
TetrisBot Documentation

Version 1.0

F.Muller - L.Ponton

mai 31, 2019

Table des matières

1	Le moteur de jeu	1
1.1	Module tetramino.py	1
1.2	Module board.py	1
1.3	Module tetris_engine.py	2
2	Les agents	5
2.1	Module agent.py	5
2.2	Module agent_human.py	5
2.3	Module agent_random1.py	6
2.4	Module agent_random2.py	6
2.5	Module agent_filtering.py	6
2.6	Module agent_evaluation.py	6
3	Algorithmes génétiques	9
3.1	Module ag_optimizer.py	9
4	Reinforcement learning	11
4.1	Module tetris_RLenv.py	11
4.2	Module qRL_optimizer.py	12
5	Divers	13
5.1	Module stats.py	13
5.2	Module textutil.py	13
	Index des modules Python	15
	Index	17

1.1 Module tetramino.py

```
class tetramino.Tetramino (id, rotations, corners)
    Classe de gestion des pièces
    copy ()
        Renvoie une copie de la pièce
    getBottomCells ()
        Renvoie les coordonnées des cellules les plus en bas
    getCorners ()
        Renvoie les coordonnées des coins de la pièce
    getLowerCell (j)
        Renvoie la ligne de la cellule la plus en bas dans la colonne j
    rotate (direction='H')
        Tourne la pièce dans la direction donnée - "H" : Sens Horaire - "T" : Sens Trigo
    setRotation (i)
        Tourne directement une pièce
    toArray ()
        Renvoie la représentation du bloc sous forme de matrice carrée
```

1.2 Module board.py

```
class board.Board (width=10, height=22)
    Classe de gestion de la grille
    columnHeight (j)
        Renvoie la hauteur de la colonne j Attention, cette fonction renvoie la hauteur et non l'indice de la dernière
        pièce
    copy ()
        Renvoie une copie de la grille
```

decodeFromInt (*n*)
Renvoie un tableau à partir d'un code entier

emptyCell (*i, j*)
Vide la cellule (*i, j*)

encodeToInt ()
Renvoie une représentation de la grille sous la forme d'un entier

getBumpiness ()
Renvoie la somme des valeurs absolues des différences de hauteurs entre les colonnes consécutives

getCell (*i, j*)
Renvoie le contenu de la cellule (*i, j*)

getColumnHeights ()
Renvoie la liste des hauteurs des colonnes

getMaxHeight ()
Renvoie la hauteur maximum des pièces du jeu

getNbHoles ()
Renvoie le nombre de trous dans la grille (en fait ici juste les cases dominées)

getSumHeights ()
Renvoie la somme des hauteurs des colonnes

isCellEmpty (*i, j*)
Teste si la cellule (*i, j*) est vide

isDominated (*i, j*)
Teste si une case est vide et est dominée par une case au-dessus

isLineFull (*i*)
Teste si une ligne est pleine

npBinaryRepresentation ()
Renvoie un numpy array contenant la grille

printInfos ()
Affiche les infos de la grille (pour tests)

processLines ()
Enlève les lignes finies et renvoie le nombre de lignes enlevées

removeLine (*i*)
Supprime la ligne *i*

setCell (*i, j, value*)
Met *value* dans la cellule (*i, j*)

updateStats ()
Met à jour tous les paramètres de la grille

1.3 Module tetris_engine.py

```
class tetris_engine.TetrisEngine(getMove=<function <lambda>>, width=10, height=22,  
                                max_blocks=0, base_blocks_bag=[<tetramino.Tetramino  
                                instance>, <tetramino.Tetramino instance>, <tetra-  
                                mino.Tetramino instance>, <tetramino.Tetramino instance>, <tetra-  
                                mino.Tetramino instance>, <tetramino.Tetramino  
                                instance>, <tetramino.Tetramino instance>], tempori-  
                                sation=0, silent=False, random_generator_seed=None,  
                                agent_name="", agent_description="")
```

Classe gérant le moteur de jeu

canPlaceBlockDirect (*column, rotation*)

