Tetris

Andrew	Constantin	Gregory	Thomas
54327	54439	54786	54341

Introduction

Nous avons implémenté une version à deux joueurs du jeu Tetris. Pour ce faire nous nous sommes basés sur un modèle simplifié du jeu de base, dans lequel nous avons ajouté un système de **client/serveur.** Le résultat est un jeu ou 2 joueurs peuvent s'affronter dans une partie de Tetris.

Description du jeu

Généralité

Tetris est principalement composé d'un champ de jeu où des pièces de formes différentes, appelées « tétrimino », descendent du haut de l'écran. Durant cette descente le joueur peut uniquement déplacer les pièces latéralement et leur faire effectuer une rotation sur elles-mêmes jusqu'à ce qu'elles touchent le bas du champ de jeu ou une pièce déjà placée. Le joueur ne peut ni ralentir la chute des pièces, ni l'empêcher, mais il peut dans certaines versions l'accélérer.

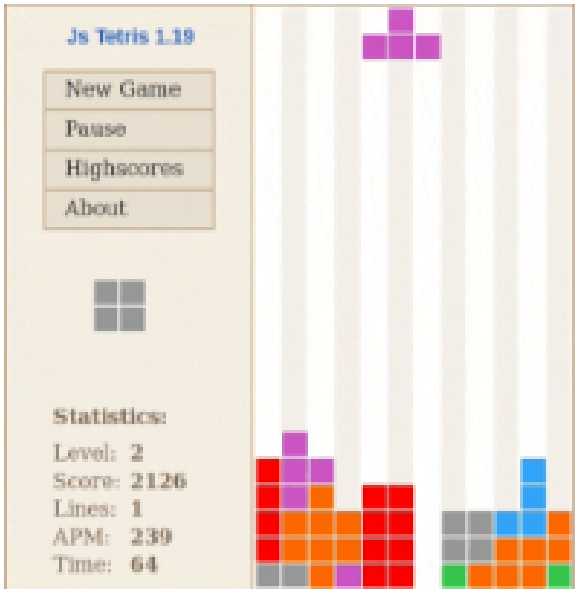


Illustration d’une pièce apparaissant

Illustration de fin de partie

Le but pour le joueur est de réaliser le plus de lignes possibles, afin de garder de l'espace pour placer les futures pièces. Une fois une ligne complétée, elle disparaît, et les blocs placés au-dessus chutent d'un rang. Si le joueur ne parvient pas à faire disparaître les lignes suffisamment vite, l'écran peut alors se remplir jusqu'en haut. Lorsqu'un tétrimino dépasse du champ de jeu, et empêche l'arrivée de tétrimino supplémentaires, la partie se termine. Le joueur obtient un score, qui dépend essentiellement du nombre de lignes réalisées lors de la partie.

Le jeu ne se termine donc jamais par la victoire du joueur. Avant de perdre, le joueur doit tenter de compléter un maximum de lignes. Pour ce faire, il peut éliminer plusieurs lignes d'un coup, ce qui lui rapporte des points de bonus dans la plupart des versions du jeu. Il est possible de compléter jusqu'à 4 lignes en même temps, grâce au tétrimino en I : ce coup est appelé un *Tetris*, comme le nom du jeu.

Au fur et à mesure que le joueur complète des lignes, son niveau augmente, accroissant le nombre de points réalisés par ligne complétée. Dans la plupart des versions sur jeu, la vitesse de chute des pièces augmente à chaque prise de niveau, laissant moins de temps au joueur pour réfléchir au placement. Le principal objectif pour le joueur consiste alors non pas à gagner, comme dans la plupart des jeux vidéo, mais à obtenir le plus grand score, via des combinaisons et de l’endurance. (Wikipédia)

Pièces

Les pièces de *Tetris*, sur lesquelles repose entièrement le jeu, sont des « tétriminos ». Il en existe sept formes différentes, toutes basées sur un assemblage de quatre carrés, d'où le mot « Tetris » du préfixe grec *tetra-* qui signifie « quatre ». Le joueur peut faire tourner plusieurs fois, à gauche et/ou à droite selon la version, de 90° n'importe quel bloc pour le poser de la façon désirée pendant qu'il descend. (Wikipédia)

