



ECOLE NATIONALE SUPÉRIEURE D'INFORMATIQUE ET D'ANALYSE DES
SYSTÈMES - RABAT

Rapport de Projet de programmation : OTHELLO

Réalisé par :

EL JAZOULY ANASS
EL KARMY OTHMANE

Encadré par :

Pr.EL HAMLAOUI MAHMOUD



Remerciements :

Nous voudrons tout d'abord adresser toute notre gratitude à notre professeur Abdellatif EL FAKER et à notre encadrant Mahmoud EL HAMLAOUI ,pour leur confiance, et surtout cette opportunité pour bien maîtriser le langage de programmation C et initier notre carrière par un aussi beau sujet.

Nous désirons aussi remercier tous ceux qui contribuaient à la réussite de ce projet, notons L'ASFIRI Halim, et bien d'autres pour leurs conseils et leurs connaissances qu'ils nous ont partagé.



Table des matières

1 Contexte général du projet	1
1.1 Présentation du projet	1
1.1.1 Sujet	1
1.1.2 Étymologie	2
1.2 Cachier des charges	3
1.2.1 Principe de jeu	3
1.2.2 Règle de jeu	3
1.2.3 Livrables	4
1.3 Problématique	4
1.4 Objectifs	4
2 Analyse et conception	5
2.1 Analyse théorique	5
2.1.1 Algorithmes	5
2.2 Conception	6
2.2.1 Conception du plateau	7
2.2.2 Conception du retour en arrière	7
2.2.3 Conception de L'accueil	8
2.2.4 Conception de la table de jeu	9
2.2.5 Sauvegarde des parties	9
2.2.6 Langage et interface	9
2.2.7 Latex	10
2.2.8 Code-blocks	10
2.2.9 Outils de collaboration et communication	10
3 Réalisation et résultat	11
3.1 Difficultés rencontrées	11
3.1.1 Le temps	11
3.1.2 L'implémentation de SDL, SDL-ttf et SDL-Mixer :	11
3.2 Fonction du code source	11
3.2.1 Directive de préprocesseur	11
3.2.2 Définition des constantes	11
3.2.3 Les structures	12
3.2.4 Fonction.c	12
4 Interface graphique	16
4.1 Menu d'accueil	16
4.2 Menu de jeu	17
4.3 Interface de jeu	19
5 Conclusion générale	21

Chapitre 1

Contexte général du projet

1.1 Presentation du projet

PROJET C s'agit de produire un programme d'environ 500 lignes de code afin de valider les compétences des cours : « Algorithmique », « Technique de programmation » et « Structures de données ». Le programme correspond à 20 heures de travail effectives en langage C. Les étudiants travaillent en binôme et bénéficient des conseils d'un professeur encadrant¹.

1.1.1 Sujet

Le Othello est un jeu de réflexion (stratégie et tactique), également connu sous le nom de Reversi, qui fait appelle à deux joueurs par tour de rôle sur un plateau de 8 cases sur 8 cases appelé othellier, chaque joueur possède des jetons de couleurs unies soit blanches, soit noires.(cf. fig. 1.1) .L'objectif du jeu est d'obtenir le plus grand nombre possible de jetons de la couleur qui vous est attribuée. Le jeu de Reversi est inventé vers 1880 en Angleterre par Lewis Waterman ou John W. Mollet². Tandis qu' Othello a été « inventé » au Japon en 1971 par Goro Hasegawa.

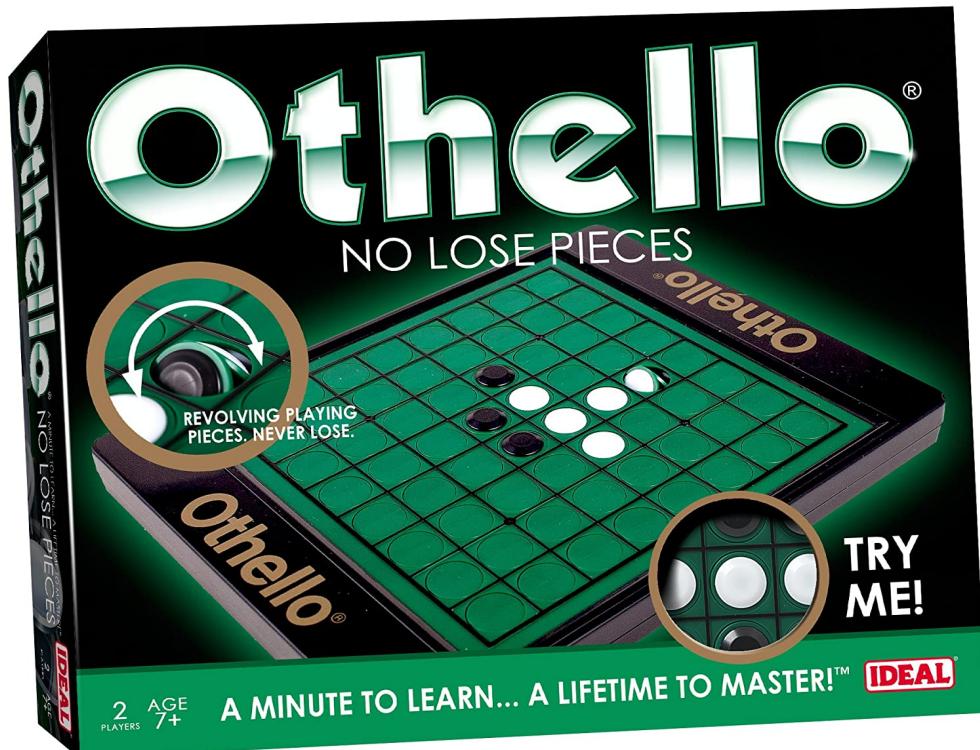


FIGURE 1.1 – Planche du jeu Othello

2. Notice du projet de programmation 2020/2021

2. il est également possible que ce soit quelqu'un d'autre qui ait inventé ce jeu, vu qu'ils s'accusent mutuellement de plagiat.

1.1.2 Étymologie

Othello est une version moderne basée sur le jeu d'Othello inventé par le britannique Lewis Waterman en 1883, et a acquis une popularité considérable au Royaume-Uni à la fin du 19e siècle.

En 1898, le célèbre éditeur de jeux allemand Ravensburger a commencé à vendre le jeu comme l'un de ses premiers jeux.

Les règles modernes sont originaires de Mito City, dans la préfecture d'Ibaraki, au Japon, dans les années 1970. Elles sont maintenant acceptées dans le monde entier. La société de jeux japonaise Tsukuda les a définies et a enregistré le jeu sous la marque "Othello" avec le slogan "Une minute pour l'apprendre... Une vie pour le maîtriser!"

Le nom a été choisi pour faire référence à la pièce de Shakespeare "Othello" et "Maure de Venise" (Maure de Venise), plus précisément :

Le conflit entre Othello Marsh et Iago, qui se décrit comme "un homme à deux visages", Et la romance entre le noir Othello et le blanc Desdemona. La couleur verte du plateau de jeu est inspirée de l'image du général Othello combattant courageusement sur l'herbe verte autour du thème de la jalousie (dans de nombreuses pièces de Shakespeare, ce thème est généralement le noyau et la clé).

