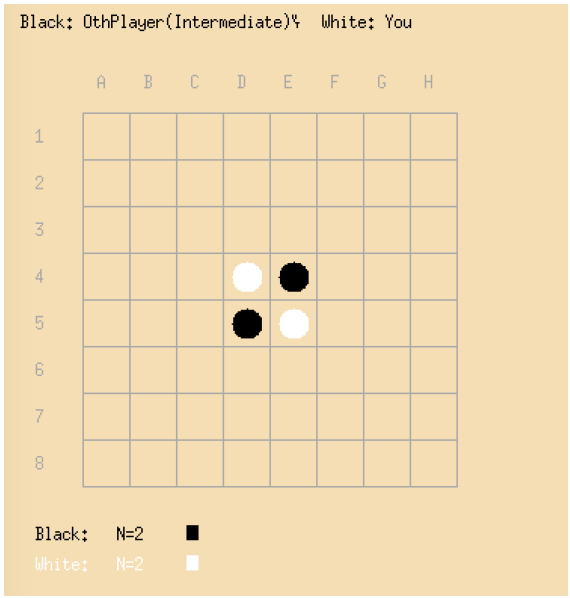
Projet informatique : Jeu d'Othello

1 Introduction

Le jeu Othello ou Reversi est un jeu de société qui se joue entre deux joueurs sur un damier de 64 cases, toutes de la même couleur. Comme au jeu d'échec, on se réfère aux cases en indiquant la colonne, repérée par une lettre, de A à H et la ligne, identifiée par un chiffre, entre 1 et 8. Les joueurs disposent de 64 pions, avec une face noire et une face blanche. Un des deux joueurs joue noir, et l'autre blanc. Quatre pions sont disposés sur le plateau au début du jeu, comme indiqué sur la figure ci-contre. Les deux joueurs jouent alternativement, en tentant d'ajouter un pion sur le plateau. C'est le joueur avec les pions noirs qui commence.



L'objectif, c'est de retourner les pions adverses, en les encadrant avec deux pions de sa couleur, selon une ligne, une colonne ou une diagonale. Si un joueur n'a aucune possibilité de poser un pion de sa couleur en retournant au moins un pion adversaire, il passe son tour. Le jeu s'arrête si toutes les cases du plateau sont occupées au si aucun des deux joueurs ne peut ajouter un pion. Le gagnant est celui qui a le plus de pions de sa couleur à l'issue du jeu. Vous trouverez des informations plus détaillées sur la page de la fédération française d'Othello ou sur la page wikipedia :

- Fédération française d'Othello : https://www.ffothello.org/othello/regles-du-jeu/
- Pages wikipedia : https://fr.wikipedia.org/wiki/Othello_(jeu)

2 Objectif du projet

Écrire un programme capable de jouer à Othello. Le programme devra au minimum satisfaire le cahier des charges suivant :

1. Avoir une représentation interne du plateau de jeu.
2. Être capable de vérifier la validité d'un mouvement, pour les noirs et les blancs.
3. Savoir faire évoluer le plateau du jeu suite à l'ajout d'un pion.

4. Pouvoir jouer pour les blancs ou les noirs, et proposer un mouvement à la suite de chaque mouvement adverse.
5. Afficher une représentation du plateau de jeu sous forme textuelle à l'écran (dans le terminal, avec des `printf()` ou `cout <<`) afin d'aider un joueur humain de jouer contre le programme.
6. Être capable de détecter la fin du jeu et déterminer le vainqueur.

Selon l'avancement de votre travail, vous pourrez améliorer votre programme en y ajoutant une représentation graphique, comme indiqué dans la section 3 ci-dessous, et/ou en évaluant les performances de votre algorithme de jeu comme détaillé dans la section 4.

3 Extension graphique

Il est possible d'améliorer le programme en le rendant capable de représenter graphiquement le plateau de jeu. Vous pouvez utiliser la micro-librairie **gfx** de tracé graphique, qui regroupe quelques fonctions simples de tracés graphiques, amplement suffisante pour ajouter une représentation graphique du plateau à votre programme de jeu Othello. Cette micro-librairie est constituée d'un fichier entête `gfx.h` , du fichier source `gfx.c` que vous pouvez ajouter à votre projet, ainsi qu'un programme principal de test `example_gfx.c` . Les commentaires dans le fichier entête et dans le programme principal, la fonction `btest()` en particulier, vous aideront à comprendre l'utilisation des fonctions et la signification de leurs arguments.

Les fonctions **gfx** utilisent la librairie X11 associée au système de fenêtrage et graphique X-Window, qui est le système graphique de base sous Linux. La librairie peut également être utilisée sur Apple MacOS, moyennant l'installation du serveur XQuartz et la librairie libX11. L'utilisation sur Microsoft Windows devrait être possible, moyennant l'installation des logiciels et librairies appropriés.

Aucune connaissance de X-Window n'est nécessaire pour l'utilisation des fonctions de **gfx**. Néanmoins, les curieux trouveront aisément plus d'informations sur X-Window sur la toile, en particulier :

- Page wikipedia pour une présentation d'ensemble :
  https://fr.wikipedia.org/wiki/X_Window_System
- Une présentation historique par un chercheur de l'INRIA :
  http://ftp.lip6.fr/pub/linux/french/echo-linux/html/intro-X/introX.html
- Le site de la fondation X.org :
  https://www.x.org/wiki/

4 Match d'Othello entre programmes d'ordinateur

Un programme de jeu Othello a été développé pour ce projet, que nous appellerons **OthPlayer** dans la suite. Vous pouvez organiser des matchs entre votre programme et **OthPlayer**. Le code du programme se trouve dans une librairie `libothmagphys.a` , les déclarations

des fonctions d'interface se trouvent dans le fichier `oth4stud.h`. Un exemple d'utilisation sous forme d'un programme principal se trouve dans le fichier `tstudoth.cc`. **OthPlayer** a 4 niveaux de jeu :

- **beg** (Beginner) : niveau débutant
- **int** (Intermediate) : niveau intermédiaire
- **adv** (Advanced) : niveau avancé
- **pro** (Professional) : niveau très avancé

Vous pouvez évaluer les performances de votre programme lors de matchs contre **OthPlayer**. Déterminer la probabilité de gain ou de perte. Dépend-elle de la couleur de jeu ?

5 Programme tothel et tothel_g

Vous pouvez tester le programme de jeu **OthPlayer** et jouer contre l'ordinateur avec le programme **tothel**. **tothel_g** est le même programme, doté d'un affichage graphique du plateau de jeu.

```
 ---- thothel.cc - Othello game
   thotel_g is the program version with graphic display
 Usage : thotel Select [ args]
  (1) thotel play comp_color comp_level
  (2) thotel autobb black_lev(0/1) white_lev(0/1) [tmw=0]
  (3) thotel autobp black/white(player) player_lev base_lev [tmw=0]
  (4) thotel autopp blackplayer_lev whiteplayer_lev [tmw=0]
* play : play against computer, specify color (black/white)
            and player level for the computer
* autobb :  computer plays against itself -
                  using base player (Othello class)
* autobp :  computer plays against itself - base player (Othello class)
                  against OthPlayer class
* autopp :  computer plays against itself - using two OthPlayer classes
 player level : 0 or 1 for the base player implemented in class Othello
      For OthPlayer: beg/int/adv/pro
            (Begginer/Intermediate/Advanced/Professional)
 tmw : wait time in units of 100 milliseconds between moves
            for auto-play,  default=0 (no wait)
   ( + two options for debug : debugprt/debugplay )
```