Patte Sébastien
Le-guellanff Quentin
22/12/2018

Java L2
Rapport du projet Tetris

Travail effectué :

Nous avons développé dans l'ordre du sujet les trois classes dont le code était à compléter. Pour les deux premières (Piece et Board), nous avons utilisé les jeux de tests unitaires proposés pour les mettre au point. Pour la dernière classe (JbrainTetris) concernant l'intelligence artificielle et le jeu avec un adversaire, nous avons testé directement en exécutant le jeu, en contrôlant les calculs par affichage de valeurs de variables (par exemple le score de la fonction BestMove), et ainsi éliminé tous les défauts que nous constations, jusqu'à obtenir un fonctionnement correct.

1) Une classe 'Piece.java' représentant une pièce du jeu de Tétris et contenant des méthodes qui permettent de :

- construire une pièce a partir d'une 'List<TPoint>', un 'String' ou une 'Piece'.
- avoir des infos sur une pièce (hauteur, largeur, skirt, prochaine rotation)
- tester si deux pièces sont égales

2) Une classe 'Board.java' représentant une grille de Tétris et contenant des méthodes qui permettent de :

- placer une pièce dans la grille
- faire un backup de l'état du board et revenir à çe backup
- supprimer les lignes remplies
- calculer la hauteur où une pièce va tomber

3) Une classe 'JbrainTetris' qui hérite de JTetris et qui ajoute la possibilité de faire jouer une intelligence artificielle. Certaines méthodes de JTetris sont modifiées :

- createControlPanel() pour ajouter le bouton 'Brain active ' et le curseur 'Adversaire'
- tick() pour déplacer la pièce au meilleur endroit selon le résultat de BestMove()
- pickNextPiece() pour choisir la pire pièce possible quand l'adversaire choisit la pièce (déterminé par le pourcentage de chance réglable avec le curseur 'Adversaire').

Contributions :

- Code de 'Piece.java', tests unitaires : Le-guellanff Quentin
- Code de 'Board.java', tests unitaires : Patte Sébastien
- Code de 'JbraintTetris.java' : Patte Sébastien
- Test général et mise au point du jeu, corrections : Patte Sébastien & Le-guellanff Quentin

Difficultés rencontrées :

'Piece.java' :
Les arguments grid, widths et heights sont utilisés dans 'PieceTest.java' mais leur statut était 'private' ce qui provoquait une erreur, il a fallu le changer en protected.

Pour la méthode ‘equals’ on pensait que la fonction devait renvoyer true quand les pièces étaient les mêmes, quelle que soit leur rotation.
Cette version de la fonction a passé les tests, mais elle a posé des problèmes plus tard car la fonction Bestmove ne testait pas les différentes rotation de la pièce (la fonction Equals renvoyait toujours true, même après rotation de la pièce).

‘Board.java’ :
Le constructeur de Board copiait les valeur de ‘this’ dans la pièce passée en paramètre alors que ça devait être l’inverse. Et comme il n’était pas utilisé dans le fichier de tests, les tests étaient validés. Il a donc fallu trouver que c’était ce constructeur qui causait des erreurs.

Deux erreurs dans la fonction dropHeight provoquaient des comportements anormaux.
Dans un premier temps les pièces se déplaçaient toutes à gauche, puis plus tard nous avons observé que BestMove ne plaçait pas les pièces de façon optimale. En fait nous nous sommes rendu compte que les calculs de score n’étaient pas corrects, car dans certains cas, le calcul ne prenait pas en compte la position la plus basse possible pour la pièce.
1) La fonction dropHeight parcourait le skirt avec un compteur i et faisait `y= getColumnHeight(i)` puis renvoyait le plus grand y. Alors qu’il faut prendre en compte ‘x’ la position où la pièce est sur le board, et donc faire `y = getColumnHeight(i+x)`.
2) Pour que la pièce se pose au plus près possible des pièces du bas (et ne reste pas suspendue une ou deux cases au dessus), il fallait prendre en compte le skirt à chaque fois en diminuant y de la valeur du skirt, soit `y = getColumnHeight(i+x)-skirt.get(i)`

‘JBrainTetris.java’ :
La copie du paramètre ‘Board board’ dans la fonction bestMove dans DefaultBrain à la ligne 23 :
`board= new Board(board);`
Avec cette copie de board il arrivait que quand une pièce apparaisse sur le terrain, le dernier essai de bestMove() apparaisse dans le jeu. Le problème a été résolu en mettant cette ligne en commentaire pour utiliser directement le board passé en paramètre.

La fonction tick déplacait d’abord la pièce vers la position de X souhaitée et faisait les rotations après ce qui pouvait faire dépasser la pièce du jeu en la tournant, ce qui entraînait un blocage. Faire les rotations avant les déplacements a résolu le problème.
Remarque : une autre solution serait d’essayer de placer la prochaine rotation de la pièce courante avec board.place() puis faire un undo(), vérifier si elle dépasse et donc faire la rotation ou non.

Exemple de résultat :
Voici ci-dessous un exemple de différence de score selon l’avantage donné à l’adversaire (796 contre 43) :