UNIVERSITÉ JEAN MONNET SAINT-ÉTIENNE

Outils Info. Tetris

Important : rendu sur Moodle (à la fin)

Vos réponses seront rendues dans une archive zip : cette archive contiendra un fichier `compte-rendu.txt` (pour les réponses aux questions précédées par une apostrophe) et vos fichiers pythons.

Pour faire une **archive zip** de votre TP pour le rendu, exécutez, dans le terminal :

```
cd ~/info-pi1/
zip -r tptetris.zip tptetris/
```

Important : mise en place (au début)

Pour tous les TPs :

- travaillez dans un dossier dédié au TP, lui même dans `~/info-pi1`,
- créez un fichier `compte-rendu.txt` pour écrire la date, votre **nom** et **groupe**, et les réponses aux questions précédées d'une apostrophe,

Pour démarrer, vous pouvez par exemple exécuter les commandes (respecter les minuscules) :

```
cd ~/info-pi1/
mkdir tptetris
cd tptetris
emacs compte-rendu.txt &
```

- [**NB**] **Travaillez dans le dossier** `tptetris` (sans sous-dossier par exercice) (*sauf instructions contraires*).
- [**NB**] Téléchargez (du site du cours) et décompressez le fichier nommé `ressources-tptetris.zip`.

NB : si vous écrivez un programme qui prend trop de temps, vous pouvez l'interrompre avec `Ctrl+C`.

Ce TP traite de **Tetris**, il est donc recommandé d'avoir déjà commencé à réfléchir au problème et/ou d'utiliser la feuille de « TD » annexe à ce sujet (voir site du cours).

Exercice 1 – Tour des Choses Fournies

Q1) Copiez le programme d'exemple avec `cp debut-tris.py tris.py` et ouvrez `tris.py`.

Q2) Copiez la bibliothèque d'exemple avec `cp debut-jeu.py jeu.py` et ouvrez `jeu.py`.

'Q3) Lancez `tris.py` et essayez d'appuyer sur les flêches gauche/droite. Que fait le programme et que font les flêches ?

'Q4) Donnez la liste des fonctions définies dans `jeu.py` (sauf les fonctions de test).

Exercice 2 – Définition et Affichage des Pièces

Q5) Dans `jeu.py`, ajouter une fonction `creer_les_pieces` qui retourne une liste de 7 pièces, chaque pièce étant une grille de 4×4 et contenant les valeurs comme dans le TD.

Q6) Dans `tris.py`, créez une fonction `piece_au_hasard` qui retourne une pièce au hasard, tirée dans la liste des 7 pièces (vous pouvez utiliser la fonction `randrange(N)` de la bibliothèque `random`).

Q7) Dans `tris.py`, créez une variable `piece` qui sera initialisée à partir d'un appel à `piece_au_hasard()`. Utilisez `print` pour vérifier que la pièce est bien une liste de liste (une grille 4×4).

Q8) Dans `tris.py`, créez une variable `piece_pos` qui est une liste à deux éléments et qui contient la position (ligne, colonne) de la pièce (de son coin inférieur gauche). Initialisez cette variable pour que la pièce soit presque tout en haut et horizontalement au milieu.

Q9) Dans la partie dédiée à l'affichage dans la boucle principale, appelez la fonction `afficher_piece` en lui donnant, entre autre, la pièce et la position courante (et la taille verticale de la grille).

Q10) Vérifiez que votre programme affiche bien la pièce.

Exercice 3 – Déplacement et Rotation des Pièces

Q11) Dans la boucle principale, faite que la pièce se déplace vers le bas à chaque pas de temps, vers la gauche quand l'utilisateur appuie sur la flèche gauche et la droit pour la flèche droite (ne plus retirer les lignes dans ces deux cas).

Q12) Vérifiez que votre programme se comporte comme voulu.

