PROJET ENCADRE INFORMATIQUE

**Fonctionnalités librairie :**

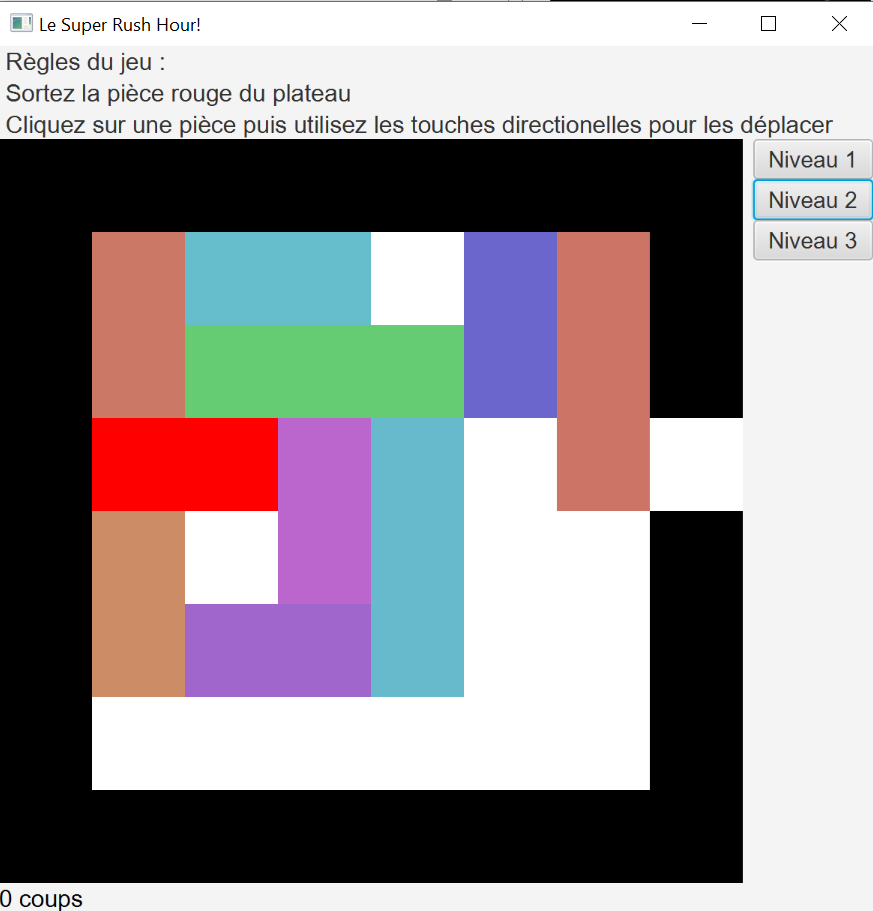
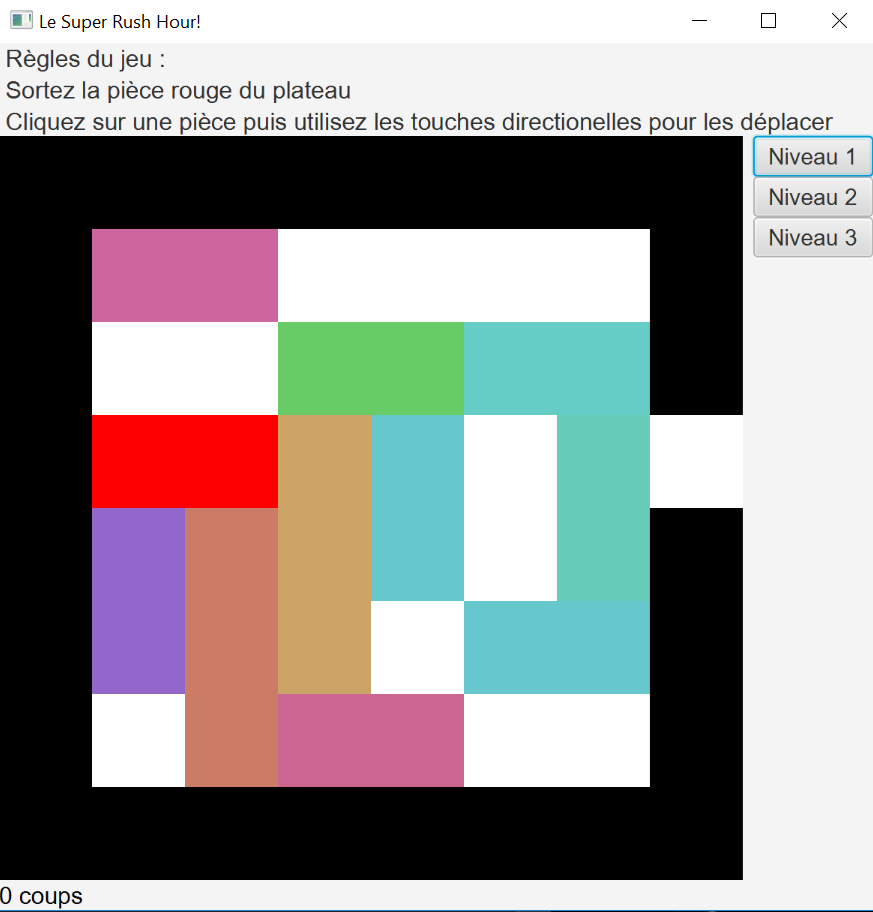
Toutes les fonctions publiques du modèle de la librairie sont les fonctionnalités de la librairie :

