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.

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 de `0` en haut.

4 La classe TetrisEngine

Chapitre 3

LES AGENTS

Deuxième partie

Optimisation par algorithmes génétiques

Troisième partie

Optimisation par reinforcement learning