LMI 1 - Semestre 2 Mars-Avril-Mai 2017

Université d'Artois
Faculté des Sciences Jean Perrin

ALGO2 – Algorithmique et Programmation 2

Projet : Tetris

Étape 2 - On agit sur les formes

Pour cette deuxième étape, l'utilisateur va pouvoir agir sur la forme : la déplacer à gauche, à droite, la faire tourner, la faire tomber plus vite… et même jouer avec plusieurs types de forme !

1 Déplacer la forme latéralement

Pour déplacer la forme latéralement, il faut que
— l'utilisateur appuie sur une touche (la flèche gauche ou la flèche droite, par exemple),
— que cet événement soit écouté par un contrôleur,
— que le contrôleur prévienne le modèle,
— que le modèle agisse (qu'il prévienne la forme de se déplacer).
La vue sera mise à jour au prochain top d'horloge, et cela fonctionne déjà dans votre application.
Il faut donc intervenir sur toutes les classes. Nous allons commencer par la forme, remonter jusqu'au contrôleur puis apporter les modifications sur la vue.

1.1 Dans la classe `Forme`

Une forme doit maintenant pouvoir se déplacer à gauche ou à droite, c'est à dire incrémenter ou décrémenter son attribut `self.__x0` … tout en prenant en compte les obstacles qui pourraient empêcher le mouvement.

D'une manière plus générale, quel que soit le mouvement qu'on voudra donner à la forme (à gauche, à droite, tourner), il faudra tester si la nouvelle position de la forme est valide, c'est-à-dire avec une abscisse et une ordonnée correctes, et sans empiéter sur une cellule qui serait déjà occupée dans le terrain. Le schéma général le plus simple pour ces fonctions de déplacement est :
— de faire le mouvement ;
— de tester si la nouvelle position est valide ;
— si elle ne l'est pas, faire le mouvement arrière.

1. Implémentez la méthode `position_valide` qui teste si chaque coordonnée absolue `(x,y)` de la forme est valide. Pour cela, la méthode doit d'abord récupérer les coordonnées absolues de la forme. Une coordonnée `(x,y)` est valide si `x` est dans un intervalle correct, si `y` est dans un intervalle correct, et si le terrain n'est pas déjà occupé en `(x,y)` ;
2. Implémentez la méthode `a_gauche` qui déplace la forme d'une colonne vers la gauche (si possible) : décrémente la valeur de `self.__x0` et teste si la nouvelle position de la forme est valide. Si ce n'est pas le cas, incrémente la valeur de `self.__x0` (`self.__x0` reprend sa valeur initiale) ;
3. De la même façon, implémentez la méthode `a_droite` qui déplace la forme d'une colonne vers la droite (si possible).

1.2 Dans la classe `ModeleTetris`

Les méthodes suivantes sont à ajouter dans la classe `ModeleTetris`.

1. Implémentez la méthode `forme_a_gauche` qui demande à la `__forme` de se déplacer à gauche ;
2. Implémentez la méthode `forme_a_droite` qui demande à la `__forme` de se déplacer à droite.

1.3 Dans la classe `Controleur`

Les deux méthodes à ajouter dans la classe `Controleur` prennent toutes les deux, en plus du paramètre `self`, un paramètre `event` (qui ne nous sert à rien dans cette application). Ce sont les deux méthodes qui seront directement appelées par une action du joueur.

1. Implémentez la méthode `forme_a_gauche(self,event)` qui demande au modèle de déplacer la forme à gauche ;
2. Implémentez la méthode `forme_a_droite(self,event)` qui demande au modèle de déplacer la forme à droite.

Il faut maintenant que les événements *appui sur la flèche gauche* et *appui sur la flèche droite* soient liés aux actions `forme_a_droite` et `forme_a_gauche`. C'est la fenêtre `Tk` qui écoute les événements, c'est elle qui doit faire le lien.

Dans le constructeur de la classe `Controleur`, juste après avoir récupéré la fenêtre de la vue dans l'attribut `self.__fen`, faites le lien pour le déplacement à gauche :

```
self.__fen.bind("<Key-Left>",self.forme_a_gauche)
```

Procédez de façon analogue pour le déplacement à droite.
Testez votre application. Vous devez pouvoir déplacer la forme à gauche et à droite.

