

- Play : pour jouer après la sélection de type de jeux et de choisir le niveau.
- Options : pour accéder au menu des options pour des simples modifications
- Help : pour l'affichage des règles de jeu.
- Quit : pour quitter.

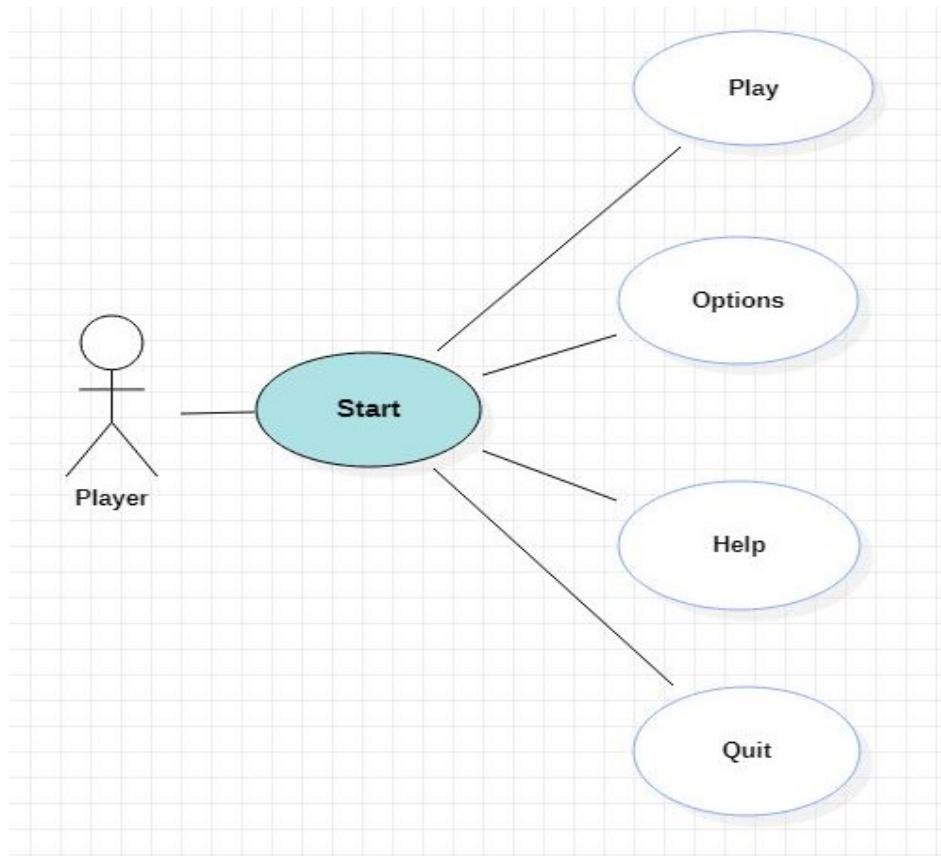


Figure 18 : diagramme de cas d'utilisation générale

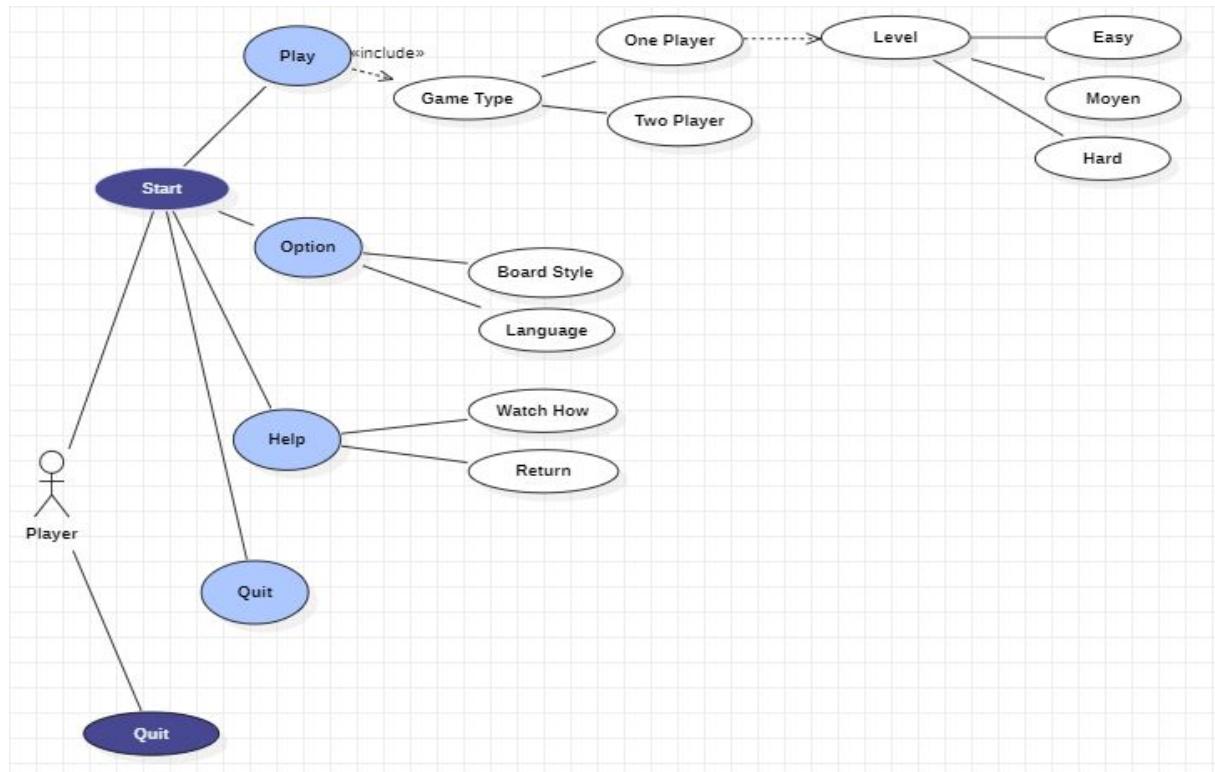


Figure 19 : diagramme de cas d'utilisation détaillé

2.2 Diagramme de séquence :

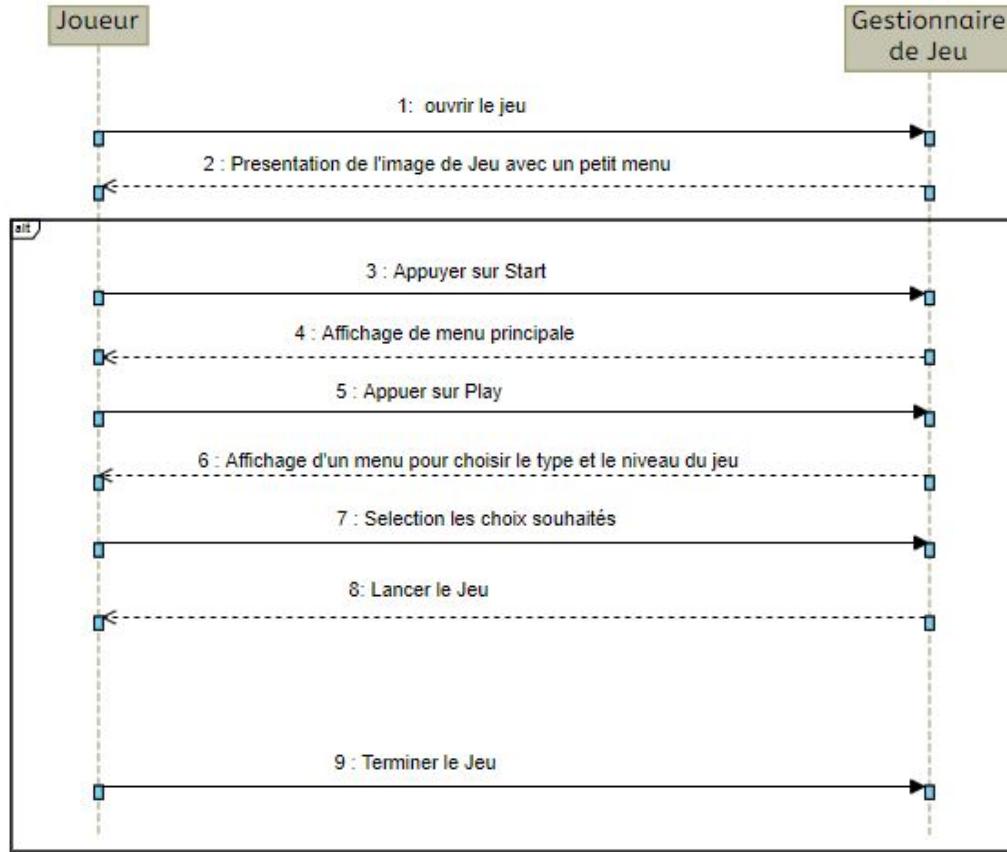


Figure 20 : Diagramme de sequence generale

Ce diagramme montre d'une façon simple le démarche de jeu si le joueur choisi à jouer.

