

Othello

Le plateau

1 Généralités

L'objectif de ce projet est de construire, à vous tous, un jeu d'Othello sur ordinateur, à l'aide d'OCaml. Votre dévoué enseignant a passé ses vacances à écrire (et réécrire) un logiciel, à le diviser et le modulariser. Cet Othello peut être joué à deux sur une machine ou sur deux machines ou encore contre une intelligence artificielle disposant de nombreux niveaux de difficulté. Vous avez maintenant sous la main toutes les briques nécessaires pour reconstruire le programme, documentation y compris. Tout, mais pas le code source.

Comment obtenir un Othello en état de marche ? Dans une invite de commande, allez dans le répertoire qui contient tous ces fichiers et tapez

```
make .cmd
```

La commande produira un fichier `reversi.exe`, sur lequel vous pouvez double-cliquer ou que vous pouvez lancer en ligne de commande. Si vous tapez

```
reversi.exe --help
```

un mini-manuel s'affichera, avec la liste des options de cet Othello.

Que devez-vous faire ? Dans l'état actuel des choses, le jeu entier a été programmé par votre tyrannique enseignant. Il s'agit d'une injustice à faire cesser aujourd'hui même, en remplaçant, pièce par pièce ce qu'il a fait.

Pour ce faire, choisissez un module, consultez sa documentation, fournie dans le fichier `index.html` du répertoire `doc`, supprimez le fichier `.cmo` correspondant, qui ne vous servira plus à rien et implantez ce module dans un fichier `.ml`.

Votre objectif est d'obtenir un fichier `.ml` tel que

```
make .cmd
```

produise un nouveau `reversi.exe` qui fonctionne aussi bien – voire mieux – que l'original.

À quoi avez-vous droit ? Vous avez 10h. Vous avez le droit à tous les documents, à Internet, à prendre des pauses, à discuter, à poser des questions. Par contre, vous n'avez pas le droit de faire faire votre travail par quelqu'un d'autre.

Vous avez le droit de travailler en binôme. Par contre, les notes seront individuelles. En d'autres termes, pour chaque fonction ou structure de données que vous écrivez, notez dans les commentaires (balise `@author`) le nom de la personne qui s'en est occupée. Seule cette personne recevra des points !

Comment serez-vous notés ? Pour chaque module, vous trouverez un barème. Pour avoir le maximum de points, il faut

- que le module compile
- que le module respecte son interface et s'intègre à un Othello fonctionnel
- que les difficultés que vous avez rencontrées et les limitations de vos algorithmes soient notés dans les commentaires
- que vos fonctions soient propres et lisibles – une fois que vous avez fait fonctionner quelque chose, n'hésitez pas à le retravailler pour améliorer la clarté

- que vos algorithmes soient documentés – ceci compte pour la moitié des points
- que vous ayez marqué votre nom devant chacune de vos fonctions écrites
- éviter de parler politique.

2 Le plateau (environ 20 points)

Othello est un jeu de plateau. Il faut donc définir quelque part ce qu'est un plateau. Pas simplement la manière dont le plateau est affiché, ce qui est le rôle de l'interface graphique, mais les dimensions du plateau, le genre de choses qu'on peut mettre sur un plateau, à quoi ressemble le plateau de départ, la notion de pions blancs et noirs, comment vérifier si le plateau est plein, comment changer une case sur le plateau, etc.

Ce module est conçu pour fonctionner sans effets de bord. Ainsi, si l'on demande au module `Plateau` d'ajouter un pion noir sur une case du plateau à l'aide de `set p ~x:3 ~y:3 Noir`, cette fonction renverra un nouveau plateau `p'` dans lequel une case a été modifiée. Le plateau original `p` lui-même n'aura pas été modifié et pourra être réutilisé.

Le module `Plateau` n'a aucune idée des règles du jeu. Il ne sait pas qu'ajouter un pion peut causer des retournements, il ne sait pas qui joue en premier, etc.

Il s'agit d'un des modules les plus simples.

Dépendances Le module `Plateau` utilise le module `Utilitaires`, que vous n'avez pas à réimplanter.

Utilisateurs Le module `Plateau` est utilisé par

- `Main`, pour initialiser le plateau et choisir les couleurs des joueurs
- `InterfaceTk`, pour consulter le contenu d'un plateau de manière à l'afficher
- `InterfaceGraphics`, pour consulter le contenu d'un plateau de manière à l'afficher
- `Vue`, pour transmettre des plateaux à afficher
- `Evaluateur`, pour que l'ordinateur puisse déterminer si une situation est bonne
- `Ia`, pour demander à l'`Evaluateur` d'évaluer des situations
- `Controle`, pour faire passer les plateaux d'un joueur à un autre
- `Client`, pour communiquer des plateaux sur le réseau
- `Serveur`, pour communiquer des plateaux sur le réseau
- `Arbitre`, pour implanter les règles du jeu
- `Encerclement`, pour implanter la notion d'encerclement et de retournement
- `Texte`, pour afficher un plateau

Conseils Commencez par les définitions de types et de fonctions les plus simples et progressez vers les plus difficiles.

- le type `couleur`
- le type `case`
- la valeur `largeur`
- la valeur `hauteur`

- le type `t` (le plus simple est d'utiliser un tableau à deux dimensions de `case`)
- la fonction `autre_couleur`
- la fonction `get`
- une fonction interne `copie`, qui copie complètement un `t`
- la fonction `set` (à l'aide de `copie`)
- la fonction `exists` (en utilisant par exemple `Utilitaires.Calcul_int_int_fini` ou `Utilitaires.Calcul_booleen_fini`)
- la fonction `pour_tous` (en utilisant par exemple `Utilitaires.Calcul_int_int_fini` ou `Utilitaires.Calcul_booleen_fini`)
- la fonction `fold`