Jeu du Puzzle Tetris

KARKASHADZE Luka, FAUQUETTE Mael
MANDIN Paul, TRANSON Alexandre

L3 Informatique Groupe 4B

4 décembre 2023

UNIVERSITÉ
CAEN
NORMANDIE

Table des matières

1 Introduction au projet **1**
1.1 Description générale de la consigne du projet 1
1.2 Description de la problématique du projet 1
1.3 Présentation du plan du rapport 1

2 Objectifs du projet **2**
2.1 Présentation des objectifs à atteindre 2

3 Fonctionnalités apportées **2**
3.1 Explications des fonctionnalités 2
3.2 Organisation du projet 2

4 Eléments techniques **3**
4.1 Description des diverses librairies utilisées 3
4.2 Description des diverses structures de données 3
4.3 Description des principaux algorithmes 4

5 Architecture du projet **4**
5.1 Diagramme des divers modules et classes 4
5.2 Explication du traitement des éléments par l'algorithme principal du projet . . . 5

6 Expérimentation et usages **6**
6.1 Cas d'utilisation 6
6.1.1 Documentation 6
6.1.2 Lancement du projet 6
6.2 Résultats quantifiables 6

7 Conclusion **7**
7.1 Récapitulatif des fonctionnalités principales 7
7.2 Propositions d'améliorations 7
7.2.1 Certains Algorithmes 7
7.2.2 Une interface plus fonctionnelle 7
7.2.3 Un plus grand nombre de fonctionnalités 7

1 Introduction au projet

1.1 Description générale de la consigne du projet

Le thème du projet est le jeu du puzzle de type "Tétris". C'est un classique des jeux vidéo. Le but du jeu est simple, il s'agit de placer les pièces du puzzle de sorte à ce que l'assemblage de ces dernères prenne le moins de case possible. C'est par le bais de ce paramètre que le score est calculé. Bien que paraissant simple au premier abord, ce jeu offre une grande variété de stratégies et peut être un excellent exercice de réflexion. Ce jeu peut être jouer que par un seul joueur.
C'est avec ce thème principal que débute notre projet avec pour consigne de faire une interface agréable et d'implémenter un bot capable de placer les pièces sur la grille de sorte à avoir le meilleur score possible.

1.2 Description de la problématique du projet

Afin de réaliser de la meilleure manière ce projet nous avons dû identifier une problématique principale à laquelle nous nous devions de répondre de façons optimales.
Après un premier temps de recherche nous avons décidé de tenter de répondre à la problématique suivante :
Comment créer un programme du jeu de puzzle de type "Tetris" optimisé et plaisant à jouer ? C'est à partir de cette dernière que nous allons construire le fil conducteur de ce rapport.

1.3 Présentation du plan du rapport

Ce rapport va se construire plusieurs grandes parties...

Dans un premier temps nous verrons et présenterons les différents objectifs qui seront fixés et qui devront être atteints. Par la même occasion nous commencerons à explorer des pistes à suivre afin d'atteindre ces objectifs.
Ensuite nous parlerons des principales fonctionnalités implémentées dans le projet afin d'avoir les premiers éléments de réponses. Nous dévoilerons également comment notre groupe s'est réparti les différentes tâches et comment chacunes des parties du groupes se sont organisées.
Suite à cela nous dévoilerons les aspects techniques de notre projet et expliquerons notre manière de programmer ce dernier par le biais des divers algorithmes mais également en présentant les diverses librairies auxquelles nous avons dû avoir recours afin de permettre la bonne exécution de nos programmes. Puis nous détaillerons les structures choisies pour l'exploitation des nombreuses données exploitées dans le jeu.
Dans la foulée nous présenterons l'architecture de nos programmes par l'intermédiaire d'un diagramme qui expliquera et définira les différents modules et classes. De plus nous verrons la manière dont sont traités les divers éléments composants notre projet par l'algorithme principal de notre jeu.
Postérieurement nous expliquerons l'utilisation à avoir de notre projet autrement dit la manière dont le jeu s'exécute, mais aussi une expliquation dans le but de générer la documentation des diverses fonctions utilisées dans les programmes des classes et packages. Nous apporterons à ceci des données importantes sur notre jeu comme par exemple le temps d'exécution de notre projet.
Nous finirons ce rapport en apportant une conclusion globale sur notre jeu de puzzle de type "Tétris" tout en apportant des axes d'améliorations qui seraient à suivre.

