L3-POO2 2013
Sujet de projet

IA pour Tetris

Travail demandé

Le travail demandé comporte une implantation en Java (ou autres langages objet). Un rapport court (de 2 à 5 pages) devra accompagner le projet en format papier. Ce rapport devra
- expliquer vos choix de modélisation et lever les ambiguïtés du sujet
- présenter la répartition du travail
- expliquer certains algorithmes
- donner un diagramme d'héritages

Implantation
- Votre programme devra fonctionner sur la machine *turing*.
- Une petite explication de l'utilisation de votre programme est aussi nécessaire.

Remise du projet

La remise du projet se fera la semaine du 16 Décembre par mail à votre responsable de TP, à savoir : Sophie Lèbre.

Une courte soutenance sera organisée durant cette semaine par Sophie Lèbre et votre chargé de cours Benoît Sonntag.

Réalisation du projet

- La réalisation de ce projet devra se faire impérativement **par groupe de deux ou trois** pour que vous vous répartissiez le travail au sein du groupe.
- Comme tout cahier des charges, celui-ci ne peut être exhaustif. En cas d'ambiguïté, préciser votre interprétation personnelle, et éventuellement les questions à poser à votre interlocuteur (responsable de projet, futurs utilisateurs, etc.). Toute solution cohérente, justifiée et non contradictoire avec le cahier des charges sera acceptée.

Sujet

L'objectif est de concevoir une IA pour Tetris. Nous ne présenterons pas ce jeu que nous connaissons tous. Mais, pour éviter toutes possibilités de lien direct entre le code du jeu et l'IA, nous allons les séparer physiquement. L'interface entre les deux applications vous sera fourni en Java.

La réalisation est en deux programmes : La première partie représente un jeu de Tetris standard, où nous allons définir quelques régles...
- Pas de visibilité de la prochaine pièce.
- Dimension 10x20.
- La rotation est dans le sens des aiguilles d'une montre.
- Faire une seule ligne ne rapporte que 40 points, alors qu'en faire 2 en rapporte 100, 3 lignes rapportent 300 et 4 lignes (le maximum) en rapportent 1200.

– Flêche droite pour aller à droite, flêche gauche pour aller à gauche, flêche haut pour une rotation, flêche bas pour faire chutter la piece.

La seconde partie représente l'IA qui n'aura rien de plus comme information que celle d'un être humain. A savoir, vous avez la possibilité d'observer à n'importe quel moment l'état du jeu sous forme matricielle, et la possibilité de simuler les 4 touches de navigation.

L'interfaçage founi fonctionne de la manière suivante : La class *tetris* prend le rôle d'un serveur au démarrage pour répondre à n'importe quel moment à une demande de l'état du jeu. Puis, son exécution se poursuit pour jouer au clavier. L'application est donc utilisable sans la partie IA.

La partie IA commence par exécuter le *tetris*, puis devient client de sa partie serveur. Ensuite, elle utilise un robot pour simuler l'action des 4 touches.

L'exemple qui vous est fourni est complétement fonctionnel. Mais, il ne s'agit pas d'un tetris. Les commentaires *"For example"* vous donnent les endroits qui ne concernent pas le tetris. Néanmoins, cet exemple vous donne comment utiliser le reste du code. Vous pouvez modifier ce code, à vous d'en faire bon usage...

Astuces

Pour calculer l'ensemble des pièces et de leurs rotations, vous pouvez utiliser ce code de mon cru. Il remplit un tableau *face* à 3 dimensions : rotation entre 0 et 3, numéro du carré entre 0 et 3 (0 :axe de rotation), coordonnée entre 0 et 1 (0 :x, 1 :y).

```
- load_face p:INTEGER <-

//  0 |  1  |  2  |  3  |  4  |  5  |  6  |
//----|-----|-----|-----|-----|-----|-----|
//  1 |     |     |     |     |     |     |
//  2 | 3 1 | 2   | 3   |   2 |     | 2   |
//  X | X   | X 1 | X   | X 1 | X 1 | X 1 |
//  3 | 2   | 3   | 2 1 | 3   | 2 3 |   3 |


( + tmp:BOOLEAN;
  + tmp1,tmp2:INTEGER;

  // First graph.
  face.put ((p!=0).to_integer) to (0,1,0);
  face.put ((p-2)*(p<4).to_integer) to (0,1,1);
  tmp1:=((p&1)<<1)-1;
  face.put ((p=4).to_integer) to (0,2,0);
  face.put tmp1               to (0,2,1);
  tmp:=(p>4);
  tmp2:=tmp.to_integer;
  face.put tmp2                           to (0,3,0);
  face.put (-tmp1*(!tmp).to_integer+tmp2) to (0,3,1);

  // 3 rotation (+pi/2).
  1.to 3 do { r:INTEGER;
    0.to 3 do { c:INTEGER;
      tmp1 := r - 1;
      face.put (-face.item (tmp1,c,1)) to (r,c,0);  // x' = -y
      face.put (+face.item (tmp1,c,0)) to (r,c,1);  // y' = +x
    };
  };
);
```

Good luck !