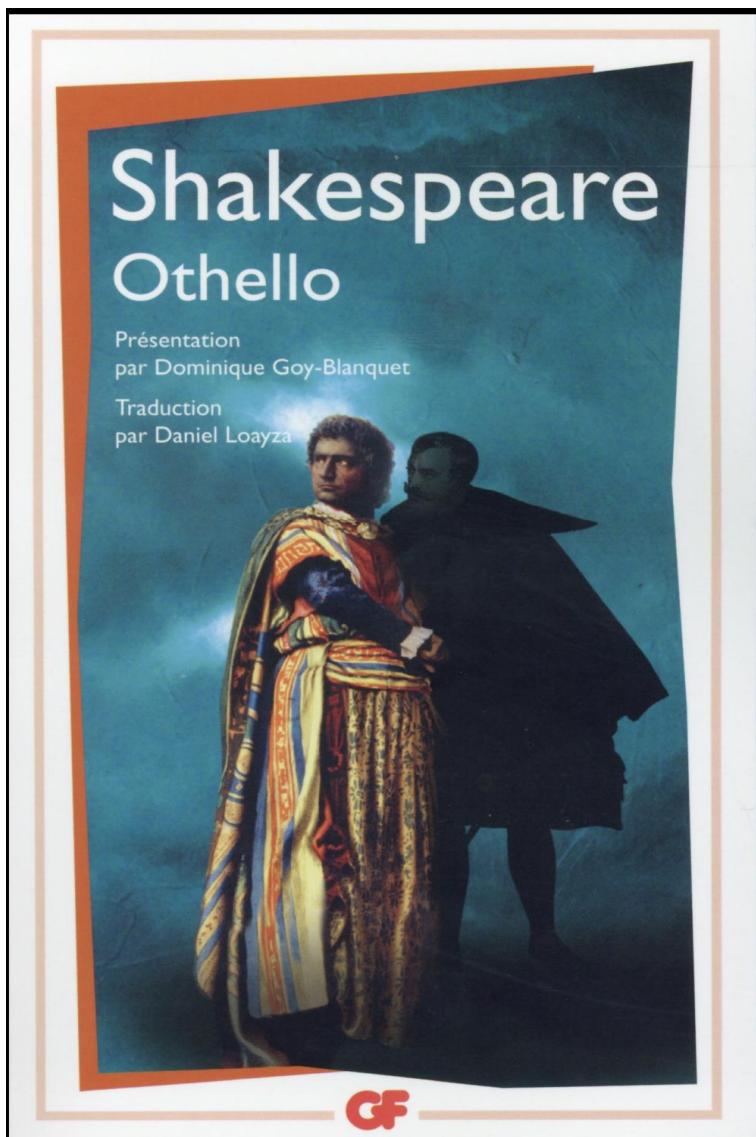


FIGURE 1.2 – Couverture de l'oeuvre de Shakespeare : "Othello"

1.2 Cachier des charges

Le cahier de charges présente l'ensemble des instructions et contraintes qui cadrent la réalisation du jeu. Il présente 3 livrable qu'on va essayer de les élaborer dans cette partie .

1.2.1 Principe de jeu

Les noirs commencent toujours le jeu. Ensuite, les joueurs se relaient pour jouer au jeu, et chaque joueur doit capturer le pion adverse pendant qu'il se déplace. Si le joueur ne peut pas capturer le pion de l'adversaire, il sera obligé de sauter le tour. Si aucun des joueurs ne peut jouer, ou si les autres joueurs n'ont plus de case vide, la partie se termine. Le gagnant à la fin du jeu est celui qui a le plus de pièces.

Lorsque le joueur place une de ses pièces à la fin de pièces successivement opposées et que l'autre extrémité est déjà occupée par l'une de ses propres pièces, la pièce sera capturée. L'alignement considéré peut être en colonne, en ligne ou en diagonale. Si le pion nouvellement placé doit fermer plusieurs routes, il capturera tous les pions adverses ainsi fermés. La capture fait inverser la pièce capturée. Ces retournements ne provoquent pas d'effet de capture en cascade : seules les pièces nouvellement placées sont prises en compte.

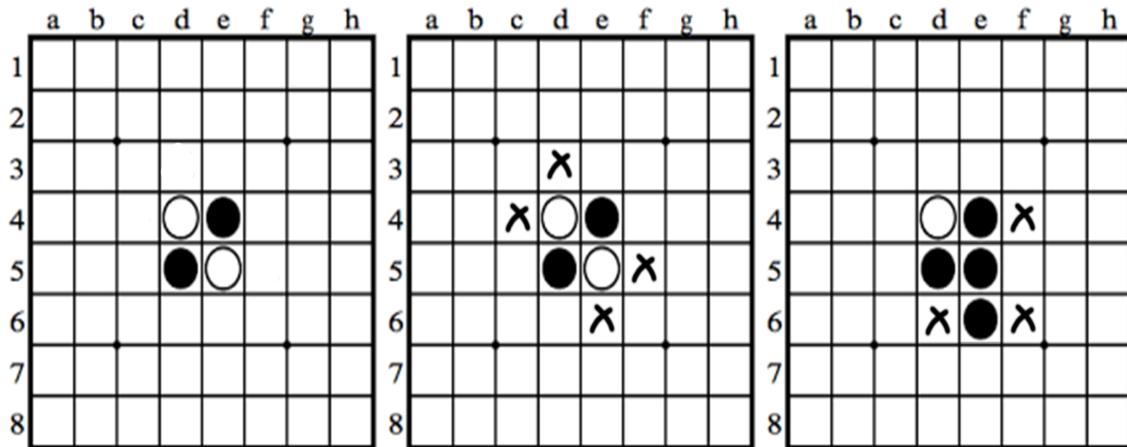


FIGURE 1.3 – Visualisation d'un premier mouvement

1.2.2 Règle de jeu

Il existe des règles qui s'appliquent sur les deux joueurs et qu'ils doivent respecter pour qu'ils puissent jouer :

- 1- Chaque joueur joue à tour de rôle en posant une pierre sur une case libre.
- 2- Si un joueur ne peut poser de pierre alors il doit passer (il ne joue pas et c'est au tour de son adversaire).
- 3- Si les deux joueurs ne peuvent plus poser de pions alors la partie est finie et le joueur ayant le plus grand nombre de pierres de sa couleur gagne.
- 4- Un coup est légal si le coup permet de capturer des pierres adverses et pour capturer des pierres il faut que ces pierres soient encadrées par des pierres adverses .

1.2.3 Livrables

Les livrables constituent un chemin pour nous guider et nous aider à définir nos objectifs pour la réalisation du projet et il en existe trois :³

— Livrable 1 :

Le programme qu'on doit réaliser doit gérer dans un premier temps :

- Recommencer à tout moment en cliquant sur le bouton "recommencer".
- Créer et enregistrer les joueurs et leurs caractéristiques (nom et score) sur fichier.
- Afficher l'historique des mouvements effectués par les joueurs.
- Permettre le chargement d'un jeu sauvegardé auparavant.
- Donner la liste des dix meilleurs scores.