2 Objectifs du projet

2.1 Présentation des objectifs à atteindre

Afin de mener à bien ce projet nous nous sommes fixer un certains nombres d'objectifs que sont :

- Concevoir une interface utilisateur intuitive et agréable pour permettre aux joueurs de naviguer facilement dans le jeu.
- Programmer un bot capable de jouer au jeu et de trouver la meilleure solution possible.
- Offrir un environnement de jeu agréable et esthétiquement plaisant aux utilisateurs.
- Documenter par le biais de contrats le processus de développement pour faciliter les futurs ajustements et améliorations du jeu.

C'est à partir de ces éléments que nous allons guider notre projet afin de le réaliser de la manière la plus optimale.

3 Fonctionnalités apportées

3.1 Explications des fonctionnalités

Notre jeu de puzzle de type "Tétris" offre plusieurs fonctionnalités avancées pour offrir une expérience de jeu optimal. Voici les principales fonctionnalités de notre application :

Génération aléatoires de la grille de puzzle : Concevoir une grille de jeu afin de permettre le positionnement des pièces de manière aléatoire. Cette grille sera la base du jeu c'est elle qui sera le "modèle" de notre projet notion sur laquelle nous reviendrons un peu plus tard dans ce rapport.

Gestion des déplacements : Faire en sorte que le jeu soit capable de gérer et de prendre en compte les déplacements des diverses pièces du jeu autrement dit être capable d'identifier si le déplacement de la pièce voulu à des coordonnées données tombe sur une case vide ou sur une case déjà prise par une autre pièce. Et donc d'agir en conséquence.

Interface graphique conviviale : Cette interface doit être claire et pratique pour que l'utilisateur ait du plaisir à jouer et puisse comprendre du premier coup ce que le programme attend de lui afin de pouvoir poursuivre le jeu.

Un bot capable de jouer : Nous avons décidé d'implémenter un bot piloté par le biais d'algorithmes.

Un double affichage : Nous avons implémenter cette fonctionnalité, un premier affichage du jeu dans le terminal et un second dans une interface graphique cliquable. Le premier de ces affichages nécessite de rentrer les coordonnées des pièces et des déplacement à la main alors que le second permet de simplement cliquer sur une pièce et de la déplacer aux coordonnées voulues à l'aide des flèches directionelles du clavier.

3.2 Organisation du projet

La liste des principales fonctionnalités implémentées dans notre projet faite nous allons pouvoir expliquer dans cette partie l'organisation que nous avons pu adopter pour réaliser ce projet. Nous avons procéder de manière à ce que notre groupe avance en harmonie. Nous avons commencé par imaginer ensemble la manière dont nous allions composer notre jeu. Nous avons ensuite commencé le développement du modèle du projet soient les pièces et le plateau qui contient les pièces. Puis nous avons continué d'avancer puis Alexandre et Luka ont laissé la

suite du développement du modèle à Paul et Maël afin de se concentrer sur la Vue terminale et graphique. Puis Maël s'est ensuite occupé du controlleur. Ce dernier aura été développé avec l'aide des fonctions de la Vue et du Modèle. Luka se sera par la suite occupé de la partie test et du fichier **Build.xml**.

4 Eléments techniques

4.1 Description des diverses librairies utilisées

Afin de réaliser ce projet nous avons eu recours à un certain nombre de librairies. Les librairies utilisées sont majoritairement celles dans les classes d'affichages et d'interfaces. Nous avons utilisé les librairies swing et event pour l'interface du jeu.

Scanner : Librairie utilisée majoritairement dans la vue Terminale et dans la classe Human. La classe Scanner fait partie du package java.util et est utilisée pour analyser les entrées de manière simple et efficace. Elle offre des méthodes permettant de lire différents types de données à partir de diverses sources notamment le clavier pour notre projet. Grâce à ses méthodes telles que nextInt(), nextDouble(), nextLine(), etc., le Scanner simplifie l'interaction avec les données d'entrée, facilitant ainsi la création de programmes interactifs.

Swing et awt : utilisées dans l'interface graphique soit les classes GamesGui, Grille et Option. Ces librairies sont des librairies graphiques qui permettent de créer des interfaces utilisateur (IU). Elles offrent un large panel de composants graphiques, tels que des boutons, des champs de texte, des tableaux, des panneaux et des fenêtres. Ces composants personnalisables peuvent être modifiés pour répondre aux besoins spécifiques de l'application. Swing et Awt utilisent le plus souvent un modèle MVC (Model-View-Controller) pour la conception de l'IU dont nous reparlerons plus tard.

