Compte-rendu

Binôme:

- Fornito Marvin

- Levy Damien

Lien GitHub: https://github.com/fornito2u/RepCo

Répartion du travail : 50/50

Marvin:

- Création et initialisation du projet (Packages Annexe / Main)
- Création des classes abstraites Joueur / EtatJeu
- Constructeur + Initialisation des classes Joueur / EtatJeu
- Méthode de comparaison d'état dans EtatReversi
- Prototype de la méthode des successeurs

(non-utilisé, Damien c'est occupé da la méthode complète fonctionnelle)

- Gestion du jeu joueur contre joueur (TP2 = Classe Reversi) + Gestion des erreurs d'entrées utilisateurs
- Méthode estUnEntier (Classe Reversi)
- Méthode eval0V2 qui est une seconde version d'eval0 qui effectue un calcul de force des positions du plateau afin de déterminer le poids.
- Méthode eval0V3 qui effectue un calcul de mobilité (nombre de coup possible depuis l'état actuel)
- Méthode minimax
- Méthode compareEval0
- Correction de la méthode évaluation (bug forçant l'ordinateur à toujours faire le même choix peu importe l'eval0 utilisé)
- Rédaction du rapport

Damien:

- Création des classes EtatReversi / JoueurReversi
- Constructeur EtatReversi / JoueurReversi
- Méthode d'initialisation du premier etat dans EtatReversi
- Méthode affichage de EtatReversi
- Méthode de gestion de la couleur du joueur dans JoueurReversi
- Méthode de gestion de la liste de joueur dans EtatJeu
- Méthode de gestion du tour dans EtatJeu
- Méthode de copie du plateau de jeu dans EtatReversi
- Main (gestion du jeu joueur contre machine)
- Gestion des successeurs : calcul et ajout des successeurs possible à la liste succ
- Méthode win() dans reversi
- Methodes pour compter et obtenir les jetons sur le plateau + Méthode pour compter jetons adverses
- Methode boolean pour connaître la couleur des joueurs
- eval0 (effectue un calcul de matérielle => combien de pièce ont chaque joueur)
- Élagage alpha beta
- Méthode max et min

Comment jouer:

- 1) Lancer l'application
- 2) Choisir le mode de jeu (Joueur contre joueur, joueur contre machine, machine contre machine) en entrant son id (Voir affichage)
- 3) Suivre les règles du mode de jeu spécifié au préalable.

Règle des différents modes de jeux :

Mode joueur contre joueur :

- Le plateau actuel est affiché en console. Les possibilités de jeu le sont également.
- Le joueur 1 choisi le numéro de plateau qui correspond au coup qu'il souhaite jouer.
- Le joueur 2 fait de même.
- Un affichage dans la console déterminera la fin de la partie et le score final.

Mode joueur contre machine :

- Le plateau actuel est affiché en console. Les possibilités de jeu le sont également.
- Le joueur 1 choisi le numéro de plateau qui correspond au coup qu'il souhaite jouer.
- La machine fait de même.
- Un affichage dans la console déterminera la fin de la partie et le score final.

Mode machine contre machine:

- Choisir la fonction d'évaluation utilisé pour la machine jouant les pions noirs (Voir l'affichage)
- Choisir la fonction d'evaluation utilisé pour la machine jouant les pions blancs (Voir l'affichage)
- Choisir le nombre de coups calculés à l'avance (Voir l'affichage)
- Les différents coups joués s'affichent en console. Les résultats sont affichés à la fin.

Difficultés rencontrées :

- Nous avons souvent travaillés sur les 2 mêmes classes en même temps : EtatReversi et le main Reversi.
- Cela à générer de nombreux conflits git lors des push/pull nous forçant à nous organiser davantage et à séparer le travail de façon spécifique
- Quelques bugs très coriaces et dure à localiser
- Le manque de temps (notamment dû aux autres projets sur lesquels nous travaillons en même temps)
- L'implémentation d'une IA basique était une première pour ma part (Marvin)

Perspectives d'amélioration :

- Une interface dynamique (système de bouton pour poser un pion, affichage du score en direct)
- Un « cheat code » activable pour afficher le meilleur coup possible au joueur.
- Un système de « timer » pour rajouter une nouvelle dimension au jeu.
- Un eval0 plus performant qui ne perd jamais!

Résultats de compareEval0 avec nos 3 fonctions : eval0, eval0V2 et eval0V3 :

```
- Profondeur 0:
               . eval0 vs eval0V2 => Egalité (renvoi 0)
               . eval0 vs eval0V3 => Egalité (renvoi 0)
               . evak0V2 vs eval0V3 => Egalité (renvoi 0)
- Profondeur 1:
               . eval0 vs eval0V2 => Egalité (renvoi 0)
               . eval0 vs eval0V3 => Egalité (renvoi 0)
               . evak0V2 vs eval0V3 => Victoire de eval0V3 (renvoi -1)
- Profondeur 2:
               . eval0 vs eval0V2 => Victoire de eval0 (renvoi 1)
               . eval0 vs eval0V3 => Egalité (renvoi 0)
               . evak0V2 vs eval0V3 => Victoire de eval0V3 (renvoi -1)
- Profondeur 3:
               . eval0 vs eval0V2 => Victoire de eval0V2 (renvoi -1)
               . eval0 vs eval0V3 => Victoire de eval03 (renvoi -1)
               . eval0V2 vs eval0V3 => Victoire de eval0V2 (renvoi 1)
```