Reversi

Objectif

Le but de ce projet est de réaliser le jeu du *Reversi*. L'utilisateur pourra jouer contre la machine ou contre un autre humain. Il y aura également la possibilité de voir la machine jouer contre elle même. L'objectif principal étant d'utiliser tous les outils et techniques appris lors du cours d'introduction à la programmation. Ce projet est organisé et géré en suivant la méthode *agile* : il vous sera demandé de réaliser des morceaux de ce projets en un temps limité. Certaines parties étant à réaliser en classe et d'autre en devoir maison. Il faudra réaliser ce projet deux fois : une fois en `C++` et une seconde fois en `Python` . Ces deux langages étant relativement similaire il s'agira, une fois l'algorithme indenté dans langage, de le traduire dans l'autre.

Voir le paragraphe **Sprints** pour le calendrier des rendus.

Principe du jeu

Reversi est un jeu de société combinatoire abstrait, qui oppose deux joueurs : Noir et Blanc.

Il se joue sur un tablier unicolore de 64 cases, 8 sur 8, appelé *othellier*. Les colonnes sont numérotées de gauche à droite par les lettres *A* à *H* ; les lignes sont numérotées de haut en bas par les chiffres *1* à *8*.

Les joueurs disposent de 64 pions bicolores, noirs d'un côté et blancs de l'autre. En début de partie, quatre pions sont déjà placés au centre de l'othellier : deux noirs, en *E4* et *D5*, et deux blancs, en *D4* et *E5*.

Chaque joueur, noir et blanc, pose l'un après l'autre un pion de sa couleur sur l'othellier selon les règles suivantes :

1. Les noirs commencent toujours la partie.
2. A chaque mouvement, le joueur doit capturer les poins adverses.
3. La capture de pions survient lorsqu'un joueur place un de ses pions à l'extrémité d'un alignement de pions adverses contigus et dont l'autre extrémité est déjà occupée par un de ses propres pions.
4. Les alignements considérés pour les captures peuvent être une colonne, une ligne, ou une diagonale.
5. Si le pion nouvellement placé vient fermer plusieurs alignements, il capture tous les pions adverses des lignes ainsi fermées.
6. La capture n'entraînent pas d'effet de capture en cascade : seul le pion nouvellement posé est pris en compte.

Le jeu s'arrête quand les deux joueurs ne peuvent plus poser de pion. On compte alors le nombre de pions. Le joueur ayant le plus grand nombre de pions de sa couleur sur l'othellier a gagné.

Constantes et variables globales

Pour réaliser ce projet nous allons définir et utiliser des variables globales et constantes.

`BLANC`, `NOIR`, `GRIS` **et** `VIDE`**.** Ces constantes, valent respectivement `'B'`, `'N'`, `'G'` et `'V'`. Elles représentent la couleur d'une case ou la couleur possible d'une case ; le `GRIS` correspond donc à la possibilité de placer les deux couleurs et le `VIDE` à l'impossibilité de placer un pion.

`OK`, `ERR_SAISIE`, `ERR_IMPOSSIBLE` **et** `ERR`**.** Ces constantes valent respectivement 0, −1, −2 et −3 et permettrons d'identifier les erreurs d'utilisation.

`j1` **et** `j2` sont des variables de type *chaine de caractères* et ont pour la valeur respective le nom du joueur 1 et le nom du joueur 2.

`c_j1` **et** `c_j2` sont des variables de type caractères et auront pour la valeur `BLANC` ou `NOIR` suivant la couleur du joueur 1 et 2.

`IA1` **et** `IA2` sont des variables de type entière et vaudront 0, 1, 2 ou 3.
- Si `IA1` vaut 0 alors le joueur 1 est un humain.
- Si `IA1` vaut 1, 2 ou 3 alors le joueur 1 est la machine (avec un niveau de difficulté croissant).

De même pour `IA2`.

`grille`**.** Cette variable représente la grille de jeu. C'est un tableau à double entrée de type caractère. La case `grille[i][j]` vaut `BLANC` si à l'intersection de la ligne `i` et de la colonne `j` il y a un pion blanc, `NOIR` s'il y a un pion noir et `VIDE` s'il n'y a aucun pion. On prendra garde. Les valeurs de `i` et `j` sont des entiers entre 0 et 7. Ainsi la position `C4` sur l'othellier correspond à la case `grille[2][3]`.

