Auteur: **Abdoulaye KATCHALA MELE**

Groupe 3 L1 Informatique

Projet : Tetris

I. Le Fonctionnement du Jeu

Explications brève des différentes classes du jeu et leurs méthodes

ModeleTetris

Dans le constructeur on a sauvegardé le nombre de ligne et le nombre de colonne dans les attributs respectifs ***self.__haut*** et ***self.__larg***. On a sauvegardé aussi la base dans l'attribut ***self.__base.*** On a crée la forme dans l'attribut ***self.__forme*** et la forme suivante dans ***self.__suivante.*** On a crée ***self.__score*** pour sauvegarder le score, ***self.__coef*** pour connaître la valeur à ajouter au score à chaque ligne complète et ***self.__delai*** pour connaître le delai par défaut de la boucle du jeu. On a crée le terrain qui est une matrice de dimension self.__haut×self.__larg composée de valeurs et que l'on a sauvegardé dans ***self.__terrain***.

```
# Attribut Terrain qui est une matrice
self.__terrain = list()
for i in range(self.__haut) :
    self.__terrain.append(list())
    for j in range(self.__larg) :
        if i < 4 :
            self.__terrain[i].append(-2)
        else :
            self.__terrain[i].append(-1)
```

La méthode ***ajoute_forme(self)*** parcours les coordonnées absolues de la forme sur le terrain et affecte l'indice de couleur de la forme aux cases ayant ses coordonnées sur self.__terrain. Cette fonction nous permettra d'ajouter la forme sur le terrain dès que il y'a une case en occupée en dessous ou que la forme à atteint sa limite

La méthode ***forme_tombe(self)*** crée une variable issu de l'appel à la fonction tombe de la classe forme afin de nous indique si il y'a eu collision ou pas. Si il y'a eu collision La méthode ajoute la forme au terrain**,** elle supprime les lignes complètes, elle affecte la forme suivante à self.__forme et crée une nouvelle forme qui sera la suivante. Si non elle retourne tout simplement Faux.

La méthode ***supprime_ligne(self,lig)*** parcours le terrain de la ligne d'indice lig à celle d'indice self.__base et parcours ces lignes pour leur affecter les valeurs des lignes au dessus et les valeurs des cases non occupées à la ligne du self.__base. Elle permet de supprimer la ligne d'indice lig.

La méthode ***supprime_lignes_complètes(self)*** parcours tout le terrain et voit si une ligne est complète. Si c'est le cas elle supprime cette ligne et incrémente le score. Ensuite si le score du joueur est compris entre un intervalle bien défini, elle rend le jeu plus rapide en diminuant self.__delai et incrémente self.__coef; ainsi le joueur gagne plus de points en supprimant des lignes.

La méthode ***ctrl_recommencer(self)*** parcours le terrain pour rendre toutes les cases inoccupées c'est à dire leur affecte la valeur -1 ou -2. Elle permet au joueur de recommencer la partie.

Forme

Dans le constructeur on a sauvegardé l'indice de couleur de la forme qui est un randint de 0 à 7 dans l'attribut ***self.__couleur***, une instance de ModeleTetris dans l'attribut ***self.__modele*** puis une liste de coordonnées provenant de la constante LES_FORMES, relative à une cellule de la forme représentant un type forme dans l'attribut ***self.__forme*** et les coordonnées sur le terrain de la case relative dans les attributs ***self.__x0(qui est un randint entre 2 et self.__larg - 2) et self.__y0(initialisé à 0).***

La méthode ***get_coords(self)*** retournent les coordonnées de la forme sur le terrain. Pour obtenir cette dernière on a crée une liste à laquelle on a mis les coordonnées relative de self.__forme plus les coordonnées de la case origine.

La méthode ***position_valide(self)*** parcours les coordonnées absolues de la forme et voit si une case est déjà occupée. Si c'est le cas elle retourne Faux, Vrai si non. Elle nous permettra de vérifier si la position de la forme est valide après un déplacement ou un retournement.

La méthode ***a_gauche(self) et a_droite(self)*** déplacent respectivement la forme à gauche et à droite en desincrémentant et en incrémentant la valeur de self.__x0 et vérifient si la position de la forme n'est pas valide. Si non elles initialisent sa valeur.