Teste si on peut placer directement un bloc dans la colonne et la rotation donnée

copy ()
Renvoie une copie de l'environnement

dropBlock ()
Fait tomber le bloc en bas

eraseBlock ()
Efface le bloc de sa position

generateNewBlock ()
Remplace le bloc courant par le suivant et fabrique un nouveau bloc suivant

generateNewBlockBag ()
Génère un nouveau sac de pièces

getNewBlock ()
Met un nouveau bloc en jeu

getPossibleMovesDirect ()
Renvoie la liste de tous les placements directs possibles sous la forme de tuples (column, rotation)

getScoreFromLines (nb_lines)
Renvoie le score suivant le nombre de lignes faites

getStrAgentName ()
Renvoie la chaîne contenant les nom et la description de l'agent

getStrInfos ()
Renvoie une chaîne contenant les informations de la partie

getStrNextBlock ()
Renvoie une chaîne pour afficher le bloc suivant

isEndGame ()
Teste la fin du jeu

isMoveInGrid (block, new_position)
Teste si le bloc reste dans la grille

isMoveOnFreeCells (block, new_position)
Teste si les nouvelles cases occupées par le bloc sont vides

isMoveValid (block, new_position)
Teste si une position est valide pour un bloc

minimalCopy ()
Renvoie une copie minimale de l'environnement (grilles et pièces) pour tester les différents coups

moveBlock (block, new_position)
Déplace un bloc vers une nouvelle position

moveBlockInDirection (direction=)
Déplace le bloc dans une direction : - "L" : Left - "R" : Right - "" : Vers le bas par défaut

placeBlock (block, position)
Place un bloc dans une position

placeBlockDirect (column, rotation)
Place directement un bloc dans la colonne et la rotation donnée

playCommand (command='N')
Joue une commande : - "L" : Move Left - "R" : Move Right - "D" : Drop - "N" : Nothing (descend d'une case) - "H" : Rotate Hours direction - "T" : Rotate Trigo direction - "P ;j :r" : Place block in column j with rotation r - "S" : Restart - "Q" : Quit

printRightColumn ()
Renvoie la chaîne correspondant au côté droit de l'affichage

rotateBlockInDirection (direction='H')
Tourne le bloc dans une direction : - "H" : Sens Horaire - "T" : Sens Trigo

run ()
Boucle principale du jeu

setBlockInitPosition()

Position initiale pour un nouveau bloc

updateTimes(*start_time*)

Met à jour les chronos

`tetris_engine.getrandbits(k)` \rightarrow x. Generates a long int with k random bits.

`tetris_engine.random()` \rightarrow x in the interval [0, 1).

2.1 Module agent.py

```
class agent.Agent (name="", description="")  
    La classe de base des agents  
    allMovesStats ()  
        Renvoie un dictionnaire contenant les stats de chaque mouvement possible Les clefs sont les mouvements  
        et les valeurs sont les stats de ce mouvement  
    commandFromMove (move)  
        Renvoie la commande d'un mouvement à passer à l'engine  
    getMoveStats (move)  
        Remplit le dictionnaire contenant les statistiques de la grille après que le mouvement move ait été joué  
  
agent.benchPlayer (player_init, nb_samples=100, max_blocks=0)  
    Réalise un bench de AgentToTest en jouant nb_samples parties  
  
agent.getrandbits (k) → x. Generates a long int with k random bits.  
  
agent.playGame (player_init, temporisation=0.1)  
    Lance des parties avec l'agent  
  
agent.plotBenchPlayer (player_init, nb_samples, nbBars=10, max_blocks=0)  
    Réalise un bench de AgentToTest en jouant nb_samples parties Affiche les résultats sous la forme d'un histo-  
gramme avec nbBars classes  
  
agent.random () → x in the interval [0, 1).
```

2.2 Module agent_human.py

```
class agent_human.AgentHuman (temporisation=0, silent=False)  
    Agent humain  
    getMove ()  
        Entre un mouvement à jouer
```

`agent_human.getrandbits(k)` → x. Generates a long int with k random bits.

`agent_human.random()` → x in the interval [0, 1).