2 Faire tomber plus vite une forme

Faire tomber plus vite une forme n'a pas d'incidence sur le modèle : la seule conséquence est visuelle, et elle consiste à appeler dans le `Controleur` la méthode `joue` plus fréquemment, autrement dit à diminuer la valeur de `self.__delai`. Il est important de remettre le délai à sa valeur initiale dès que la forme s'est posée.

Pour cette action, tout se passe dans la classe `Controleur`.

1. Implémentez une méthode `forme_tombe(self,event)` qui modifie la valeur de l'attribut `__delai` (je vous conseille une valeur entre 160 et 200) ;
2. Dans le constructeur, liez l'événement *appui sur la flèche bas* `"<Key-Down>"` à la méthode `forme_tombe` ;
3. Modifiez légèrement la méthode `affichage` : vous devez maintenant récupérer le booléen retourné par l'appel à la méthode `forme_tombe` du modèle. Si la forme n'est pas tombée (il y a eu collision et elle s'est posée), remettez l'attribut `self.__delai` à sa valeur initiale.

Testez votre application. L'appui sur la flèche bas doit faite tomber plus vite une forme, et la vitesse normale doit être rétablie à la forme suivante.

3 Tourner une forme

Lorsqu'on tourne une forme, les coordonnées relatives qui décrivent la forme changent. La figure 1 montre les différentes positions de la forme que nous utilisons actuellement lorsqu'on la fait tourner d'un quart de tour à droite (dans le sens horaire).

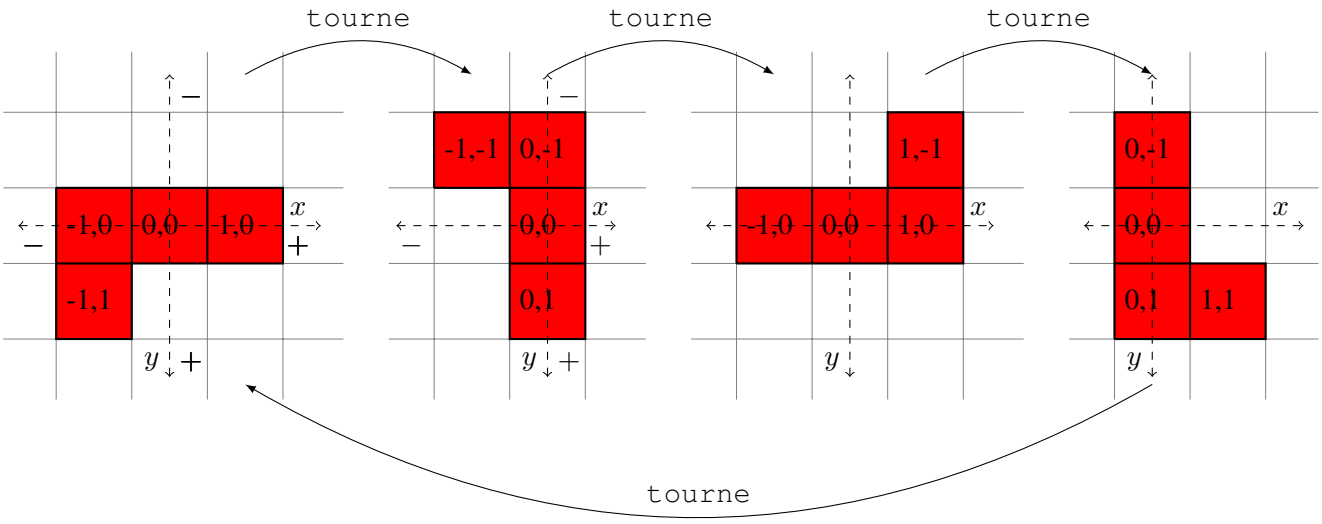

FIGURE 1 – Rotation d'une forme

Nous faisons des rotations de 90 ° par rapport à l'origine de notre repère... Vous pouvez constater que l'image d'une cellule aux coordonnées `(x,y)` par une rotation de 90 ° dans le sens horaire est une cellule aux coordonnées `(-y,x)`.

