Tetris

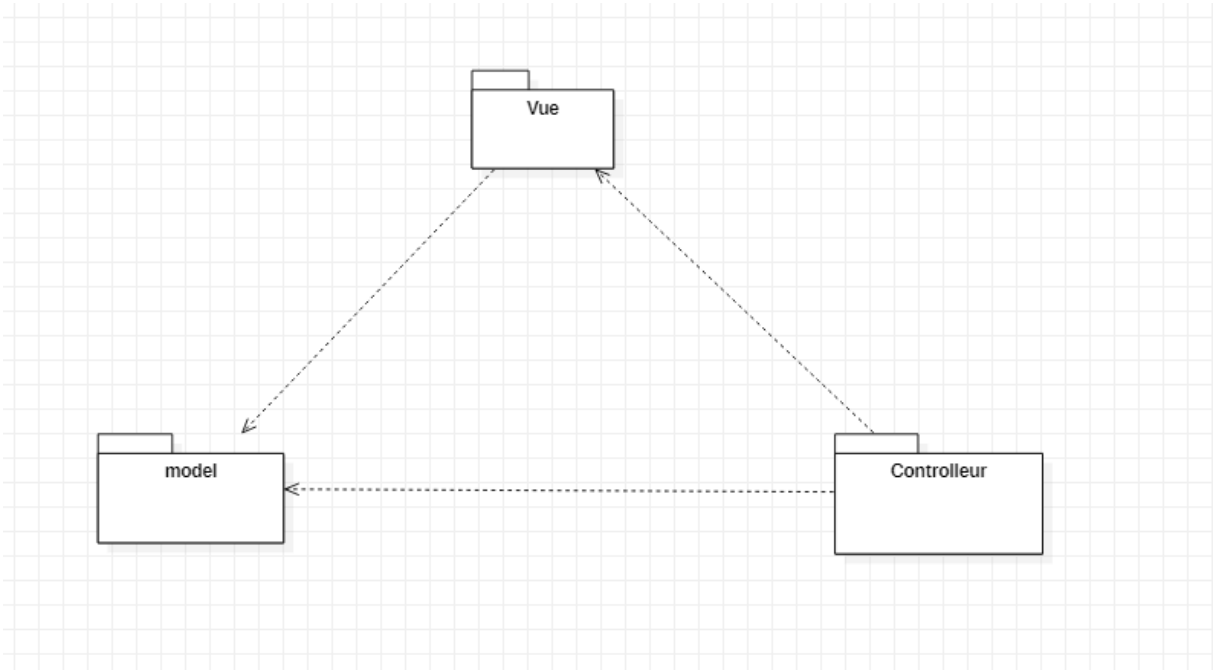
Youssef El Ouahabi et Arnold Oumbe

Qu’est-ce que « Tetris » ?

Tetris est un jeu de puzzle où les joueurs empilent des blocs géométriques appelés tétriminos qui tombent à des vitesses croissantes. Le but est de compléter des lignes horizontales sans laisser d'espaces vides. Lorsqu'une ligne est complétée, elle disparaît, libérant de l'espace et marquant des points. Le jeu se termine lorsque les blocs s'empilent jusqu'au haut de l'écran et qu'aucun autre bloc ne peut tomber.

Modélisation.

Pour notre modélisation, nous avons choisi le design pattern MVC, qui divise l'application en trois composants interconnectés : le Modèle, la Vue, et le Contrôleur. Le Modèle gère les données et la logique métier du jeu, la Vue présente ces données à l'utilisateur avec une interface graphique, et le Contrôleur interprète les entrées de l'utilisateur, en les relayant au Modèle pour mettre à jour les données, et à la Vue pour actualiser l'affichage. Ce schéma favorise une conception modulaire et facilite la maintenance et l'évolution du logiciel.



Model :

Classe TetrisGame :

La classe sert de cœur central pour le jeu Tetris. Elle coordonne les règles du jeu, maintient l'état du jeu, et gère l'interaction entre les différents composants du modèle, la vue et les contrôles utilisateur.