2.3 Module agent_random1.py

class `agent_random1.AgentRandom1` (*temporisation=0.1, silent=False*)

Agent aléatoire jouant avec les touches du clavier

getMove ()

Renvoie un mouvement de touche aléatoire

`agent_random1.getrandbits(k)` → x. Generates a long int with k random bits.

`agent_random1.random()` → x in the interval [0, 1).

2.4 Module agent_random2.py

class `agent_random2.AgentRandom2` (*temporisation=0.1, silent=False*)

Agent aléatoire jouant directement les pièces

getMove ()

Renvoie un mouvement direct aléatoire

`agent_random2.getrandbits(k)` → x. Generates a long int with k random bits.

`agent_random2.random()` → x in the interval [0, 1).

2.5 Module agent_filtering.py

class `agent_filtering.AgentFiltering` (*temporisation=0.1, silent=False, order=['holes', 'sum_heights', 'bumpiness', 'lines']*)

Agent procédant par filtrage des coups

filterMoves (*stat, value*)

Filtre les mouvements en récupérant uniquement ceux dont la stat vaut value

getMove ()

Optimisation en filtrant successivement les mouvements suivant les différentes stats

maxStat (*stat*)

Renvoie la valeur maxi d'une stat

minStat (*stat*)

Renvoie la valeur mini d'une stat

`agent_filtering.getrandbits(k)` → x. Generates a long int with k random bits.

`agent_filtering.random()` → x in the interval [0, 1).

2.6 Module agent_evaluation.py

class `agent_evaluation.AgentEvaluation` (*eval_coeffs=[0.548, 0.5218, 0.6267, 0.1862], temporisation=0.1, silent=False*)

Agent procédant par évaluation des coups

getMove()

Optimisation à partir de la fonction d'évaluation

moveEvaluation (*move*)

Évalue le mouvement *move*=(j, r)

`agent_evaluation.getrandbits(k)` → x. Generates a long int with k random bits.

`agent_evaluation.playGameWithAgentEvaluation(coeffs, temporisation=0)`

Joue des parties avec l'agent par évaluation et les coeffs donnés

`agent_evaluation.random()` → x in the interval [0, 1).

3.1 Module `ag_optimizer.py`

```
class ag_optimizer.AGOptimizer (population_size=20, nb_generations=2, nb_bits=16,  
                                max_nb_blocks=5, nb_games_played=1, proba_mutation=0.05,  
                                mutation_rate=0.2, percentage_for_tournament=0.1, per-  
                                centage_new_offspring=0.3, elitism_percentage=0.2, vec-  
                                tor_encoding='float', parents_selection_method='tournament',  
                                old_generation_policy='best', evaluation_criteria='lines')
```

Optimisation des coefficients par algorithme génétique

crossover (*parent1, parent2*)

Renvoie le ou les enfants de parent1 et parent2

deleteWorst ()

Enlève les moins bons éléments de la population

fitness (*vector*)

Fitness de l'individu : score total sur nb_games_played parties

generateNewOffspring ()

Renvoie la nouvelle génération

initPopulation ()

Initialise la population

keepOnlyElite ()

Garde les meilleurs éléments de la génération précédente

makeNewGeneration ()

Crée une nouvelle génération

mutate (*individu*)

Mute un individu

mutateBinVector (*bin_vector*)

Mute un vecteur

mutateFloatVector (*vector*)

Mute un individu (son vecteur)

plotStats ()
Courbes de statistiques

process ()
Boucle principale de l'optimisation

randomBinaryList ()
Renvoie une liste de self.nb_bits chiffres binaires aléatoires

randomBinaryVector ()
Renvoie un vecteur binaire aléatoire

randomFloatVector ()
Renvoie un vecteur aléatoire normé

scoreOnOneGame (vector)
Score sur une partie

sortPopulationDescending ()
Trie la population par ordre décroissant de scores

stringOfParameters ()
Renvoie la chaîne des paramètres de l'algorithme génétique

tournamentSelection ()
Sélection par tournoi

updateBinaryIndividu (individu)
Met à jour les paramètres d'un individu à partir de son vecteur binaire

updateScore (individu)
Met à jour le score de l'individu

updateStats ()
Met à jours les statistiques

wheelSelection ()
Sélection d'un individu avec une roulette

ag_optimizer.binToFloat (bits)
Renvoie la représentation entre 0 et 1 d'une liste binaire Le chiffre des unités étant considéré comme le 1er élément de la liste (ça n'a aucune importance vu qu'on va partir de listes aléatoires)

ag_optimizer.binVectorToFloat (bin_vector)
Convertit un vecteur de listes binaires en vecteur de float

ag_optimizer.getrandbits (k) → x. Generates a long int with k random bits.

