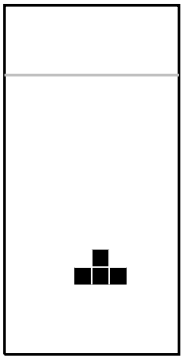
Projet – Tétris

Ce projet est à faire en binôme et à remettre le lundi 6 janvier 2010 durant le cours.

1. Présentation

Tétris est un jeu de type puzzle où les pièces sont des **tétraminos** (un assemblage de 4 carrés élémentaires). Les tétraminos, tirés aléatoirement, tombent 1 à 1 dans le **puits**.



Le joueur peut déplacer le tétramino tombant vers la gauche ou vers la droite avec les flèches correspondantes du clavier ou lui faire faire une rotation avec la flèche du haut. S'il le souhaite, il peut accelérer la chute du tétramino avec la flèche du bas. Une fois que le tétramino ne peut plus tomber, le joueur peut le déplacer latéralement ou lui faire faire des rotations avant que celui-ci ne soit définitivement **verrouillé**. Si une ligne complète du puits est recouverte par des morceaux de tétraminos, elle est retirée, le joueur marque des points et tout ce qui est au-dessus de cette ligne dans le puits tombe.

La partie est perdue lorsque un tétramino est verrouillé complètement au-dessus de la ligne de vie. Plus le joueur complète de lignes, plus son score, son niveau et la vitesse de chute augmentent.

Le but de ce TP est de coder un jeu tétris fonctionnel. Un squelette de code contenant la gestion de l'interface graphique, la gestion des évènements, les différentes structures et la trame du code que vous aurez à écrire vous est fourni sur mon site au LMPA.

Télécharger et décompresser le squelette du projet correspondant à votre OS.

2. Structures

Notre version du tétris sera basée sur 3 structures : `Puits`, `Tetramino` et `Tetris`. Celles-ci sont définies et partiellement initialisées dans le fichier `tetris.h`.

Il y a 7 formes différentes de tétramino, numérotées de 1 à 7. Chaque tétramino est composé d'un carré central (dessiner en noir sur le tableau suivant) et de 3 carrés périphériques. Le tableau suivant illustre les 7 formes de tétramino ainsi que leurs rotations possibles. Le nom des tétraminos n'est pas utilisé dans le programme. La structure `Tetramino` est alors composée de

- `forme` : la forme du tétramino (de 1 à 7) ;
- `rotation` : pour l'angle de rotation, multiple de 90° (initialisé à `0`) ;
- `x,y` : les coordonnées du centre du tétramino (initialisé à `(0,0)`) ;
- `x1,y1,x2,y2,x3,y3` : les coordonnées relatives (au centre) des 3 autres carrés.

forme	nom	rotation=0	rotation=1	rotation=2	rotation=3
1	I				
2	J				
3	L				
4	O				
5	S				
6	T				
7	Z				

La structure `Puits` représente un ensemble de carrées et est composée de
- `hauteur` : hauteur totale du puits, en nombre de carrés (initialisé à `28`) ;
- `largeur` : largeur totale du puits, en nombre de carrés (initialisé à `10`) ;
- `grille` : les valeurs des carrés du puits.

Le carré à la position (x, y) de la grille vaut `0` s'il est vide ou vaut le numéro de la forme du tétramino ayant déposé le carré. Pour accéder aux valeurs de la grille on a les fonctions :
- `lire_position(Puits p,int x,int y)` retourne la valeur à la position `(x,y)` du puits `p`.
- `ecrire_position(Puits p,int x,int y,int v)` assigne la valeur `v` à la position `(x,y)` du puits `p`.

La structure `Tetris` est composée de
- `couleurs` un tableau de 8 couleurs. Celle d'indice 0 est noire, celles d'indice $i > 0$ correspond au tétramino de forme i ;
- `puits` : le puits du tétris ;
- `tetramino_courant` : le tétramino en train de tomber ;
- `prochain_tetramino` : le prochain tétramino à tomber ;
- `dx`, `dy` : la position du point en bas à gauche pour le dessin du puits (initialisés à `(10,10)`) ;
- `taille_carre` : la taille des carrés à dessiner (initialisé à `20`) ;
- `ligne_vie` : position de la ligne de vie dans le puits (initialisé à `22`) ;
- `lignes_terminees` : nombre de lignes terminées (initialisé à `0`) ;
- `niveau` : niveau du joueur, de 1 à 20 ;
- `score` : score du joueur (initialisé à `0`) ;
- `game_over` : vaut `true` si le jeu est perdu (initialisé à `false`).

3. Ce qu'il faut faire

Exercice 1. Modifier la fonction `affichage_tetris(Tetris& tetris)` pour que celle-ci affiche le contour du puits `tetris.puits` en blanc et la ligne de vie en gris à l'aide des fonctions `dessine_ligne(int x1,int y1,int x2,int y2)` et `defini_couleur_dessin(Couleur c)`. Les variables `blanc` et `gris` de type `Couleur` sont déjà définies.

Exercice 2. Modifier la fonction `initialisation_tetramino(Tetramino& t,int f)` qui initialise le tétramino `t` à la forme d'indice `f` pour une rotation de `0`.