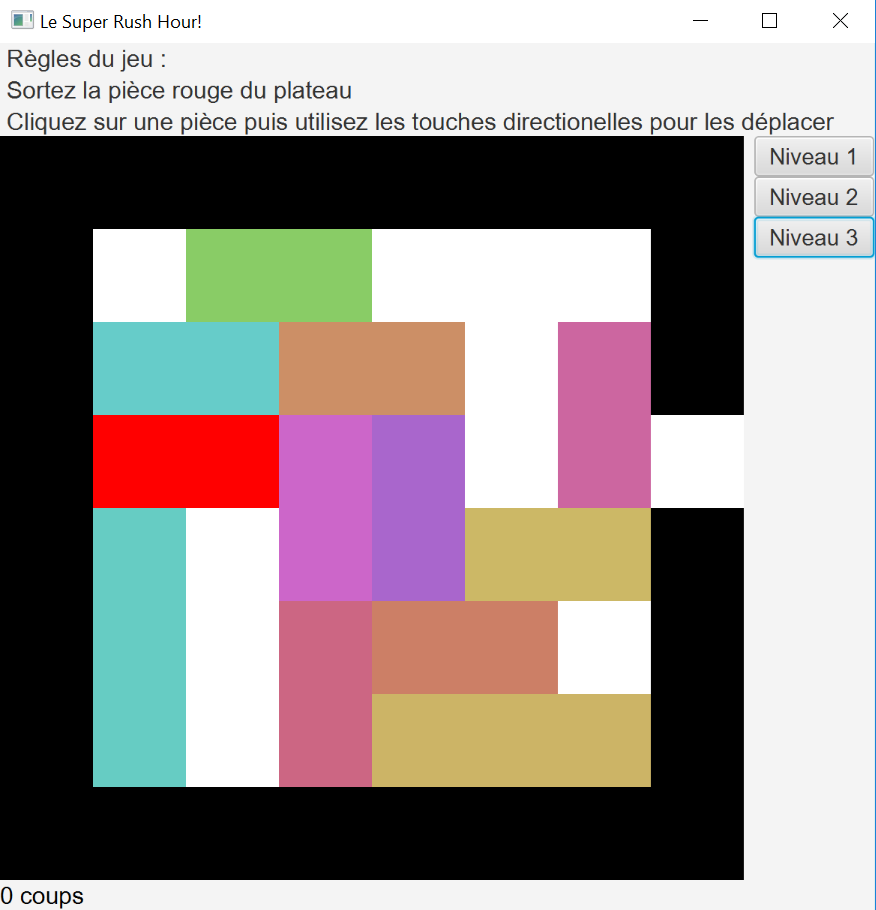
* Création du modèle
* Pose de Piece : différentes manières de l’utiliser (en spécifiant ou non la couleur ou les déplacements possibles de la pièce
* Sélection d’une pièce aux coordonnées spécifiées ou la dernière pièce crées.
* Déplacer une pièce
* Pivoter pièce
* Récupérer la couleur d’une pièce.
* Changer les déplacements autorisés d’une pièce
* Vider une case du plateau (supprimer la case de la pièce s’il y en a une)
* Supprimer une ligne du plateau et faire descendre les pièces au-dessus (pour le Tetris)

**RUSH HOUR :**

Présentation :

L’application propose au joueur différents niveaux de difficulté du jeu. Le principe reste le même, à savoir déplacer les différentes pièces du plateau afin de positionner l’unique pièce rouge en dehors du plateau.