Event : La librairie "event" en Java est un ensemble de classes et d'interfaces qui sont utilisées pour implémenter le modèle de conception "Observateur/Observé" (Observer/Observable) dans les programmes Java. Ce concepte vise à informer certains objets du changement d'états d'autres objets. Autrement dit dans notre projet l'affichage doit être au courant de l'état de chaque case de la grille. Par exemple si un coup est joué sur une case vide alors l'affichage doit être au courant que cette case est vide et qu'un coup à été joué afin de mettre un point rouge sur cette case et montrer à l'utilisateur qu'elle ne peut plus être jouée. Ce concepte est utilisé dans le but de permettre une meilleure gestion des événements dans le programme.

4.2 Description des diverses structures de données

Nous allons exposer dans cette partie la manière dont sont traitées les diverses données nécessaires au bon fonctionnement du jeu.

PiecesPuzzle : Un objet de type PiecesPuzzle représente une pièce qui contient un id unique, une couleur au format hexadecimal, un array à 2 dimensions constitué de true et false pour sa forme et enfin la position du pixel en haut à gauche pour le placer dans le plateau.

PlateauPuzzle : Un PlateauPuzzle représente le plateau sur lequel sera disposé les pièces vu sidessus. Il a besoin d'une longueur et d'une largeur ainsi qu'une liste d'objet PiecesPuzzle.

Grille : La classe grille est la classe dans laquelle est contenue la matrice dans laquelle se déroule le jeu c'est à partir de cette dernière que l'affichage des pièces se fait. Le principe de cette classe est de colorier la case contenant une pièce par la couleur de la pièce

contenue. Ainsi le joueur peut visualiser aisément les formes de chaque pièces. Chaque pièces ayant une couleur différente.

4.3 Description des principaux algorithmes

Nous allons voir ici les principaux algorithmes sur lesquels reposent notre projet. Nous expliquerons comment ils fonctionnent et nous expliquerons leur utilité et surtout pourquoi nous avons décidé de les programmer ainsi.

Gestion de la grille : Chacune des fonctions de gestion de la grille se déroulent de la même manière. Nous prenons les coordonnées des pièces à déplacer, les coordonnées sont toujours exactes car elles sont triées au préalable dans les controllers. Ensuite nous modifions les objets dans le tableau selon l'action (déplacement d'une pièce, rotation d'une pièce).

Gestion des collisions : Pour pouvoir placer une piece il faut qu'elle ne sorte pas du plateau et que aucun de ses pixels ne se superposent sur une autre piece. Pour cela on parcours la forme de la piece sur le tableau où elle placée et regarde ainsi pixels par pixels si il y a une collision.

Mise à jour de la vue interface2D : Dans les grandes lignes, notre interface 2D (GameGUI) possède une Grille qui est un Jpannel affiché dans la Frame. A chaque modification du plateau, le signal est envoyé et le GameGUI repaint() (met à jour) la grille. Cet affichage des grilles se passe en plusieurs étapes. Il affiche dans un premier temps les lignes de la grille. Ensuite il parcourt la grille et paint les cases en fonction de leur état (Libre/Prise).

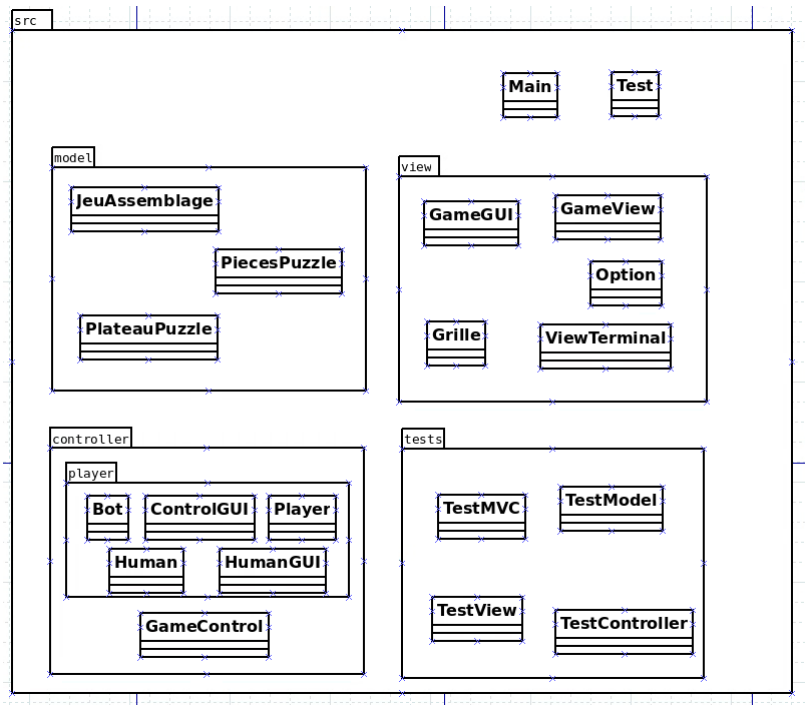
Déplacement : Pour déplacer une piece on garde en mémoire sa position de base et ensuite surpprime cette piece pour éviter qu'elle puisse rentrer en collision avec elle-même. On vérifie ensuite qu'il n'y ai pas de collision si on place la pièce à la nouvelle position avec une certaine rotation. Si il y a une collision, on annule la rotation et replace la pièce à sa position initiale. Si il n'y a pas de collision alors on ajoute la nouvelle pièce.

