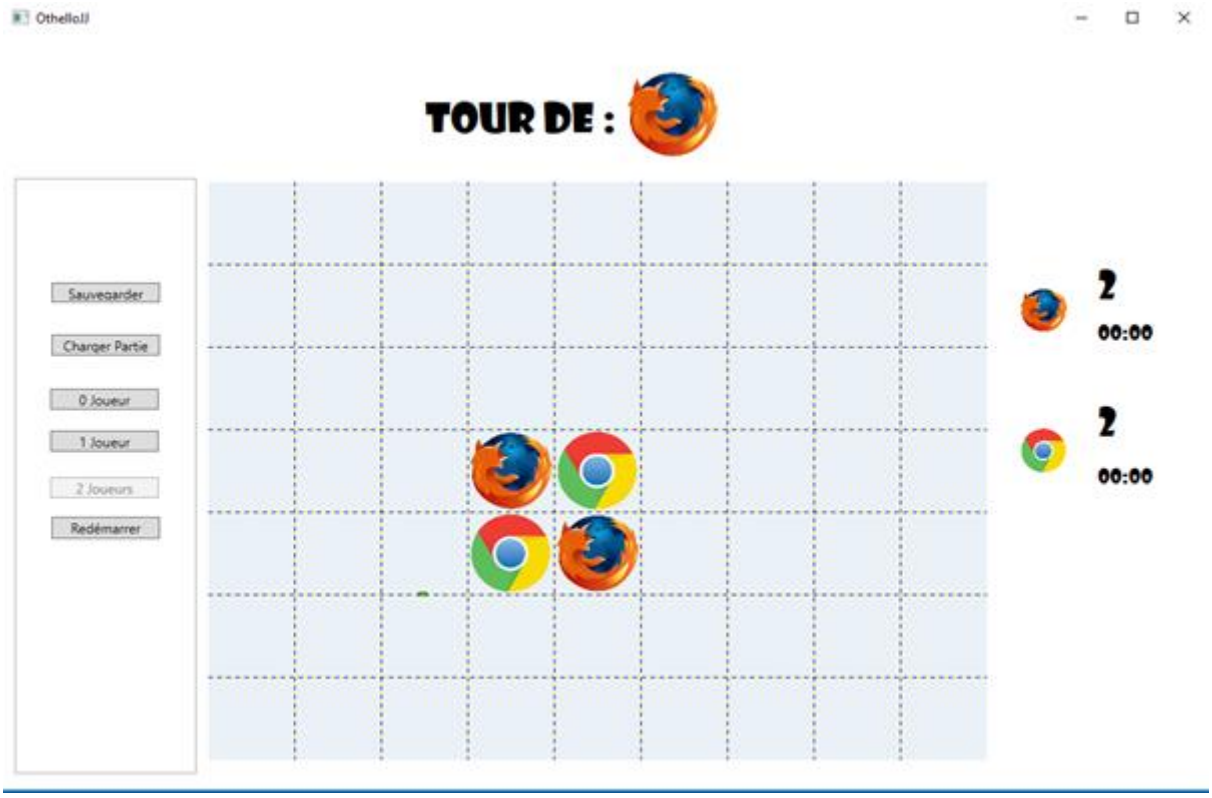
OthelloJJ

1. Interface



Le jeu permet de jouer contre une IA basique ou joueur contre joueur. Il est possible de redémarrer une partie, de la sauvegarder ou de la charger. Le score et le temps de chaque joueur sont affichés à droite. Le joueur qui doit jouer ce tour est affiché en haut.

2. Gestion de la grille

La grille du jeu est une *grid* dont les lignes et les colonnes sont créées en C# au démarrage. Pour savoir sur quelle cellule nous avons cliqué, la classe *MainWindow* possède une fonction, déclenchée au clic sur la *grid* qui regarde la case qui correspond à la position de la souris.
Les pions sont ajoutés dans des *Controls.Image* qui sont directement ajoutés dans la *grid*, les cases vides ne contiennent rien.

3. Update

Dans la classe *Game,* une méthode *Update* permet de passer au tour suivant et de définir les prochains coups visibles (voir prochain point). La grille du jeu est ensuite nettoyée et les images sont toutes recréées d'après le tableau de valeur qui a été modifié.

4. Gestion des coups possibles

Les coups possibles sont définis dans le tableau qui reflète notre grille. Pour trouver si un coup est possible, toutes les cases sont testées, en partant dans toutes les directions. Si dans l'une des directions il y a des pions à retourner alors l'emplacement est possible et il est marqué.

5. Gestion du temps

Pour la gestion du temps de chaque joueur, nous avons employé un *DispatcherTimer*. Le DispatcherTimer est exécuté dans le *UI-Thread* ce qui permet de directement changer l'affichage des temps dans l'interface utilisateur. Il faut faire attention lors de la désérialisation, car la méthode dans le *delegate Tick* reste celle de la classe avant le désérialisation. Il faut donc enlever la méthode dans le delegate et mettre celle de la classe désérialiser après la désérialisation.

6. Sauvegarde et chargement d'une partie.

Pour sauvegarder une partie, nous avons employé la Sérialisation binaire. La Sérialisation consiste à mettre à la suite des informations. Dans notre cas on sérialise une classe, donc tous ses attributs sont mis à la suite dans un format binaire et sauvegarder dans un fichier. jj. L'attribut ImageSource de notre class data n'est pas sérialisable. Il faut donc implémenter une méthode marquée [*OnDeserialized*] qui s'exécute après la désérialisation et qui permet entre autres de remettre les images dans les attributs non sérialisable et de mettre la bonne méthode de la *delegate* du DispatcherTimer. Pour le chargement de la partie, nous désérialisons un fichier.jj et affectons les données à l'objet *Game*. Il faut aussi affecter le nouvel objet *Game* au *DataContext* pour le *binding*.