PPOO - Tetris (partie 4)

Pierre-Yves Derbaix et Raphael Javaux

La commande pour exécuter le programme est : `ant run`

Améliorations par rapport à l'étape précédente

Pièces différentes dans les modes multi-joueurs

La génération des pièces est à présent différente pour chaque joueur des modes multi-joueurs, à l'exception du mode classique.
Nous utilisons simplement une « seed » distincte pour initialiser les générateurs de nombres aléatoires de chaque joueur.

Trou des lignes de pénalités dans le mode multi classique

Le l'emplacement du trou pour les lignes de pénalités dans le mode multijoueur classique est maintenant déterminé au début de chaque partie et non de manière aléatoire à chaque ligne à ajouter.

GUI recréée manuellement

Lors de l'étape précédente, par manque de temps, nous avions généré la vue du menu à l'aide de NetBeans. La vue a maintenant été recréée à la main.

Game over pour tous les joueurs à la fin d'une partie

Lorsqu'une partie se termine, le jeu de chaque joueur s'arrête, un gagnant est désigné et une boite de dialogue s'affiche pour proposer de réessayer ou de retourner au menu principal.

Gestion des touches simultanées

Les touches sont maintenant gérées de manière simultanées. Lorsqu'une touche est enfoncée, celle-ci est ajoutée à un ensemble de touche actives (`activeKeys`). Ensuite, il suffit de vérifier pour chaque joueur que que des touches qui leur sont associés ne sont pas présentes dans cet ensemble. Si c'est le cas, il suffit d'appeler la méthode correspondante du `GameController`.

Vue et gestion du clavier génériques par rapport au nombre de joueurs

Tout d'abord, nous avons refactorisé les contrôleurs multi-joueurs du jeu (sous-classes de `MultiGamePlay`) pour gérer le le déroulement du jeu de manière indépendante du nombre de joueurs.

La vue `TwoPlayersSwingView` a été renommée en `MultiPlayerSwingView` . Celle-ci est construite à l'aide d'une liste de `GamePlay` (instanciés par `MultiGamePlay`). Le nombre correspondant de `GamePanel`.

Concernant la gestion du clavier, deux classes ont été créées : `KeySet` et `KeyboardHandler`.

La classe `KeySet` représente les touches associées à chaque action dans le jeu (déplacements latéraux, rotation, « soft drop » et « hard drop »). Chaque touche est représentée par un entier qui correspond à son code clavier (voir `KeyEvent`).

La classe `KeyboardHandler`, qui comprend un `KeySet`, effectue le mapping entre l'ensemble des touches actuellement enfoncés et le (ou les) `GameController`(s) associés à chaque joueur.

La classe `SinglePlayerSwingView` et la `MultiPlayerSwingView` possèdent chacun, respectivement, un `KeyboardHandler` ou une liste de `KeyboardHandlers`. Lorsqu'une touche est enfoncée, celle-ci est ajoutée à l'ensemble des touches actives et chaque `KeyboardHandler` vérifie si les touches qui lui sont attribuées ne sont pas présentes dans cet ensemble. Si c'est le cas, la ou les méthodes adéquate de chaque `GameController` sont appelées.

Une amélioration possible serait de créer un thread qui « monitorerait » à intervale régulier les "activeKeys". Ceci permettrait de ne plus dépendre uniquement sur les évènements générées par le clavier pour faire appel aux méthodes des `GameController`.

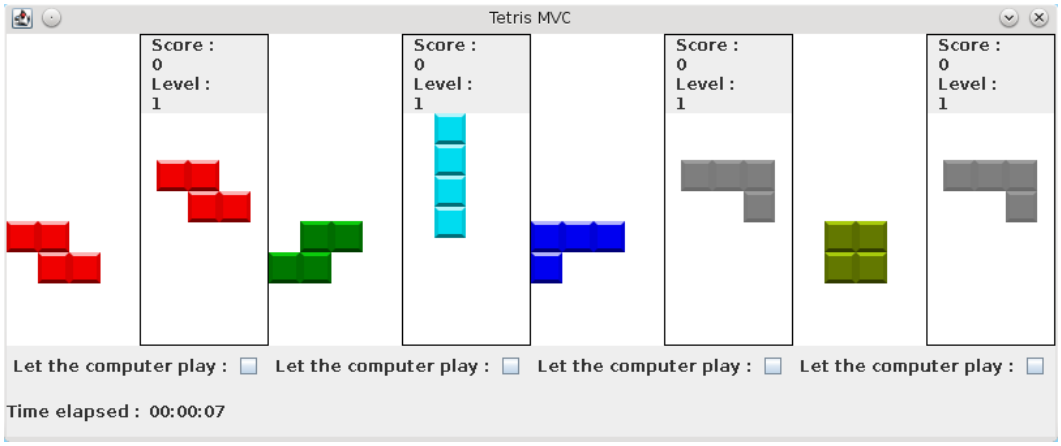
Taille dynamique de la grille

Nous avons tout d'abord créé une classe `Configuration` qui retient la hauteur et la largeur de la grille. Elle comprend également si les pièces sont dessinées à l'aide d'images, le nombre de joueurs en mode multijoueur ainsi que les `KeySet` de chaque joueur.