— Livrable 2 :

Au niveau du deuxième livrable on doit permettre au joueur de jouer contre un non humain ;

- Avec des positions aléatoires.
- Avec la mise en place d'un algorithme permettant d'optimiser la recherche du meilleur coup, en limitant le nombre de cases visitées. Il est possible d'utiliser l'algorithme min-max avec l'élagage Alpha-Beta .

— Livrable 3 :

Il est également recommandé de considérer :

- La sécurité des accès à l'aide de mots de passe,
- La réalisation d'une interface de jeu conviviale.

1.3 Problématique

A quel niveau serions-nous capables de traduire le jeu à la réalité et de le réaliser sous la forme d'une interface graphique jouable et facile à manipuler ?

1.4 Objectifs

Nous avons définis des objectifs pour ce projet en les classifiant selon leurs importances :

- Renforcer nos connaissances au langage C et s'initier à la réalisation de projets, et valider nos compétences acquises.
- Respecter le cahier des charges tout en étant efficace.
- Soigner le plus possible l'affichage et améliorer nos compétences en design.
- Réussir la combinaison entre la théorie et la pratique.
- Travailler en collaboration pour la réalisation d'un seul projet, et s'adapter au travail en groupe.

3. Cahier des charges distribué par notre encadrant

Chapitre 2

Analyse et conception

Ce chapitre permet de faire une analyse théorique du jeu Othello. En effet, cette conception est cruciale afin de comprendre la totalité des principes et les coder en se basant sur le cahier de charges fourni. Ainsi que la conception que nous avons adopter.

2.1 Analyse théorique

On va diviser la conception théorique sous 3 parties savoir les algorithmes qu'on a adopter, la méthode de résolution graphique ainsi que les formules mathématiques utilisées.

2.1.1 Algorithmes

- Les Algorithmes pour "highlight" : Il en existe deux majeures fonctions pour highlight une case, et c'est une étape cruciale puisque c'est elle qui va nous permettre de décider si un mouvement est autorisé par la suite .

Algorithm 1 contour

```
Directions ← [tout – les – directions – possibles];  
direction;  
for direction in Directions do  
    SWITCH(direction)  
    - case :(une direction donnée)  
    - - chercher(ligne,colonne,joueur-adverse,joueur,direction)  
    - - break ;  
end for
```

L'algorithme contour permet de parcourir tout les direction possible, et il fait appelle à l'algorithme de la recherche suivant permettant de rechercher un joueur ou le vide en se basant sur le paramètre "joueur-adverse".

Algorithm 2 chercher(ligne,colonne,joueur-adverse,joueur,direction)

```
if joueur-adverse = NOIR ou BLANC then  
    chercher(ligne,colonne,VIDE,joueur,direction)  
else if joueur-adverse = VIDE then  
    chercher le Vide en suivant la direction tant qu'on est au sein du plateau  
    if Vide trouve then  
        Marquer la position  
        SORTIR  
    end if  
else  
    Afficher(joueur-adversse incorrecte)  
end if
```

- Algorithme pour la justification d'un coup legal : Comme cité dans la partie précédente cette algorithme utilise principalement le fait qu'une case est légale ou non en vérifiant si elle est marquée ou pas .

Algorithm 3 coup legal

```
if Case=Marqueur(marquee) then
    colorier suivant tout les direction
    SORTIR
end if
```

- Algorithme de coloriage : Cet algorithme presente la manifestation des regles selon un joueur capture les pions ennemis.

Algorithm 4 coloriage

```
for chaque direction do
    parcourir la direction chercher un pion allie tant qu'on a pas recontre un vide
    if cette condition est verifie then
        colorier par la couleur du joueur
    end if
end for
```

- Algorithme elage Alpha-Beta cet algorithm a ete monsione dans le 2 eme livrable du cahier des charges ,il consiste a choisir le meilleur coup base sur la combinaison de l'anticipation des mouvements, en suivant une fonction evaluation determine par l'utilisateur.[3][4]

Algorithm 5 Fonction ALPHA BETA(p, A, B,opers)

```
if Feuille(p) = vrai then
    retourner f(p)
else
    if TypeNoeud(p) = MIN then
        Beta(p)=MAX VAL
        for chaque op dans opers do
            fils = successeur(p, op)
            Val = ALPHA BETA(fils, A, Min(B,Beta(p)),opers)
            Beta(p)= Min(Beta(p), Val)
            if A >= Beta(p) then
                retourner Beta(p)
            end if
        end for
        retourner Beta(p)
    else
        Alpha(p)= -MAX VAL
        for chacune op dans opers do
            fils = successeur(p, op)
            Val = ALPHA BETA(fils, Max(A,Alpha(p)), B,opers)

            Alpha(p) = Max(Alpha(p), Val)
            if alpha(p) >= B then
                retourner Alpha(p)
            end if
        end for
    end if
end if
```

2.2 Conception

Le programme devra permettre la gestion d'une partie entre deux joueurs humains (ou on demandera au programme de simuler un joueur). Il devra savoir si un coup demande par l'utilisateur est autorisé ou non, si il est autorisé alors il va devoir traduire les règles du jeu et changer le plateau.

2.2.1 Conception du plateau

La conception du plateau la plus simple est celle d'une liste à deux dimensions dans lequel les cases sont NOIRE, BLANCHE ou VIDE, on modélisera les pièces noires par 1, les pièces blanches par -1 et les vides par 0, ainsi notre tableau devra ressembler à la figure ci-dessous .(cf. fig. 2.1)

```
board[8][8]={ {0,0,0,0,0,0,0,0}, //0
              {0,0,0,0,0,0,0,0}, //1
              {0,0,0,0,0,0,0,0}, //2
              {0,0,0,WHITE,BLACK,0,0,0},//3
              {0,0,0,BLACK,WHITE,0,0,0},//4
              {0,0,0,0,0,0,0,0}, //5
              {0,0,0,0,0,0,0,0}, //6
              {0,0,0,0,0,0,0,0}}; //7*/
```

FIGURE 2.1 – Tableau du jeu

2.2.2 Conception du retour en arrière

La conception du retour en arrière a été réalisée par l'implémentation du liste doublement chaînée inverse et ainsi on peut retourner en arrière ou aller vers l'avant. Une liste doublement chaînée est une liste à laquelle chaque élément peut accéder au moyen de pointeurs vers les éléments positionnés immédiatement avant et après dans la liste. Le chaînage se fait donc dans les deux sens, ce qui permet de parcourir la liste en avant et en arrière, ce qui n'était pas possible avec la liste simple. De plus, s'il est facile d'ajouter des éléments à chaque extrémité d'une simple liste, il l'est beaucoup moins lorsqu'il s'agit de supprimer l'élément en fin de liste (au sens de chaînage). La liste doublement chaînée nous facilitera la tâche. En sauvegardant les plateaux et le joueur qui en a joué.