Champ de jeu

Le champ de jeu, ou « matrice », est l'espace dans lequel tombent les tétriminos. Il dispose toujours d'une grille en arrière-plan, visible ou non, dont les cases sont de la même grandeur que les carrés des tétrimino, et que ceux-ci suivent dans leur chute. Ce champ de jeu est de dix cases de largeur et de vingt-deux cases de hauteur selon les *Tetris Guidelines*. (Wikipédia)

Chute des tétrimino

Dans le champ de jeu, les tétrimino chutent à partir du haut, avec une vitesse déterminée par le niveau de la partie. Plus le niveau est élevé, plus les pièces tombent vite, obligeant le joueur à placer de plus en plus vite les tétrimino. La plupart des versions accordent un bonus de points lorsque le joueur accélère la vitesse de la chute d'un bloc (appelé *soft drop*). Ce bonus est habituellement proportionnel au temps d'appui. De la même façon, le mouvement *hard drop permet* de poser directement le tétrimino dans la colonne et la position dans laquelle elle est lors de la chute. Cette action rapporte également un bonus de points dans la plupart des versions. (Wikipédia)

La formule utilisé pour calculer la vitesse de descente de Tetrimino est la suivante

$$(0.8 - ((level - 1) * 0.007))^{level-1}$$

Informations supplémentaires

Au-delà du champ de jeu, la majorité des versions de *Tetris* disposent de plusieurs informations contextuelles. L’élément le plus courant est l'affichage du nombre de lignes complétées depuis le début du jeu, le score (en fonction du nombre de lignes complétées et la vitesse de placement du joueur) et le niveau courant qui détermine la vitesse de chute des pièces.

Ce cadre d'information peut également indiquer le tétrimino suivant, permettant au joueur d'anticiper le coup suivant. Les versions récentes du jeu intègrent une réserve dans laquelle une pièce peut être stockée temporairement, affichée à côté du champ de jeu. (Wikipédia)

Action	Score	Description
Single	100 x level	1 ligne détruite
Double	300 x level	2 lignes détruites
Triple	500 x level	3 lignes détruites
Tetris	800 x level	4 lignes détruites
Soft Drop	1	Soft drop de une ligne
Hard Drop	2 x n	Hard drop de n lignes

Mode multijoueur

Certaines versions intègrent des modes de jeu à plusieurs. Tout comme dans la version solo, le but est de contenir les pièces dans le champ de jeu le plus longtemps possible. Le gagnant peut être déterminé selon plusieurs critères : nombres de points, de lignes réalisées, etc. Sur Game Boy, par exemple, le vainqueur est le premier à réaliser 30 lignes ou à faire perdre l'adversaire.

Les adversaires peuvent également voir l'avancée de l'un par rapport à l'autre. Sur consoles de salon, où tous les concurrents jouent sur un même écran, il est possible de voir le jeu complet de chaque joueur et donc de prévoir quand un adversaire s'apprête à faire un *Tetris*

