LMI 1 - Semestre 2 Mars-Avril-Mai 2017

Université d'Artois
Faculté des Sciences Jean Perrin

ALGO2 – Algorithmique et Programmation 2

Projet : Tetris

Étape 3 - Suppression des lignes et affichage de la prochaine forme

Pour cette troisième étape, on s'attaque aux deux dernières fonctionnalités qui permettent de jouer : d'une part, lorsqu'une ligne est complète, elle est supprimée, d'autre part, le jeu affiche la prochaine forme qui va tomber pour que l'utilisateur puisse anticiper ses actions.

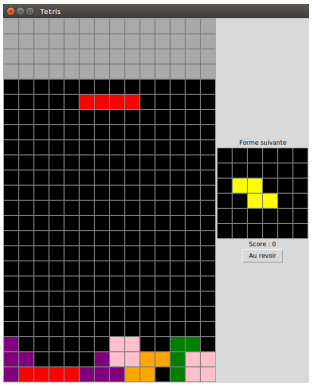

FIGURE 1 – À la fin de l'étape 3

1 Suppression des lignes pleines

Cette action ne concerne que le modèle. Il s'agit de supprimer du terrain les lignes pleines, en faisant *tomber* les lignes supérieures. Les étapes suivantes vous guident. Dans la classe `ModeleTetris` :

1. Ajoutez un attribut `__score` initialisé à 0 dans le constructeur :
2. Implémentez la méthode `est_ligne_complete(self, lig)` qui teste si la ligne d'indice `lig` sur le terrain est complète ;
3. Implémentez la méthode `supprime_ligne(self,lig)` qui supprime la ligne d'indice `lig` sur le terrain. Toutes les valeurs des lignes de `self.__base` à `lig-1` inclus *descendent* d'un cran. Vous ferez attention à **copier** les éléments de chaque ligne. En ligne d'indice `self.__base`, une ligne ne contenant que des -1 apparaît ;
4. Implémentez la méthode `supprime_lignes_completes(self)` qui supprime toutes les lignes complètes de `self.__base` à `self.__hauteur` (exclu). Chaque ligne supprimée augmente de 1 la valeur de `__score` ;
5. Cette dernière méthode doit être systématiquement appelée dans la méthode `forme_tombe`.

Testez votre jeu !

2 Afficher le score

Nous allons maintenant afficher le score obtenu par le joueur, c'est-à-dire le nombre de lignes supprimées.

Dans le modèle

Dans la classe `ModeleTetris`, implémentez une méthode `get_score(self)` qui retourne la valeur du score.

Dans la vue

Dans la classe `VueTetris` :

1. Dans le constructeur, au-dessus du bouton `quitter`, dans la même `Frame`, placez un `Label` dont le texte initial sera `"Score : 0"`. Ce `Label` doit être conservé dans un attribut nommé `__lbl_score`.
2. Implémentez une méthode `met_a_jour_score(self, val)` qui prend un paramètre `val` de type `int` et change le texte de `__lbl_score` pour afficher `val` dans le score.

Dans le contrôleur

Dans la méthode `affichage` de la classe `Controleur`, après avoir fait tomber la forme sur le modèle, redessiner le terrain puis la forme sur la vue, le contrôleur doit demander au modèle la valeur du score et demander à la vue de mettre à jour son affichage.

3 Afficher la forme suivante

Cette partie concerne plus particulièrement la vue. Il s'agit de faire apparaître, dans la zone à droite du terrain de jeu, la prochaine forme qui apparaîtra.

Dans la vue

Dans la classe `VueTetris` :

1. Définissez une constante `SUIVANT` dont la valeur est 6 ;
2. Dans le constructeur, au-dessus de `__lbl_score` :
   (a) créez un `Label` dont le texte sera `Forme suivante :`. Ce composant ne sera pas modifé par l'application, il n'a donc pas à être conservé dans un attribut ;
   (b) créez un `Canvas`, similaire à celui qui permet de dessiner le terrain, mais d'une taille prévue pour contenir `SUIVANT`×`SUIVANT` carrés de côté `DIM`. Ce `Canvas` sera conservé dans un attribut `self.__can_fsuivante` ;
   (c) à l'instar de ce que vous avez fait pour `__can_terrain`, créez `SUIVANT`×`SUIVANT` carrés noirs que vous mémoriserez dans un attribut `__les_suivants` qui sera une liste de listes d'objets créés par la méthode `create_rectangle`. Vous pouvez tester votre jeu et vous devez avoir l'affichage suivant à peu près identique à celui présenté sur 2.
3. Implémentez une méthode `dessine_case_suivante(x,y,coul)`, qui fonctionne comme la méthode `dessine_case` mais qui concerne cette fois `__can_fsuivante` et `__les_suivants` ;
4. Implémentez une méthode `nettoie_forme_suivante` qui remet du noir sur tous les carrés de `__can_fsuivante` ;
5. À l'instar de la méthode `dessine_forme`, implémentez une méthode `dessine_forme_suivante(coords,coul)`
   qui dessine la forme dont les coordonnées et la couleur sont données en paramètre. Il y a deux autres modifications par rapport à `dessine_forme` :
   — la première chose à faire est de nettoyer `__can_fsuivante` ;

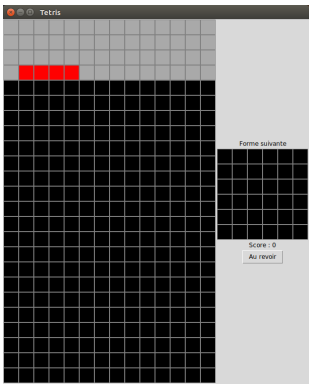

FIGURE 2 – Emplacement de la forme suivante

— ensuite vous redessinerez les carrés de coordonnées `(x+2,y+2)`, et non pas `(x,y)`. En fait, le contrôleur va vous transmettrres les coordonnées relatives de la forme à afficher sur ce panneau. Pour que la forme soit posée à peu près au milieu, il faut décaler chacun des carrés la constituant de deux lignes et de deux colonnes.

Dans le modèle

1. Dans la classe `Forme`, ajoutez une méthode `get_coords_relatives` qui retourne une copie de la liste des coordonnées relatives de la forme (i.e. `__forme`) ;
2. Dans la classe `ModeleTetris` :
   (a) dans le constructeur, ajoutez un nouvel attribut `__suivante` et initialisez-le comme une nouvelle forme (comme ce que vous avez fait pour `__forme`) ;
   (b) implémentez une méthode `get_coords_suivante` qui retourne les coordonnées relatives de `__suivante` ;
   (c) implémentez une méthode `get_couleur_suivante` qui retourne la couleur de `__suivante` ;
   (d) modifiez la méthode `forme_tombe` : maintenant, quand il y a eu collision et que vous avez ajouté la forme au terrain, alors `__forme` prend la valeur de `__suivante`, et `__suivante` est réinitialisé par une nouvelle `Forme`.

Dans le contrôleur

Dans la méthode `affichage`, si la forme courante s'est posée sur le terrain, alors il faut que le contrôleur récupère auprès du modèle les coordonnées relatives de la forme suivante ainsi que la couleur, et qu'il demande à la vue d'afficher la forme suivante.

Testez votre jeu, c'est fini pour cette étape !