5 Architecture du projet

5.1 Diagramme des divers modules et classes

Nous allons d'abord parlé de l'arborescence de nos packages. Nous avons organisés notre projet avec l'arborescence suivante :
Nous avons décidé de diviser nos packages ainsi afin de faciliter la compréhension de tous. En effet nous pouvons comprendre directement quel package englobe quelle partie de notre projet. Car pour rappel notre jeu utilise le concept de MVC... Ainsi avec le package "controller" nous savons que les classes contenues remplissent la fontion de contrôle du modèle. Le package "model" quant à lui est le plus important il prend en compte tous les éléments indispensables au bon fonctionnement du jeu. Et le package "View" est celui qui concerne l'affichage du jeu par le biais de l'interface. Le fichier "Main" est celui qui permet le lancement de tout le jeu complet il regroupe et gère l'entièreté des autres packages dans le but d'assurer un déroulement correct de la partie du jeu.

Maintenant que nous avons vu l'arborescence de notre projet nous allons nous attarder sur ce que contient chacun des packages par le biais de ce diagramme :
insérer un diagramme qui présente tout ça



5.2 Explication du traitement des éléments par l'algorithme principal du projet

Nous avons basé notre projet sur le modèle de programmation dit MVC (Model-View-Controller). Ce concept est un modèle de conception largement utilisé dans le développement de logiciels. Il permet de séparer les responsabilités et les tâches d'une application en trois types de modules :

Le modèle : qui représente les données de l'application. Il s'agit de la partie de l'application qui gère les interactions avec les données, effectue des calculs et assure le bon fonctionnement de l'application.

La vue : qui représente l'interface utilisateur de l'application. Elle affiche les données du modèle à l'utilisateur et lui permet d'interagir avec l'application. La vue est responsable de la présentation des données à l'utilisateur.

le contrôleur : qui est responsable de la gestion des interactions entre le modèle et la vue. Il reçoit les actions de l'utilisateur à partir de la vue et utilise le modèle pour répondre à ces actions. Il met également à jour la vue en conséquence.

C'est sur ce modèle de programmation que nous nous sommes basés. Nous pouvons expliquer point par point les différents éléments de notre projet.

Le modèle : correspond aux PlateauPuzzle dans lequel on a une liste de pièces qui sera le jeu c'est cette classe qui va gérer les colisions entre les différentes pièces mais également le score. Qui lui est calculé en fonction de la taille que prend l'assemblage des pièces

La vue : est l'interface que l'utilisateur voit et avec laquelle il est en interaction. Il peut cliquer sur une pièce de la grille afin de la sélectionner et donc de pouvoir modifier sa position soit en la faisant tourner soit en la déplaçant par le biais des flèces du clavier.

le contrôleur : est la partie qui fait le lien entre la grille de l'interface avec laquelle le joueur intéragit et le modèle qui est modifié en fonction des intéractions du joueur. Les contrôleurs vont se contenter de prendre en compte les changements de position d'une pièce envoyés par l'utilisateur et de les transmettre à la grille.

Ainsi dans notre contexte pour le jeu de Puzzle de type "Tétris", le modèle MVC offre une clarté de l'architecture ce qui nous a simplifié les differentes tâches en tant que développeur. Cela nous a permis également de nous répartir les tâches de sorte à ce que certains membre de notre groupe puisse continuer le développement du modèle tout en laissant les autres avancé sur la vue. Cette forme de programmation été particulièrement utile pour gérer efficacement les interactions entre l'utilisateur, l'application et puis la vue.

