Projet Tutoré

Othello

Travail à rendre avant le 19/12/2013 à 17h

1. Objectif :

Vous devez réaliser en groupe le jeu d'Othello en version 2 joueurs (l'un contre l'autre). Vous trouverez ci-dessous les règles du jeu ainsi que des consignes d'organisation et de réalisation.

LES REGLES DU JEU

Othello est un jeu de stratégie à deux joueurs : Noir et Blanc. Il se joue sur un plateau unicolore de 64 cases, 8 sur 8, appelé othellier. Ces joueurs disposent de 64 pions bicolores, noirs d'un côté et blancs de l'autre. Par commodité, chaque joueur a devant lui 32 pions mais ils ne lui appartiennent pas et il doit en donner à son adversaire si celui-ci n'en a plus. Un pion est noir si sa face noire est visible et blanc si sa face blanche est sur le dessus.

But du jeu

Avoir plus de pions de sa couleur que l'adversaire à la fin de la partie. Celle-ci s'achève lorsque aucun des deux joueurs ne peut plus jouer de coup légal. Cela intervient généralement lorsque les 64 cases sont occupées.

Position de départ

Au début de la partie, deux pions noirs sont placés en e4 et d5 et deux pions blancs sont placés en d4 et e5 (voir figure 1).

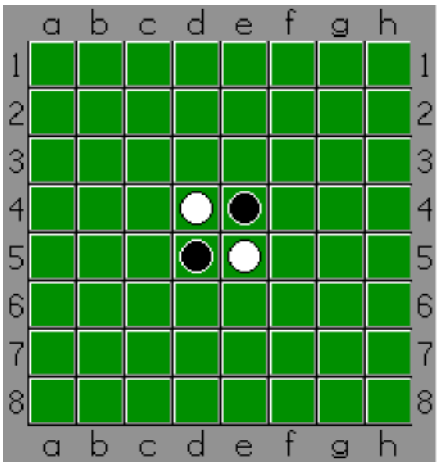

fig. 1 : la position de départ

Noir commence toujours et les deux adversaires jouent ensuite à tour de rôle.

La pose d'un pion

A son tour de jeu, le joueur doit poser un pion de sa couleur sur une case vide de l'othellier, adjacente à un pion adverse. Il doit également, en posant son pion, encadrer un ou plusieurs pions adverses entre le pion qu'il pose et un pion à sa couleur, déjà placé sur l'othellier. Il retourne alors de sa couleur le ou les pions qu'il vient d'encadrer. Les pions ne sont ni retirés de l'othellier, ni déplacés d'une case à l'autre.

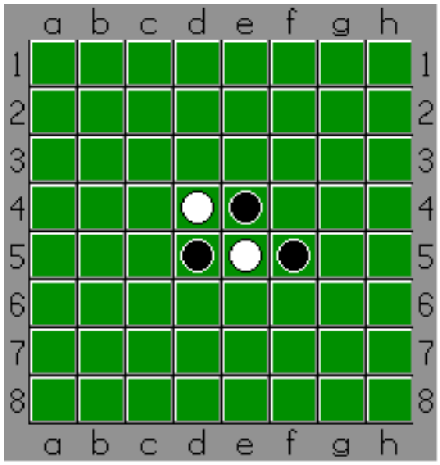

fig. 2 : Noir joue f5 ...

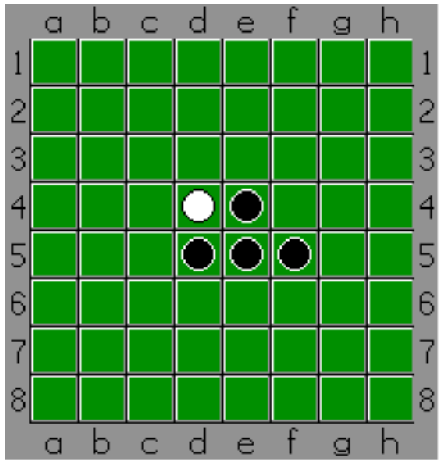

fig. 3 : et retourne e5 !

