Rapport Graphe Puissance 4

Groupe 1 : Bastien Didier, Gilles Labrousse, Thomas Palis

Objectif :

Réaliser un programme permettant de jouer contre une IA au jeu puissance 4.

Lancer le programme :

Pour exécuter le programme vous devez posséder Java est vous pouvez lancer le fichier .JAR avec la commande avec en argument le fichier d'entrée au format convenue en cours (Chaque tiret est une case et les joueurs sont représentés par des X et des O) :

```
java -jar <fichier.txt>
```

Les résultats du coup jouer sont écris dans un fichier game.txt

Base du programme :

Notre programme regarde les 3 meilleurs coup possible en suivant notre fonction d'évaluations puis regarde les 3 meilleurs coup possible du joueur d'après et ainsi de suite avec un niveaux de récursivité de trois tour des 2 joueurs. Vous trouverez l'algorithme utilisée ainsi que son implémentations en 5eme et 6eme partie.

Fonction d'évaluation :

Nous faisons la somme des jetons alignés dans chaque directions.Plus la somme est élevée plus la fonction d'évaluation renvoie un score élevée.De plus si la somme contient un 3 ou un 4 elle renverra un score plus élevée qu'une fonction contenant uniquement des 2 ....Cette fonction renvoie le même score en négatif pour simuler le joueur adverse. De plus si le joueur adverse est sur le point de gagner et que nous n'allons pas gagner ce tour-ci la priorité seras de le bloquer en le mettant a 100.

Difficulté rencontré :

Les deux principales difficultés on était de créer une fonction d'évaluations pour savoir quel coups était le meilleurs. Après plusieurs essaie et ajustement nous avons finalement trouvés une fonction d'évaluations que réponds a nos attente. L'autre difficultés a était l'implémentation de l'algorithme même si les principales idées et la créations de l'algorithme on était assez simple a trouvez l'implémentation de celle ci a était plus compliqué mais nous y sommes tout de même parvenue.

Algorithme Intelligence Artificielle

```
Fonction bestPlay :
si derniere branch et niveau de recursivité est sup a 6
  bestPlay(nextKey,0)
sinon
  si contient hashmap [branchkey] -15 -10 -9 -12 -11
    bestPlay(nextKey,0)
   sinon
    si contient hashmap[branchkey] 15 10 9 12 11
      playBest(branchkey,hashmap)
      return
     sinon
      si recursivitélevel%2==0
        si recursivité level=6
          playadv
          playbest nextbranchkey,recursivitélevel=0)
         sinon
          playadv
          playBeset(branchKey,recursitélevel+1)
        sinon
          si branchkey==null
            get 3 best possibilities
            for each possibilities
              put score into hashmap
              bestPlay(branchkey,recursivité level=2)
           sinon
            get 3 best possibilities
              score dans hashmap[branch] => premiere possibilité
              for each possibilities
                score dans hashmap[branch]possibilities
                bestplay(branchkey,recursivité level)
```

Fonction java :

```java
            int totdir1=1;
            int totdir2=1;
            int totdir3=1;
            int totdir4=1;
            int somme=0;
            int gau=evaluationDirection(plateau, coup, "G");
            int d=evaluationDirection(plateau, coup, "D");
            totdir1+=gau+d;
            result.put("G+D",totdir1);
            somme+=result.get("G+D");
            gau=evaluationDirection(plateau, coup, "GN");
            d=evaluationDirection(plateau, coup, "DS");
            totdir2+=gau+d;
            result.put("GS+DS",totdir2);
            somme+=totdir2;
```

Bastien Didier, Gilles Labrousse, Thomas Palis

```
            gau=evaluationDirection(plateau, coup, "DN");
            d=evaluationDirection(plateau, coup, "GS");
            totdir3+=gau+d;
            result.put("DN+GS",totdir3);
            somme+=result.get("DN+GS");
            gau=evaluationDirection(plateau, coup, "N");
            d=evaluationDirection(plateau, coup, "S");
            totdir4+=gau+d;
            result.put("NS", totdir4);
            somme+=totdir4;
            //System.out.println("somme : "+somme+" coup  :
"+Arrays.toString(coup));
            if(result.containsValue(4)) {
                  return 15;
            }else {
                  switch (somme) {
                  case 4:
                        return 0;
                  case 5:
                        return 1;
                  case 6:
                        if(!result.containsValue(3))
                              return 2;
                        else
                              return 5;
                  case 7:
                              if(result.containsValue(3))
                                    return 6;
                              else
                                    return 4;
                  case 8:
                        if(result.containsValue(3))
                              return 7;
                        else
                              return 4;
                  case 9:
                        if(result.containsValue(1)) {
                              return 9;
                        }else {
                              return 8;
                        }
                  case 10:
                        return 10;
                  case 11:
                        return 11;
                  case 12:
                        return 12;
                  default:
                        return 0;
```