

Othello

Programme principal

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.

Le programme principal (environ 60 points)

Le programme principal (module `Main`, fichier `main.ml`) gouverne le lancement du jeu. En fonction des arguments qu'on lui passe en ligne de commande, il va soit lancer une partie à deux joueurs sur le même ordinateur, soit lancer une partie contre l'ordinateur, soit encore lancer une partie contre le réseau, voire lancer un serveur.

Le module `Main` joue le rôle du chef d'orchestre. Toutes les fonctionnalités existent déjà dans les divers modules, il ne reste qu'à les assembler.

Il s'agit du module le plus confus et qui demande le plus de discipline, puisqu'il a accès à toutes les fonctionnalités d'Othello – et qu'il ne faut pas se perdre entre toutes ces fonctionnalités.

Dépendances Le module `Main` dépend de

- `Plateau` (pour choisir une couleur lorsqu'on joue contre l'ordinateur)
- `InterfaceTk` (pour initialiser l'interface graphique Tk)
- `InterfaceGraphics` (pour initialiser l'interface graphique Graphics)
- `Ia` (pour initialiser l'IA)
- `Evaluateur` (pour initialiser l'IA)
- `Vue` (pour brancher une interface graphique au contrôleur)
- `Controle` (pour initialiser le contrôleur de l'interface graphique)
- `Client` (pour lancer le jeu sur réseau)
- `Serveur` (pour lancer l'arbitre réseau)
- `Arbitre` (pour déterminer qui joue en premier)

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 Aucun module n'utilise le module `Main`.

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 Main.XXX n'est pas encore implantée"
```
- Commencez par écrire une version de ce module qui lance systématiquement une partie à deux joueurs sur le même ordinateur avec l'interface `LabTk`. Vous ajouterez le reste au fur et à mesure.
- Pour demander à OCaml de lancer une fonction (mettons la fonction `f : unit -> unit`) à chaque fois que ce module est chargé, il vous suffit d'ajouter quelque chose comme

```
let resultat = f()
```

quelque part dans le module. Par la suite, lorsque vous lancerez `reversi.exe`, la fonction `f` sera automatiquement exécutée. C'est l'équivalent d'ajouter une fonction `public static final void main(String arg[])` en Java, sauf que c'est plus simple – et vous pouvez en mettre plusieurs.
- Lorsque vous aurez besoin que `Main` comprenne les arguments de la ligne de commande, consultez la documentation du module `Arg` dans le manuel de OCaml.

BON COURAGE !