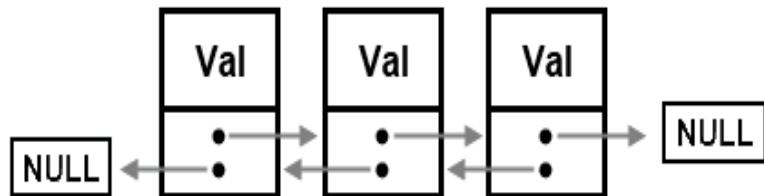


FIGURE 2.2 – Conception d'une Liste doublement chaînée

2.2.3 Conception de L'accueil

La conception de L'accueil suit l'organigramme suivant . Les deux actions sign-in et log-in font appel a

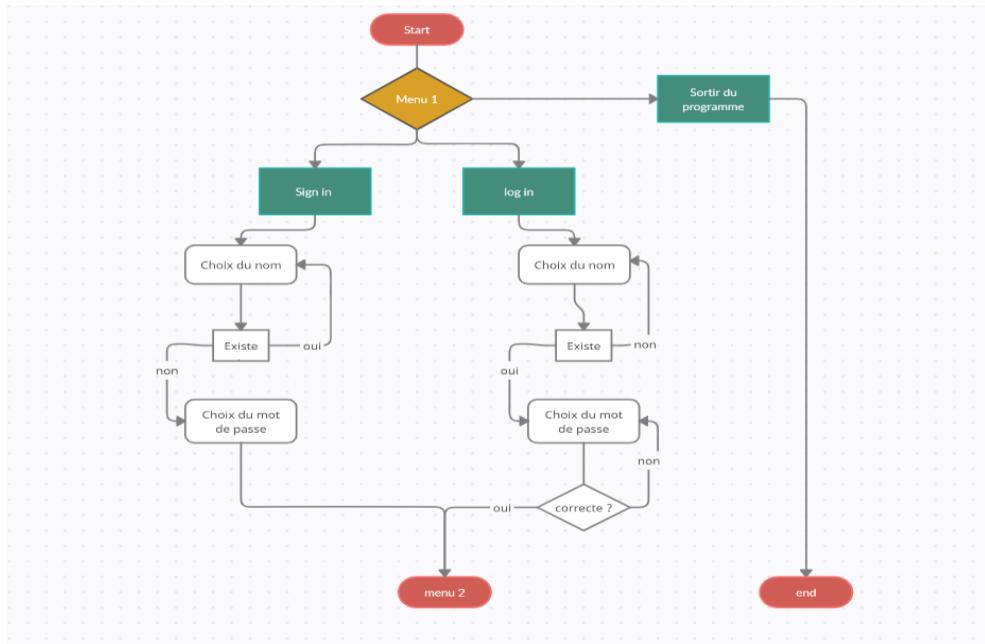


FIGURE 2.3 – Organigramme Menu D'accueil

l'utilisation de fichier où le contenu sera stocker ou retirer sans oublier de dechiffrer les donnees pour que leur accès soit plus ou moins difficile . Une fois que L'utilisateur a suivi les instructions il sera redirige vers le menu 2 ou il trouvera les options du jeu a savoir jouer contre un humain ou contre L'ordinateur .Le shemas suivant explique en details ceci.

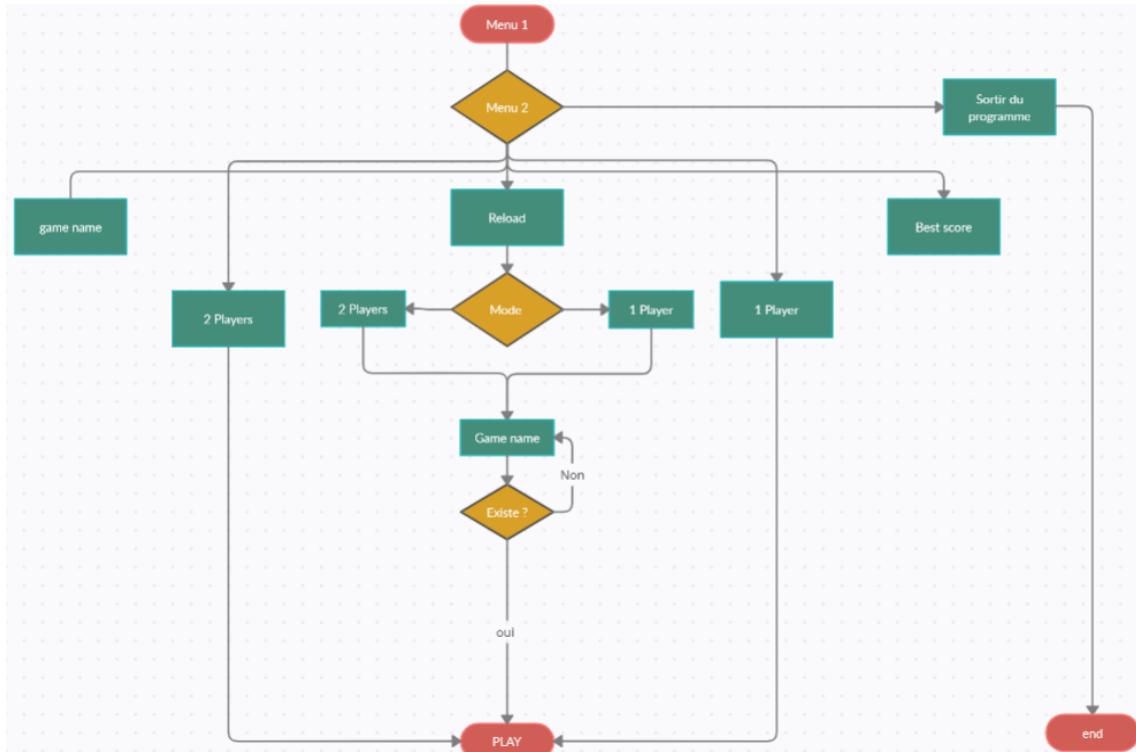


FIGURE 2.4 – Organigramme Menu De jeu

2.2.4 Conception de la table de jeu

Une fois le joueur a réussi à créer un compte il va être redirigé vers l'interface où il va jouer , Ce diagramme explique son fonctionnement.