2.3 Diagramme de classe

Le diagramme ci-dessus représente le diagramme de classe de ce projet. Il est constitué de plusieurs classes tel que :

- **GtkWindow** : Cette une fenêtre de niveau supérieur qui peut contenir d'autres widgets.
- **Game** : Cette classe concerne les mouvements, le niveau de difficulté selon le choix du joueur, et le fonctionnement des boutons qui sont apparu

à côté du plateau de jeu et d'autre qui ont été évoquées dans le tableau ci-dessous.

Nom des fonctions
SpaceClicked
RandomFindMove
NewBoard
DoMove
copyBoard
getMoveList
ChangeTiles
flipTiles
heuristic
Ord Vs Ord
minimax
Alpha_beta
iterativeDeepening
On_Cancel_Clicked
On_Stop_Clicked
On_Restart_Clicked
On_Return_Clicked
Menu_About
Menu_Principale
Set_Timer

Tableau 1 : Fonction de la classe Game

- Board : Cette classe est relative à la création de damier, le déplacement des pions, le calcul de score et d'autre.

Nom des fonctions
GetTilePicture
SetTilePicture
setBoard
getBoard
Score
setTiles
getTileStatus
setTileStatus
checkFlip
checkIfAnyValidMoves

validMove
flipPieces
gameOver

Tableau 2 : Fonction de la classe Board

- Help_b : Cette classe concerne le bouton Help .

Nom des fonctions
On_GoBack_Clicked
On_Watch_Clicked

Tableau 3 : Fonction de la classe Help_b

- Start_b : il est à propos de bouton Start .

Nom des fonctions
On_done_clicked
On_game_Clicked
On_howToPlay_Clicked
On_Options_Clicked
Disable_Choices
Activate_Choices

Tableau 4 : Fonction de la classe Start_b

- Option_b : il est à propos de bouton Option :

Nom des fonctions
On_return_clicked
On_done_clicked

Tableau 5 : Fonction de la classe Option_b

- Fenetre : concerne le premier menu apparut lorsqu'on lance le jeu.

Nom des fonctions
On_Start_Clicked

Tableau 6 : Fonction de la classe Fenetre

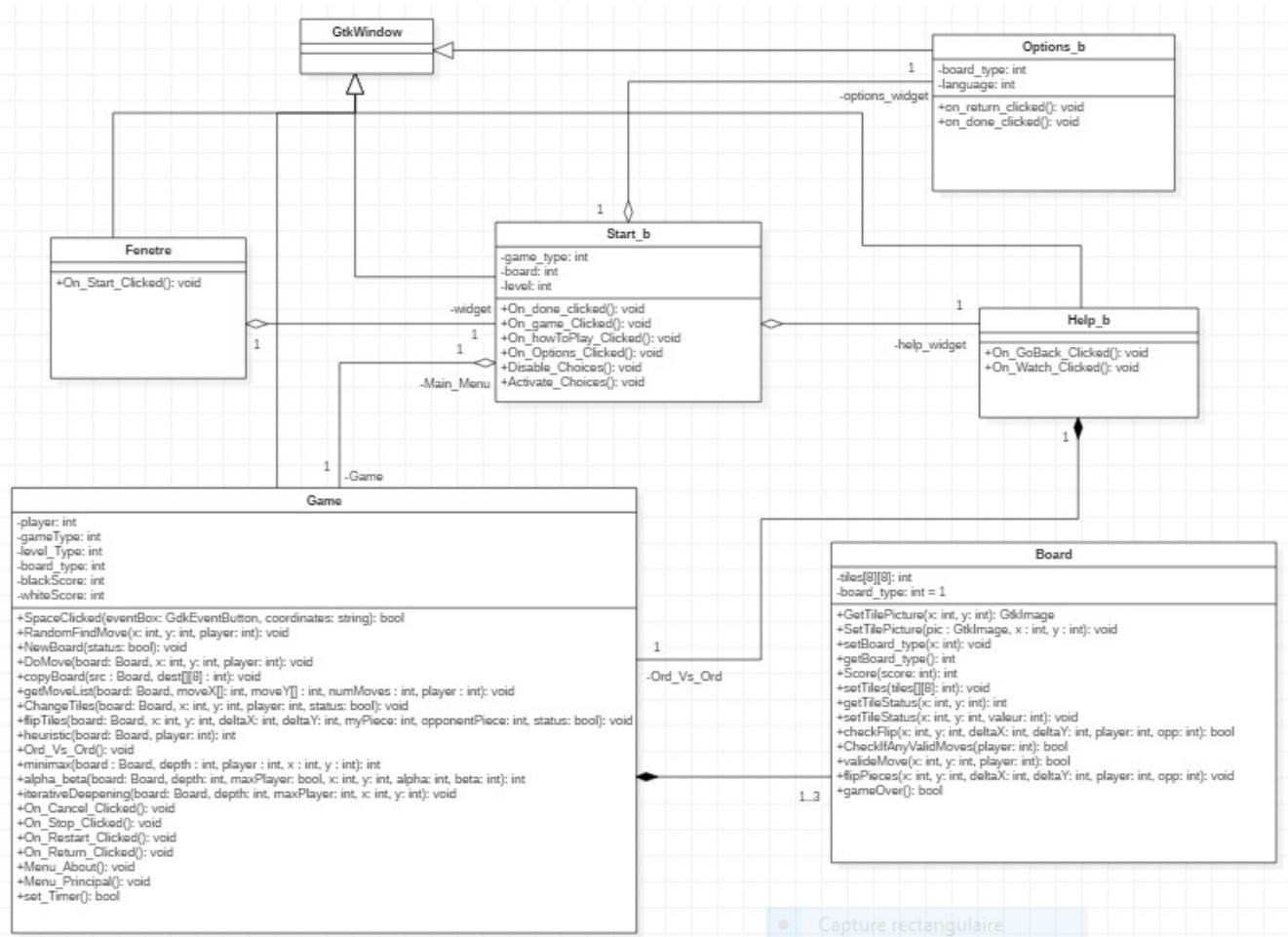


Figure 21 : Diagramme de classe

Chapitre 3 :

Etude technique et environnement

Dans ce chapitre on présente l'étude technique, puis s'articule sur l'architecture adoptée et le langage utilisé pour la réalisation de ce jeu. Le chapitre s'achève par la présentation des Frameworks exploitées.

1. Capture des besoins techniques :

Afin de réaliser ce projet nous avons besoin d'un langage de programmation efficace et portable sur plusieurs systèmes d'exploitation, en plus il doit être rapide et performant au niveau des calculs. Généralement, il doit utiliser de manière optimale les ressources matérielles comme le temps d'utilisation des processeurs, gestion de mémoire... .Après plusieurs recherches d'un langage possèdant ces fonctionnalités, nous avons opté pour le langage C++.

Ayant l'objectif de concevoir des interfaces ergonomiques et simples à manier, nous avons besoin d'une interface qui est facile à utiliser et bien adaptée avec CPP. Parmi les meilleurs toolkits qu'on a trouvé nous choisissons gtkmm. gtkmm est l'interface C++ officielle pour la bibliothèque graphique populaire GTK+.

2. Architecture adoptée :

La POO a été définie par Ole-Johan Dahl et Kristen Nygaard au début de 1960. En 1970, l'américain Alan Kay a appliqué des ajustements sur leurs travaux.

La programmation oriente objet est un paradigme au sein de la programmation informatique. Elle consiste à concevoir une application sous la forme d'un ensemble d'objets reliés entre eux par des relations. Elle a quatre concepts principaux qui sont :

- ✓ Héritage : l'utilisation de ce principe permet de gagner du temps, l'héritage permet d'une classe B d'hériter des attributs et des méthodes d'une classe A.
- ✓ Polymorphisme : ce principe signifie qu'on peut utiliser des méthodes ou des variables selon nos besoins d'eux.
- ✓ Abstraction : ce concept consiste à cacher les détails inutiles à l'utilisateur.
- ✓ Encapsulation : permet de définir des niveaux de visibilité des éléments de la classe. Ces niveaux de visibilité définissent les droits d'accès aux données.

Le langage C++ est un super ensemble de programmation C avec une implémentation supplémentaire de concepts orientés objet.

