**Nom du projet : Othello**

**Professeur : Olivier Hüsser**

**Elèves : Schaffo Raphaël, Gygi Damien**

# **Introduction**

Dans le cadre du cours de « .Net » de troisième année de l’école d’ingénieurs à Neuchâtel, nous devons créer un Othello (en anglais Reversi) fonctionnel respectant toutes les règles officielles du jeu. En plus de cela, nous devons suivre plusieurs contraintes que vous pouvez trouver dans le cahier des charges.

# **Réalisation**

Lors de la phase de réalisation du projet, nous avons très rapidement identifié certains points chauds qui pourrait être plus difficile à implémenter. Voici une liste de ceux-ci :

* Comment vérifier de manière efficace si une case est jouable ou non ?
* Comment retourner toutes les bonnes cases suite à l’ajout d’une pièce ?
* Comment binder un widget et le score ?
* Comment enregistrer et ouvrir un état de jeu ?

Pour chacune de ces questions, nous avons tenté d’élaborer la solution la plus efficace, compréhensible et simple à implémenter (dans l’idée de la convention « Dry & Kiss »).

# **La case est-elle jouable ?**

Cette fonction doit vérifier, tout dépendant la couleur du joueur actuel, quelles cases sont jouables ou non. Pour savoir si une case est jouable, il faut vérifier ses 8 voisins (donc 8 directions). Cela demande beaucoup de réflexion si on veut éviter un grand nombre de tests. Pour éviter ce grand nombre de tests et faciliter un maximum le code, nous avons eu l’idée de créer un tableau qui contient les 8 directions et en une seule boucle on vérifie ces 8 directions. En peu de lignes on obtient le résultat désiré. Pour encore optimiser notre implémentation, dès qu’une direction indique qu’une case est jouable on ne vérifie plus les autres directions. On aura donc moins de tests lors de l’exécution.

# **Le « Flip » de la pièce**

Pour retourner les pièces, nous avons dû simplement réutiliser la boucle de vérification des 8 directions qu’on a créées auparavant pour savoir si une case et jouable ou non et récupérer grâce à cette boucle toutes les cases qui doivent être retournées. Après avoir parcouru toutes les directions on retourne toutes les pièces nécessaires.

# **Enregistrer et recharger une partie**

Pour faciliter un maximum l’enregistrement et le chargement de partie, nous avons simplement crée un fichier qui contient l’état de chaque case sous forme d’un caractère (e=libre, b=black, w=white, i=jouable) tous séparés par un point-virgule. A la fin de cette chaîne de textes contenant l’état des cases, on ajoute le caractère « f « si c’est le tour du joueur noir et « t » si c’est le tour du joueur blanc. Quand on recharge le jeu on lit cette chaîne de textes, on l’interprète, on met les cases du jeu dans le bon état on donne la main au bon joueur et on rafraichit l’affichage graphique.

# **Binder avec un widget**

Il nous a suffi d’un exemple sur le net qu’on a suivi pour se rendre compte que le « data binding » est fort simple à implémenté et performant.

# **Bonus**

Nous avons ajouté comme bonus la possibilité de changer la couleur des cases.