DEV4 - Rapport Tetris Remise 3 Implémentation GUI

Enzo RENARD - 60904 - D211 & Julien DELCOMBEL - 60189 - D212
Maître-assistant : Jonas Beleho

Introduction

Cette remise marque la dernière version de notre projet. Dans ce rapport, nous détaillerons les modifications apportées. Cela inclut les corrections par rapport aux remarques sur l'itération 2, les nouveaux ajouts fonctionnels et les ajustements pour développer l'Interface Graphique. Nous discuterons également des problèmes rencontrés et des solutions que nous avons mises en place. Nous examinerons les bugs encore présents, ainsi que les avertissements émis par le compilateur, afin de proposer des pistes de solution. Enfin, nous fournirons une estimation du temps passé sur cette première partie du projet.

Corrections effectuées par rapport à la remise console:

Model

Commençons tout d'abord par les modifications les moins notables du modèle.

GameSettings et GameStatus:

- Nous avons créé deux structures qui reprennent les différents paramètres et conditions de victoire afin d'alléger la classe Game et limiter son nombre d'attributs.

Controller

- Nous avons déplacé la méthode askForInt(), qui est chargée de récupérer les choix de l'utilisateur concernant les différents paramètres, de la classe GameView vers la classe ApplicationTetris afin de respecter le pattern MVC. Ainsi le package VIEW se charge uniquement de l'affichage et le package Controller, de l'interaction entre le model et l'utilisateur.

Ajout des classes pour produire l'interface graphique:

View

Nous avons décidé de séparer en classe chaque élément important qui ensemble forment l'affichage complet du jeu.

BoardBox:

- Nous avons utilisé un QWidget afin de pouvoir l'insérer dans un layout principal. Nous nous donnons un GameController pour qu'il obtienne les informations nécessaires à sa mission d'afficher le plateau de jeu.
- Nous utilisons les outils fournis par Qt tel que QPainter et QKeyEvent afin de dessiner et réagir aux actions du Tetris

InfoBox:

- Nous utilisons également un QWidget pour former avec BoardBox l'ensemble des éléments du jeu dans TetrisView

TetrisConfiguration:

- Il s'agit de la fenêtre qui s'ouvre en premier, permettant ainsi à l'utilisateur de configurer son jeu avant de lancer une partie. Nous avons choisi un QFormLayout car il s'agit simplement de cocher et choisir différents paramètres définis.
- Nous utilisons des ComboBox lorsque nous voulons donner un choix prédéfinis à l'utilisateur et des LineEdit pour un choix plus large.

TetrisGameOver:

- Cette fenêtre apparaîtra au Game Over pour bien marquer la fin du jeu. Nous utilisons un QWidget pour l'incorporer facilement et y ajouter différents éléments que nous placerons selon notre envie.

TetroView:

- Un QWidget qui va se peindre lui-même sur le plateau de jeu. Cela nous permet plus de facilité de gestion d'affichage des tetrominos.

TetrisView:

- Il s'agit de la vue principale de l'application. C'est également notre observeur afin de mettre à jour tous les éléments en fonction des changements dans le jeu.

Modification du model pour l'interface graphique:

Model

Pour cette itération, nous avons dû implémenter la notion de descente automatique des tétrominos. Afin de répondre à ce besoin, nous avons ajouté un Thread dans notre modèle

Game:

- Implémentation de Observer/Observable pour update en temps réel la vue graphique
- launchAutoDown(): afin d'implémenter la chute automatique des tetrominos, nous avons ajouté une méthode utilisant les Threads. Nous avons eu besoin d'une méthode privée getTimeBetweenDown() pour adapter la vitesse en fonction du niveau
- Nous avons ajouté des méthodes getter tels que getTime(), getBeforeLastTetromino() pour obtenir des informations à afficher et effectuer des animations dans notre interface graphique

Bugs restants:

N/A

Warning restants:

N/A

Problèmes rencontrés et solutions:

- Gestion des Threads et concurrence d'accès
- Attention particulière à la gestion de la mémoire. Beaucoup de new pour les éléments de l'interface graphique. Mise en place des destructeurs qui font les deletes nécessaires

Estimation du temps de travail:

Nous nous sommes basés sur le nombre de commits. Nous prenons une moyenne de 30 minutes par commit et estimons donc le temps de travail nécessaire:

$$100 * 30\,min = 3000\,min = 50\,h$$