Attributs :

board (Board): C'est l'instance qui représente le plateau de jeu où les pièces de Tetris tombent et sont positionnées. Il gère la grille qui est composée de cellules pouvant être remplies par des pièces de jeu.

isGameOver (boolean): Cet indicateur est essentiel pour déterminer le flux du jeu. Quand il passe à TRUE, cela signifie que le jeu doit s'arrêter, soit parce que le plateau est rempli et qu'aucune nouvelle pièce ne peut être placée, soit parce qu'une condition de victoire a été atteinte.

isGameWon (boolean): Cet indicateur spécifie si l'arrêt du jeu est dû à une victoire du joueur. Si le joueur atteint un score spécifique, complète un certain nombre de lignes, ou survit pendant un temps défini, alors ISGAMEWON devient TRUE.

level (Integer): Représente le niveau de difficulté actuel du jeu, qui augmente progressivement à mesure que le joueur efface des lignes. Un niveau plus élevé signifie généralement une vitesse de chute des pièces plus rapide, augmentant ainsi la difficulté du jeu.

Score (Integer): Le score est un chiffre qui augmente en fonction de la performance du joueur, par exemple lors de la suppression de lignes ou la réalisation de manœuvres difficiles.

linesCleared (Integer): Ce compteur suit le nombre total de lignes que le joueur a réussi à effacer pendant la partie. Il est souvent utilisé pour calculer le score et déterminer quand augmenter le niveau du jeu.

timer (Timer): Le timer orchestre la cadence du jeu. Il déclenche des événements à intervalles réguliers, comme la descente automatique de la pièce actuelle.

factory (Factory): La FACTORY est responsable de la création de nouvelles pièces de Tetris. Elle assure que les nouvelles pièces sont générées de façon aléatoire et introduites dans le jeu de manière équitable.

Méthodes :

TetrisGame(): Le constructeur initialise les composants essentiels du jeu, tels que le plateau, le score initial, et prépare le jeu à démarrer.

startGame(): Cette méthode lance toutes les activités nécessaires pour commencer une nouvelle partie de Tetris, y compris la génération de la première pièce et le démarrage du timer.

endGame(): Elle est appelée pour terminer la partie en cours. Cela peut être dû à une défaite du joueur ou à la réalisation d'une condition de victoire.

checkVictoryConditions(): Cette méthode vérifie après chaque action importante du jeu si le joueur a atteint une condition de victoire. Si c'est le cas, elle met à jour l'état du jeu en conséquence.

updateScore(): Elle calcule le score en fonction des lignes effacées ou d'autres facteurs de score définis par le jeu. Elle met à jour l'attribut score avec la nouvelle valeur.

increaseLevel(): Elle augmente le niveau de jeu, ce qui est généralement associé à une augmentation de la vitesse de descente des pièces, rendant le jeu plus difficile.

moveCurrentPieceDown(): Elle déplace la pièce actuelle vers le bas. Si le mouvement est impossible (en raison de l'atteinte du fond du plateau ou d'un blocage par des pièces déjà placées), la pièce est fixée en place et une nouvelle pièce est générée.

generateNewPiece(): Elle demande à la FACTORY de créer une nouvelle pièce de Tetris, qui est ensuite placée en haut du plateau de jeu. Si la pièce ne peut pas être placée, cela signifie généralement que le jeu est terminé.

movePiecePosition(): Cette méthode déplace la pièce sur le plateau de jeu.

getPossibleMoves: Cette méthode renvoie la liste de position possible pour une pièce pour un déplacement s'effectue dans une direction.

Classe Board :

La classe représente le plateau de jeu dans Tetris, où toutes les actions de placement et de manipulation des pièces. Cette classe est continuellement mise à jour à mesure que le joueur déplace et fait pivoter les pièces.