3. Choix des langages :

C++ est un langage de programmation de niveau intermédiaire développé par Bjarne Stroustrup à partir de 1979 aux Bell Labs. CPP est le langage de programmation le plus largement utilisé dans la programmation d'applications systèmes. Il est présent dans presque tous les domaines. Le C++ donne une compréhension claire de la programmation orientée objet. Il donne aussi la possibilité de travailler à bas niveau ce qui offre le contrôle en termes de gestion de la mémoire, de meilleure performance. Le tableau ci-dessus présente une comparaison de quelques langages de programmation :

	JAVA	CPP	PYTHON
Déploiement stable	Oui	Oui	Oui
Access au Niveau bas	Limite	Fort	Bon
Support multimédia	Moyen	Fort	Moyen
Performance du programme	Fort	Meilleur	Moyen
Support OS	Linux Windows Android	Linux Windows Android iOS	Linux Windows

Tableau 7 : comparaison de quelques langages de programmation

En plus, il existe de nombreux compilateurs C++ :

- Clang C++
- Cygwin (GNU C++)
- MINGW
- source GNU CC
- Intel C++

- Microsoft Visual C++
- Oracle C++
- HP C++
- ...

Et pour le code on peut l'écrire et l'exécuter dans plusieurs IDEs comme Dev-C++, CodeBlocks

4. Framework utilisé :

4.1 GLIBMM :

Glibmm est une surcouche de la bibliothèque GLIB (la base de GTK+). Elle est responsable de la boucle d'évènement, utilisation des threads et système orienté objet ..., Glibmm fournit une API sans interface utilisateur qui n'est pas disponible dans le C++ standard et permet à gtkmm d'encapsuler les API basée sur GObject (une interface de programmation et une bibliothèque logicielle multiplateforme).

4.2 GTKMM :

GTKMM permet de créer des interfaces graphiques en utilisant les mécanismes du C++, GTKMM est une surcouche de GTK+, GTKMM inclut moins de code que GTK, qui utilise des noms de fonctions préfixés et de nombreuses macros de distribution. GTKMM a plusieurs avantages, il est :

- fonctionne sur la plupart des systèmes d'exploitation : Linux, Windows, MacOs, autres.... .
- Utilise des bibliothèques C++ standard, y compris pour les chaînes, conteneurs et itérateurs.
- Utilise des espaces de noms C++.
- Utilisation de l'héritage pour créer des éléments graphiques personnalisés.
-

En conclusion gtkmm rassemble un ensemble des classes par l'instruction suivante : `#include<gtkmm.h>`, mais il est préférable d'inclure seulement celles dont nous avons besoin. Pour que l'exécutable soit moins volumineux et la compilation moins longue. On a le choix entre divers classes proposées par cette bibliothèque, parmi les classes les plus utilisées :

objet	classe
Main	<code>#include<gtkmm/main.h></code>
Fenêtre	<code>#include<gtkmm/window.h></code>
Bouton	<code>#include <gtkmm/button.h></code>
Tableau	<code>#include<gtkmm/table.h></code>
Menu	<code>#include <gtkmm/menu.h></code>
Image	<code>#include<gtkmm/image.h></code>
....	...

Tableau 8 : Quelque Classe De GTKMM

4.3 CSS :

CSS est apparu dans les années 90, c'est un langage informatique utilisé pour la mise en forme.

CSS est utilisé dans des documents web, type de page HTML ou XML. Dans notre cas nous avons utilisé le CSS dans la réalisation des interfaces graphique de C++, il est simple à ajouter en utilisant `#include <gtkmm/cssprovider.h>` au début. Ce header nous permet de créer une interface plus intelligible et plus intéressante.

Chapitre 4 :

Réalisation, Interfaces, Test

Ce chapitre est dédié principalement à exposer l'environnement matériel puis les logiciels, les outils utilisés et les interfaces. A la fin de ce chapitre, on présente des tests de réalisation de ce jeu.

1 Environnement matériels :

On a besoin de rien de particulier pour la réalisation du jeu ni pour le développement et encore moins l'exécution. On a seulement besoin d'un ordinateur portable ou d'une machine virtuelle VM. De plus, les logiciels et les outils utilisés sont disponibles pour Windows, Linux, MacOs.

2 Les logiciels et les outils :

2.1 CodeBlocks :

CodeBlocks est un environnement de développement. Il fait partie des logiciels de type IDE, il est orienté langage C et C++, sa première version multiplateforme est lancé en 2008. Il est gratuit et possède des versions qui intègrent des compilateurs comme le compilateur GCC.

2.2 GTKMM :

Gtkmm est un logiciel libre distribué sous la licence publique générale pour la bibliothèque GNU. Il est disponible sur plusieurs systèmes d'exploitations.

2.3 MSYS2 :

Msys2 est une plate-forme de distribution et de développement, il contient une collection d'outils et de bibliothèque qui offert un environnement facile à utiliser pour créer, installer et exécuter des logiciels Windows natif. Donc à partir de msys2 l'application gtkmm peut être compilé et exécuter.

2.4 StarUML

StarUML est un logiciel de modélisation utilisé dans le développement et dans la conception oriente objet, il est considéré comme un outil open source. Il permet aux utilisateurs de créer des diagrammes UML. Généralement StarUML est complet, robuste et présente une profusion d'outils et de paramètres.

2.5 GanttProject

GanttProject est un logiciel libre, il permet de représenter l'organisation et la distribution des tâches d'un projet pendant une période. On peut lire de gauche à droite l'avancement du projet dans le temps

2.6 Sublime Text 3

C'est tout simplement un éditeur de texte. Dans le cadre de ce projet, on l'utilise pour créer des fichiers en langage CSS.

3 Interface :

Au début lorsque on lance le jeu voilà ce qu'il apparait :

- Deux boutons, le premier c'est *Start* qui indique au joueur de la presser s'il veut jouer, et l'autre *Quit* qui indique au joueur de la presser s'il veut quitter totalement le jeu.



Figure 22 : Interface De Jeu

Si le joueur clique sur *Start*, Il voit un menu contenant quatre boutons :

- Play : pour jouer.
- Options : pour modifier les paramètres du jeu.

- Help : pour savoir comment jouer et savoir aussi l'ensemble des règles à suivre pour gagner.
- Quit : pour quitter le jeu.



Figure 23 : Interface Si Le Joueur Clique Sur Strat

Puis si le joueur clique sur *Play*, il a le choix de sélectionner le type de jeu :

- ✓ One Player : dans le cas d'un joueur (Humain Vs Ordinateur).
- ✓ Two Player : dans le cas de deux joueurs (Humain Vs Humain).

Ensuite il va choisir le niveau de difficulté de jeu, s'il choisit précédemment le type d'One Player, sinon le match se déroulera entre deux humains (Two Player).

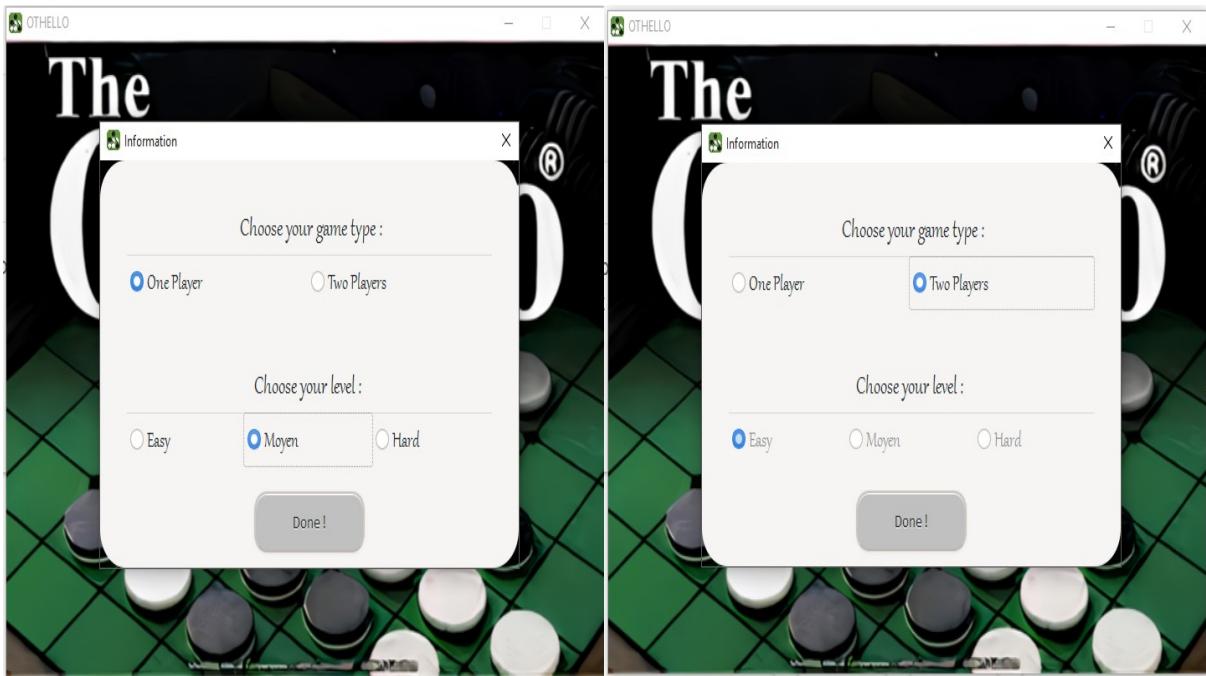


Figure 24 : Interface S'il choisit One Player ou Two Player

Une fois le joueur frappe le bouton *done !* Il va commencer le jeu.