ag_optimizer.linearCombination (a1, vector1, a2, vector2)
Renvoie la combinaison linéaire de deux vecteurs

ag_optimizer.normalize (vector)
Normalise un vecteur

ag_optimizer.random () → x in the interval [0, 1).

4.1 Module tetris_RLenv.py

```
class tetris_RLenv.TetrisEnv (width=10,          height=22,          max_blocks=0,
                             base_blocks_bag=[<tetramino.Tetramino instance>, <tetra-
                             mino.Tetramino instance>, <tetramino.Tetramino instance>,
                             <tetramino.Tetramino instance>, <tetramino.Tetramino ins-
                             tance>, <tetramino.Tetramino instance>, <tetramino.Tetramino
                             instance>], random_generator_seed=None, agent_name="",
                             agent_description="")
```

Environnement à la OpenAI Gym pour implémenter le reinforcement learning

getState ()

Renvoie l'état de la grille

getStateCode ()

Renvoie de code entier de l'état de la grille

render ()

Affiche le jeu

reset ()

Réinitialise l'environnement

sampleAction ()

Renvoie une action aléatoire

step (action)

Effectue une action (joue un coup) Met à jour les informations (done, state)

`tetris_RLenv.getrandbits (k)` → x. Generates a long int with k random bits.

`tetris_RLenv.random ()` → x in the interval [0, 1).

4.2 Module qRL_optimizer.py

```
class qRL_optimizer.QRLOptimizer (width=5, height=5, base_blocks_bag=[<tetramino.Tetramino
                                instance>], max_episodes=2000, max_blocks=500, al-
                                pha=0.1, gamma=0.9, epsilon_min=0.01, epsilon_max=1,
                                epsilon_delta=0.001)
    Optimisation par Q-learning sur une configuration simple
getQIndexes ()
    Renvoie un tableau dans lequel chaque cellule est le nombre de fois qu'elle apparaît dans les indices de la
    Q-Table avec des Q-Values toutes non nulles
initQValue (s)
    Initialise la Q-value de l'état s avec des 0 si s n'est pas encore dans la table
learn ()
    Lance l'apprentissage
play ()
    Joue la partie avec la Q-table créée
printQIndexes ()
    Affiche le nombre de fois que chaque cellule apparaît dans les indices de la Q-Table en nuances de gris
reinit ()
    Réinitialise l'environnement
update (s, a)
    Met à jour la Q-table de l'état s sur une action a

qRL_optimizer.argmax (liste)
    Renvoie l'indice de la valeur max de liste ou un indice aléatoire si la liste ne contient que des 0

qRL_optimizer.getrandbits (k) → x. Generates a long int with k random bits.

qRL_optimizer.random () → x in the interval [0, 1).

qRL_optimizer.total_size (o)
    Renvoie la taille totale d'un objet en mémoire Code récupéré sur : https://code.activestate.com/recipes/577504/
    et adapté au projet
```


5.1 Module stats.py

```
class stats.Stats (data=None, filename="", mean_time=0, nbBars=10, title="")
```

Représentation statistique des parties

```
getAllStats ()
```

Récupère tous les indicateurs statistiques

```
getEffectif ()
```

Renvoie le nombre de données

```
getMaxi ()
```

Renvoie le maximum des données

```
getMean ()
```

Renvoie la moyenne des données

```
getMini ()
```

Renvoie le minimum des données

```
getQuartiles ()
```

Renvoie un tuple (Q1, Médiane, Q3)

```
getSigma ()
```

Renvoie l'écart-type des données

```
histogram (save=True)
```

Crée et affiche l'histogramme

```
loadData ()
```

Charge les données à partir d'un fichier texte

```
saveData ()
```

Sauvegarde les données dans un fichier texte

5.2 Module textutil.py

```
textutil.boxed (text, prefix="", window_width=0, window_height=0)
```

Affiche chaque ligne de texte précédée d'un préfixe dans une boîte de largeur window_width