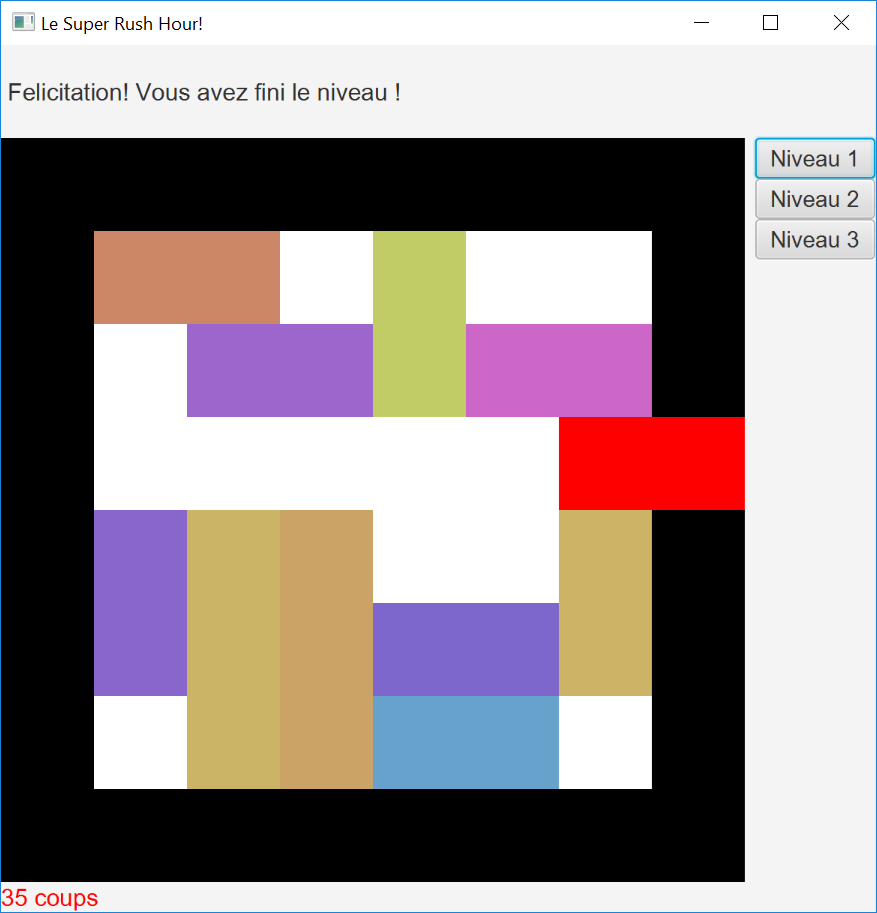
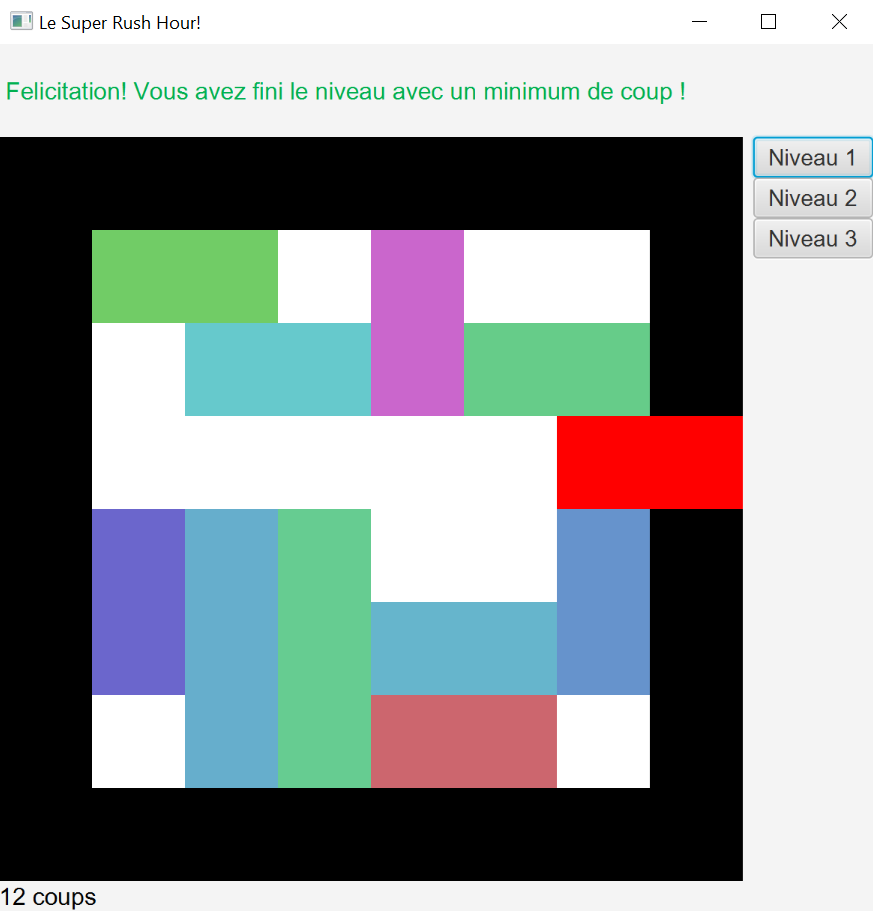