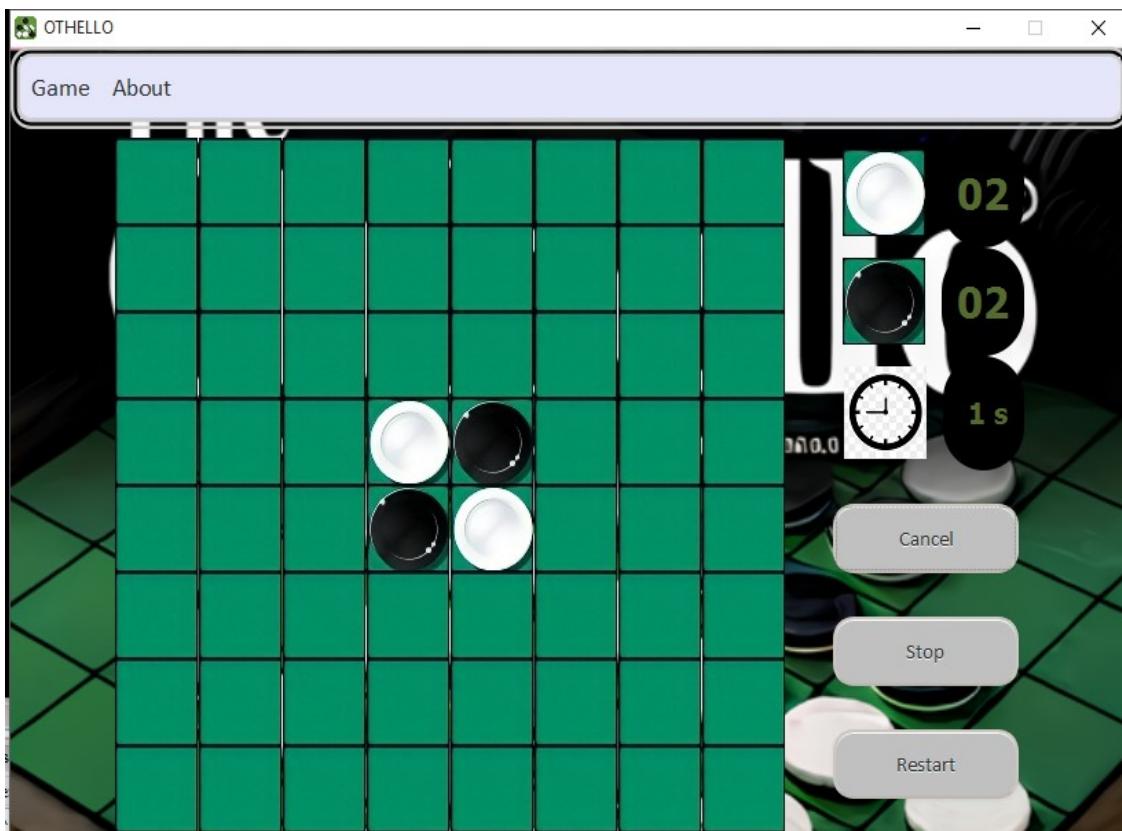


Figure 25 : Début De Jeu

A droite on affiche le score des joueurs selon la couleur des pions plus trois boutons :

- ✓ Cancel : pour revenir en arrière d'un seul mouvement.
- ✓ Stop : pour mettre en pause le minuteur.
- ✓ Restart : pour restaurer le damier comme au début de jeu.

Et en haut on a une barre qui inclut :

- ✓ Game : contenant les trois boutons précédents (Cancel, stop, Restart) et l'option de Main Menu qui sert de revenir au menu principale et Quit qui permet de quitter le jeu.
- ✓ About : contenant les noms des créateurs de jeu et un mail à contacter.

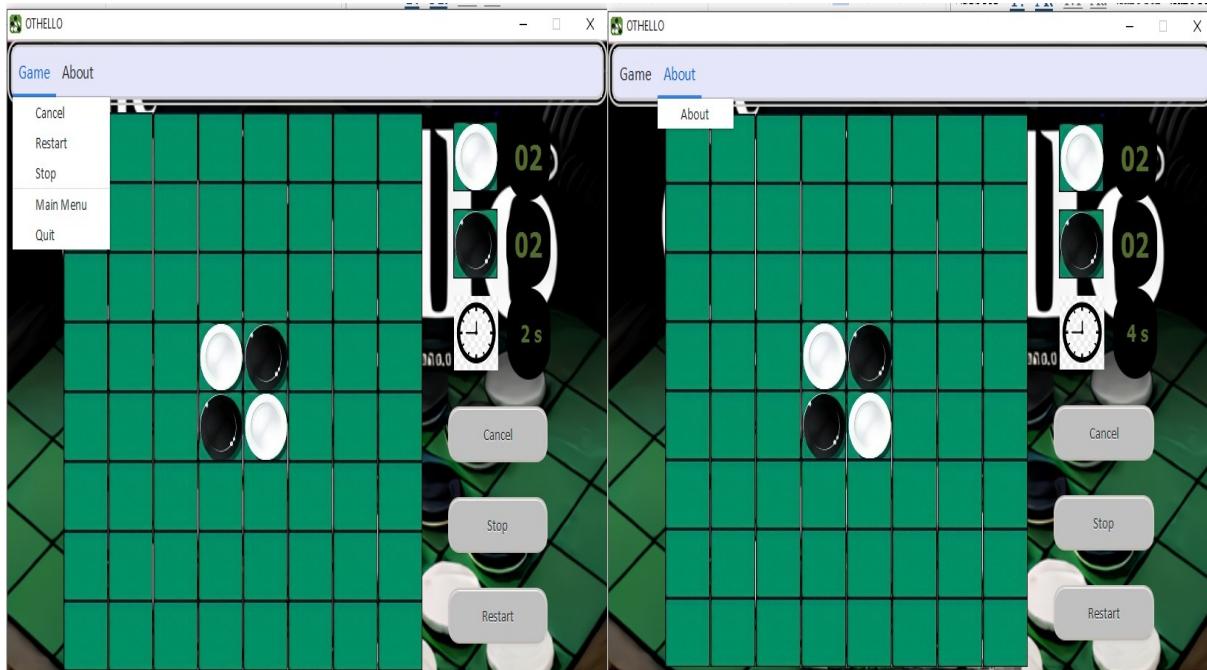


Figure 26 : Barre d'outils

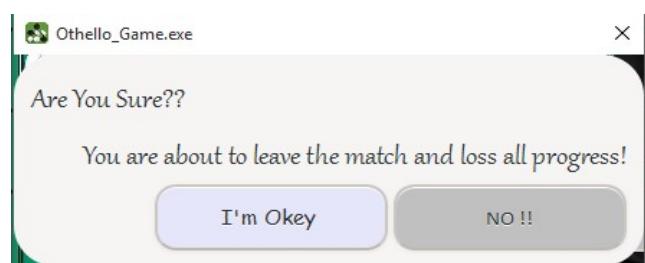


Figure 27 S'il veut Revenir Au Menu Principale

Quand le jeu termine, un message apparaît pour informer les deux joueurs qui est le gagnant.