La méthode ***tourne(self)*** parcours self.__forme pour remplacer les coordonnées de la forme par leurs rotation de 90° . Si la position n'est pas valide, elle initialise la valeur de self.__forme en lui affectant la valeur de forme_prec (qu'on a enregistré au début afin de garder la valeur de self.__forme)

```
# Memorise sa valeur dans forme_prec
forme_prec = self.__forme[:]

# Parcours self.__forme
for i in range(len(self.__forme)) :
    # Remplace les coordonnées par leur rotation
    self.__forme[i] = (-forme_prec[i][1],forme_prec[i][0])
# Si la position n'est pas valide elle le remet à sa valeur initiale
if not self.position_valide() :
    self.__forme = forme_prec
```

VueTetris

Dans le constructeur, on a sauvegardé le modele dans l'attribut ***self.__modele*** et un booléen nous permettant de mettre la partie en pause dans ***self.__pause***. On a crée la fenêtre qu'on a sauvegardé dans l'attribut ***self.__fen***. On a crée une frame pour mettre certains éléments qu'on a placé à droite. On a sauvegardé dans l'attribut ***self.__can_terrain*** un canvas de largeur 14 fois DIM et de 24 fois DIM et dans l'attribut ***self.__can_fsuivante*** un canvas de largeur et de hauteur SUIVANT*DIM. On a créé un bouton Au revoir pour quitter, le bouton Pause dans ***self.__btn_ctrl*** et un label pour afficher le score et les avons placés dans la frame. On a créé des rectangles dans les deux canvas en fonction de leurs dimensions. On a sauvegardé chaque rectangle dans deux Matrices de même dimension que leurs canvas dans l'attribut ***self.__les_cases et self.__les_suivants***

La méthode ***dessine_terrain(self)*** parcours les rectangles du canvas et leur met la couleur des valeur des case de mêmes coordonnées que ces rectangles dans le terrain. Ceci se fait par l'appel de la méthode precedente.

La méthode ***dessine_forme(self,coords,couleur) et dessine_forme_suivante(self,coords,coul)*** parcours les coordonnées en paramètre et change respectivement la couleur des rectangles de mêmes coordonnées à la couleur passée en paramètre et la couleur des rectangles de même coordonnées (mais 2 fois incrémenté) à la couleur passée en paramètre après avoir nettoyé le canvas des suivants. On l'utilisera pour dessiner la forme et la forme_suivante.

La méthode ***nettoie_forme_suivante(self)*** change la couleurs de toutes les rectangles de self.__les_suivants en noir. Elle nous permettra de nettoyer le canvas pour mettre la prochaine forme suivante.

La méthode ***ctrl_pause_recom(self)*** permet de mettre le jeu en marche en mettant self.__pause à Faux. Dans ce cas elle réécrit le texte du bouton Pause qui était ***Commencer*** en ***Pause***; et si le jeu était fini, elle va appeler la méthode ctrl_recommencer du modele pour recommencer le jeu. Si le jeu est en cours, elle permet de lettre le jeu en pause et réécrit le texte du bouton en ***Reprendre***.

Controleur

Dans le constructeur, on a sauvegardé le modele dans l'attribut ***self.__modele,*** la vue dans l'attribut ***self.__vue*** et la fenêtre de la classe VueTetris. On a demandé à la vue de dessiner la forme et la forme suivante sur le terrain. On a sauvegardé le délai de l'avancement de la forme dans l'attribut ***self.__delai***. On a lancé un appel à la méthode joue et le lancement de la boucle des évènements sur la fenêtre. On a ajouter des évènements avec la fonction bind sur la ***flèche gauche*** déplaçant la forme vers la gauche, la ***flèche droite*** déplaçant la forme vers la droite, ***la flèche bas*** augmentant la vitesse du jeu et ***la flèche haut*** la faisant diminuer, La touche ***espace*** tournant la forme de 90° et la touche ***Entrée*** qui fait démarrer le jeu et met la pause.

```
# Un Clique sur la flèche gauche deplace la forme vers la gauche
self.__fen.bind("<Key-Left>",self.forme_a_gauche)
# Un Clique sur la flèche droite deplace la forme vers la droite
self.__fen.bind("<Key-Right>",self.forme_a_droite)
# Un Clique sur la flèche bas deplace la forme plus rapidement vers le bas
self.__fen.bind("<Key-Down>",self.forme_tombe)
# Un Clique sur la flèche haut fait deplace la forme moins rapidement
self.__fen.bind("<Key-Up>",self.forme_remonte)
# Un Clique sur espace tourne la forme
self.__fen.bind("<space>",self.forme_tourne)
# Un Clique sur Entrée met le jeu en pause ou fait recommencer le jeu
self.__fen.bind("<Return>",self.ctrl_pause_recom)

# demande à la vue de dessiner la forme du modele et la forme suivante
self.__vue.dessine_forme(self.__modele.get_coords_forme(),self.__modele.get_couleur_forme())
self.__vue.dessine_forme_suivante(self.__modele.get_coords_suivante(),self.__modele.get_couleur_suivante())
```

La méthode ***joue(self)*** appelle la méthode affichage de la classe Controleur si le jeu n'est pas fini et que il n'est pas en pause; si non si le jeu est fini elle réécrit le texte du btn_ctrl en ***Recommencer.*** Ensuite elle se rappelle après le delai sauvegardé précédemment.