`textutil.center (string, length)`

Centre la chaîne string sur la longueur

`textutil.cleanLine (string)`

Renvoie la chaîne sans les caractères spéciaux de couleur

`textutil.dateNow ()`

Renvoie une chaîne avec la date et l'heure courante

`textutil.mergeChains (string1, string2)`

Fusionne deux chaînes côte à côte pour l'affichage

`textutil.textColor (string, bg=15, fg=232)`

Renvoie une chaîne contenant le texte coloré avec bg pour la couleur de fond et fg pour la couleur du texte.
Utilise les codes ANSI/VT100.

a

ag_optimizer, 9
agent, 5
agent_evaluation, 6
agent_filtering, 6
agent_human, 5
agent_random1, 6
agent_random2, 6

b

board, 1

q

qRL_optimizer, 12

s

stats, 13

t

tetramino, 1
tetris_engine, 2
tetris_RLenv, 11
textutil, 13

A

ag_optimizer (module), 9
 Agent (classe dans agent), 5
 agent (module), 5
 agent_evaluation (module), 6
 agent_filtering (module), 6
 agent_human (module), 5
 agent_random1 (module), 6
 agent_random2 (module), 6
 AgentEvaluation (classe dans agent_evaluation), 6
 AgentFiltering (classe dans agent_filtering), 6
 AgentHuman (classe dans agent_human), 5
 AgentRandom1 (classe dans agent_random1), 6
 AgentRandom2 (classe dans agent_random2), 6
 AGOptimizer (classe dans ag_optimizer), 9
 allMovesStats() (méthode agent.Agent), 5
 argmax() (dans le module qRL_optimizer), 12

B

benchPlayer() (dans le module agent), 5
 binToFloat() (dans le module ag_optimizer), 10
 binVectorToFloat() (dans le module ag_optimizer), 10
 Board (classe dans board), 1
 board (module), 1
 boxed() (dans le module textutil), 13

C

canPlaceBlockDirect() (méthode
 tris_engine.TetrisEngine), 2
 center() (dans le module textutil), 14
 cleanLine() (dans le module textutil), 14
 columnHeight() (méthode board.Board), 1
 commandFromMove() (méthode agent.Agent), 5
 copy() (méthode board.Board), 1
 copy() (méthode tetramino.Tetramino), 1
 copy() (méthode tetris_engine.TetrisEngine), 2
 crossover() (méthode ag_optimizer.AGOptimizer), 9

D

dateNow() (dans le module textutil), 14

decodeFromInt() (méthode board.Board), 1
 deleteWorst() (méthode ag_optimizer.AGOptimizer), 9
 dropBlock() (méthode tetris_engine.TetrisEngine), 3

E

emptyCell() (méthode board.Board), 2
 encodeToInt() (méthode board.Board), 2
 eraseBlock() (méthode tetris_engine.TetrisEngine), 3

F

filterMoves() (méthode agent_filtering.AgentFiltering), 6
 fitness() (méthode ag_optimizer.AGOptimizer), 9

G

generateNewBlock() (méthode
 tris_engine.TetrisEngine), 3
 generateNewBlockBag() (méthode
 tris_engine.TetrisEngine), 3
 generateNewOffspring() (méthode
 ag_optimizer.AGOptimizer), 9
 getAllStats() (méthode stats.Stats), 13
 getBottomCells() (méthode tetramino.Tetramino), 1
 getBumpiness() (méthode board.Board), 2
 getCell() (méthode board.Board), 2
 getColumnHeights() (méthode board.Board), 2
 getCorners() (méthode tetramino.Tetramino), 1
 getEffectif() (méthode stats.Stats), 13
 getLowerCell() (méthode tetramino.Tetramino), 1
 getMaxHeight() (méthode board.Board), 2
 getMaxi() (méthode stats.Stats), 13
 getMean() (méthode stats.Stats), 13
 getMini() (méthode stats.Stats), 13
 getMove() (méthode agent_evaluation.AgentEvaluation),
 6
 getMove() (méthode agent_filtering.AgentFiltering), 6
 getMove() (méthode agent_human.AgentHuman), 5
 getMove() (méthode agent_random1.AgentRandom1), 6
 getMove() (méthode agent_random2.AgentRandom2), 6
 getMoveStats() (méthode agent.Agent), 5