1. Implémentez la méthode `tourne` de la classe `Forme`. Cette méthode mémorise la valeur de `__forme` (i.e. la liste des coordonnées relatives de la forme) dans une variable `forme_prec`. Puis elle réinitialise `__forme` en calculant pour chaque coordonnée `(x,y)` de `forme_prec` son image par la rotation, c'est-à-dire `(-y,x)`. La méthode regarde ensuite si la nouvelle position de la forme est valide. Si ce n'est pas le cas, `__forme` reprend sa valeur précédente (qui a été mémorisée dans `forme_prec`);
2. Implémentez la méthode `forme_tourne` de la classe `ModeleTetris`. Cette méthode demande à la forme de tourner;
3. Implémentez la méthode `forme_tourne(self,event)` de la classe `Controleur` qui demande au modèle de faire tourner la forme;
4. Faites le lien dans le constructeur de la classe `Controleur` entre l'action *appui sur la flèche haut* `"<Key-Up>"` et la méthode `forme_tourne` de la classe `Controleur`.

Testez votre application, vous devez pouvoir faire tourner votre forme.

4 Plusieurs formes

Nous avons maintenant tous les éléments pour jouer avec plusieurs formes. Les sept formes qu'on trouve habituellement dans le jeu sont présentées dans 2.

1. Avec un papier et un crayon, déterminez sur chaque forme une case centrale (celle qui sera en `(0,0)`, qui sera le pivot), et les adresses relatives des autres cases;
2. Dans le fichier `modele.py`, initialisez une constante `LES_FORMES` qui sera une liste de liste de couples `(int,int)`, c'est-à-dire qui contiendra la liste des coordonnées relatives de chaque forme que vous avez dessinée :

   ```
   LES_FORMES = [[(-1,1),(-1,0),(0,0),(1,0)], ... ] # a completer
   ```

3. Dans le constructeur de la classe `Forme`, tirez un indice au hasard dans l'intervalle des indices possibles de la liste `LES_FORMES`. Cet indice deviendra la valeur de `__couleur`, et l'élément

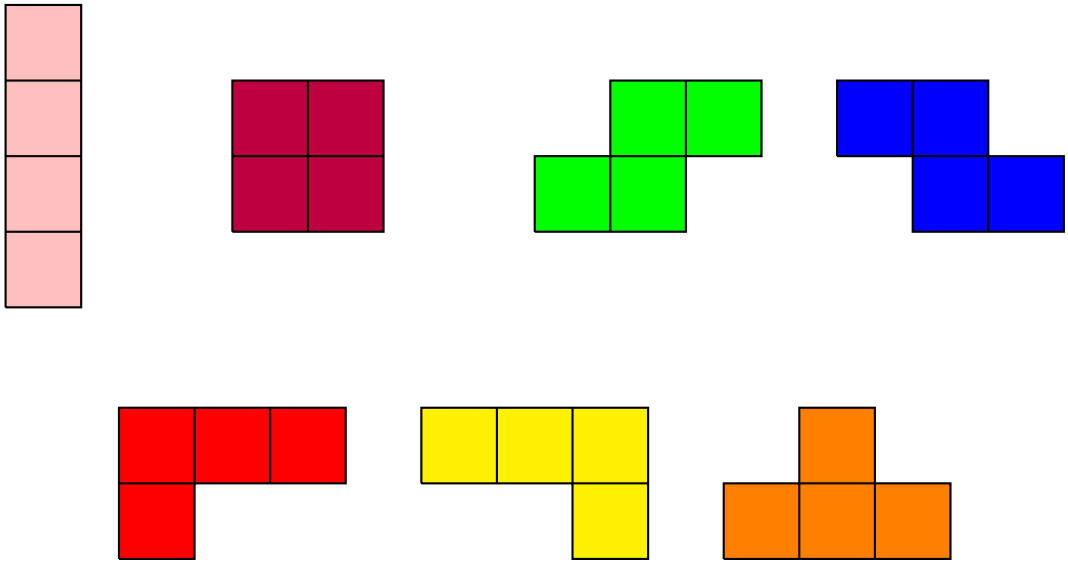

FIGURE 2 – Les différentes formes du jeu

correspondant dans `LES_FORMES` vous donnera la valeur de `__forme`. Vous devez donc avoir un nombre de couleurs dans la constante `COULEURS` du module `vue` cohérent avec le nombre d'éléments de `LES_FORMES`. C'est le cas si vous avez suivi précisément le sujet.

Testez votre application , jouez... C'est fini pour cette étape !