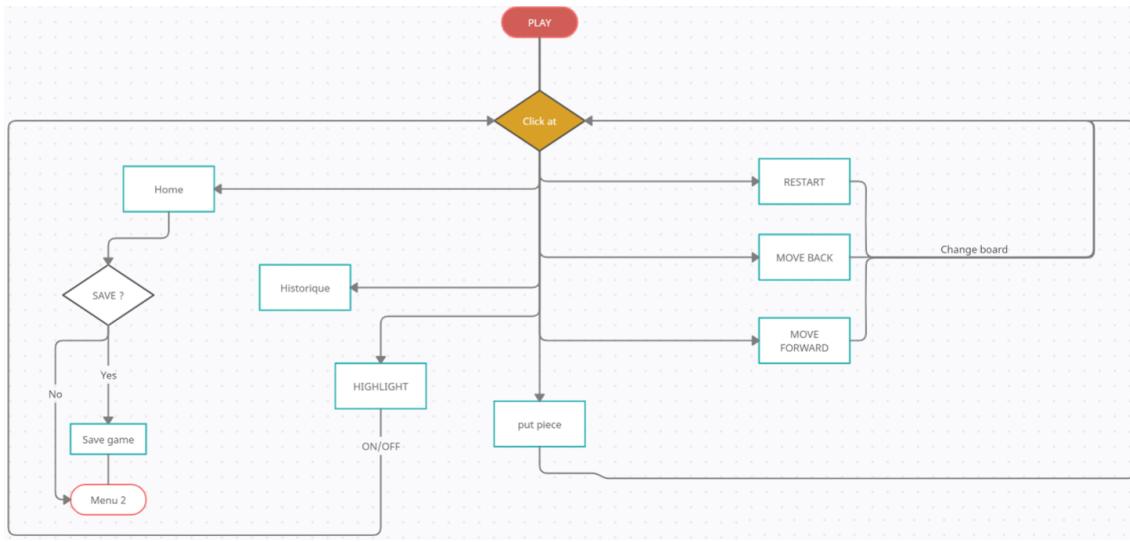


FIGURE 2.5 – Organigramme Menu De jeu

2.2.5 Sauvegarde des parties

Afin de pouvoir sauvegarder la partie en cours et rappeler la partie précédemment enregistrée, la méthode de stockage de fichiers doit être utilisée. En effet, nous avons établi la sauvegarde des plateaux , des joueurs et leurs caractéristiques, de L'historique des mouvements et des utilisateurs et leur coordonnées(nom et mot de passe), puis de revenir et charger n'importe quel partie il a sauvegarde. Si le joueur souhaite reprendre fermer le programme ou se déplacer à l'accueil, il est également recommandé au joueur de choisir de sauvegarder la partie ou non .

Dans la sauvegarde on a utiliser le mode texte et le mode binaire tout en optimisant chaque mode et sa fonction , ainsi que la lisibilité et la sécurité des données.

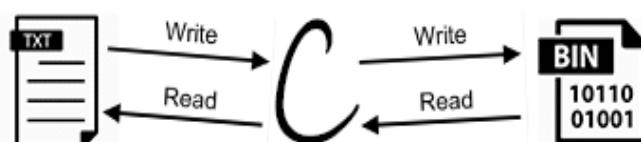


FIGURE 2.6 – Management des fichiers

2.2.6 Language et interface

Le langage qu'on va utiliser dans notre projet est le langage C , puisque c'est le seul langage qu'on a appris et exerce pendant les séances de cours afin de représenter graphiquement l'affichage du code source. Nous avons essayé d'assurer une meilleure organisation et le maximum de clarté en ajoutant des commentaires le long du projet et en divisant notre code sur plusieurs fichiers ".c" et ".h"(mais le temps nous a permis de l'accomplir et donc on l'a abandonnée) nous avons de même essayé de manipuler la bibliothèque SDL[1] (Simple

DirectMedia Layer). Sa decouverte etait penible vu que le language C n'est plus utilise pour la realisation des interfaces graphiques .



FIGURE 2.7 – Logo du Language C



FIGURE 2.8 – Logo de la librairie SDL

2.2.7 Latex

Les documents professionnels nécessitent un logiciel de traitement et LaTex est l'outils parfait pour cette tâche. Il etait donc inévitable de travailler avec ce dernier, en plus que nous voulons profiter du maximum de cette opportunité et ainsi explorer LaTex.[5]



FIGURE 2.9 – Logo du logiciel LaTex

2.2.8 Code-blocks

Au niveau du choix du IDE(integrated development environment) ,on a opte pour Code-blocks vu sa facilite de manipulation ainsi que sa rapidite (on aurait aime travailler avec VS-code mais on devait apprendre le language java-script et plus precisement JSON ce qui n'est pas un tache facile).

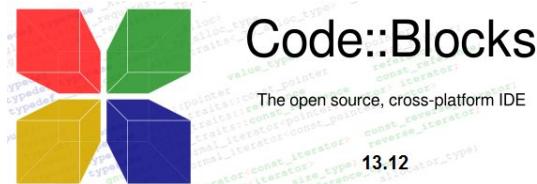


FIGURE 2.10 – Logo du IDE CODE : :BLOCKS

2.2.9 Outils de collaboration et communication

Les outils de collaborations qu'on a utilise pour ce projet sont **Github** et **Microsoft teams**, et dont l'email institutionnel nous a ete tres benefique pour l'obtention des versions PRO des programme.



FIGURE 2.11 – Logo de Github



FIGURE 2.12 – Logo de Microsoft teams

Chapitre 3

Realisation et resultat

Dans ce chapitre, nous énumérerons les plus grandes difficultés rencontrées dans le processus de réalisation du projet. Par conséquent, nous expliquerons le rôle des fonctions apparaissant dans le code source

3.1 Difficultes rencontrées

Cette section détaille le processus d'exécution du projet et les problèmes qui en découlent.

3.1.1 Le temps

Le temps était la contrainte la plus dur à gérer vu qu'on vient de s'initier à la réalisation du projet, et le délai était insuffisant voire même trop serré pour perfectionner notre projet, sa réalisation dans cette durée était un vrai challenge.

3.1.2 L'implementation de SDL, SDL-ttf et SDL-Mixer :

Vu que c'est notre première fois qu'on travaille avec des library importées, L'inclusion de SDL et ses sub-libraries était généralement difficile sans oublier que les tutoriels ou les explications n'existaient presque pas [2][10].

3.2 Fonction du code source

Dans cette partie, nous développerons chaque fonction utilisée dans le code source.

3.2.1 Directive de preprocesseur

Ces directives ont comme rôle l'importation des bibliothèques et les fichiers header contenant les fonctions à utiliser dans le main.

```
1 #include <stdlib.h>
2 #include <stdio.h>
3 #include <string.h>
4 #include <SDL.h>
5 #include <SDL_ttf.h>
6 #include <unistd.h>
7 #include <sys/stat.h>
8 #include <SDL_mixer.h>
```

3.2.2 Définition des constantes

```
1 #define largeur 600 //largeur de la fenetre
2 #define hauteur 600 //hauteur de la fenetre
3 #define BLACK 1 //modélisation des pièces noirs
4 #define WHITE -1 //modélisation des pièces blanches
5 #define EMPTY_SPACE 0 //modélisation du vide
6 #define HIGHLIGHTER 2 //le marqueur
7 #define infinity 1000 //pour L'IA
```

3.2.3 Les structures

Pour modeliser la structure du retour en arriere ou en avant nous avons implementer la structure suivante.

```
1 typedef struct _board_list
2 {
3     int _last_turn;
4     struct _board_list *previous;
5     int board_m[8][8];
6     struct _board_list *next;
7 }
8 }board_list;
```

La structure utilisee pour la realisation de L'IA mode hard .

```
1 typedef struct _list_mouvement
2 {
3     int list_mouvement_possible[28][2]; //possible moves can't be higher than 28 in one turn [8]
4     int indice_list_possible;
5 }list_mouvement;
```

Une structure permettant de stocker une evalution et les coordonnes qui ont mene a son obtention.

```
1 typedef struct _evaluation
2 {
3     int _eval;
4     int coord[2];
5 }
6 }eval;
```

