TP3 — « OTHELLO »

Dans ce travail, tu devras apporter des améliorations à une version très élémentaire d'un jeu de stratégie Othello.

Il y aura beaucoup de travail à faire et tu peux modifier à peu près tout dans le projet de base qui t'es donné (dossier « **Othello – version de base** »). Tu dois par contre garder en tête qu'il faut avoir une raison valable pour changer du code qui fonctionne déjà (un principe important à respecter en entreprise) ou la structure de ce code.

RÈGLES DU JEU

Voici les règles que nous utiliserons pour le jeu. Elles sont inspirées principalement du site http://www.ffothello.org/othello/regles-du-jeu/.

- Le jeu se joue sur une grille de 64 cases (8 par 8).
    - Les coordonnées des cases vont de A à H de gauche à droite et de 1 à 8 du haut vers le bas.
- Au début de la partie …
    - Deux pions blancs sont placés en D4 et E5.
    - Deux pions noirs sont placés en E4 et D5.
- Le premier joueur est toujours celui qui contrôle les pions noirs.
- Il y a deux conditions à vérifier pour qu'un coup soit considéré comme étant légal.
    - Le pion doit être placé sur une **case vide qui est adjacente à un pion adverse**.
    - Le pion doit de plus **encadrer au moins un pion de l'adversaire**.
        - Par « encadrer », on entend qu'au moins un pion de la couleur de l'adversaire se retrouvera positionné entre le nouveau pion du joueur et un pion de ce même joueur déjà présent en jeu. Ceci peut être fait sur un axe horizontal, vertical ou diagonal.
- Quand un coup est joué, tous les pions de l'adversaire qui sont encadrés **changent de couleur** et prennent celle du joueur qui vient de jouer son coup.
- Si aucun coup légal n'existe pour un joueur, il **passe son tour**.
    - Si les deux joueurs passent leur tour l'un après l'autre, **la partie se termine**.
- Lorsqu'une partie est terminée, le joueur ayant la couleur avec le plus de pions en jeu gagne.

TRAVAIL À FAIRE

Bien que les améliorations à apporter soient présentées une par une, ta solution doit fonctionner comme un tout cohérent, pas une série de modifications qui s'ignorent les unes et les autres. Autrement dit, garde une vue d'ensemble sur ta solution, car il se peut très bien que plusieurs améliorations puissent se faire ensemble et en harmonie.

REFACTORISATIONS

- La classe Point a été utilisée à plusieurs endroits dans le code. Point utilise le type *double* alors que notre application ne nécessite que des variables de type *int*.
    - Tu dois créer une nouvelle classe qu'on appellera **Coordonnee** et qui fera exactement comme Point, mais avec des ***int***.
    - Point ne devra plus être utilisé dans l'application.
- La couleur (blanc ou noir) est représentée par une chaine de caractères. Il n'y a pourtant que deux valeurs possibles dans Othello et ça ne changera jamais.
    - Tu dois créer une **énumération pour représenter ces couleurs**.
    - Le format string ne devra plus être utilisé pour représenter des couleurs dans l'application.
- L'apparence de la grille de jeu est présentement déterminée par le XAML de JeuOthelloControl. Bien que nous soyons certains que le nombre de cases ne changera jamais, la taille des cases, elle, pourrait changer.
    - Pour que ce soit possible, l'interface de la grille de jeu devra être définie dans le *code behind*.
    - La méthode qui fera le travail sera nommée **DefinirGrid**.
- À quelques endroits, on compare GrilleJeu.EstCaseBlanche à null pour déterminer s'il y a quelque chose dans la case. Il y a **moyen de mieux faire**, et c'est à toi de corriger la situation.

MODIFICATIONS

- Il faut ajouter à l'interface une indication du **nombre de pièces que chaque joueur contrôle**.
    - L'apparence de cet affichage doit respecter le style graphique du jeu lui-même (y correspondre).
    - Cet affichage doit faire partie de la grille de jeu.
- Il faut ajouter à l'interface un bouton qui permet de commencer une nouvelle partie.
    - Ce bouton lancera un écran de démarrage de partie.
    - L'écran de démarrage de partie devra respecter les critères suivants :
        - Il est fait à l'aide d'un **UserControl**.
        - Il est affiché **au-dessus** de tous les autres éléments d'interface, ce qui fait qu'il bloque l'accès au reste de l'interface.
        - Il a une section, au centre, dans laquelle on retrouve les éléments d'interface.
        - Autour de ce centre, l'espace inutilisé est occupé par un **rectangle d'une couleur noire**, mais avec une **opacité** qui permet d'entrevoir le contenu qui se trouve en dessous.
    - Si une partie est en cours, il faudra **demander à l'utilisateur** s'il veut bien démarrer une nouvelle partie et perdre celle qui est en cours.
        - Une partie est considérée comme étant « en cours » si au moins un coup a été joué et la partie n'est pas terminée.
    - On devra avoir la possibilité de **changer la taille des cases de jeu** lors du démarrage d'une partie.
        - Il faudra déterminer un intervalle des valeurs acceptées pour la taille des cases. Cet intervalle devra être clair et toujours respecté.
        - La taille **minimum** permise devra être suffisamment grande pour que le jeu soit utilisable.
        - La taille **maximale** permise devra être raisonnable.
        - **Par défaut**, la taille sera celle que l'application utilise présentement.
    - On devra avoir la possibilité de choisir un **jeu de couleur** pour les pièces du jeu.
        - Il faudra offrir **trois** jeux de couleurs prédéterminés à l'utilisateur.
        - Un « jeu de couleur » est une paire de couleurs utilisée pour les pièces des deux joueurs.
        - Les jeux de couleurs offerts devront permettre de clairement différencier les pièces de chaque joueur. Il faut éviter les jeux de couleurs désagréables à regarder.
        - Tu peux ajouter un quatrième jeu de couleur farfelu, si ça te fait plaisir !
        - La combinaison de **blanc et noir** doit faire partie des choix offerts et c'est le choix par défaut.
    - Il devra être possible de choisir d'**annuler** le lancement d'une nouvelle partie.
    - Au lancement de l'application, l'écran de démarrage de partie sera la première chose affichée. **Dans ce cas** particulier, il ne sera **pas possible d'annuler** le lancement de la partie puisqu'il n'y en a aucune en cours.
    - Les parties opposeront toujours le joueur au jeu (l'IA).