Figure 28 : Message Indique Le Gagnant

Concernant le bouton *options* il permet de choisir la couleur du damier et la langue :



Figure 29 : Les Paramètres de Jeu à Modifier

Et pour *Help*, il contient un tutoriel du jeu avec des images explicatives. A la fin, il existe deux boutons : Return pour retourner au menu principale et Watch how pour voir comment jouer :

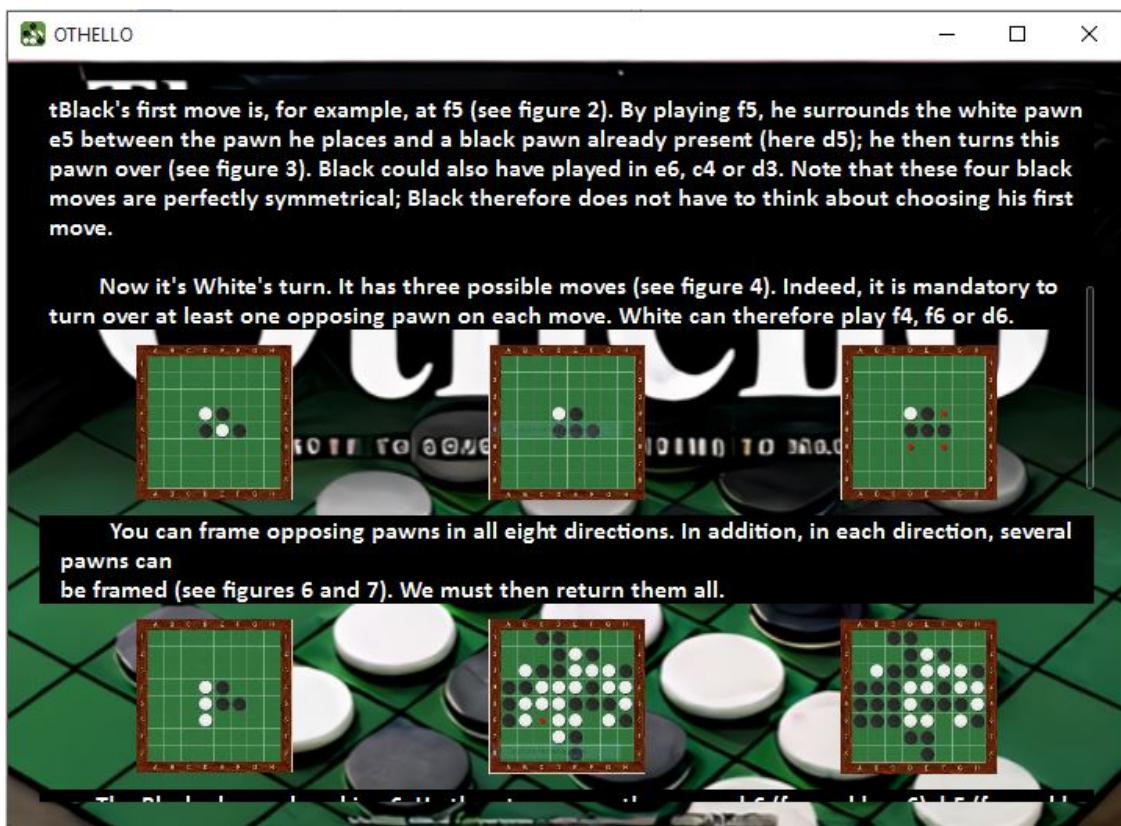
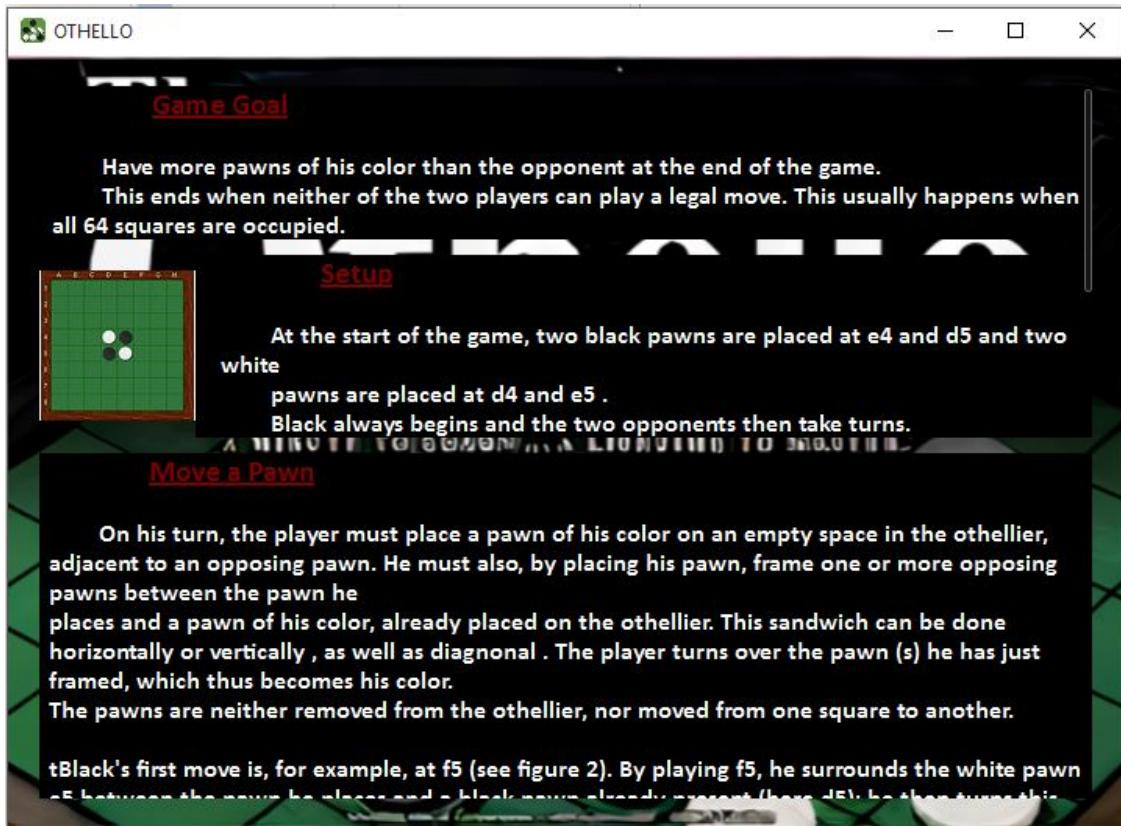


Figure 30 : Tutoriel de Jeu

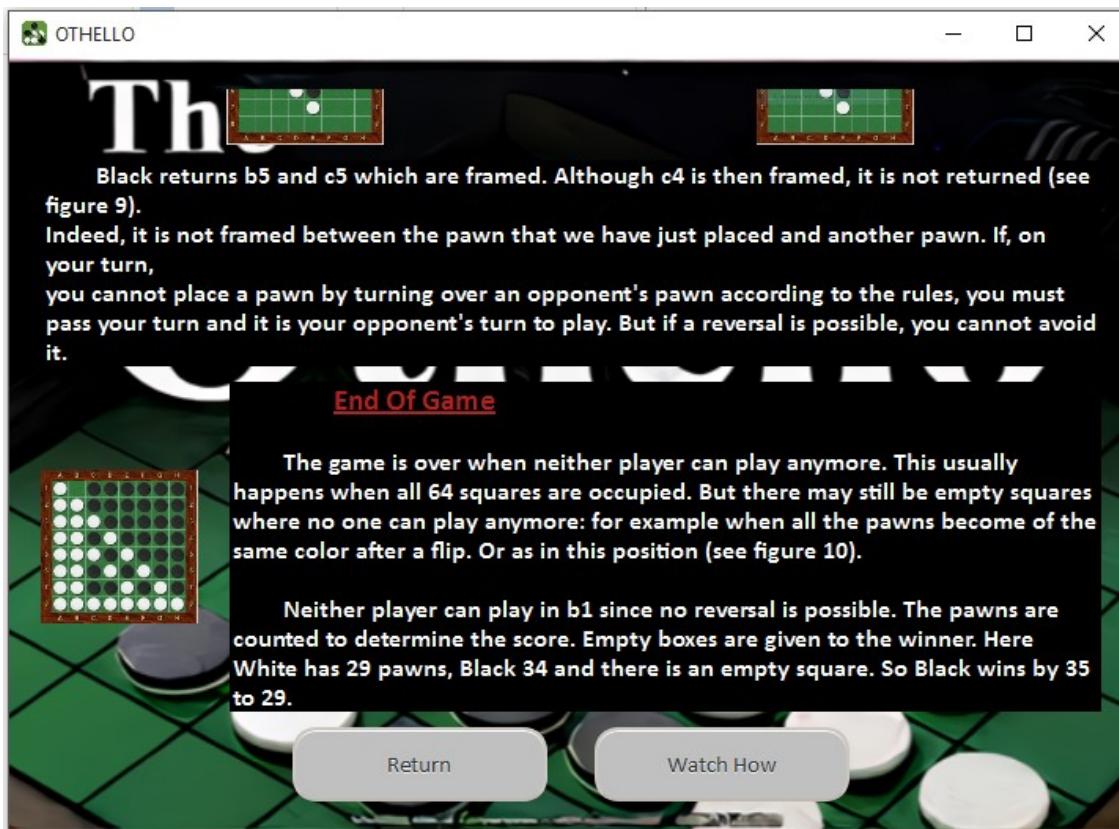


Figure 31 : la fin de tutoriel

Si le joueur ne comprend pas bien le tutoriel, il peut regarder un match (Ordinateur Vs Ordinateur) par une clique sur le bouton Watch How