Le premier coup de Noir est, par exemple, en f5 (voir figure 2). En jouant f5, il encadre le pion blanc e5 entre le pion qu'il pose et un pion noir déjà présent (ici d5) ; il retourne alors ce pion (voir figure 3). Noir aurait aussi pu jouer en e6, c4 ou d3. Notons que ces quatre coups de Noir sont parfaitement symétriques ; Noir n'a donc pas à réfléchir pour choisir son premier coup.

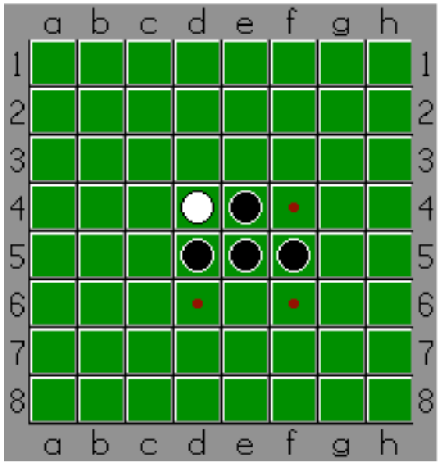

fig. 4 : Blanc f4, f6 ou d6

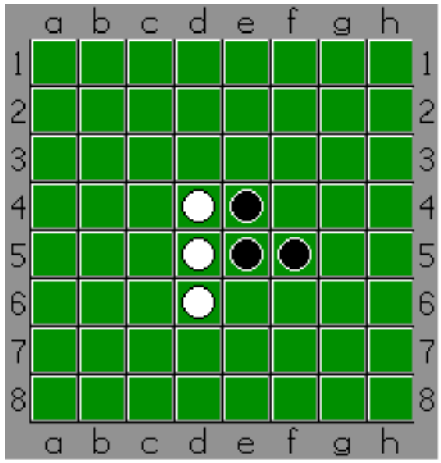

fig. 5 : si Blanc joue d6

C'est maintenant à Blanc de jouer. Il a trois coups possibles (voir figure 4). En effet, il est obligatoire de retourner au moins un pion adverse à chaque coup. Blanc peut donc jouer en f4, f6 ou en d6. On peut encadrer des pions adverses dans les huit directions. Par ailleurs, dans chaque direction, plusieurs pions peuvent être encadrés (voir figures 6 et 7). On doit alors tous les retourner.

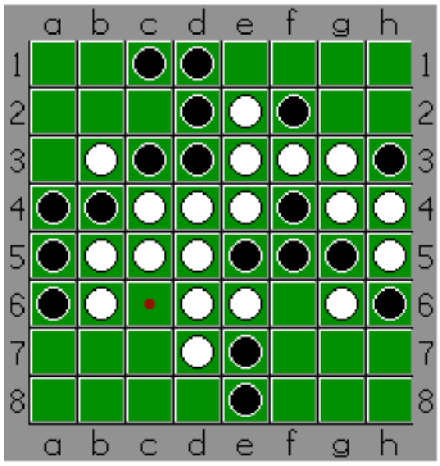

fig. 6 : Noir joue c6 ...

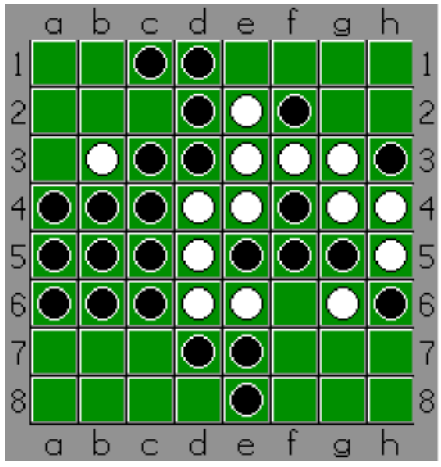

fig. 7 : et le résultat

Le joueur Noir a joué en c6. Il retourne alors les pions b6 (encadré grâce à a6), b5 (encadré grâce à a4), d7 (encadré grâce à e8), c5 et c4 (encadrés grâce à c3). Notons que ni d6, ni e6 ne sont retournés à cause de la case vide en f6.

