

Othello

Encerclements, retournements et trahisons

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 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.

Encerclements et retournements (environ 30 points)

Othello est un jeu d'encerclements et de retournements de situation et de pions. Tout cela est traité dans le module `Encerclement` (fichier `encerclement.ml`), qui permet à l'arbitre de déterminer si le fait de placer un pion sur une case du plateau va provoquer un encerclement et, le cas échéant, de déterminer quels pions adverses doivent être retournés.

Ce module est conçu pour être utilisé uniquement par `Arbitre`.

Dépendances Le module `Encerclement` utilise

- `Direction`
- `Plateau`

Vous êtes donc encouragés à consulter la documentation de ces modules et à vous servir de leur contenu. Vous n'avez pas à les réimplanter.

Utilisateurs Le module `Encerclement` est utilisé par `Arbitre`

Suggestions

- Commencez par faire la liste des fonctions que vous devez implanter. En attendant de commencer chaque fonction, remplacez leur corps par

```
failwith "la fonction Encerclement.XXX n'est pas encore implantée"
```
- Commencez par implanter les fonctions, valeurs et types les plus simples et progressez en direction des constructions les plus compliquées. Voici une suggestion d'ordre. N'essayez pas de tout faire, ni même de tout comprendre, d'un coup.
 1. `chercher_encerclement_selon_direction`
 2. `chercher_encerclement`, à `chercher_encerclement_selon_direction` et de `Direction.fold`
 3. `placer_selon_direction`