3.2.4 Fonction.c

Les prototypes de tout les fonctions .

```
1 void first_page(SDL_Window *window);
2 void menu(SDL_Window *window,SDL_Event event,int *num_menu);
3 void putpion(SDL_Window *window,int last_turn);
4 void putpion_with_hint(SDL_Window *window,int last_turn);
5 void fondgame(SDL_Renderer *renderer,int last_turn);
6 int if_click_highlight(SDL_Event event);
7 void initialisation_des_coordeunes(SDL_Rect coordeunes_case[8][8]);
8 void coordeune_click(int *ligne,int *colonne,SDL_Rect coordeunes_case[8][8],SDL_Event event);
9 int if_click_replay(SDL_Event event);
10 void restart_game();
11 int if_click_back(SDL_Event event);
12 int if_click_next(SDL_Event event);
13 int if_click_home(SDL_Event event);
14 int if_click_historic(SDL_Window *Window);
15 void save_the_game(SDL_Window *window,int hint_etat,int *programme,int *stay_in_menu,int type,int last_turn);
16 void show(char * text,SDL_Renderer *renderer);
17 void bestscores(SDL_Window *window);
18 SDL_Renderer* fond_first_page(SDL_Window *window);
19 void fond(SDL_Renderer *renderer,char* surface_name);
20 void fond_errornameparty(SDL_Renderer *renderer);
21 void fond_nameparty(SDL_Renderer *renderer);
22 void fond_username(SDL_Renderer *renderer);
23 void fond_password(SDL_Renderer *renderer);
24 void fond_errorusername(SDL_Renderer *renderer);
25 void fond_usernamenotfound(SDL_Renderer *renderer);
26 void fond_errorpassword(SDL_Renderer *renderer);
27 void fond_chooseparty(SDL_Renderer *renderer);
28 void fond_chosemode(SDL_Renderer *renderer);
29 void fond_menu(SDL_Renderer *renderer);
30 void fond_historic(SDL_Renderer *renderer);
31 void rules(SDL_Window *window);
32 int choosemode_inreload(SDL_Window *window,int *last_turn);
33 int chooseparty(SDL_Renderer *renderer,int *last_turn);
34 int player_turn(int last_turn,int *pass_turn);
35 int contour (int pos_l,int pos_c,int player);
36 void click_at(int player,int click_l,int click_c);
37 void reset_h();
38 void set_color(int pos_l,int pos_c,int player);
39 int still_in_board(int pos_l,int pos_c);
40 int search_player(int pos_l,int pos_c,int player,int player_origin,int dl,int dc);
41 void mark_position(int pos_l,int pos_c);
42 void display();
```

```

43 void display_h();
44 void update_list_board(board_list **ptr_list_boards, int last_turn);
45 void free_ptr(board_list **ptr_list_boards);
46 void move_backward(board_list **ptr_list_boards, int last_turn);
47 void move_forward(board_list **ptr_list_boards, int last_turn);
48 int sign_in(SDL_Renderer *renderer);
49 int log_in(SDL_Renderer *renderer);
50 int decode(char *str, int key);
51 int code(char *str, int key);
52 int creat_folder(char *dir_name);
53 int isDirectoryExists(const char *path);
54 int sauvegarde(int *last_turn);
55 int reload(int *last_turn);
56 int save(int *last_turn, SDL_Renderer *renderer);
57 void save_history(int pos_l, int pos_c, int player);
58 void increment_decrement(int i);
59 void best_scores(int score_black);
60 void simple_IA(int *pos_l, int *pos_c);
61 void remove_files();
62 int max(int a, int b);[1]
63 int min(int a, int b);
64 list_mouvement AI();
65 int evaluation(int board_[8][8]);
66 eval minimax(int board_[8][8], int depth, int alpha, int beta, int maximizingPlayer);
67 void set_color_IA(int board_IA[8][8], int pos_l, int pos_c, int player); //coloration //v but we
    should optimize it more
68 void free_ptr(board_list **ptr_list_boards);
69 int still_in_board(int pos_l, int pos_c);
70 void reset_h();
71 void move_backward(board_list **ptr_list_boards, int last_turn);
72 int main(int argc, char *argv[])

```

1

— Fonctions reliées au fonctionnement :

```
1 int contour ( int pos_l , int pos_c , int player ) ;
```

Input : position par rapport à la ligne et la colonne et le joueur actuel. Output : un nombre positif correspondant à la somme des directions auxquelles il a trouvé un coup légal.

```
1 int tell_turn ( int player );
```

Cette fonction permet de donner la main à un joueur donné lorsqu'il est permis de jouer pendant ce tour. Input : le joueur qui a joué pendant le tour précédent. Output : le joueur qui doit jouer dans le tour actuel.

```
1 void click_at ( int player , int click_l , int click_c ) ;
```

Cette fonction permet de changer le tableau en sachant que le mouvement est légal. Input : le joueur qui a cliqué, la ligne et la colonne.

```
1 void reset_h();
```

Permet de réinitialiser Highlight-board;

```
1 void set_color ( int pos_l, int pos_c, int player );
```

Permet de modifier le tableau après l'ajout d'une pièce. Input : ligne, colonne, joueur.

```
1 int search_player ( int pos_l , int pos_c , int player , int player_origin , int
    dl , int dc ) ;
```

C'est la fonction dont l'algorithme a été annoncé dans la partie 2, algorithme 2. Input : ligne, colonne, joueur adverse, joueur, (dl,dc) constitue la direction.

```
1 void move_backward(board_list **ptr_list_boards, int last_turn);
```

Cette fonction permet de revenir en arrière. Input : Structure de la liste de retour en arrière, le dernier joueur.

```
1 void move_forward(board_list **ptr_list_boards, int last_turn);
```

Cette fonction permet d'aller vers l'avant après un retour en arrière. Input : Structure de la liste de retour en arrière, le dernier joueur.

```
1 void update_list_board (board_list **ptr_list_boards, int last_turn) ;
```

- on réaliser les programmes de IA mais on a pu implémenter une version facile dans le code source.

Cette fonction permet d'update la structure de retour en arriere.Input double poiteur sur Structure de la liste de retour en arriere, le dernier joueur.

— Fonctions de sauvegarde :

```
1 int code (char *str ,int key);  
2 int decode (char *str ,int key);
```

Chiffrer et dechiffrer le contenu a l'aide d'une cle.Input :la chaine de character et la cle.

```
1 int creat_folder(char *dir_name);
```

Cree un dossier .Input : chaine de caractere correspondant au nom du dossier.

```
1 int isDirectoryExists(const char *path);
```

Verifie si un dossier existe ou pas.Input : chaine de caractere correspondant au chemin du dossier .

```
1 int save(int *last_turn ,SDL_Renderer *renderer);
```

Fonction qui sauvegarder une partie .Input : le dernier joueur ,SDL-renderer qui est un rendu .

```
1 int reload(int *last_turn);
```

Charger une partie deja sauvegarder.Input :le dernier joueur.

```
1 int sauvegarde (int *last_turn) ;
```

Sauvegarde un utilisateur.