6 Expérimentation et usages

6.1 Cas d'utilisation

Voici quelques petits conseils pour faciliter l'utilisation et la compréhension de notre projet.

6.1.1 Documentation

Afin de permettre une meilleure analyse de notre projet nous avons écrit un certain nombres de contrats pour chaque fonction utile au jeu. Tout cela dans le but de permettre de générer une documentation complète permettant de savoir l'utilité de chaque fonction, ce qu'elle prend en argument et ce qu'elle renvoie.Pour générer cette documentation, il suffit de se placer dans le dossier "mandin-transon-fauquette-karkashadze" puis de rentrer la commande "ant javadoc" dans le terminal. Cela générera la documentation automatiquement à partir des contrats précédemment évoqués et sera consultable directement sur un navigateur web.

6.1.2 Lancement du projet

Pour lancer le jeu il faut dans un premier temps le compiler dans un dossier. Afin de faire ceci vous pouvez utiliser la commande suivante dans un terminale à la racine du projet :
ant compile
Une fois compilé le programme du jeu est prêt à être lancé pour cela il vous faut rentrer cette commande toujours à la racine du projet : **ant main** ou **ant**

6.2 Résultats quantifiables

Nous pouvons prendre comme test celui du temps d'exécution du projet avant de faire débuter la partie. Autrement dit la création de la grille ainsi que la mise en place de l'ordinateurs ainsi que des bâteaux. Afin de mesure le temps nous avons mis en place deux variables de mesure de temps par le biais de la méthode "System.currentTimeMillis()"
Lors de l'exécution du programme nous avons cette ligne de code :

```
1 long startTime = System.currentTimeMillis();
```

Cette ligne permet de mesurer le temps au moment où est exécuté le programme.
Puis une fois que la partie est prête nous avons cette ligne de code :

```
1 long endTime = System.currentTimeMillis();
2 long totalTime = endTime - startTime;
3 System.out.println("Temps␣d'execution␣:␣" + totalTime
4 + "␣ms.");
```

Ici nous avons une variable qui mesure le temps à la fin de la configuration de la partie cette valeur est ensuite soustraite à celle qui est déclarée au lancement du programme afin d'avoir le temps pris par l'ordinateur pour générer la partie avec un bot puis l'interface graphique. Une fois exécutée nous avons obtenus ce résultat :

```
Temps d'execution : 27ms
```

Ce qui est un temps assez court et satisfaisant afin de garantir les meilleures performances possibles au chargement de la partie.

7 Conclusion

7.1 Récapitulatif des fonctionnalités principales

Dans notre rendu final nous avons réussi à implémenter les principales fonctionnalités souhaitées que sont :

- La grille généré aléatoirement avec les différents états de chaque case.
- Le bot qui n'utilise pas d'aléatoire.
- Une gestion des colisions entre chaque pièce
- Une interface propre et agréable à parcourir.
- L'utilisation du modèle MVC par l'entièreté du projet
- Un résultat en fonction de la performance en jeu

7.2 Propositions d'améliorations

Nous avons tout de même certains points sur lesquels nous pourrions améliorer notre projet :

7.2.1 Certains Algorithmes

En effet si la totalité des algorithmes sont fonctionnels et utiles à notre projet nous aurions un axe sur lequel nous pourrions améliorer nos programme notamment sur certains algorithmes qui présentent une forte complexité que ce soit en espace comme en temps. Mais aussi sur la lisibilité du code de certains autres programmes.

7.2.2 Une interface plus fonctionnelle

Nous avons certes une interface convenable mais qui pourrait être améliorée grâce à l'apport de nouvelles fonctionnalités ou bien par un esthétisme un peu plus travaillé. C'est à dire afficher plus clairement les pièces avec éventuellement un JPanel présentant les pièces qui ne sont pas encore placées dans la grille.

7.2.3 Un plus grand nombre de fonctionnalités

- Le BOT qui pourrait être amélioré et donner des réponses bien plus convainquantes.
- Un menu option permettant de choisir diverses options directement sur l'interface au lieu de les demander sur le terminal.
- Un alogrithme qui calcule le meilleur plateau final possible pour avoir une meilleure base de calcule pour notre fonction résultat.