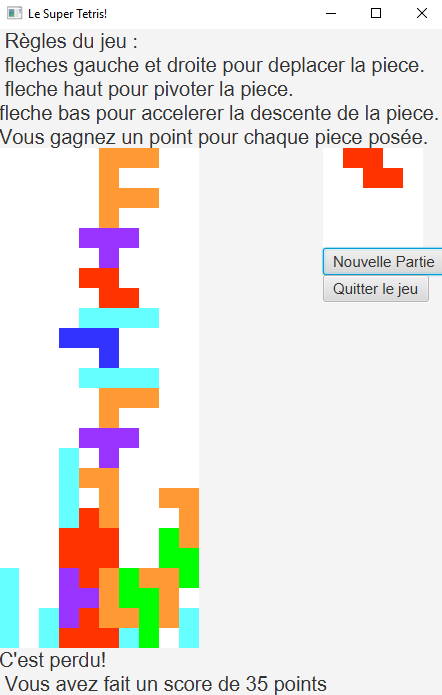
Les pièces ne se déplacent qu’horizontalement ou verticalement. Le nombre de déplacement effectué par le joueur est comptabilisé pour pouvoir lui fournir un score. Si le joueur dépasse le nombre de coup minimum pour avoir le score maximal (spécifique à chaque niveau) celui-ci apparait en rouge.



Suivant le nombre de déplacement fait par le joueur, un message de fin de partie différent apparait.



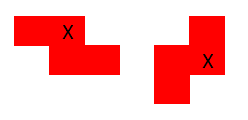
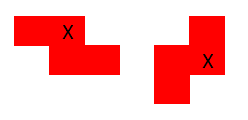
**TETRIS :**



Présentation :

Lorsque le joueur lance une nouvelle partie, une pièce apparait en haut du plateau de jeu. Celle-ci descend automatiquement, et peut être déplacée à gauche, à droite et en bas pour la faire descendre plus rapidement grâce aux flèches directionnelles. Cependant la pièce ne peut pas être remontée, si le joueur appuie sur la flèche du haut la pièce pivote dans le sens anti-horaire autours d’une case pivot (représentée ici par un x). Voici un exemple :

Forme initiale : Forme pivotée :

Lorsque la pièce rentre en collision avec une autre pièce déjà positionnée ou qu’elle atteint le bas du plateau celle-ci se fige et ne peut plus être déplacée et une autre apparait aléatoirement.

A chaque fois qu’une pièce se fige le joueur gagne un point de score.

Quand une pièce se fige et qu’elle remplie une ligne entière alors cette ligne se supprime et fait descendre les cases supérieures.

**Justification analyse :**

**Justification du modèle :**

Une Case contient deux valeurs x et y correspond à des coordonnées en 2D.

Une pièce est représentée par une collection de Case, dont les coordonnées sont relatives à la case pivot, une case définie a la création de la pièce.

Un plateau contient des attributs définissant sa taille et une collection de Pieces.

Le choix de la classe Case a été fait pour faciliter la suppression de case d’une pièce et aussi le pivotement (grâce aux coordonnées relatives au centre de rotation)

Le Plateau se charge de faire les vérifications de collision des pièces. Si un déplacement ou un pivotement n’est pas possible alors une Exception est retournée.

**Justification du modèle MVC :**

Dans la libraire les communications entre la vu et le modèle respectent bien le modèle MVC, par le biais des Observer.

La librairie n’a pas de contrôleurs par défaut mais la vue, a sa création, initialise un gridPane qui contient les différentes cases du plateau. Ce gridPane peut ainsi être récupéré par la vue d’un jeu afin de définir des actions a exécuter lors d’un clic de souris sur une case.

Ainsi la communication entre nos jeux et la librairie ne respecte pas un modèle MVC strict puisqu’il y a contact direct entre la vue du jeu et la vue de la librairie. Ce compromis a été fait car JavaFx n’est pas pensée pour un modèle MVC strict et donc il aurait été difficile à implémenter.

Dans nos jeux, la vue et les contrôleurs sont réunis dans le même fichier.

**Documentation UML :**

Librairie :

Rush hour hour

(tetris)