Introduction à la Programmation

—

Examen final sur machine

—

G1: mathis.petrovich(at)enpc.fr
G2: thomas.belos(at)enpc.fr
G3: clement.riu(at)enpc.fr
G4: nicolas.audebert(at)cnam.fr
G5: pascal.monasse(at)enpc.fr
G6: laurent.bulteau(at)u-pem.fr
G7: abderahmane.bedouhene(at)enpc.fr

—

15/01/21

—

1 Enoncé

1.1 Jeu 10×10

Nous allons programmer le jeu 10×10, une variante de Tetris mais sans animation ni gravité. On a des pièces de différentes formes qu'il faut placer sur une grille, de taille 10×10 normalement. Toute ligne ou colonne de la grille remplie est détruite et rapporte des points. Le jeu se termine quand un nouvelle pièce apparaît et qu'il n'y a plus d'espace libre où la déposer. Utilisez un projet Imagine++ de base. Reportez dans le code en commentaire les questions auxquelles vous répondez (`Q1`, `Q2`, etc).

Il est important de livrer un code clair (commenté et indenté) et qui compile sans warning, même s'il ne répond pas à toutes les questions. Pour cela, vérifiez à chaque étape que votre programme compile et se lance correctement. À la fin de l'examen, nettoyez votre code (indentation...) et vérifiez encore qu'il compile. Créez alors une archive contenant votre code et le fichier `CMakeLists.txt`.

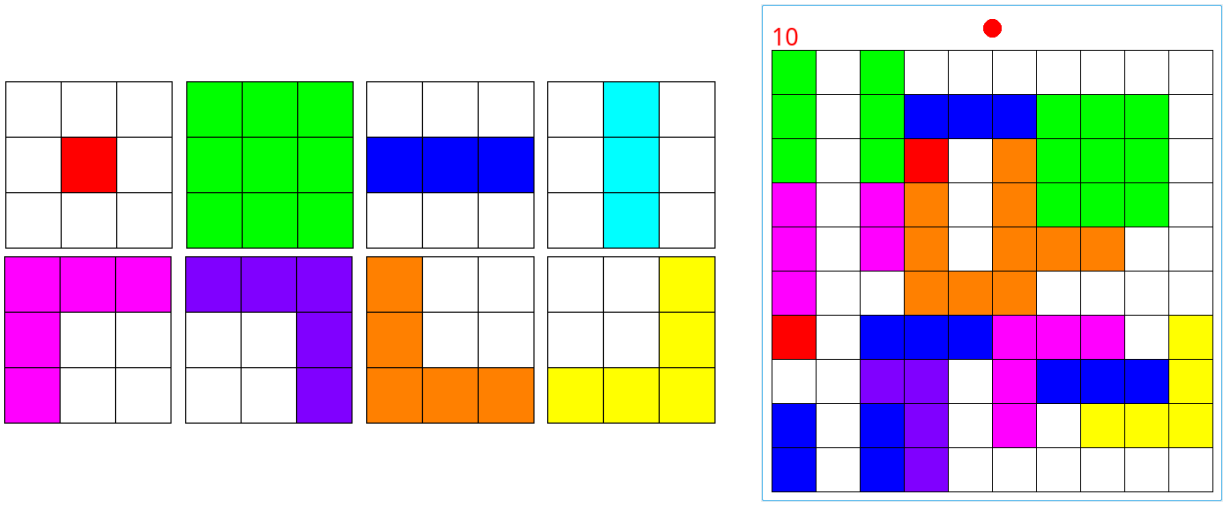

FIGURE 1 – Les 8 pièces du jeu 10×10, composées d'au maximum 3×3 cases (cases blanches non occupées par la pièce), chacune avec sa couleur. À droite, le jeu en cours de partie. Une colonne remplie du plateau 10×10 a été détruite, d'où un score affiché de 10 points. Le voyant rouge en haut indique que la pièce courante (un L orange), située au centre, ne peut être déposée là car elle recouvre au moins une case occupée. Le joueur doit la déplacer à un emplacement libre avant de la déposer.

1.2 Pièces

1. On laisse de chaque côté `marge`= 10 pixels avec le bord de la fenêtre, sauf en haut avec `margeScore`= 50 pixels. On a une grille carrée de `size`= 10 cases de côté et chaque case occupe $z \times z$ pixels avec $z = 50$. Ouvrir une fenêtre de taille correcte pouvant afficher tout cela.
2. Implémenter une fonction `dessine_grille` qui dessine la grille.
3. Dans un fichier séparé, créer une classe `Piece`, composée de jusqu'à 3×3 cases (un booléen indique si la case est remplie), une couleur et une position (x, y) de la case centrale dans la grille.
4. Ajouter un constructeur à la classe. Celui-ci prend une des 8 formes de pièce au hasard (à chaque forme est associée une couleur unique). Sa position est à peu près centrée dans la grille. Pour les 4 pièces en forme de L, essayez de ne pas faire 4 fois des codes similaires.
5. Ajouter une méthode `bloc(i,j)` ($-1 \leq i, j \leq 1$) permettant de savoir si la case centrale $+(i, j)$ est occupée et une méthode `couleur`.
6. Ajouter une méthode `dessine`.
7. Tester cette classe dans une fonction `test_deplacement` : les touches flèches du clavier déplacent une pièce d'une case dans la direction indiquée (on ne vérifie pas si on sort de la grille), la touche "Entrée" termine la fonction. Utiliser la fonction `getKey` d'Imagine++ qui renvoie le code de la touche appuyée par l'utilisateur.

1.3 Plateau

8. Créer une classe `Plateau` composé de $w \times h$ cases. Potentiellement $w \neq h$ et ces dimensions ne sont pas fixées a priori, même si nous n'utiliserons qu'un plateau avec $w = h =$`size`.
9. Écrire constructeur et destructeur de cette classe. Les cases sont initialement blanches (cases libres).
10. Écrire sa méthode `dessine`. Les méthodes qui suivent prennent une pièce en paramètre, que nous appelons p.
11. La méthode `contient`, à implémenter, indique si p ne déborde pas du plateau.
12. La méthode `verifie` indique si p n'occupe que des cases libres (blanches).
13. La méthode `positions` compte le nombre d'emplacements possibles de p sans recouvrement.
14. La méthode `absorbe` "fige" p en copiant sa couleur dans les cases du plateau qu'elle occupe.
15. La fonction `jeu` est une première version qui affiche une nouvelle pièce, laisse le joueur la déplacer avec les flèches du clavier, la fige (si l'emplacement est libre) lors de l'appui sur la touche "Entrée" et passe à la pièce suivante. Le jeu s'arrête lorsque la nouvelle pièce n'a aucun emplacement libre possible.

1.4 Jeu final

16. Écrire une méthode `Plateau::detruis` qui compte les lignes et colonnes remplies, les détruit (remises à blanc) et renvoie le nombre effectif de cases détruites (attention, ne pas compter double une case participant au remplissage d'une ligne et d'une colonne).
17. Incorporer cette fonctionnalité dans le jeu.
18. Tenir le compte et afficher le score (nombre de cases détruites) dans le jeu.
19. Indiquer par un disque vert ou rouge si l'emplacement courant de la pièce est libre ou pas.