Actions

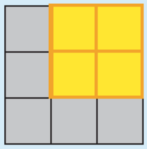
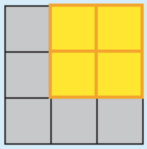
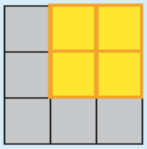
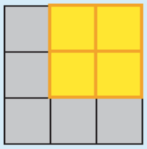
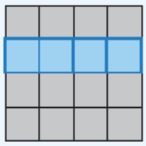
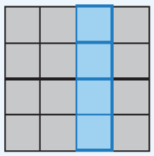
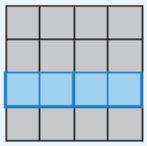
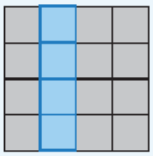
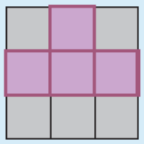
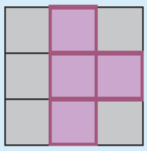
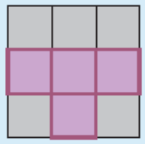
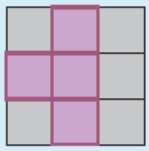
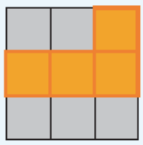
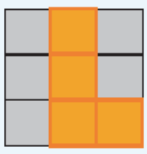
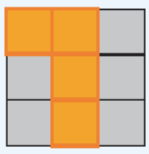
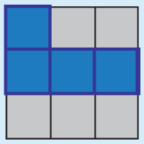
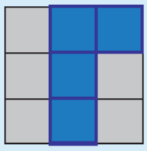
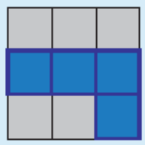
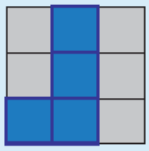
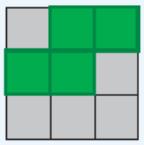
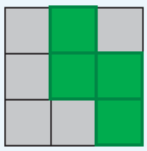
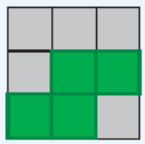
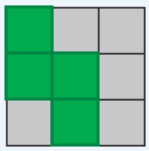
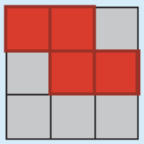
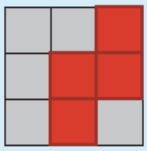
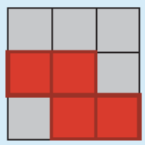
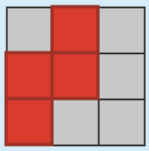
Voice les différentes actions qui peuvent être réaliser dans Tetris.

▼ **Déplacement :**

- **RIGHT** : translate le tétrimino vers la droite.
- **LEFT** : translate le tétrimino vers la gauche.
- **SOFTDROP** : accéléré la chute du tétrimino.
- **HARDROP** : pose directement le tétrimino dans la colonne et la position dans laquelle elle est lors de la chute.

▼ **ROTATE** :

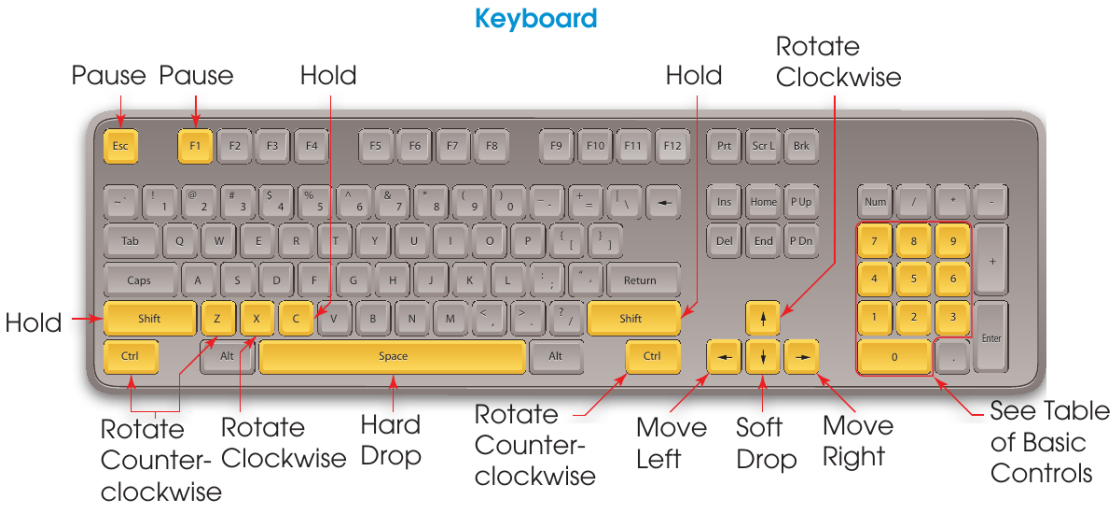
Fait tourner le tétriminos dans le sens horaire ou antihoraire.