Exercice 3. Terminer la fonction `affichage_tetramino(Tetris& tetris, Tetramino& t)` permettant d'afficher un tétramino dans le puits. Vous pourrez tester vos fonctions de création et d'affichage de tétramino en dé-commentant la partie correspondante dans `affichage_tetris`. N'oubliez pas de re-commenter le code-test avant de poursuivre.

Exercice 4. La structure `Tetris` comporte 2 éléments de type `Tetramino` : `tetramino_courant` et `prochain_tetramino`. Le premier est le tétramino tombant dans le puits et le deuxième est le prochain à tomber. Lors de la création d'un nouveau tétramino on assigne `prochain_tetramino` à `tetramino_courant`. On modifie ensuite la position `(x,y)` de `tetramino_courant` pour qu'il soit affiché au milieu en haut du puits. On crée alors un nouveau tétramino de forme aléatoire dans `prochain_tetramino` et on modifie sa position `(x,y)` pour qu'il soit affiché en-dessous du mot `Suivant`. C'est le but de la fonction `nouveau_tetramino(Tetris& tetris)`.

Exercice 5. Modifier la fonction `deplace_tetramino(Tetramino& t,int mx,int my)` pour qu'elle retourne une copie du tétramino `t` translaté de `(mx,my)`.

Exercice 6. Modifier la fonction `teste_position(Tetris& tetris,int x,int y)` pour qu'elle retourne `true` si la case à la position `(x,y)` est une case vide du puits `tetris.puits`, et non une case occupée ou à l'extérieur du puits.

Exercice 7. Modifier la fonction `teste_tetramino(Tetris& tetris,Tetramino& t)` qui retourne `true` si le tétramino `t` est entièrement à l'intérieur du puits `tetris.puits` et si sa place est libre.

Exercice 8. Modifier la fonction `evenement_deplacement(Tetris& tetris,int direction)` qui déplace le tétramino `tetris.tetramino_courant` d'une case vers la gauche (`direction=-1`) ou vers la droite (`direction=1`) si possible. On utilisera les fonctions `deplace_tetramino` et `teste_tetramino`.

Exercice 9. Modifier la fonction `evenement_chute(Tetris& tetris)` qui fait tomber le tétramino `tetris.tetramino_courant` d'une case si possible. La fonction retournera `true` si la chute a pu être réalisée et `false` sinon.

Exercice 10. Modifier la fonction `tourne_tetramino(Tetramino& t)` pour qu'elle retourne une copie du tétramino `t` après rotation de 90° dans le sens trigonométrique (voir tableau). Les tétraminos `I` et `O` seront traités séparément. Pour les autres on pourra utiliser la forme matricielle d'une rotation d'angle α dans le repère standard :

$$\begin{bmatrix} \cos(\alpha) & -\sin(\alpha) \\ \sin(\alpha) & \cos(\alpha) \end{bmatrix}$$

Exercice 11. Modifier la fonction `evenement_rotation(Tetris& tetris)` qui fait tourner le tétramino `tetris.tetramino_courant` si possible.

Exercice 12. Modifier la fonction `verouillage_tetramino(Tetris& tetris)` qui verouille le tétramino `tetris.tetramino_courant` dans la grille : les carrés de la grille recouverts par le tétramino prennent la valeur de la forme de celui-ci.

Exercice 13. Modifier la fonction `verouillage_tetramino(Tetris& tetris)` qui teste si le tétramino `tetris.tetramino_courant` est entièrement au-dessus de la ligne de vie. On commencera par déterminer la plus petite ordonnée des carrés du tétramino, qu'on stockera dans `ymin`. Si `ymin` est plus grand que `tetris.ligne_vie` on arme `tetris.game_over` à `true` et on quitte la fonction. Sinon on crée un nouveau tétramino à l'aide de la fonction `nouveau_tetramino`.

Exercice 14. Modifier la fonction `verouillage_tetramino(Tetris& tetris)` pour qu'elle supprime et détermine le nombre de lignes finies, qu'on stockera dans `nombre_lignes_finies`.

Exercice 15. Modifier la fonction `verouillage_tetramino(Tetris& tetris)` pour qu'elle mette à jour `tetris.lignes_terminees` comptant le nombre de lignes terminées depuis le début de la partie, `tetris.niveau` pour le niveau du joueur et `tetris.score` pour le score du joueur.

Le niveau du joueur augmente de 1 si le joueur a terminé plus de $10 \times$ `niveau` lignes, le niveau maximal étant de 20. L'augmentation du score dépend du nombre `nombre_lignes_finies` de lignes finies en verouillant le dernier tétramino (entre 0 et 4), de la hauteur du tétramino `ymin` et du `niveau` du joueur suivant par la formule suivante :

$$\texttt{ajout score} = (\texttt{tetris.niveau} + \texttt{ymin}) \times \texttt{multiple}$$

où `multiple` dépend de `nombre_lignes_finies` :

`nombre_lignes_finies`	0	1	2	3	4
`multiple`	0	100	300	500	800