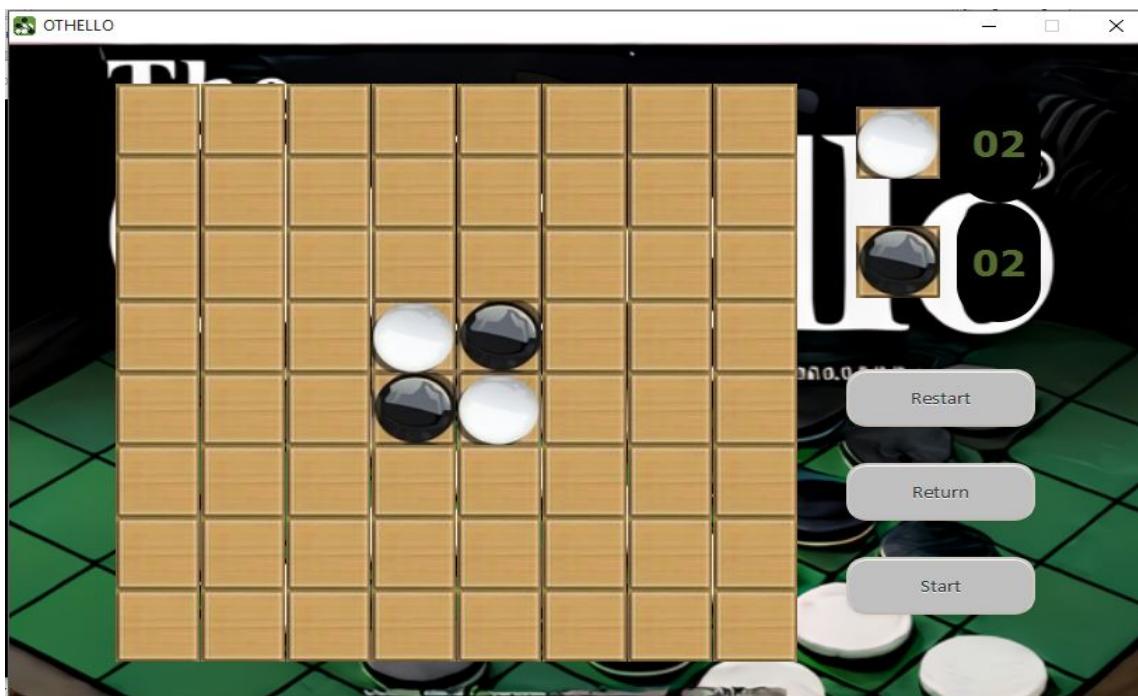


Figure 32 : Regarder un match (Ordinateur Vs Ordinateur)

4 Tests :

Pour tester notre jeu et la bonne fonctionnaliste des algorithmes on va faire une teste sur les différents types de jeu (Player Vs ordinateur ou Player Vs Player), et les niveaux (facile, moyenne, difficile).

4.1 Game type => One Palyer :

Le premier cas qui nous allons tester est si le joueur sélectionner de jouer contre l'ordinateur :

a. Niveau facile :

Dans ce niveau, on a utilisé un algorithme qui donne au finale une liste des mouvements possibles. Puis il se donne une position au hasard parmi la liste précédente où l'ordinateur va poser leur pion.

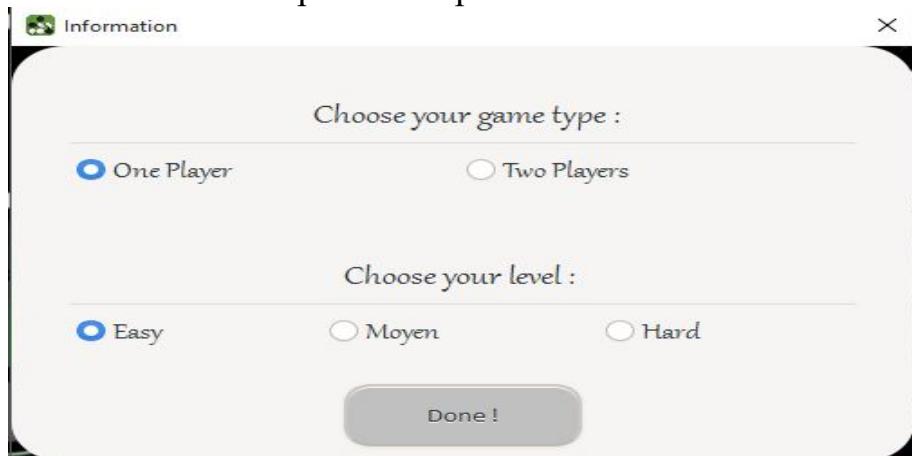


Figure 33 : Choisir le niveau ‘Easy’

Les pions noirs représente le joueur et les pions blancs représente l'ordinateur, les figures suivantes représente le jeu au début et au finale :

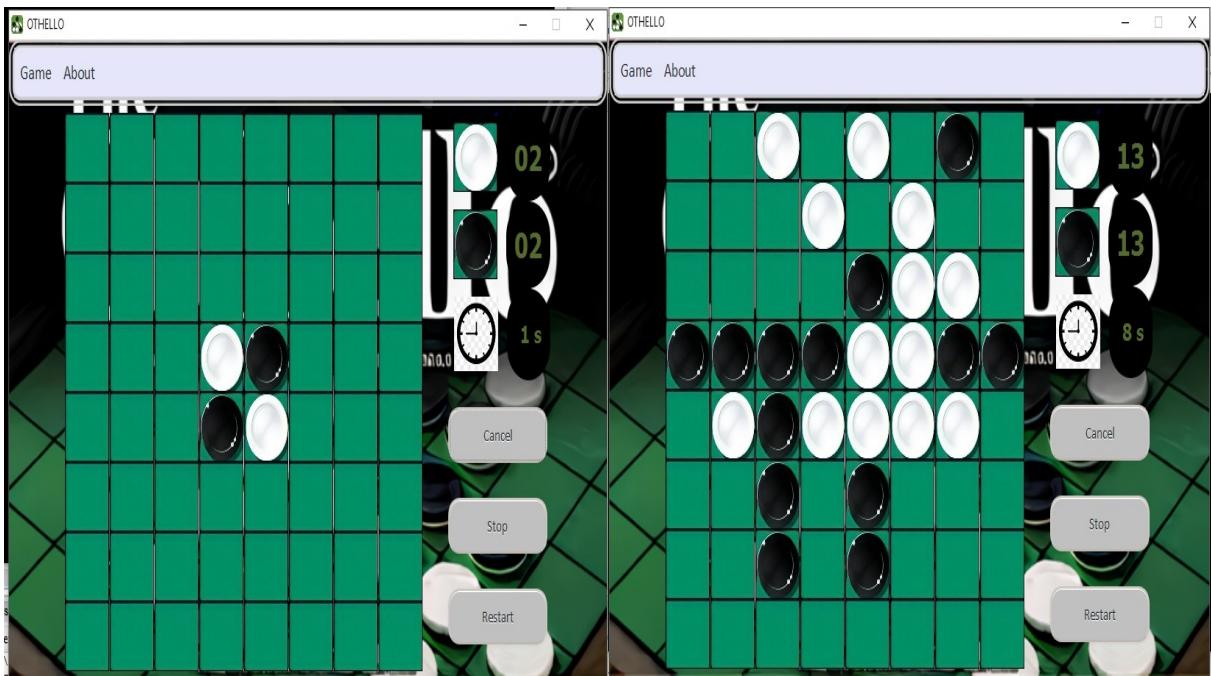


Figure 34 : Le jeu au début et au milieu

Dans ce cas le gagnant est le joueur qui a les pions noirs (score de joueur = 44, score d'ordinateur = 20).