— Fonctions liees aux interfaces :

```
1 void first_page(SDL_Window *window);
```

l'interface initiale, constituée de "sign-in" et "log-in".

```
1 void menu(SDL_Window *window ,SDL_Event event ,int *num_menu);
```

l'interface de menu où on choisit l'action souhaitée (1player : jouer contre une machine,2players : jouer en mode 2joueurs,reload : charger une partie sauvegardée, best score : voir les meilleurs score, rules : savoir les regles ; précisemment comment fonctionne le changement de pièces.

```
1 void putpion(SDL_Window *window);
```

afficher la table modifiée par "set-color" sans indices.

```
1 void putpion_with_hint(SDL_Window *window);
```

afficher la table modifiée par "set-color" avec indices.

```
1 void fondgame(SDL_Renderer *renderer);
```

l'interface au cours de jeu.

```
1 int if_click_highlight(SDL_Event event);
```

cliquer sur le bouton qui permet d'afficher les indices.

```
1 int if_click_back(SDL_Event event);
```

cliquer sur le bouton qui permet de retourner à l'etape précédente.

```
1 int if_click_next(SDL_Event event);
```

cliquer sur le bouton qui permet de retourner à l'etape suivante.

```
1 int if_click_home(SDL_Event event);
```

cliquer sur le bouton qui permet de retourner au menu.

```
1 int if_click_historic(SDL_Window *Window);
```

cliquer sur le bouton qui permet d'afficher l'historique.

```
1 int if_click_replay(SDL_Event event);
```

cliquer sur le bouton qui permet de rejouer depuis le début.

```
1 void initialisation_des_cordonnes(SDL_Rect coordonnes_case[8][8]);
```

faire lier les coordonnes du tableau avec l'interface graphique.

```
1 void coordonne_click(int *ligne ,int *colonne ,SDL_Rect coordonnes_case[8][8] ,SDL_Event event);
```

interpréter la zone cliquée en case du tableau.

```
1 void restart_game();
```

reinitialisation du jeu.

```
1 void save_the_game(SDL_Window *window, int hint_etat, int *programme, int *stay_in_menu, int type, int last_turn);
```

sauvegarder le jeu dans un fichier binaire.

```
1 void show(char * text,SDL_Renderer *renderer);
```

apparaître le contenu d'un fichier.

```
1 void bestscores(SDL_Window *window);
```

afficher les meilleurs scores.

```
1 void fond(SDL_Renderer *renderer,char* surface_name);
2 void fond_errornameparty(SDL_Renderer *renderer);
3 void fond_nameparty(SDL_Renderer *renderer);
4 void fond_username(SDL_Renderer *renderer);
5 void fond_password(SDL_Renderer *renderer);
6 void fond_errorusername(SDL_Renderer *renderer);
7 void fond_usernamenotfound(SDL_Renderer *renderer);
8 void fond_errorpassword(SDL_Renderer *renderer);
9 void fond_chooseparty(SDL_Renderer *renderer);
10 void fond_choosemode(SDL_Renderer *renderer);
11 void fond_menu(SDL_Renderer *renderer);
12 void fond_historic(SDL_Renderer *renderer);
13 void rules(SDL_Window *window);
```

les fonds de l'interface graphique(erreur au niveau du nom,mot de passe ,choisir une partie à télécharger,l'historique...). int choosemode-inreload(SDL-Window *window,int *last-turn); choisir le mode du partie à télécharger (2joueurs ou 1joueur). int chooseparty(SDL-Renderer *renderer,int *last-turn); choisir la partie à télécharger.

Chapitre 4

Interface graphique

Dans ce chapitre nous allons valider les attentes du projet à travers une interface graphique en utilisant la bibliothèque SDL .

4.1 Menu d'accueil

Dans cette partie nous allons réaliser l'interface graphique du menu au'on a déjà initialisé dans le chapitre 2 .(cf. fig. 2.3)

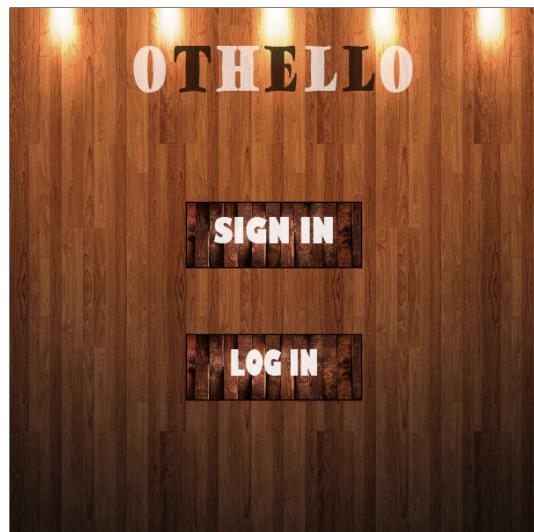


FIGURE 4.1 – Visualisation graphique du premier menu

— Sign in - Log in :

Lorsque l'utilisateur click sur sign in ou log in ,il lui montre les deux interface pour la connexion ou l'inscription en se basant sur la figure (cf. fig. 2.3), avec la possibilité d'annuler a tout moment a l'aide du croix en haut a droite.

On effectue la saisie a l'aide de SDL-ttf.

Si le nom existe déjà lors de Sign-in on affichera le message d'erreur et si le nom n'existe pas lors de log-in on affichera l'erreur suivante. (cf. fig. 4.4 et 4.5)



FIGURE 4.2 – Interface Saisie du nom



FIGURE 4.3 – Interface saisie du password



FIGURE 4.4 – Interface Sign-in erreur nom existe déjà



FIGURE 4.5 – Interface Log-in erreur nom n'existe pas

Si le mot de passe entre pendant le log-in on affichera aussi un message.



FIGURE 4.6 – Interface Log-in erreur mot de passe incorrecte

4.2 Menu de jeu

Apres que joueur a passe le premier menu on presentera le menu du jeu . Ce menu est celui presente dans le chapitre 2 (cf. fig. 2.4) .



FIGURE 4.7 – Visualisation graphique du premier menu

le bouton "1 player" et "2 players" permettent au joueur de jouer contre l'ordinateur ou un "1vs1". On affichera une interface dans laquelle le joueur choisirera le nom de la partie. Si le nom existe déjà on affichera une erreur.

Le bouton RELOAD permet de charger une partie, en choisissant le mode en un premier temps et dans un deuxième, on affichera une interface permettant la **navigation** a l'aide des flèches, lorsque le joueur veut choisir une partie il suffit qu'il click dessus et le programme s'en chargera pour la charger la partie .



FIGURE 4.8 – Interface Saisie du nom



FIGURE 4.9 – Interface choix de la partie



FIGURE 4.10 – Interface Saisie du nom



FIGURE 4.11 – Interface choix de la partie

Le bouton BEST SCORE permet d'afficher les meilleurs scores obtenus par l'utilisateur.

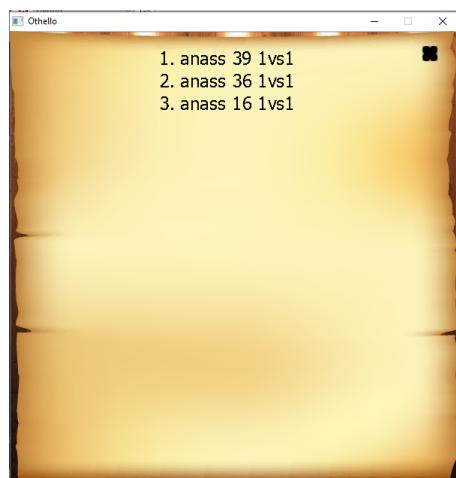


FIGURE 4.12 – Interface de Best-scores

Le bouton Rules permet d'afficher les règles du jeu sous la forme d'une interface graphique.



FIGURE 4.13 – Interface de Rules

4.3 Interface de jeu

Une fois le joueur a choisi le mode. Il sera rediriger vers l'interface de jeu (cf. fig.4.14).