La méthode ***affichage(self)*** fait tomber la forme d'une case, redessiner le terrain puis la forme sur le terrain , demande le score au modèle et le met à jour. Si il y'a collision, elle remet la valeur du delai à la valeur du delai par défaut dans le modèle et dessine la forme suivante sur le canvas des suivants.

La méthode ***forme_tombe(self) et forme_remonte(self)*** font respectivement diminuer et augmenter la valeur de self.__delai de 90 (pour que la dorme tombe plus vite ou moins vite) à chaque fois qu'on les appelle.

La méthode ***ctrl_pause_recom(self,event)*** nous permet d'associer au bouton entré la commande ***self.__vue.ctrl_pause_recom()*** sans pour autant la modifier.

Résumé du fonctionnement du programme du jeu

Pour lancer le jeu, il suffit d'exécuter le contrôleur dans la console et d'appuyer sur la touche ***Entrée*** ou le bouton ***Commencer***.

Le contrôleur enregistre le modèle qui crée la forme, la forme suivante et le terrain et ensuite le contrôleur crée la vue, lance la boucle des évènements et appelle sa méthode ***joue***. La méthode joue est la boucle

principale du jeu; elle vérifie si la partie est finie ou que la partie est en pause, si oui elle modifie le bouton ***Pause en Recommencer***, si non elle appelle la méthode ***affichage***.

L'appel à la méthode affichage permettra de faire tomber la forme d'une case en appelant la méthode ***forme_tombe*** du modele qui, à son tour, appelle la méthode ***tombe*** de la classe forme.

Si il n'y a pas eu pas collision (la forme n'a pas atteint la limite ou n'a pas atteint une autre forme sur le terrain), la méthode forme_tombe retourne False et la méthode affichage va dessiner le terrain et la forme à sa nouvelle position et va demander le score au modèle pour le mettre à jour sur la vue. Ensuite la méthode joue va se rappeler après le delai inscrit; et donc la boucle recommence.

Si non, si il y'a collision la méthode forme_tombe va ajouter la forme sur le terrain en appelant la méthode ***ajoute_forme,*** elle va demander à la méthode ***supprime_ligne_complètes*** de supprimer les lignes complètes**.** Cette dernière méthode va quant à elle supprimer les lignes concernées, ajouter au score le **coefficient** et va rendre le jeu plus rapide en regardant le score du joueur.

Par la suite la méthode forme_tombe va affecter la forme suivante à self.__forme puis va créer une autre forme à self.__suivant. Ensuite la méthode affichage va remettre la valeur du delai à la valeur du delai par défaut dans le modèle, dessiner la forme suivante sur le canvas des suivants, ensuite redessiner le terrain mis à jour et la nouvelle forme et enfin recuperer le score pour le mettre à jour sur la vue. Ensuite la méthode joue va se rappeler après le délai ; donc la boucle recommence.

```
    # Elle remet self.__delai à sa valeur en fonction du score
    self.__delai = self.__modele.get_delai()
    # Elle demande à la vue de dessiner la forme suivante en recuperant sa couleur et ses coordonnées au près du modele
    self.__vue.dessine_forme_suivante(self.__modele.get_coords_suivante(),self.__modele.get_couleur_suivante())

# La vue redessine son terrain
self.__vue.dessine_terrain()
# La vue redessine la forme
self.__vue.dessine_forme(self.__modele.get_coords_forme(),self.__modele.get_couleur_forme())
# Demande au modele le score
val = self.__modele.get_score()
# Met le score à jour
self.__vue.met_a_jour_score(val)
```

la partie est finie dès que les formes du terrain atteignent une case de la dernière ligne et dans ce cas, le joueur peut recommencer la partie en appuyant soit sur ***Entrée ou Recommencer***. Notons que plus le score du jeu augmente plus le jeu devient rapide et compliqué, aussi le joueur gagne beaucoup plus de point à ce moment. Les formes ne débordent pas dans le terrain car j'ai pris les cases d'origine relative tout en haut de la forme.

Un clique sur la ***flèche gauche*** appelle la méthode ***forme_a_gauche*** déplaçant la forme vers la gauche.

Un clique sur la ***flèche droite*** appelle la méthode ***forme_a_droite*** déplaçant la forme vers la droite.

Un clique sur ***la flèche bas*** appelle la méthode ***forme_tombe*** augmentant la vitesse du jeu.

Un clique sur ***la flèche haut*** appelle la méthode ***forme_remonte*** faisant diminuera vitesse du jeu,

Un clique sur la touche ***espace*** tourne la forme de 90° et sur la touche ***Entrée*** fait appel à ***ctrl_pause_recom*** qui démarre le jeu et met la pause.