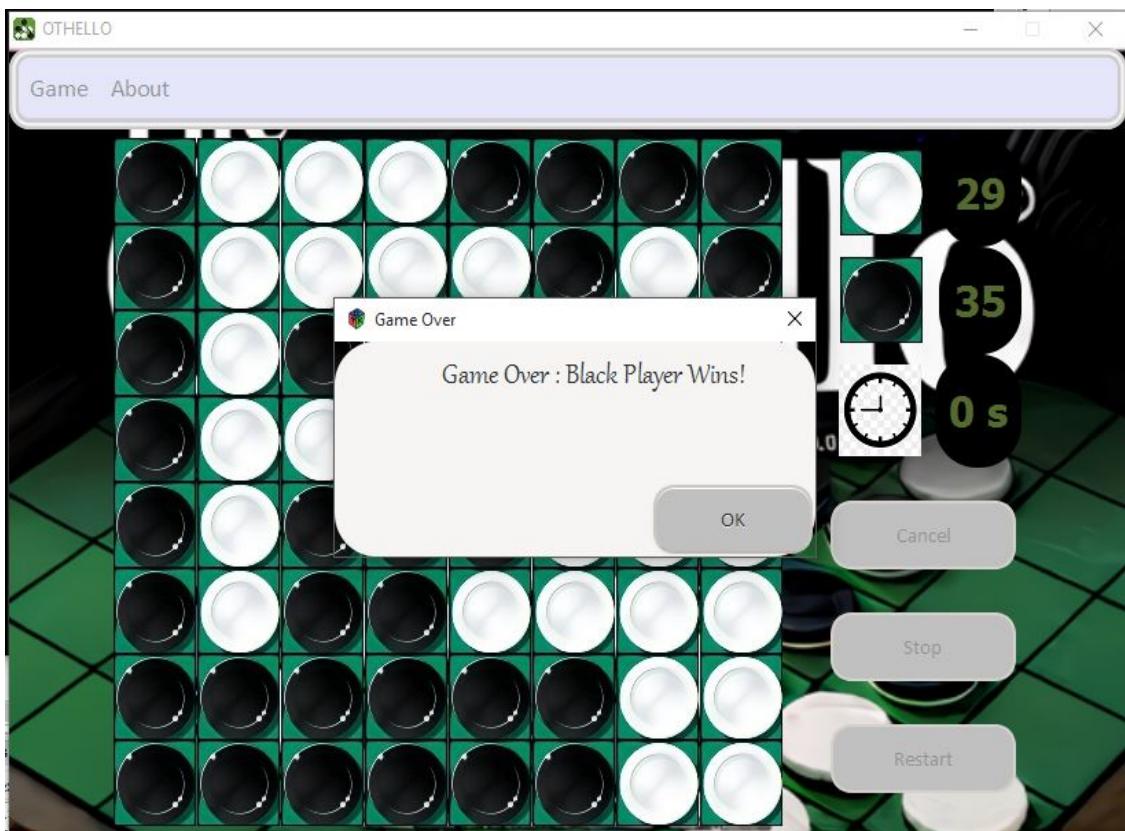


Figure 35 : le joueur gagne (score = 35)

b. Niveau moyenne :

Dans ce niveau, on a utilisé l'algorithme MinMax. Le niveau de difficulté dans cette étape est plus que le niveau de difficulté dans l'étape précédente.



Figure 36 : Choisir le niveau 'Moyen'

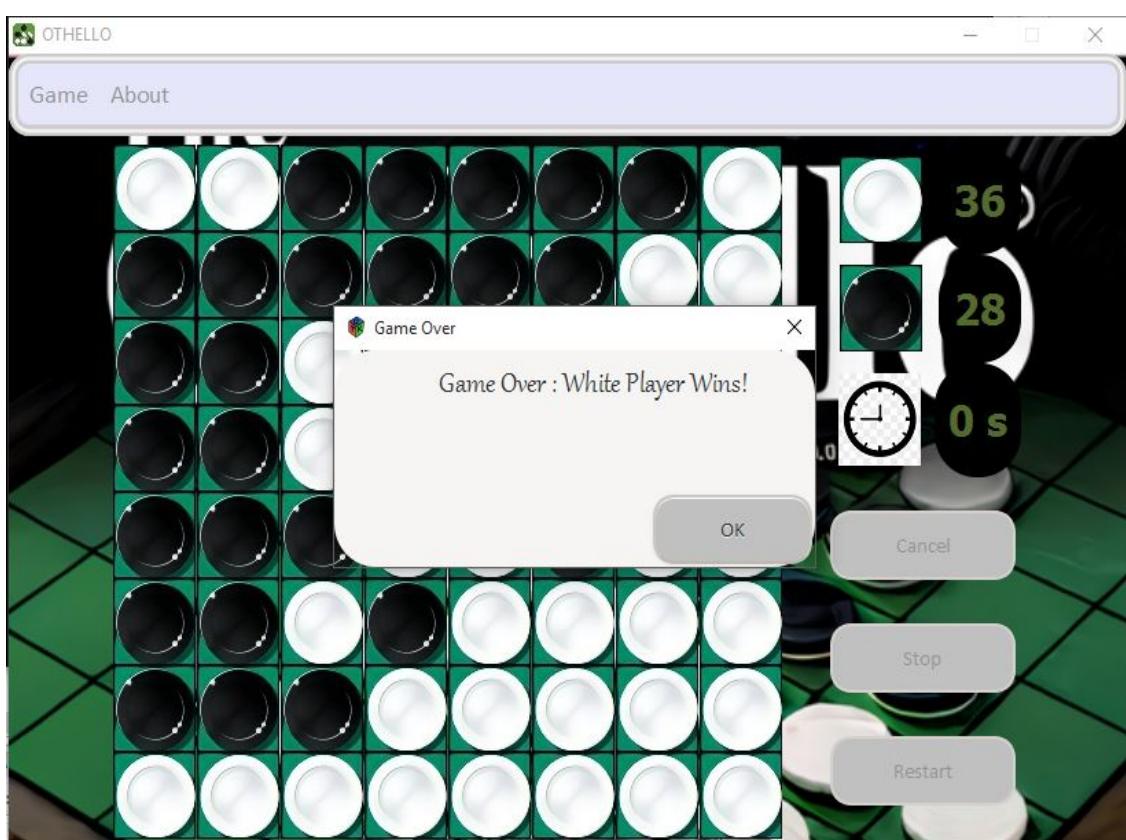


Figure 37 : l'ordinateur gagne (score = 36)

c. Niveau difficile :

Dans ce niveau, on a utilisé l'algorithme AlphaBeta, le niveau de difficulté est plus élevé par rapport au les autre niveaux.

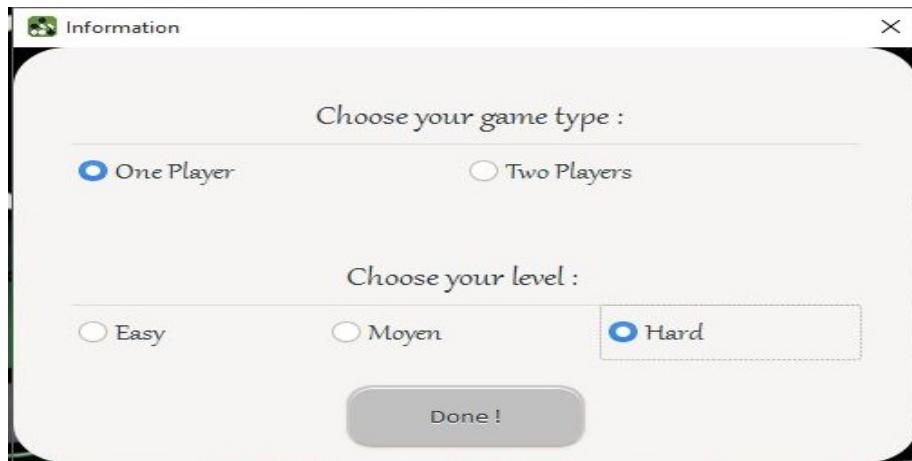


Figure 38 : Choisir le niveau 'Hard'

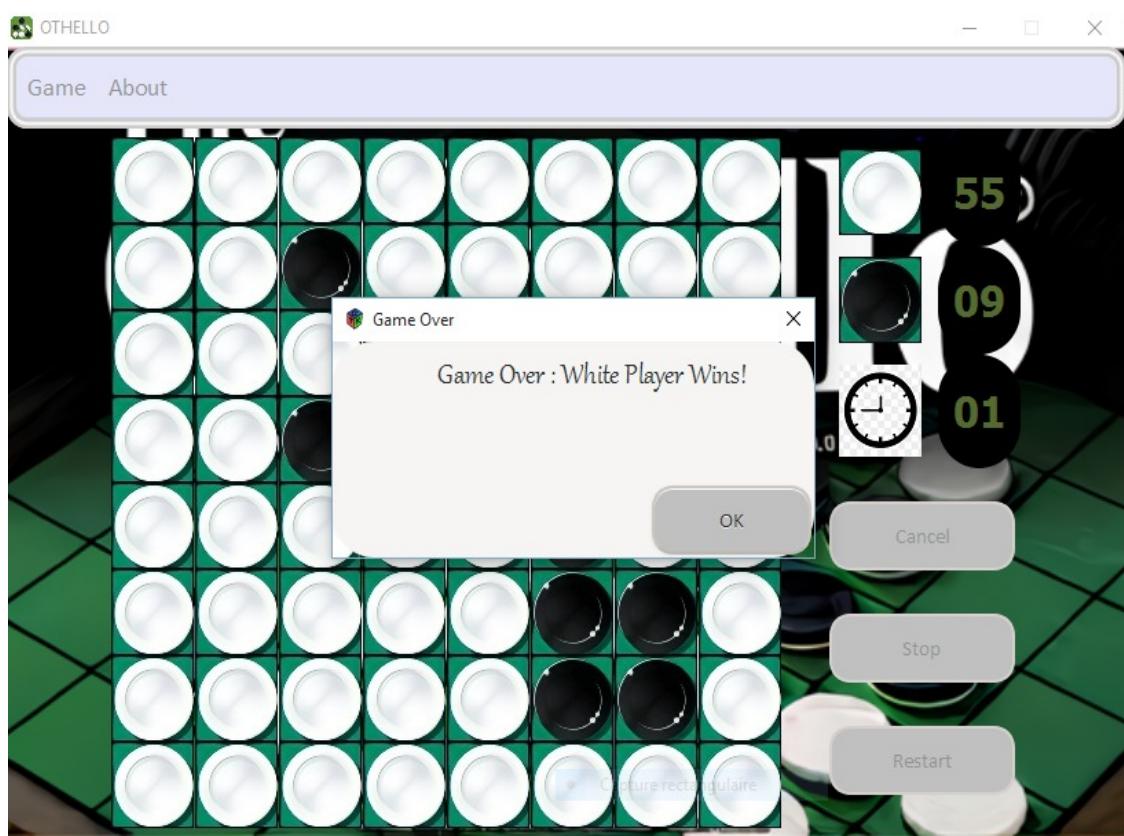


Figure 39 : L'ordinateur gagne (score = 55)

4.2 Game type => Two Player :

Dans ce cas, le joueur choisit de jouer contre un humain. Alors le résultat final ça dépend de l'intelligence du joueur.