Tetrimino	North	East	South	West
O-Tetrimino				
I-Tetrimino				
T-Tetrimino				
L-Tetrimino				
J-Tetrimino				
S-Tetrimino				
Z-Tetrimino				

▼ **HOLD** :

Permet de garder la pièce tombante en réserver. Cette action peut être réalisée une seule fois par chute c'est à dire que lorsqu'on a mis de côté une pièce il faut au moins placer une pièce pour à nouveau pouvoir utiliser cette action. Si une pièce est déjà en réserve, cette action va switcher la pièce tombante avec celle en réserve.

Bouton de contrôle



Voici les inputs qui sont acceptés par le jeu lier à l'action qu'ils réalisent. Ils sont pris en compte uniquement durant une partie de jeu.

- Déplacement « **LEFT** » : Flèche de gauche, 4 pavée numérique
- Déplacement « **RIGHT** » : Flèche de droite, 6 pavée numérique
- Déplacement « **SOFTDROP** » : Flèche du bas, 2 pavée numérique
- Déplacement « **HARDROP** » : Espace, 8 pavée numérique
- **ROTATE** « horaire » : Flèche du haut, X, 3 et 5 pavée numérique
- **ROTATE** « antihoraire » : CTRL, 1 et 9 pavée numérique
- **HOLD** : SHIFT, C, 0 pavée numérique