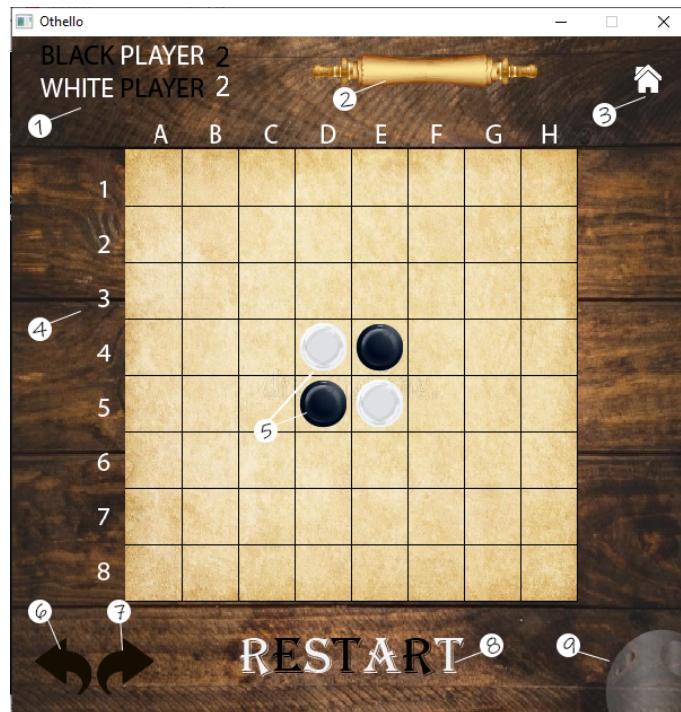


FIGURE 4.14 – Interface de Jeu initialement

1. Score.
2. Historique.
3. Home(retour au menu de jeu).
4. Board.
5. Les pieces.
6. retour en arriere.
7. aller en avant.
8. recommencer.
9. Activer/desactiver Highlight.

Pendant la partie les pieces s'afficheront dans les emplacement dans lesquels chaque joueur click dessus, avec le highlight on obtient le resultat suivant.(Cf fig.4.15)

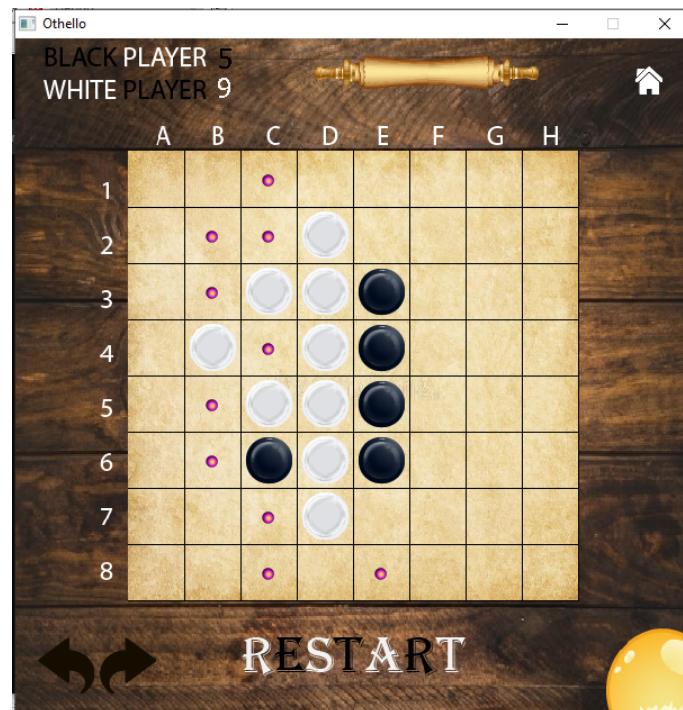


FIGURE 4.15 – Interface de Jeu pendant la partie

Une fois le jeu termine on va afficher l'historique des mouvements.

B:E-6	W:F-6	W:A-1
W:D-6	B:A-8	W:A-2
B:C-6	W:F-5	
W:D-7	B:F-4	
B:C-5	W:G-4	*****
W:B-4	B:G-6	GAME
B:C-3	W:G-5	OVER
W:D-3	B:G-7	
B:E-3	W:G-8	
W:D-2	B:H-8	*****
B:E-8	W:H-7	BLACK
W:D-8	B:H-6	WINS
B:C-7	W:H-5	
W:C-8	B:H-4	
B:B-7	W:H-3	*****
W:A-7	B:G-3	WHITE
B:B-6	W:F-3	:25
W:B-5	B:F-2	BLACK
B:A-5	W:G-2	:39
W:A-4	B:H-2	
B:B-3	W:G-1	
W:B-2	B:F-1	
B:A-3	W:E-2	
W:C-4	B:E-1	
B:A-6	W:D-1	
W:E-7	B:C-1	
B:F-8	W:C-2	
W:F-7	B:B-1	
B:B-8	W:H-1	

FIGURE 4.16 – Interface de Jeu à la fin

Chapitre 5

Conclusion generale

En conclusion , nous devons avouer que retrospectivement nous sommes satisfaits de ce mini-projet puisque nous avons atteint des nouveaux objectifs en apprenant beaucoup de chose .

En effet ce projet nous a permis de mieux comprendre et de bien maitriser le language C et la demarche pour entamer un projet. Sans oublier l'experience et la motivation acquise .

Pour realisation de ce projet nous avons proceder par une demarche parallele en invoquant le graphique avec l'implementation des algorithmes et ca a ete la cle pour la reussite de ce projet .

Enfin nous esperons que cette oportunité se reproduirera une autre fois dans le future pour ainsi accumuler de plus en plus d'expériences .

Si vous etes curieux de decouvrir l'enchainement des evenements ,nous vous recommanderons vivement de visiter notre lien Github du projet.[9]^{1 2}

^{2.} si vous avez une question n'hésitez pas à nous contacter via notre email :othmane.elkarmy@um5r.ac.ma ou anass.eljazouly@um5r.ac.ma

Bibliographie

- [1] <<https://www.libsdl.org/>>, 2020. [Online ; accessed 15-January-2021].
- [2] <https://www.libsdl.org/projects/SDL_ttf/>, 2020. [Online ; accessed 25-January-2021].
- [3] <https://www.libsdl.org/projects/SDL_mixer/>, 2020. [Online ; accessed 25-January-2021].
- [4] <<https://pastebin.com/rZg1Mz9G>>, 2020. [Online ; accessed 5-February-2021].
- [5] <<https://www.geeksforgeeks.org/minimax-algorithm-in-game-theory-set-4-alpha-beta-pruning/>>, 2020. [Online ; accessed 5-February-2021].
- [6] <<https://www.overleaf.com/project>>, 2020. [Online ; accessed 4-February-2021].
- [7] <<https://www.github.com/>>, 2020. [Online ; accessed 25-Junuary-2021].
- [8] <<https://www.microsoft.com/en-ww/microsoft-teams/group-chat-software/>>, 2020. [Online ; accessed 25-January-2020].
- [9] <<https://puzzling.stackexchange.com/questions/31896/othello-most-number-of-legal-moves-in-a-given-board>>, 2020. [Online ; accessed 25-January-2020].
- [10] <<https://github.com/anasseljazouly/projet-C-othello>>, 2020.