Il n'y a pas de réaction en chaîne : les pions retournés ne peuvent pas servir à en retourner d'autres lors du même tour de jeu.

Ainsi, sur la figure 8, Noir joue en a5 : il retourne b5 et c5 qui sont encadrés. Bien que c4 soit alors encadré, il n'est pas retourné (voir figure 9).

En effet, il n'est pas encadré entre le pion que l'on vient de poser et un autre pion.

Si, à votre tour de jeu, vous ne pouvez pas poser un pion en retournant un pion adverse suivant les règles, vous devez passer votre tour et c'est à votre adversaire de jouer *(variante possible :* vous pouvez poser un pion mais forcément dans une case voisine d'une case déjà occupée).

Mais si un retournement est possible, vous ne pouvez vous y soustraire (prise obligatoire, comme aux dames)

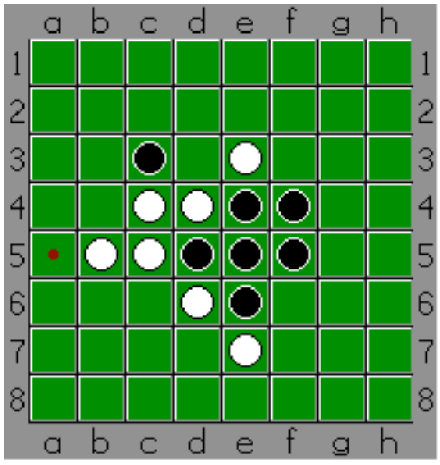

fig. 8 : Noir joue a5...

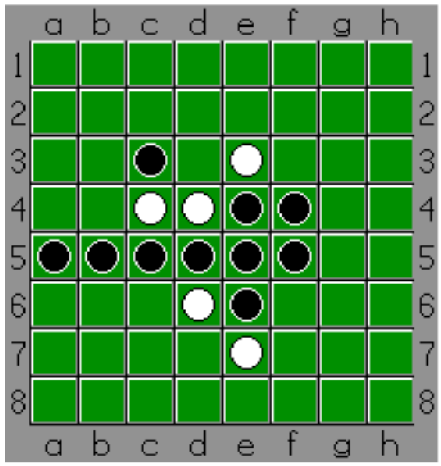

fig. 9 : c4 reste Blanc

Fin de la partie

La partie est terminée lorsqu'aucun des deux joueurs ne peut plus jouer. Cela arrive généralement lorsque les 64 cases sont occupées. Mais il se peut qu'il reste des cases vides où personne ne peut plus jouer : par exemple lorsque tous les pions deviennent d'une même couleur après un retournement. Ou bien comme sur cette position (voir figure 10).

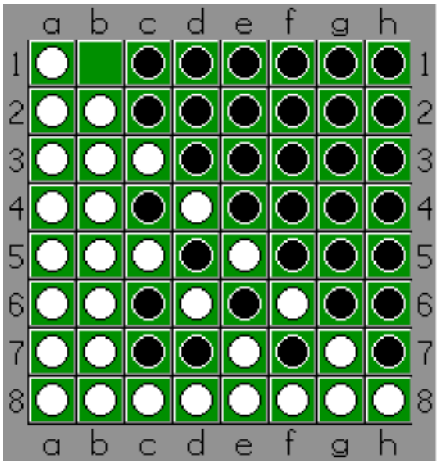

fig. 10 : la partie est finie !