Attributs :

grid (Square[][]) : Un tableau bidimensionnel d'objets Square qui compose le plateau de jeu de Tetris. Chaque Square peut être vide ou rempli comme partie d'une pièce de Tetris. La grille est utilisée pour suivre les positions des pièces et vérifier les collisions et les lignes complètes.

currentPiece (Brick) : L'objet Brick actuel que le joueur contrôle. C'est la pièce qui est activement manipulée et peut être déplacée ou tournée jusqu'à ce qu'elle soit verrouillée en place sur la grille.

row (integer) : Un entier représentant la ligne actuelle où se trouve la currentPiece. Cela est souvent utilisé pour manipuler la position verticale de la pièce lors des déplacements.

column (integer) : Un entier représentant la colonne actuelle où se trouve la currentPiece, utilisé pour les déplacements horizontaux de la pièce sur la grille.

Méthodes :

Board() : Le constructeur de la classe Board qui initialise le plateau de jeu. Il crée la grille et définit les valeurs initiales pour la position actuelle (row et column).

getCurrentBrick() : Brick - Renvoie une référence à la currentPiece, permettant aux autres composants du jeu, tels que le contrôleur, de savoir quelle pièce est actuellement en jeu.

setCurrentBrick() : void - Définit la currentPiece sur le plateau de jeu. Cette méthode est utilisée pour placer une nouvelle pièce sur le plateau lorsque la pièce précédente a été verrouillée en place.

contains(pos: Position) : boolean - Vérifie si la Position spécifiée contient une partie d'une pièce (ou un Square non vide). Elle est utilisée pour la détection des collisions lors du déplacement des pièces.

isFree(pos: Position) : boolean - Détermine si une position donnée sur la grille est libre, c'est-à-dire qu'elle ne contient pas une partie d'une brique déjà placée. Cette méthode est cruciale pour valider les mouvements ou les rotations de la currentPiece.

ClearLines(): void - Cette méthode parcourt la grille du jeu pour identifier et supprimer toutes les lignes complètes.

Classe Square :

La classe Square est conçu pour gérer l'état individuel de chaque case sur le plateau de jeu de Tetris.

Méthodes :

Square() : Le constructeur de la classe Square qui initialise un carré du plateau de jeu de Tetris.

getBrick() : Brick - Renvoie la référence à la brique (Brick) que ce carré contient. Si le carré est vide.

setBrick(brick: Brick) : void - Assigne une brique à ce carré. Si une brique est passée à cette méthode, le carré est considéré comme rempli. Si null est passé, le carré est considéré comme vide.

isFree() : boolean - Renvoie un booléen indiquant si le carré est libre (c'est-à-dire ne contenant pas de brique) ou non.

Classe Brick :

La classe Brick dans jeu Tetris est une représentation d'une pièce de Tetris.

Attributs :

squares (List<Square>) : Une liste de Square qui compose cette brique. Dans Tetris, une brique est composée de 4 carrés. Chaque Square peut être occupé ou non, indiquant la forme de la brique.

Méthodes :

rotateClockwise()void - Fait pivoter la brique de 90 degrés dans le sens horaire.

rotateNoClockwise()void - Fait pivoter la brique de 90 degrés dans le sens anti-horaire.

Brick(type: String) : Le constructeur de la classe Brick qui initialise une nouvelle brique avec le type spécifié.

getType(): String - Renvoie le type de la brique, ce qui est utile pour déterminer la forme de la brique lors du dessin sur le plateau de jeu ou pour d'autres logiques de jeu.

Classe Factory :

La classe dans le contexte d'un jeu Tetris est responsable de la génération des pièces de Tetris, qui sont utilisées par le joueur pendant la partie.

Attributs :

defaultBag (List<BrickFactory >): Un sac par défaut contenant les types de briques disponibles dans le jeu Tetris, représentés par l'énumération BrickFactory. Chaque valeur de l'énumération représente un type distinct de brique (par exemple, I, O, T, S, Z, J, L).

customBag (List< BrickFactory >): Un sac personnalisé qui peut être configuré par le joueur ou le développeur pour inclure différentes formes de briques ou des fréquences différentes pour certaines briques, permettant une personnalisation plus poussée du jeu.

currentBag (List< BrickFactory >): Le sac actuel utilisé pour générer les nouvelles briques. Il est mélangé pour s'assurer que la distribution des briques est équitable et imprévisible.