4.3 Conclusion :

Le tableau ci-dessus montre le nombre de fois où le joueur a gagné, le nombre de fois l'ordinateur a gagné et le nombre de fois où ils ont marqué une égalité pendant 6 itérations :

	Joueur gagne	Ordinateur gagne	Marquer une égalité
Niveau 'Easy'	4	1	1
Niveau 'Moyen'	0	5	1
Niveau 'Hard'	0	6	0

Tableau 9 : comparaison de gain par rapport les niveaux

Pour une vue plus claire et une comparaison complète, Nous présentons le graphe suivant qui illustre ce qui est indiqué dans le tableau ci-dessus :

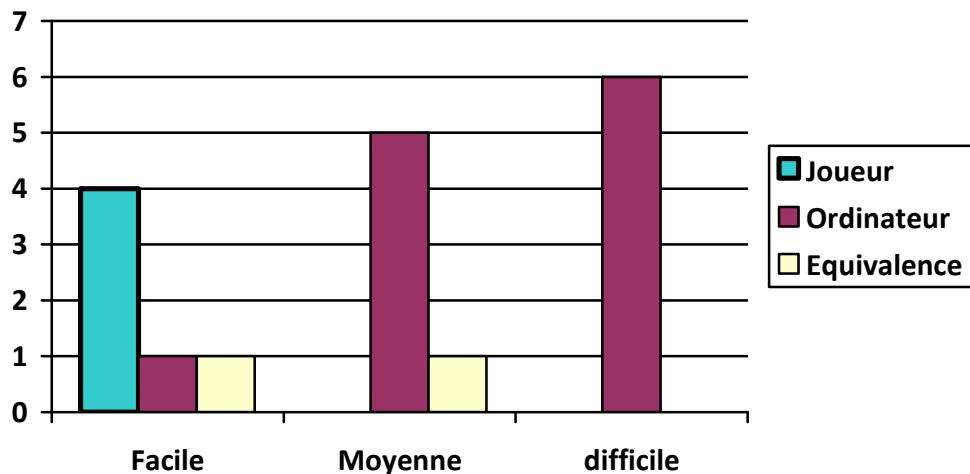


Figure 40 : graphe illustratif

Conclusion générale et perspective

Ce projet de fin d'étude, visé de concevoir et de réaliser le jeu vidéo Othello. Au terme de ce travail de 3 mois, nous avons pu concrétiser ce jeu qui est impossible de le battre par un humain.

Nous avons atteint notre objectif dans ce délai en nous basant sur les directives de notre encadrement et par suivre une méthodologie de travail.

Pour les réaliser, nous avons commencé tout d'abord par des recherches sur le sujet du projet et une étude fonctionnelle détaillée des besoins du projet. Ensuite, Nous avons réalisé l'étude technique de notre projet en établissant l'architecture globale et choisissant les technologies et les outils de réalisation.

Nous avons utilisé les diagrammes UML pour la conception. Pour la partie de réalisation nous avons utilisé le langage de programmation C++, pour les interfaces GTKMM, GLIBMM et CSS.

En outre, ce projet a été une occasion unique pour notre équipe pour acquérir beaucoup de compétences différentes qui nous seront d'une utilité primordiale dans le futur proche et lointain.

L'élaboration de ce projet nous a permis d'élargir notre connaissance et d'utiliser différentes pratiques et concepts que nous avons acquis tout au long de notre année académique.

Bibliographie

Annexes

1 Algorithme MinMax en C++ :

```
int Game::minimax(Board *board, int depth, int player,int &x,int &y) {  
    int moveX[60], moveY[60];  
    int opp=-1;  
    if(player== -1) opp=1;  
    int numMoves;  
    if (depth == 0 || board->gameOver()) {  
        return heuristic(board,player);  
    }  
    getMoveList(board, moveX, moveY, numMoves, player );  
    if (numMoves == 0)  
    {  
        x = -1;  
        y = -1;  
    }  
    else {  
  
        if (player==1) {  
            int bestMoveVal = -99999;  
            int bestX = moveX[0], bestY = moveY[0];  
            for (int i = 0; i < numMoves; i++)  
            {  
                int tempBoard[8][8];  
                copyBoard(board,tempBoard);  
                temp1->setTiles(tempBoard);  
                DoMove(temp1, moveX[i], moveY[i], player);  
                int v = minimax(temp1, depth - 1,opp,x,y);  
                if (v > bestMoveVal) {  
                    bestMoveVal = v;  
                    bestX = moveX[i];  
                    bestY = moveY[i];  
                }  
                x = bestX;  
                y = bestY;  
            }  
            return bestMoveVal;  
        }  
        else {  
            int bestMoveVal = 99999;  
            int bestX = moveX[0], bestY = moveY[0];  
            for (int i = 0; i < numMoves; i++)  
            {  
                int tempBoard[8][8];  
                copyBoard(board,tempBoard);  
            }  
        }  
    }  
}
```

```

temp2->setTiles(tempBoard);
DoMove(temp2, moveX[i], moveY[i], player);
int v = minimax(temp2, depth - 1, opp, x, y);
if (v < bestMoveVal) {
    bestMoveVal = v;
    bestX = moveX[i];
    bestY = moveY[i];
}
x = bestX;
y = bestY;
}
return bestMoveVal;
}
}
}

```

2 Algorithme AlphaBeta en C++ :

2.1 AlphaBeta :

```

int Game::alpha_beta(Board *board,int depth,bool maxPlayer,int &x,int &y,int alpha,int beta )
{
    int moveX[60], moveY[60];
    int numMoves;
    int player;
    if(maxPlayer) player=1;
    else player=-1;

    if (depth == 0 || board->gameOver()) {
        return heuristic(board,player);
    }
    getMoveList(board, moveX, moveY, numMoves, player );
    if (numMoves == 0)
    {
        x = -1;
        y = -1;
    }
    else {

        if (maxPlayer) {
            int bestMoveVal = -99999;
            int bestX = moveX[0], bestY = moveY[0];
            for (int i = 0; i < numMoves; i++)
            {
                int tempBoard[8][8];
                copyBoard(board,tempBoard);

```

```

temp1->setTiles(tempBoard);
DoMove(temp1, moveX[i], moveY[i], player);
int v = alpha_beta(temp1, depth - 1, false, x, y, alpha, beta);
if (v > bestMoveVal) {
    bestMoveVal = v;
    bestX = moveX[i];
    bestY = moveY[i];
}

x = bestX;
y = bestY;
if( v > alpha ) alpha=v;
if(beta <= alpha) break;
}
return bestMoveVal;
}
else {
    int bestMoveVal = 99999;
    int bestX = moveX[0], bestY = moveY[0];
    for (int i = 0; i < numMoves; i++)
    {
        int tempBoard[8][8];
        copyBoard(board,tempBoard);
        temp2->setTiles(tempBoard);
        DoMove(temp2, moveX[i], moveY[i], player);
        int v = alpha_beta(temp2, depth - 1, true, x, y, alpha, beta);
        if (v < bestMoveVal) {
            bestMoveVal = v;
        }
        if( v < beta ) beta=v;
        if(beta <= alpha) break;;
    }
    return bestMoveVal;
}
}
}

```

2.2 Iterative deepening :

```

void Game :: iterativeDeepening (Board *board , int depth , int maxPlayer , int &x , int &y)
{
    int alpha=-99999,beta=99999;
    for(int i=0;i<=depth;i++)
    {
        alpha_beta(board,i,maxPlayer,x,y,alpha,beta);
    }
}

```