- Il est présentement possible de jouer des coups illégaux.
    - L'intelligence artificielle (**IA**) respecte la règle qui demande qu'un coup soit joué sur une case adjacente, mais **ne respecte pas l'obligation d'encadrer** au moins une pièce de l'adversaire.
    - Les coups du **joueur** ne sont sujets à **aucune vérification**, pour le moment.
    - Puisque les vérifications à faire pour les deux joueurs sont les mêmes, il serait préférable de ne **pas dupliquer** le code.
- L'IA est présentement plutôt simpliste. Son choix est fait purement par le hasard.
    - Tu dois **améliorer son fonctionnement**. Aucun algorithme n'est imposé, mais tu dois trouver une logique qui va au-delà de « choisir un coup au hasard parmi les coups légaux ».
    - Soyons clairs. Il n'est pas question de faire une IA compétitive, il suffit de dépasser le niveau présent de complexité. Du moment que ton algorithme est une amélioration et qu'il fonctionne sans erreurs, il sera considéré comme acceptable.
    - Tu dois fournir des **explications sur le fonctionnement** de cet algorithme dans un **fichier Word**. Les explications doivent me **permettre de comprendre ta logique**. Elles doivent donc être claires et complètes.
- Pour le moment, quand on clique sur une case occupée, la pièce est retournée. Cette fonctionnalité n'existe que parce que le jeu ne fait pas déjà ce travail.
    - Lors de l'exécution d'un coup, que ce soit un coup du joueur ou de l'IA, le jeu devrait **lui-même déterminer quelles sont les pièces à retourner** et les retourner lui-même.
    - Puisque le jeu va maintenant s'occuper de retourner les pièces, les clics sur les cases occupées devraient être **ignorés**.
- Les commentaires ont été oubliés dans l'application.
    - Tu dois **ajouter les commentaires manquants** en respectant les exigences indiquées cette session à ce sujet.
    - Il faut bien sûr tenir en compte que ces commentaires devront expliquer ce que fait la version finale du code. Autrement dit, si tu apportes des modifications dans le code, les commentaires doivent refléter ces modifications.
- Finalement, l'interface a besoin de **raffinement**. Par exemple, mais ne te limite pas seulement à cet exemple, le titre de la fenêtre est encore « MainWindow ».

BONUS (5%)

Tu peux, si tu le désires, ajouter une fonctionnalité au jeu. Cette fonctionnalité sera évaluée sur une combinaison de sa **complexité**, la **qualité de son implémentation** et son **originalité**. Un seul ajout sera accepté pour ce bonus.

Si tu choisis d'ajouter une telle fonctionnalité, tu dois la nommer et fournir une **explication** à son sujet dans le même **fichier Word** qui contient tes explications sur l'intelligence artificielle.

PONDÉRATION

Élément	Valeur
Refactorisations	20 %
Fonctionnement des modifications demandées	15 %
Ensemble du fonctionnement du lancement d'une nouvelle partie	30 %
Intelligence artificielle	25 %
Commentaires	10 %
Bonus	+5 %

Puisque des points sont attribués pour les commentaires, il n'y a pas de pénalité associée à ceux-ci.

Le **français** doit être bien écrit (et est sujet à la pénalité habituelle de 10 %).

REMISE

La remise se fera le **lundi 15 mai**, à la **fin du cours** (à 14 h 20 au plus tard).

- Tu dois remettre ton projet au complet dans un dossier nommé « **TP3** » suivi d'un soulignement et de tes **initiales**.
  (ex. : TP3_ID pour Ivan Desjardins)
- Tu dois aussi remettre le fichier Word contenant les explications demandées dans ce même dossier. Le fichier sera nommé « **Explications** », suivi d'un soulignement et de tes **initiales**.
  (ex. : Explications_ID.docx pour Ivan Desjardins)
- Le projet doit compiler sur la version de Visual Studio du collège.
  Si tu utilises une version différente à la maison, assure-toi que tout compile et fonctionne correctement au Cégep.