TétrisBot

Frédéric Muller - Lionel Ponton

Pojet maths-infos du DU 2ème année - 2017-2018

INTRODUCTION

TABLE DES MATIÈRES

Première partie

Le moteur de jeu

Chapitre 1

LE JEU TÉTRIS

1 Histoire

2 Règles du jeu adaptées au projet

Chapitre 2

IMPLÉMENTATION DU MOTEUR DE JEU

1 Structures de données

Le moteur de jeu est construit autour de trois classes :

- `Tetramino` : les blocs
- `Board` : la grille de jeu
- `TetrisEngine` : le moteur de jeu qui fait le lien entre les deux classes précédentes

Notons que toutes ces classes implémentent une méthode `copy(self)` qui envoie une copie de l'objet en utilisant le module `deepcopy`.

2 La classe Tetramino

La classe `Tetramino` est responsable de le gestion des blocs et de leurs rotations. Elle est implémentée dans le fichier `tetramino.py`.
Un bloc est défini par :

- Un id, `self.id`, qui permet d'identifier son type.
- Un glyphe de base, `self.base_glyph`, qui représente la pièce sans rotation dans une matrice carrée. Chaque case occupée par le bloc est codée par son `id` et les cases vides par 0.
- Le nombre de rotations possibles de la pièce, `self.nb_rotations` (par exemple le O n'a qu'une seul rotation, le I en a deux et le T en a quatre).

Les rotations sont crées par la méthode `makeRotations(self)` au moment de la construction de la pièce et sont stockées dans une liste de glyphs, `self.rotations`.
Pour gérer les rotations, on utilise un attribut `self.glyph_index` qui donne l'indice de la rotation courante, ainsi que le glyph courant, `self.glyph`.
Enfin Deux méthodes permettent de faire tourner la pièce :

- `rotate(self, direction='H')` : met à jour le glyph avec celui de sa rotation dans le sens donné (`'H'` pour le sens horaire et `'T'` pour le sens trigonométrique).
- `setRotation(self, i)` : tourne directement la pièce dans la rotation d'indice `i`.

Notons également la méthode `getBoundingBox(self)` qui renvoie les coordonnées des coins des cases de la pièce réellement utilisées.
Cette classe admet enfin quelques méthodes utiles :

- `copy(self)` : renvoie une copie du bloc
- `__str__(self)` : renvoie une chaîne de caractères pour afficher la pièce à des fin de test uniquement ici.

Dans ce fichier on définit également les pièces qui vont être utilisées, ainsi que la liste de toutes ces pièces, `BLOCK_BAG`.

3 La classe Board

La classe `Board` implémente la grille, ses méthodes de gestion (mise à jour des cellules, traitement des lignes,...) ainsi que les outils statistiques (nombre de trous, hauteur maximum,...). Elle est implémentée dans le fichier `board.py`.

3.1 Constructeur

Le constructeur de la classe `board` admet deux paramètres :

- Sa largeur : `width`
- Sa hauteur : `height`

La grille en elle-même est stockée dans la liste double `self.grid` qui a pour largeur `width` et pour hauteur `height+2` (avec les deux lignes invisibles du dessus).
Enfin, les différents indicateurs statistiques sont initialisés.

3.2 Gestion des cellules

La gestion des cellules de la grille sont gérées par des getters et setters qui présentent peu d'intérêt et dont les noms sont explicites.
Notons toutefois les méthodes suivantes qui seront utilisées lors de la suppression des lignes pleines :

- `isLineFull(self, i)` qui teste si la ligne `i` est vide
- `removeLine(self, i)` qui supprime la ligne `i` de la grille et rajoute une ligne vide en haut.

3.3 Récupération des caractéristiques de la grille

Les méthodes suivantes permettent de récupérer, dans des attributs spécifiques les différentes caractéristiques de la grille :

- `columnHeight(self, j)` : renvoie la hauteur de la colonne `j`. Le principe de l'algorithme est de partir du haut de la grille et de décrémenter cette hauteur tant que la cellule visitée est vide :

```
i = hauteur de la grille
Tant que i>=0 et la cellule (i,j) est vide :
 i = i-1
Renvoyer i+1
```

  Notons que cette fonction renvoie bien la hauteur de la colonne et non l'indice de la ligne de la case la plus haute.
- `getColumnHeights(self)` : renvoie la liste des hauteurs des colonnes.
- `getMaxHeight(self)` et `getSumHeights(self)` : renvoient respectivement la hauteur maximum et la somme des hauteurs des colonnes.

- `getBumpiness(self)` : renvoie la somme des valeurs absolues des différences de hauteur entre les colonnes consécutives.
- Pour la détermination du nombre de trous, on procède de la façon suivante : un trou est défini comme une cellule vide dans une colonne qui contient un cellule pleine au-dessus (cellule dominée).
  - `isDominated(self, i, j)` : teste si la cellule `(i,j)` est dominée.

```
Si la cellule (i,j) n'est pas vide :
 Renvoyer Faux
Sinon :
 Pour k allant de i+1 à (hauteur de la colonne j) - 1 :
   Si la cellule (i,j) n'est pas vide :
     Renvoyer Vrai
 Renvoyer Faux
```

  - `getNbHoles(self)` : renvoie le nombre de trous en comptant pour chaque cellule si elle est dominée.
- `updateStats(self)` : met à jour toutes les caractéristiques de la grille

3.4 Gestion des lignes

La méthode `processLines(self)` supprime les lignes pleines et renvoie le nombre de lignes supprimées :

```
nb_lignes = 0
hauteur_max = hauteur de la grille
i = 0
Tant que i <= hauteur_max :
 Si la ligne i est pleine :
   Enlever la ligne i
   hauteur_max = hauteur_max-1
   nb_lignes = nb_lignes +1
 Sinon :
   i = i + 1
Renvoyer nb_lignes
```

3.5 Méthodes utilitaires

- `copy(self)` : renvoie une copie de la grille.
- `__str__(self)` : renvoie une chaîne de caractère pour affichage de la grille.
- `printInfos(self)` : affiche les caractéristiques de la grille (pour les tests).

4 La classe TetrisEngine

Cette classe fait le lien entre les deux précédentes et implémente le déroulement de la partie. Elle est implémentée dans le fichier `tetris_engine.py`.

4.1 Constructeur et attributs

Les paramètres du constructeur sont les suivants :

- `width` et `height` : les dimensions de la grille
- `max_blocks` : le nombre maximum de blocs à jouer pour limiter les temps des essais (`0` pour jouer jusqu'à la fin de la partie)
- `temporisation` : en secondes, un temps de pause entre deux mouvements
- `silent` : si `True`, ne produit aucun affichage (pour les essais sur plusieurs parties)
- `getMove` : c'est la fonction qui va renvoyer, à chaque tour, le coup à jouer. C'est cette fonction que vont implémenter les agents (fonction de callback).

Le constructeur va également définir les attributs suivants :

- Gestion de la grille :
  - `self.board` : la grille de jeu dans laquelle les pièces évoluent.
  - `self.fixed_board` : la grille de jeu statique dans laquelle les pièces sont placées. Cette grille est une copie de `self.board` à la fin de chaque coup.
- Gestion des blocs :
  - `self.block_bag` : la sac contenant les pièces suivantes
  - `self.block` : le bloc courant en jeu
  - `self.block_position` : la position du coin en haut à gauche du bloc courant
  - `self.next_block` : le bloc suivant
  - `self.nb_blocks_played` : le nombre de pièces déjà jouées

Chapitre 3

LES AGENTS

Deuxième partie

Optimisation par algorithmes génétiques

Troisième partie

Optimisation par reinforcement learning