Afzal Zeeshan
Ducobu Alexandre
M. Mélot
Faculté des Sciences, UMONS
16 mai 2014

Projet d'informatique

Boîte à jeux: Tic-Tac-Toe, Puissance 4 et Othello

Faculté des Sciences

UMONS Université de Mons

Répartition des tâches

Le travail a été réparti de la manière suivante: Zeeshan s'est occupé des intelligences artificielles du Puissance 4 ainsi que de Ant et Alexandre a réalisé le Tic-Tac-Toe, le Puissance 4 et Othello. Ensuite, nous avons travaillé ensemble sur les tests unitaires et sur les intelligences du Puissance 4 qu'il nous a fallu retravailler.

Points faibles

Lors d'une partie continue entre deux intelligences artificielles (IA), les différents coups ne sont pas visibles. En effet, il faut attendre la fin de la partie pour que la grille du jeu soit mise à jour et que l'on puisse voir quelles sont les cellules qui ont été jouées.

La répétition de certains bouts de code est un autre point faible de notre projet. Par exemple, pour la comparaison des IA, le code est sensiblement similaire à celui utilisé lors d'une partie. Nous avons eu le même souci pour l'initialisation de la grille du jeu que nous avons été obligés de dupliquer.

Description des choix

- Le jeu:

  Nous avons préféré commencer le jeu afin de pouvoir y jouer dans le terminal. Lorsque les jeux ont été finis (à part les IA), nous nous sommes mis à l'interface graphique. Avec l'aide de M. Absil, nous avons retravaillé notre code dans le but d'afficher le jeu dans une fenêtre. Nous avions le choix entre des boutons ou des panels, et nous avons choisi les boutons car il y avait plus d'informations dans le cours à leur propos.
  Entre chaque jeu, nous avons essayé de rester le plus cohérent possible et nous avons donc utilisé certaines parties du code pour chacun des trois jeux.
  Ensuite, viennent les tests unitaires que nous avons écrits pour vérifier les conditions de victoires de chaque jeux. Nous avons donc quatorze tests:
  - Tic-Tac-Toe:
    1. Égalité, 5 pions X et 4 pions O;
    2. Victoire de X, trois pions X alignés;
    3. Défaite de X, trois pions O alignés;
  - Puissance 4:
    4. Égalité, 21 pions X et 21 pions O;
    5. Victoire de X, trois pions X alignés;
    6. Défaite de X, trois pions O alignés;
  - Othello:
    7. Égalité lorsque la grille est complète, 32 pions X et 32 pions O;
    8. Victoire de X lorsque la grille est complète, 39 pions X et 25 pions O;
    9. Défaite de X lorsque la grille est complète, 17 pions X et 47 pions O;
    10. Victoire de X lorsque O n'a plus de pions;
    11. Défaite de X lorsque X n'a plus de pions;
    12. Égalité lorsque les deux joueurs sont bloqués, 24 pions X et 24 pions O;
    13. Victoire de X lorsque les deux joueurs sont bloqués, 35 pions X et 28 pions O;
    14. Défaite de X lorsque les deux joueurs sont bloqués, 1 pions X et 59 pions O;

  Puisque nous préférions les terminer avant la date limite, nous n'avons pas ajouté d'effets ou d'images aux différents éléments présents dans ce projet.

- Les IA:

  - Pour le Tic-Tac-Toe, l'intelligence faible joue au hasard tant qu'il n'y a pas de gagnant et qu'il y a des cellules libres. Nous pensons qu'ainsi, elle est simple et remplit correctement son travail. La forte prend le centre s'il est libre, ainsi, elle a la capacité d'aligner trois pions dans trois directions. Elle essaye aussi de jouer dans les coins: elle peut ainsi soit gagner en diagonale, soit horizontalement ou verticalement. Et, bien sûr, si deux pions sont alignés (par elle ou le concurrent), elle s'interpose. Ainsi, elle bloque la majorité des tentatives de l'autre joueur même si elle peut perdre contre une stratégie plus complexe.

  - Pour le puissance 4, la faible joue au hasard: elle choisit une colonne et joue dans la rangée libre la plus basse. Nous avons décidé de ne pas la laisser jouer tout le temps dans la même colonne pour corser le jeu et la rendre plus "humaine".
    La forte joue la deuxième cellule ou l'avant-dernière cellule de la grille si elles sont libres, ainsi elle peut empêcher l'autre joueur d'aligner quatre pions horizontalement sur la première ligne. Après, elle joue au hasard en vérifiant toujours de pouvoir jouer dans la colonne choisie. Dès que trois pions sont alignés horizontalement, verticalement ou en oblique, elle vérifie si elle peut aligner son pion soit pour gagner soit pour empêcher son opposant de gagner. Pour cela, elle vérifie, dans les cas horizontaux et obliques, si elle peut s'aligner aux trois autres tout en ayant un (des) pion(s) sous elle, sauf si elle se trouve dans la rangée du bas.

  - Pour Othello, la faible et la forte ont à peu près la même implémentation: elles jouent au hasard sur les cellules où il est possible de jouer.
    Pour rendre la forte plus forte, nous avons choisi de vérifier si les coins étaient jouables. Dans l'affirmative, l'IA prend le coin ce qui lui donne un avantage non négligeable pour gagner puisqu'elle contrôle alors deux côtés de la grille ainsi qu'une des diagonales principales. De plus, les coins sont les seules cellules qui, une fois prises par un joueur, ne peuvent être prises par l'autre. Prenons par exemple que le joueur Blanc, représenté par O, contrôle 60 cellules (aucune n'est un coin) et que le joueur Noir, représenté par X, contrôle un des 4 coins. O ne peut plus jouer mais X peut: chaque coin est une cellule disponible pour X. À chaque coup, X prend un côté de la grille ou une diagonale et O ne peut toujours pas jouer. Qui gagne ? C'est X avec 40 pions contre 24 pour O.

Apports positifs

Grâce à ce projet, nous connaissons bien la matière puisque nous en avons eu besoin tout au long de sa conception. De plus, nous devions retourner dans les slides lorsque nous avions besoin de quelque chose. Et, comme nous apprenions le langage en même temps, tout cela était bien frais dans notre tête.
Les TP nous ont aussi aidés pour le projet, car ils touchaient la matière que nous n'avions pas toujours vue au cours. Nous pensons ici aux packages, à la lecture et à l'écriture dans des fichiers et les interfaces graphiques.

Apports négatifs

Lorsque nous travaillions sur le projet, nous perdions du temps par rapport aux autres matières et, de manière identique, nous perdions du temps pour le projet quand nous travaillions sur les autres matières. Ce qui nous à causé quelques difficultés et contretemps.
Apprendre Java pendant que nous devions coder notre projet avait, comme dit plus tôt, des points positifs mais voici le négatif: nous ne connaissions pas toujours ce qui nous était nécessaire au moment où nous en avions besoin, ce qui nous freinait.

Sources

Puisque nous n'avons pas trouvé comment ne garder que les premières décimales d'un nombre dans le cours, nous avons trouvé le code ci-dessous sur ce site: http://www.infres.enst.fr/~hudry/coursJava/exercices/ecritDouble.html.

```
DecimalFormat f = new DecimalFormat();
f.setMaximumFractionDigits(2);
```

Pour les boîtes de dialogues et la barre de menu, nous nous sommes inspirés de ce qui était mis à notre disposition à travers le chapitre 15 du cours de Programmation et Algorithmique II.

Annexe

Voici le diagramme de classe UML du projet.