getNbHoles() (méthode board.Board), 2
getNewBlock() (méthode tetris_engine.TetrisEngine), 3
getPossibleMovesDirect() (méthode tetris_engine.TetrisEngine), 3
getQIndexes() (méthode qRL_optimizer.QRLOptimizer), 12
getQuartiles() (méthode stats.Stats), 13
getrandbits() (dans le module ag_optimizer), 10
getrandbits() (dans le module agent), 5
getrandbits() (dans le module agent_evaluation), 7
getrandbits() (dans le module agent_filtering), 6
getrandbits() (dans le module agent_human), 6
getrandbits() (dans le module agent_random1), 6
getrandbits() (dans le module agent_random2), 6
getrandbits() (dans le module qRL_optimizer), 12
getrandbits() (dans le module tetris_engine), 4
getrandbits() (dans le module tetris_RLenv), 11
getScoreFromLines() (méthode tetris_engine.TetrisEngine), 3
getSigma() (méthode stats.Stats), 13
getState() (méthode tetris_RLenv.TetrisEnv), 11
getStateCode() (méthode tetris_RLenv.TetrisEnv), 11
getStrAgentName() (méthode tetris_engine.TetrisEngine), 3
getStrInfos() (méthode tetris_engine.TetrisEngine), 3
getStrNextBlock() (méthode tetris_engine.TetrisEngine), 3
getSumHeights() (méthode board.Board), 2

H

histogram() (méthode stats.Stats), 13

I

initPopulation() (méthode ag_optimizer.AGOptimizer), 9
initQValue() (méthode qRL_optimizer.QRLOptimizer), 12
isCellEmpty() (méthode board.Board), 2
isDominated() (méthode board.Board), 2
isEndGame() (méthode tetris_engine.TetrisEngine), 3
isLineFull() (méthode board.Board), 2
isMoveInGrid() (méthode tetris_engine.TetrisEngine), 3
isMoveOnFreeCells() (méthode tetris_engine.TetrisEngine), 3
isMoveValid() (méthode tetris_engine.TetrisEngine), 3

K

keepOnlyElite() (méthode ag_optimizer.AGOptimizer), 9

L

learn() (méthode qRL_optimizer.QRLOptimizer), 12
linearCombination() (dans le module ag_optimizer), 10
loadData() (méthode stats.Stats), 13

M

makeNewGeneration() (méthode ag_optimizer.AGOptimizer), 9
maxStat() (méthode agent_filtering.AgentFiltering), 6
mergeChains() (dans le module textutil), 14
minimalCopy() (méthode tetris_engine.TetrisEngine), 3
minStat() (méthode agent_filtering.AgentFiltering), 6
moveBlock() (méthode tetris_engine.TetrisEngine), 3
moveBlockInDirection() (méthode tetris_engine.TetrisEngine), 3
moveEvaluation() (méthode agent_evaluation.AgentEvaluation), 7
mutate() (méthode ag_optimizer.AGOptimizer), 9
mutateBinVector() (méthode ag_optimizer.AGOptimizer), 9
mutateFloatVector() (méthode ag_optimizer.AGOptimizer), 9

N

normalize() (dans le module ag_optimizer), 10
npBinaryRepresentation() (méthode board.Board), 2

P

placeBlock() (méthode tetris_engine.TetrisEngine), 3
placeBlockDirect() (méthode tetris_engine.TetrisEngine), 3
play() (méthode qRL_optimizer.QRLOptimizer), 12
playCommand() (méthode tetris_engine.TetrisEngine), 3
playGame() (dans le module agent), 5
playGameWithAgentEvaluation() (dans le module agent_evaluation), 7
plotBenchPlayer() (dans le module agent), 5
plotStats() (méthode ag_optimizer.AGOptimizer), 9
printInfos() (méthode board.Board), 2
printQIndexes() (méthode qRL_optimizer.QRLOptimizer), 12
printRightColumn() (méthode tetris_engine.TetrisEngine), 3
process() (méthode ag_optimizer.AGOptimizer), 10
processLines() (méthode board.Board), 2