Aucun des deux joueurs ne peut jouer en b1 puisque aucun retournement n'est possible. On compte les pions pour déterminer le score. Les cases vides sont données au vainqueur. Ici, Blanc a 29 pions, Noir 34 et il y a une case vide. Donc Noir gagne par 35 à 29.

2. Implémentation désirée :

Votre travail est d'implémenter ce jeu en mode 2 joueurs ***humains*** (on ne demande pas de faire jouer la machine), et en utilisant des classes.

Vous êtes libres de compléter le travail par les améliorations qui vous semblent intéressantes. Par exemple permettre de gérer plusieurs parties : sauvegarder et charger une partie en cours, et si gérer un fichier des meilleurs joueurs (classés par nombre de victoires). NB : les fichiers ne seront abordés en cours qu'en fin de semestre.

3. Organisation du travail :

Ce projet sollicite 2 compétences acquises au cours du S1 en algo/C++:

1. Compétences algorithmiques : le test de validité d'un coup, la détermination de l'ensemble des pions à retourner, ainsi que le test de fin de partie représentent des difficultés algorithmiques.
2. Compétences en programmation objet : décomposer un problème en sous-problèmes, concevoir et définir des classes bien adaptées au problème à traiter, gérer les cas d'erreur.

Ajouter à cela les compétences pour mener un projet à bien un projet en autonomie et en groupe.

Les compétences en programmation objet n'étant acquises qu'au cours de la 2ème partie du semestre, la 1ère partie du déroulement du projet tutoré se focalisera sur les aspects algorithmiques, sans se préoccuper de définir les classes.

A - Partie algorithmique :
partie la plus complexe, à commencer dès la distribution du sujet

- tester la validité d'un coup donné : le jeu est en cours, et le plateau de jeu est partiellement rempli. C'est au tour d'un des joueurs de jouer. Il veut jouer dans une case donnée. Comment faire pour déterminer que son coup est possible ?

  Indice : il faut envisager les 8 directions possibles (nord, sud, est, ouest, nord-est, nord-ouest, sud-est, sud-ouest).. Retenir la 1ère direction qui est valide : elle sera utile par la suite pour savoir quels pions retourner.
- Jouer un coup : retourner les pions dans le cas d'un coup supposé possible et dont on dispose de la direction
- Fin de partie : déterminer si le jeu est bloqué (plus aucun coup possible) ou si la partie est terminée car toutes les cases sont occupées.

Pour implémenter et tester vos sous- algorithmes avant l'implémentation des classes :

Chacun des sous-algorithmes sera implémenté sous la forme d'un sous-programme prenant en paramètre entre autres un tableau de char à 2 dimensions représentant le plateau de jeu. Des jeux d'essais seront réalisés en écrivant un programme principal dans lequel on initialise le tableau à sa déclaration. On peut ainsi tester facilement différentes configurations (en reprenant celles données dans l'énoncé par exemple).

Exemple : déclaration d'un tableau `pessai` représentant un plateau dans sa configuration initiale :

```
const char B='B';
const char N='N';
const char V='X';  // pour représenter une case vide
const int MAX=8;
char pessai[][MAX]={
    {V,V,V,V,V,V},
    {V,V,V,V,V,V},
    {V,V,B,N,V,V},
    {V,V,B,B,V,V},
    {V,V,V,V,V,V},
    {V,V,V,V,V,V}
};
```

Des explications sur les algorithmes de ces fonctions devront être incluses dans le document à rendre.

B- Partie programmation objet : *à commencer dès le 29 novembre*

Ne tardez pas à vous y mettre : il y a pas mal de code à écrire et il faut se répartir le travail

Définir au moins les 3 classes Plateau, Joueur et Jeu:

- **`Plateau`**, avec au minimum comme attributs :

  `char p[MAX][MAX]` ; // chaque case du plateau contient `'B' ou 'N' ou 'X'`

  MAX vaut 8 dans la règle de base, mais rien n'empêche les plus avancés d'envisager des « niveaux » de difficulté en introduisant un attribut `difficulte`.