Un bouton "Options" a été ajouté au menu principal. Une pression sur celui-ci déclenche l'ouverture d'une `OptionsView`. Cette vue permet de configurer la taille de la grille, le nombre de joueurs en mode multi ainsi que les touches associées à ceux-ci.

Les touches de chaque joueur sont configurées dans une autre vue : `KeySetDialog`. Cette boite de dialogue contient un "text field" par touche. Il suffit de cliquer dans un de ces "text field" et d'appuyer sur une touche pour associer cette touche à l'action correspondante.

Par exemple, pour une grille de 4 sur 10 et 4 joueurs en mode multi, nous obtenons le résultat suivant :



Dans le menu "options", le nombre de joueurs maximum a été limité à 4, car, au delà, il devient difficile de trouver les touches à assigner a tous les joueurs. Les valeurs pour la hauteur et la largeur de la grille sont également limitées.

Joueur automatique

Le joueur automatique est implémenté par la classe `ai.ArtificialIntelligence`.
La classe est un listener du plateau de jeu (comme les vues). Le choix et le déplacement de la pièce se fait lorsque la grille introduit une nouvelle pièce dans le jeu.

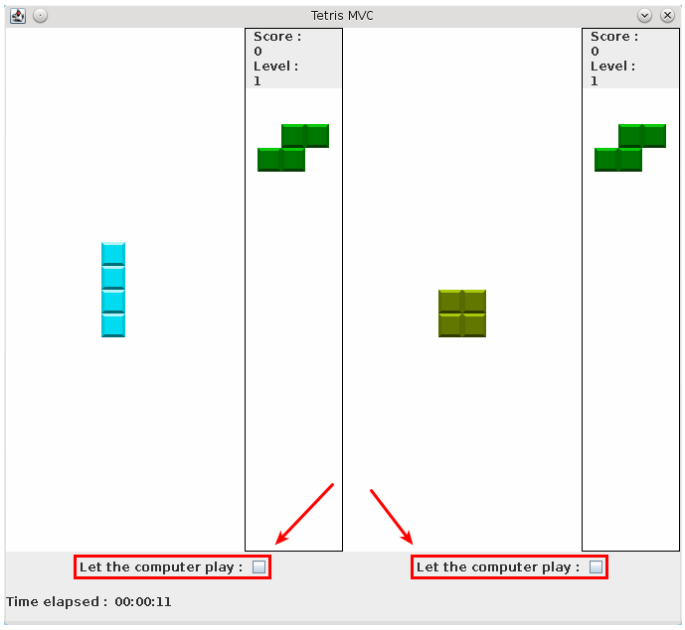
La stratégie, assez simple, consiste à tester les différentes colonnes avec différentes rotations. L'intelligence artificielle n'est pas capable de gérer le « glissement » des pièces, c'est à dire d'aller placer une pièce sous une autre.

L'heuristique pour le décision de la colonne et de la rotation est faite à partir du choix minimisant la combinaison linéaire de trois critères suivante :

$$\begin{aligned} pénalité \;=\; \alpha \times nombre\ de\ cellule\ vides\ bloquees \;-\; \beta \times nombre\ de\ lignes\ effacees\ par\ la\ piece \\ +\gamma \,/\, \sqrt{distance\ du\ sommet\ de\ la\ piece\ par\ rapport\ a\ la\ bordure\ supérieure} \end{aligned}$$

Le premier terme concerne le nombre de cellules vides qui ne seront plus « remplissables » sans avoir au préalable à supprimer au moins une ligne.
Le second terme prend en compte le nombre de lignes qui seront effacées par la pièce.
Le dernier terme tente de placer les pièces le plus bas possible dans la grille. La racine carrée permet de donner plus d'importance à ce critère lorsque la grille commence à être fortement remplie (lorsque la distance entre le sommet de la pièce et de la bordure supérieure du jeu devient faible).

Les trois paramètres α, β et γ ont été fixés à 1, 1 et 4, respectivement. Il serait intéressant d'écrire un programme testant différentes valeurs de ces paramètres afin d'optimiser ceux-ci pour les performances de l'IA.



Interface

Au niveau de la vue, nous avons rajouté au `GamePanel` une checkbox pour permettre d'activer/désactiver le joueur automatique à n'importe quel moment durant une partie. Le joueur automatique est disponible pour l'ensemble des modes de jeu.

Une fois la checkbox sélectionnée, le joueur automatique prend la main et le `KeyboardHandler` est désactivé afin d'empêcher l'utilisateur de "gêner" l'IA. Le joueur peut reprendre la main à n'importe quel moment de la partie.