Q

qRL_optimizer (module), 12
QRLOptimizer (classe dans qRL_optimizer), 12

R

random() (dans le module ag_optimizer), 10
random() (dans le module agent), 5
random() (dans le module agent_evaluation), 7
random() (dans le module agent_filtering), 6
random() (dans le module agent_human), 6
random() (dans le module agent_random1), 6
random() (dans le module agent_random2), 6

[random\(\)](#) (dans le module `qRL_optimizer`), 12
[random\(\)](#) (dans le module `tetris_engine`), 4
[random\(\)](#) (dans le module `tetris_RLenv`), 11
[randomBinaryList\(\)](#) (méthode `ag_optimizer.AGOptimizer`), 10
[randomBinaryVector\(\)](#) (méthode `ag_optimizer.AGOptimizer`), 10
[randomFloatVector\(\)](#) (méthode `ag_optimizer.AGOptimizer`), 10
[reinit\(\)](#) (méthode `qRL_optimizer.QRLOptimizer`), 12
[removeLine\(\)](#) (méthode `board.Board`), 2
[render\(\)](#) (méthode `tetris_RLenv.TetrisEnv`), 11
[reset\(\)](#) (méthode `tetris_RLenv.TetrisEnv`), 11
[rotate\(\)](#) (méthode `tetramino.Tetramino`), 1
[rotateBlockInDirection\(\)](#) (méthode `tetris_engine.TetrisEngine`), 3
[run\(\)](#) (méthode `tetris_engine.TetrisEngine`), 3

S

[sampleAction\(\)](#) (méthode `tetris_RLenv.TetrisEnv`), 11
[saveData\(\)](#) (méthode `stats.Stats`), 13
[scoreOnOneGame\(\)](#) (méthode `ag_optimizer.AGOptimizer`), 10
[setBlockInitPosition\(\)](#) (méthode `tetris_engine.TetrisEngine`), 3
[setCell\(\)](#) (méthode `board.Board`), 2
[setRotation\(\)](#) (méthode `tetramino.Tetramino`), 1
[sortPopulationDescending\(\)](#) (méthode `ag_optimizer.AGOptimizer`), 10
[Stats](#) (classe dans `stats`), 13
[stats](#) (module), 13
[step\(\)](#) (méthode `tetris_RLenv.TetrisEnv`), 11
[stringOfParameters\(\)](#) (méthode `ag_optimizer.AGOptimizer`), 10

T

[Tetramino](#) (classe dans `tetramino`), 1
[tetramino](#) (module), 1
[tetris_engine](#) (module), 2
[tetris_RLenv](#) (module), 11
[TetrisEngine](#) (classe dans `tetris_engine`), 2
[TetrisEnv](#) (classe dans `tetris_RLenv`), 11
[textColor\(\)](#) (dans le module `textutil`), 14
[textutil](#) (module), 13
[toArray\(\)](#) (méthode `tetramino.Tetramino`), 1
[total_size\(\)](#) (dans le module `qRL_optimizer`), 12
[tournamentSelection\(\)](#) (méthode `ag_optimizer.AGOptimizer`), 10

U

[update\(\)](#) (méthode `qRL_optimizer.QRLOptimizer`), 12
[updateBinaryIndividu\(\)](#) (méthode `ag_optimizer.AGOptimizer`), 10
[updateScore\(\)](#) (méthode `ag_optimizer.AGOptimizer`), 10

[updateStats\(\)](#) (méthode `ag_optimizer.AGOptimizer`), 10
[updateStats\(\)](#) (méthode `board.Board`), 2
[updateTimes\(\)](#) (méthode `tetris_engine.TetrisEngine`), 4

W

[wheelSelection\(\)](#) (méthode `ag_optimizer.AGOptimizer`), 10