Méthodes :

CreatePiece(): Brick - Génère une nouvelle brique en sélectionnant aléatoirement parmi les briques disponibles dans le currentBag. Une fois qu'une brique est sélectionnée, elle est retirée de ce sac, et lorsque le sac est vide, il est remélangé.

Factory(customBag: List<BrickFactory>): Le constructeur de la classe Factory qui permet de créer une Factory avec un sac personnalisé de briques si fourni, sinon le sac par défaut est utilisé.

shuffleBag(): void - Mélange le sac de briques pour changer l'ordre de distribution des briques. C'est une étape cruciale pour garantir la justesse du jeu en évitant les séquences prévisibles de briques.

initializeBags(): void - Initialise les sacs de briques. Si un sac personnalisé est fourni, il est utilisé pour remplir le currentBag ; sinon, le defaultBag est utilisé. Cette méthode prépare également la Factory pour la génération des briques en mélangeant le currentBag.

Classe : Position :

Attributs :

row (integer) : Un entier qui représente la ligne dans la grille de Tetris. C'est l'indice vertical qui détermine la position d'une brique ou d'un carré sur le plateau de jeu.

column (integer) : Un entier qui représente la colonne dans la grille de Tetris. C'est l'indice horizontal pour la position d'une brique ou d'un carré sur le plateau.

Méthodes :

Position(row: integer, column: integer) : Le constructeur de la classe Position qui initialise une nouvelle Position avec les indices de ligne et de colonne spécifiés.

getRow(): integer - Renvoie l'indice de la ligne de la position. Cette méthode est utilisée pour obtenir l'indice vertical de la position d'une pièce ou d'un carré sur le plateau de jeu.

getColumn(): integer - Renvoie l'indice de la colonne de la position. Cette méthode est utilisée pour obtenir l'indice horizontal de la position d'une pièce ou d'un carré sur le plateau de jeu.

Next(Direction) - Cette méthode renvoie une liste de position contenant une position pour chaque case d'une pièce dans la direction indiqué.

Énumération Direction :

Énumeratuer : Nord(N) ;Ouest(O) ;Est(E)
Cette énumération représente les directions de mouvement des pièces.

Énumération BrickFactory :

Énumeratuer : T ;L ;I ;S;Z;J;O
Cette énumération représente les différents types de formes utilisés dans le jeu Tetris.

VIEW :

Classe View :

Attributs :

game: Une référence au modèle de jeu, utilisée pour interagir et obtenir des informations sur l'état actuel du jeu.

Méthodes :

View(model: TetrisGame) : Le constructeur de la classe View qui initialise la vue avec une référence au modèle de jeu TetrisGame.

DisplayBoard(): void - Affiche l'état actuel du plateau de jeu, y compris la position des pièces et les lignes complétées. Cette méthode peut impliquer la redessiner de la grille et la mise à jour des informations à l'écran.

showGameOver(): void - Affiche une notification ou un écran indiquant que le jeu est terminé, souvent après qu'un joueur a perdu la partie.

showVictory(): void - Affiche une notification ou un écran célébrant la victoire du joueur, déclenchée lorsque le joueur remplit une condition de victoire.

InputUser(): String - Recueille et renvoie les entrées de l'utilisateur, telles que les mouvements de pièce ou les commandes de menu. Cette méthode peut être utilisée pour écouter les entrées clavier ou les interactions avec l'interface utilisateur.

Controlleur :

Classe Controlleur :

Attributs :

game (TetrisGame) : Une référence au modèle du jeu, qui contient la logique de jeu et l'état actuel du jeu Tetris.

view (View) : Une référence à la vue, qui est chargée de l'affichage et de la présentation de l'interface utilisateur du jeu.

Méthodes :

Contrôleur(game: TetrisGame, view: View) : Le constructeur de la classe Contrôleur qui initialise le contrôleur avec des références au modèle et à la vue du jeu Tetris.

play(): void - Démarre le jeu et initialise la boucle principale du jeu. C'est dans cette méthode que les entrées de l'utilisateur sont traitées et que les mises à jour de l'état du jeu sont effectuées, en fonction des actions du joueur.