Compilation et execution

Make + CMake build

```
$ make release
```

Documentation

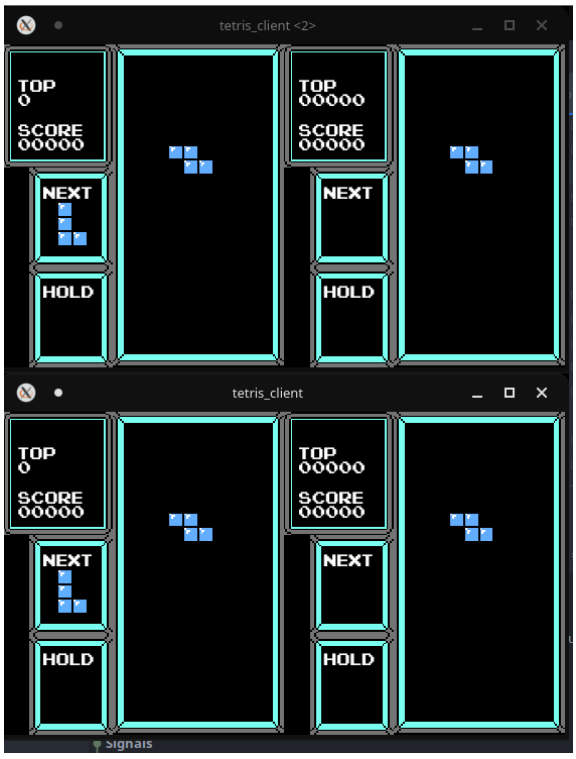
```
$ make docs
```

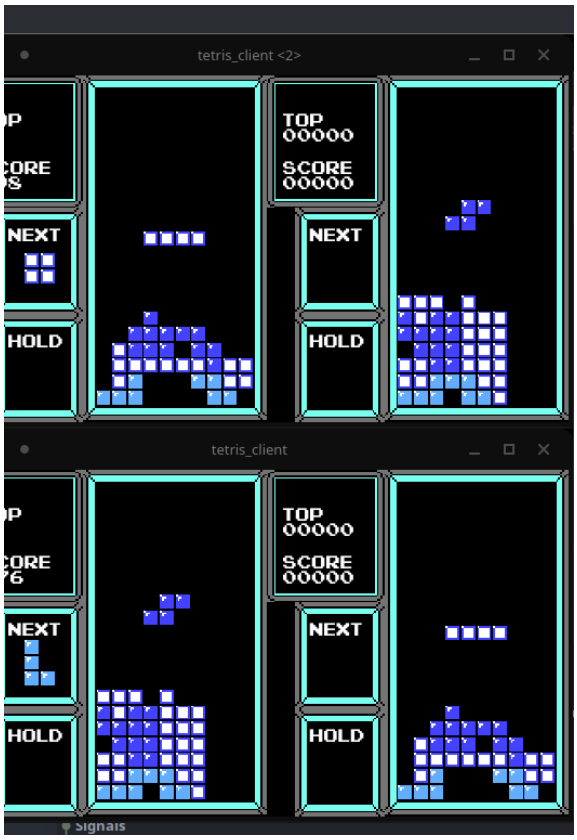
Unit Testing

```
$ make test
```

Implémentations

Client





Model

Game

Pour implémenter le jeu, nous avons utilisé le StatePattern . Celui-ci nous permet d'avoir un *ongoingGame* avec plusieurs états.

Les 5 états différents :

- Not started

  Le not started state définis l'état du jeu quand il n'est pas encore commencé. Comme on peut le voir la seul méthode disponible dans cette état est start(). Elle permet de démarrer le jeu, elle balance le jeu en *falling state*.

- Falling

  Le falling state définis l'état dans lequel un tetrimino peut tombé. Un timer existe pour faire tombé la pièce en fonction de la puissance de gravité à appliquer dessus. Dans cette état on peut *rotate, move, hold, softDrop et hardDrop*. Dès que la pièce n'arrive plus à descendre, le jeu bascule en *locked down* state.

- Locked down

  Le locked down state définis l'état dans lequel un tetrimino a 50 millisecondes pour se placer. L'utilisateur peut encore *move* et *rotate,* si celui-ci est permit, le jeu retourne en falling state. Après écoulement des 50 millisecondes, le tetrimino se place dans la matrice et c'est au prochaine de commencé a tombé.

- Blocked out

  Si a la création du prochain tetrimino, un tetrimino déjà placé se trouve a la coordonée ou le nouveau tetrimino va tombé, le jeu bascule en blocked out state, le joueur à perdu. Aucune méthode a part l'arrêt du jeu est disponible.

- Locked out

  Si au lock d'un tetrimino, il se trouve entièrement dans la zone de buffer (les 2 premières lignes de jeu) le jeu bascule en locked out state, le joueur a perdu. Aucune méthode à part l'arrêt du jeu est disponible.

- Stopped

  Si le joueur demande d'arrêter le jeu, le jeu bascule en stopped state.

C'est dans le ongoingGame qu'on peut appelé des actions (rotate, move, softDorp, hardDrop, hold) sur le jeu. Il les appliqueras en fonction du state. Le ongoingGame envoy 9 types de signal différents vers la vue et le client pour signaler des changements dans la partie.

Server

Nous avons un implémenté une version client/server basé sur QTcpSocket et QTcpServer de la librairie QT6. Le server se lance sur base de 2 argument, l'ip et le

port de lancement, et attends la connexion de client. Lorsque le premier client se connecte, il sera mis en attente. Lors de la connexion du 2 ème joueur, Le server crée un Match qui va géré la partie entre 2 joueurs. D'autres joueurs peuvent se connecté et le server lancera un nouveau match avec une nouvelle pair de joueur.

Le match se charge de transmettre les échanges entre les 2 joueurs.

Les 2 Clients communiquent l'un avec l'autre en notifiant le server avec des notifications. Ces objets sont basé sur un format JSON ensuite il sont écrit dans le socket et le server se cherche de les transféré à l'autre client. Les notifications regroupent tout les types de mouvements des pièces ainsi que les messages altérant le statues d'une partie (lancement, fin,...)

Lorsque les joueurs termine la partie, le server clôture le match

Utils

Une simple classe qui représentes des coordonnées d'un point en deux dimensions.