Laboratoire 2 – récursivité

Objectifs ASD

- Se familiariser avec les algorithmes récursifs.
- Etudier l'algorithme minimax

Objectifs C++

- Créer une classe implémentant une interface

Consignes

Le but du laboratoire est de vous familiariser avec la notion de récursivité en implémentant les algorithmes récursifs pour les jeux Tic Tac Toe et Puissance 4.

L'algorithme minimax est un algorithme qui s'applique à la théorie des jeux pour les jeux à deux joueurs à somme nulle consistant à minimiser la perte maximum (c'est-à-dire dans le pire des cas). Parmi les applications de cet algorithme on retrouve les jeux Tic Tac Toe et Puissance 4.

Tic Tac Toe

Pour implémenter le jeu Tic Tac Toe (X & O) vous avez à disposition un fichier .cpp qui fournit les fonctions de base et un programme principal permettant de le tester. Les variables globales et les fonctions de base sont décrites de manière détaillée et la plupart sont implémentées.

Votre tâche sera d'implémenter la fonction ai qui utilise une fonction score dont le pseudo-code figure dans votre support de cours. Pour tester le fonctionnement, le programme de test qui vous est fourni permet de choisir parmi trois manières de jouer :

0. aucun joueur humain (la machine contre la machine) : il y aura toujours égalité
1. la machine contre l'humain (choix aléatoire de celui qui commence) : égalité si l'humain joue bien
2. l'humain contre l'humain : le résultat du jeu dépend de la manière de jouer des humains

Le premier joueur (humain ou machine) sera toujours le X. Dans les cas où la machine joue contre la machine ou la machine joue contre l'humain, la fonction ai est appelée.

Dans un premier temps vous allez nous rendre la fonction ai et score du programme de Tic Tac Toe. Il faut qu'elle passe le test de CodeCheck où elle joue contre la nôtre. Elle doit forcer l'égalité si nous jouons bien, et nous battre si nous faisons une erreur.

Puissance 4

Par la suite, vous allez implémenter le jeu Puissance 4. Le but de ce jeu est d'aligner 4 pions sur une grille comptant 6 rangées et 7 colonnes. Chaque joueur dispose de 21 pions d'une couleur (par convention, en général jaune ou rouge). Tour à tour les deux joueurs placent un pion dans la colonne de leur choix, le pion coulisse alors jusqu'à la position la plus basse possible dans ladite colonne à la suite de quoi c'est à l'adversaire de jouer. Le vainqueur est le joueur qui réalise le premier un alignement (horizontal, vertical ou diagonal) d'au moins quatre pions de sa couleur. Si, alors que toutes les cases de la grille de jeu sont remplies, aucun des deux joueurs n'a réalisé un tel alignement, la partie est déclarée nulle.

Pour votre ordinateur, jouer à puissance 4 n'est pas fondamentalement différent de jouer à Tic Tac Toe. L'algorithme utilisé sera toujours minimax (ou negamax ou une variante). Il convient essentiellement de lui expliquer les coups jouables (7 colonnes au lieu de 9 cases) et la règle permettant de savoir si un joueur a gagné (4 alignés plutôt que 3).

Il y a pourtant une différence majeure. Il est ici exclus d'explorer les 4000 milliards de jeux possibles pour décider du prochain coup. Il faut donc se limiter à un nombre de coups donné et ajouter une condition d'arrêt supplémentaire à l'algorithme récursif : avoir atteint la profondeur d'exploration requise. Il faudra donc adapter la fonction score que vous avez écrite précédemment.

Vous mettrez en oeuvre ce jeu sous la forme d'une classe dont l'interface public vous est imposée, mais dont vous avez toute liberté de définir le contenu privé. Cette classe sera utilisée par un programme qui vous est également fourni et gère l'interaction avec 0, 1 ou 2 joueurs humains face à votre algorithme.

Vous devez soumettre uniquement les fichiers .cpp et .h définissant votre classe, proprement documentés, avec le moins de duplication de code possible, mais aussi le plus efficace possible en temps de calcul.

Contacts

- Prof. Olivier Cuisenaire : olivier.cuisenaire@heig-vd.ch
- Prof. Mireille Goud : mireille.goud@heig-vd.ch
- Prof. Laura Elena Raileanu : laura.raileanu@heig-vd.ch

- Dominique Jollien : dominique.jollien@heig-vd.ch
- Racine Raphaël : raphael.racine@heig-vd.ch