Année 2020–2021

Projet informatique (PI4) – L2

Jeux de pions

François Laroussinie
francoisl@irif.fr

Résumé : L'objectif de ce projet est de réaliser un programme permettant de jouer à des jeux de type "gomoku" ou "puissance 4".

1 Les règles du jeu

Le Gomoku se joue à deux joueurs A et B sur un plateau carré correspondant à un quadrillage de dimension 19×19. A chaque tour, l'un des joueurs pose un pion sur une case libre du plateau : Alternativement, A pose un pion `X` puis B pose un pion `O`. Le jeu s'arrête lorsqu'il n'y a plus de case libre (match nul) ou lorsqu'un des joueurs a réussi à aligner 5 pions horizontalement, verticalement ou diagonalement et ce joueur remporte alors la partie.

Ici nous allons faire un programme qui permet de choisir la taille du plateau, notée $N \times M$ dans la suite, et le nombre de pions à aligner, noté K. Enfin, on pourra aussi choisir de jouer façon Gomoku où il est possible de poser un pion sur n'importe quelle case vide ou façon Puissance 4 où les pions « tombent » et se placent sur la dernière case vide d'une colonne (NB : dans ce jeu, un coup est donc juste un numéro de colonne).

2 Algorithme

Le programme à réaliser devra permettre à un utilisateur de jouer contre l'ordinateur : l'utilisateur pourra indiquer la case (ou la colonne pour la version Puissance 4) sur laquelle il place un pion, puis l'ordinateur jouera à son tour, etc... jusqu'à la fin de la partie. Le programme contiendra donc, en plus d'un module d'initialisation, un module gérant l'interface avec l'utilisateur (saisie des coups et affichage du plateau) et un module pour choisir un coup en fonction d'une position donnée. Cette section a pour objet la présentation de l'algorithme de choix d'un coup par le programme. Cet algorithme est notamment utilisé dans le Gomoku du système GNU-EMACS.
Le principe de cet algorithme est de faire une évaluation statique de la position du jeu et de choisir un des coups ayant obtenu la meilleure évaluation.

On appelle *K-uplet* un ensemble de K cases alignées sur le plateau. L'idée est d'attribuer une évaluation (score) à chaque K-uplet. Ce score sera d'autant plus grand que ce K-uplet sera intéressant pour gagner la partie. Par exemple, supposons que le programme joue avec les pions `O`. Si un K-uplet contient quatre pions `O` et une case vide, alors il recevra une note très élevée : jouer sur la case vide pourrait permettre au programme de remporter la partie (car il obtiendrait un K-uplet gagnant). D'autre part un K-uplet comprenant quatre pions `X` et une case vide recevra une note élevée (car le programme a intérêt à jouer dans ce K-uplet, i.e. dans la case libre, pour empêcher l'utilisateur de gagner) mais moins que le K-uplet

précédent. En revanche un K-uplet contenant au moins un pion `X` et un pion `O` recevra la note 0 : jamais ce K-uplet ne pourra être gagnant ni pour l'utilisateur, ni pour le programme. On considèrera ainsi les différents cas possibles (un, deux, trois, ... ou $K-1$ `X`, un, deux, trois, ... ou $K-1$ `O`, au moins un `X` et un `O`, et le cas où le K-uplet est vide).

Une fois que tous les K-uplets ont été évalués, on évalue chaque case libre (i, j) du plateau en additionnant les notes de tous les K-uplets contenant la case (i, j). Ainsi une case recevra une note d'autant plus élevée qu'elle appartient à des K-uplets intéressants. Finalement il reste à choisir aléatoirement une case libre parmi celles ayant obtenu la plus forte évaluation. Le reste du document présente la structure de données à utiliser et certains choix d'implémentation à faire pour la construction du programme.

3 Travail à réaliser

Pour représenter le lien entre les K-uplet et les cases et pour stocker leurs évaluations, on utilisera les deux classes suivantes :

- Les K-uplets seront représentés par une classe `K_uplet` contenant (entre autres) un tableau des K cases contenues dans un K-uplet et un score correspondant à l'évaluation de ce K-uplet.
- Les cases seront représentées par une classe `Case` contenant (entre autres) son état (vide, X ou O), la liste des K-uplets contentant la case et le score de la case.

On utilisera aussi une classe `Jeu` contenant un tableau de cases, la liste des K-uplets du plateau et le trait (à qui est-ce le tour de jouer ?).

Il y a bien sûr à gérer l'affichage et l'interaction avec l'utilisateur (NB : aucune interface graphique n'est exigée).

4 Compléments

Une fois que le programme initial marchera, on pourra considérer une ou des extensions parmi :

- un algorithme de type min-max (ou $\alpha-\beta$) pour améliorer les coups ;
- sauvegarder des parties ;
- permettre le retour en arrière ;
- ajouter une interface graphique ;
- jouer sur un cylindre ou une sphère ;
- ...