  En dehors de l'affichage de la grille, cette classe implémentera la plupart des fonctions algorithmiques citées ci-dessus

- **`Joueur`**, avec au minimum comme attributs :
  `string nom ;` // le nom du joueur
  `char coul ;` // caractère représentant le joueur : 'N' ou 'B'
  `int score ;` // le score du joueur pour la partie en cours
  avec constructeur, accesseurs et modifieur pour le score

- Définir également une classe **`Jeu`** qui contiendra comme attribut des objets des 2 classes Plateau et Joueur :

  ```
  Plateau p; // le plateau de jeu
  Joueur j1,j2;  // les 2 joueurs
  bool joueurCourant ;  //vrai si joueur 1, faux sinon
  ```

  Les méthodes de la classe **`Jeu`** utiliseront les méthodes des classes **`Plateau`** et **`Joueur`**. La classe **`Jeu`** implémentera en particulier des méthodes indiquant si le jeu est bloqué ou si il est fini (plus de case libre). Son constructeur mettra le jeu dans la situation initiale (plateau de départ), demandera les noms des 2 joueurs (qui auront un score nul au départ). Le premier joueur est le joueur Blanc.

Conseil d'organisation : commencez par écrire ensemble les fichiers de déclaration des 3 classes (les "h"), puis répartissez-vous l'écriture des méthodes. Vous pourrez ainsi travailler en parallèle.

4. Séances de questions/réponses :

Vous devez travailler en autonomie. Quelques séances de questions/réponses avec les enseignants qui encadrent ces projets seront organisées. La première séance aura lieu lors de la semaine de rentrée des vacances de la Toussaint, le **mardi 5 novembre**, entre 12h30-14h. Vous devez donc commencer travailler dès maintenant pour arriver avec des questions ! Ces séances sont facultatives. Vous pouvez venir juste pour poser votre question, ou bien rester plus longtemps et écouter celles des autres.

L'information sur le lieu de la première séance, ainsi que sur les horaires et les lieux des séances suivantes sera donnée ultérieurement.

5. Soutenances :

La présentation du travail réalisé fera l'objet d'une soutenance lors de la semaine finale du S1 (après les DS). Chaque étudiant exposera une partie du travail. Il vous appartient de choisir ce que vous souhaitez présenter et comment vous vous répartissez les présentations pour valoriser à la fois votre travail personnel et votre travail de groupe (5mn par personne). La soutenance se terminera par une démonstration (5 mn pas plus)

6. Travail à rendre : avant le 19/12/2013 à 17h :

Vous devez rendre (mail + impression du dossier) à l'enseignant qui corrigera votre projet une archive contenant :

- L'ensemble des fichiers constituant le programme. Les noms des programmeurs de chaque classe doivent apparaître obligatoirement en commentaire, en tête de chaque fichier.
- Un dossier comprenant, au minimum :
  - Une présentation de votre projet
  - Les explications sur les algorithmes de la partie A
  - La description des classes de la partie B
  - les jeux d'essais associés aux tests des méthodes correspondant à la partie algorithmique.
  - Une conclusion dans laquelle seront indiqués au moins les éléments suivants :
    - Organisation du travail dans le temps et entre les membres du groupe,
    - Avancement plus ou moins complet du travail, les difficultés rencontrées,
    - Les extensions éventuelles apportées,
    - Les améliorations envisageables,
    - Une conclusion personnelle sur ce que le projet vous a apporté.

7. Evaluation :

L'évaluation du projet tiendra compte non seulement de la réalisation mais aussi la qualité du travail en équipe (mise en valeur en particulier lors de la soutenance.)

Ainsi, les éléments qui participeront à l'évaluation seront notamment :

- Qualité du travail en équipe
- Contenu du dossier
- Qualité de la programmation
- Soin des affichages
- Gestion des cas d'erreurs
- Soutenance