Q13) Dans la boucle principale, gérez l'événement flèche du haut (`16777235`) qui doit appeler la fonction `tourner_piece_droite` pour faire tourner la pièce à droite. Cette fonction est pré-remplie mais nécessite d'être modifiée pour effectivement fonctionner (elle copie pour l'instant la pièce sans la tourner).

Q14) Vérifiez que la nouvelle fonctionnalité est bien opérationnelle.

Q15) Faite de même pour la flèche du bas (`16777237`) qui doit appeler la fonction `tourner_piece_gauche` que vous devez créer.

Q16) Vérifiez que la nouvelle fonctionnalité est bien opérationnelle.

Exercice 4 – Arrêt des Pièces

Q17) Dans la boucle principale, à chaque pas de simulation, après avoir fait descendre la pièce, appelez la fonction `probleme()` et si elle renvoie vraie (c'est à dire que la pièce est à un endroit interdit) remontez la pièce. La fonction `probleme` renvoie pour l'instant toujours « faux » donc le comportement ne doit pas changer.

Q18) Remplissez la fonction `probleme`, en utilisant la même structure que pour `afficher_piece`, sauf que chaque bloc de la pièce doit non pas être affiché mais testé comme étant « valide ». Dès qu'un bloc est invalide, `probleme` doit renvoyer « faux ». Un bloc est invalide si il est trop à gauche, trop en bas, trop à droite ou si il est superposé à une case non vide du niveau.

Q19) Vérifiez que la pièce ne descend plus quand elle rencontre le bas du niveau ou un obstacle.

Q20) Dans la boucle principale, sur le modèle de ce que vous avez fait à chaque pas de simulation, faite que chaque déplacement/rotation de la pièce soit suivi d'une détection de problème et d'une « annulation » de l'action qui a été faite, en cas de problème.

Q21) Vérifiez que la pièce ne peut plus sortir du jeu ou traverser les obstacles.

Exercice 5 – Intégration des Pièces

Q22) Quand une pièce est arrêtée pour cause de problème (lorsqu'elle descend automatiquement), appeler la fonction `integrer_piece`, puis retirer une pièce au hasard et ré-initialisez sa position.

Q23) Vérifiez qu'une nouvelle pièce est bien tirée (et que pour l'instant la pièce d'avant disparaît).

Q24) Remplissez la fonction `integrer_piece`, en utilisant la même structure que pour `afficher_piece`, sauf que chaque bloc de la pièce doit non pas être affiché mais ajouté dans le niveau.

Q25) Vérifiez que la pièce est maintenant bien ajoutée au niveau.

Exercice 6 – Retrait des Lignes pleines

Q26) Après l'appel à `integrer_piece`, appelez la fonction `retirer_lignes_pleines` et utilisez `print` pour afficher sa valeur de retour (pour l'instant elle ne fait rien).

Q27) Remplissez la fonction `retirer_lignes_pleines`, qui doit : parcourir les lignes de haut en bas et pour chacune, si elle est pleine (toutes ses valeurs sont non-nulles), appeler `retirer_ligne` pour enlever la ligne. La fonction doit aussi renvoyer le nombre de lignes retirées.

Q28) Vérifiez que les bons nombres s'affichent lorsque vous complétez une ou plusieurs lignes en jouant (vous pouvez modifier à volonté la fonction `remplir_pour_tester` pour créer un plateau initial plus favorable).

Exercice 7 – Options

Dans le désordre, vous pouvez :

Q29) Colorer la pièce courante (sans copier coller tous les appels à `couleur`.

Q30) Ajouter un score, qui augmente de 1 pour chaque ligne faite, de 4 pour 2 lignes, de 9 pour 3, de 16 pour 4.

Q31) Faire accélérer la partie à chaque niveau (un nombre de points donné).

Q32) Tirer à l'avance et afficher la pièce suivante.

Q33) Ajouter un fantôme montrant où la pièce se trouverait si on la laissait aller jusqu'en bas.

Q34) Faire que la barre d'espace (code `32`) envoie directement la pièce en bas.

Q35) Faire que la touche A (code `65`) fasse descendre la pièce d'une case vers le bas.