`test`. Cette variable est également un tableau à double entrée de type caractère (on sera, de la même manière que pour `grille`, attentif au décalage des indices). Cette variable indique la possibilité de placer un pion. Précisément si `test[i][j]` vaut :
- `VIDE`, c'est qu'on ne peut placer aucun pion à la position `i`, `j`.
- `BLANC`, c'est qu'on ne peut placer qu'un pion blanc à la position `i`, `j`.
- `NOIR`, c'est qu'on ne peut placer qu'un pion noir à la position `i`, `j`.
- `GRIS`, c'est qu'on peut placer un pion blanc ou noir à la position `i`, `j`.

Fonctions et procédures

Pour vous aider le projet est divisé en divers fonctions et procédures simplifiant la réalisation de la fonction principale. Elles sont un minimum mais il vous est permis de compléter cette liste pour simplifier votre travail.
- `Plateau`. Cette procédure ne prend pas de paramètre. Elle affiche l'othellier. Elle vous est distribuée.
- `Calcul_Test`. Cette procédure ne prend aucun paramètre. Elle calcul la variable `test`. Précisément, en n'utilisant que la variable `grille` on détermine `test`.
- `Init`. Cette procédure ne prend aucun paramètre. Elle initialise la `grille` et le `test` au début de partie (placement des 4 pions au centre).
- `Compte_Couleur`. Cette fonction prend en paramètre un caractère (qui correspondra à `BLANC` ou `NOIR`) et retournera le nombre de pion de l'othellier ayant la couleur passé en paramètre.
- `Fin`. Cette fonction ne prend pas de paramètre. Elle renvoie `true` si la partie est fini (c'est à dire que plus aucun des deux joueurs ne peut jouer) et `false` sinon.
- `Test_tour`. Cette fonction prend en paramètre un caractère (qui correspondra à `BLANC` ou `NOIR`) et renverra `true` si le joueur de la couleur passé en paramètre peut jouer.
- `Placement`. Cette procédure prend en paramètre deux entiers et un caractères. Le premier entier `i` représentant l'indice de ligne, le second `j` l'indice de colonne et le caractère `c` la couleur. Cette procédure va placer en `i`, `j` la couleur `c`. Les variables `grille` et `test` sont alors mise à jour.
- `Placer`. Cette procédure prend en paramètre un caractère `c` (correspondant à la couleur) et un entier `IA` correspondant à l'adversaire (voir le paragraphes **Constantes et variables globales**). Si `IA` vaut :
  1. 0, alors je joueur est humain. La fonction l'invite donc à saisir une position pour jouer.
  2. 1, 2 ou 3, alors le joueur est la machine. La fonction va sélectionner une place pour jouer en fonction du niveau de difficulté.

  Une fois la position de jeu sélectionnée, la fonction va dans un premier temps tester s'il y a une erreur dans la saisie ou dans la possibilité de jouer.
  1. S'il y a une erreur dans la saisie, la fonction renvoie `ERR_SAISIE`
  2. S'il est impossible de jouer à la place sélectionné, la fonction renvoie `ERR_IMPOSSIBLE`
  3. Sinon, la fonction place le pion (par un simple appel à la fonction `Placement`) et renvoie `OK`.

Sprints

Sprint 1 . Pour vendredi 24 mars à 14h. Sur feuille, rendre :
- Le fonction `Compte_Couleur` rédigée en `C++`
- La fonction `Fin` rédigée en `Python`
- La fonction `Test_Tour` rédigée en `C++`

Sprint 2. Lors de la séance du vendredi 24 mars vous aurez 2h pour rédiger, dans le langage de votre choix parmi `C++` et `Python` , la fonction `Calcul_Test` et la procédure `Init`.

Sprint 3. A la fin de la séance du jeudi 30 mars vous devrez rendre la fonction `Placement` et la procédure `Placer` rédigées dans le langage de votre choix parmi `C++` et `Python` .

Sprint 4. Pour le vendredi 31 mars 23h59, le projet fini est à envoyer à `hebert.iut@gmail.com`. Chaque heure de retard entrainera une pénalité de 1 point sur la note.