Rapport méthodes de conception

Créer un jeu de puzzle facon Tetris sous Java

DAVID Matthias
LE BRIS Ilan
MARCHERON Bastien
PARCHEMINER Nolann

4 décembre 2023

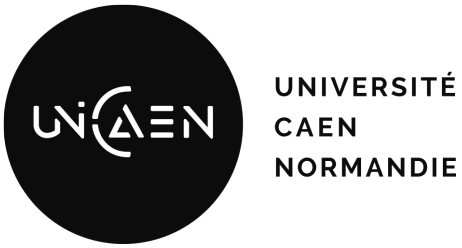

Table des matières

1 Introduction

Dans le cadre de l'unité d'enseignement méthodes de conception, nous devions réaliser une application en java par groupe de 4.
Le sujet donné consistait à développer un puzzle a base de polyominos (Figure que l'on retrouve dans le célébre jeu Tétris). L'objectif principal étant d'assembler ces formes ensembles afin qu'elles occupent le moins de place possible.

2 Mise en place du projet

2.1 Analyse

Il nous a été demandé de réaliser ce projet en mettant en place le design pattern MVC (Modèle Vue Controlleur). Pour la conception de l'application, nous avons donc identifié 2 entités majeures, La première représentera une pièce de jeu (appelée piecePuzzle). La seconde permettras de définir le plateau de jeu (appelée jeuAssemblage). Chacune de ces deux entités se verra composée d'un répertoire regroupant les modèles, un autre les vues ainsi que les controlleurs et enfin un dernier répertoire regroupant les tests.

2.2 Répartition des tâches

Afin de se rapprocher de l'environnement de travail que l'on trouve dans le monde professionnel, nous nous sommes répartis les tâches de manière à avoir une quantité de travail équivalente tout en collaborant sur certains points de manière efficace. Chaque membre du groupe a choisi la partie sur laquelle il voulait travailler. Les tâches ont été réparties de la manière suivante :

— DAVID Matthias : Conception des modèles

— LE BRIS Ilan : Conception de l'interface graphique, des vues et controlleurs

— MARCHERON Bastien : Élaboration du rapport, aide sur différente partie

— PARCHEMINER Nolann : Gestion de tous les tests ainsi que des fichiers d'aumatisations de tâches (ant)

3 Conception

3.1 Structure

Comme dis plutot notre projet vas se composer de 2 package, piecePuzzle et jeuAssemblage(renommé en livraison)

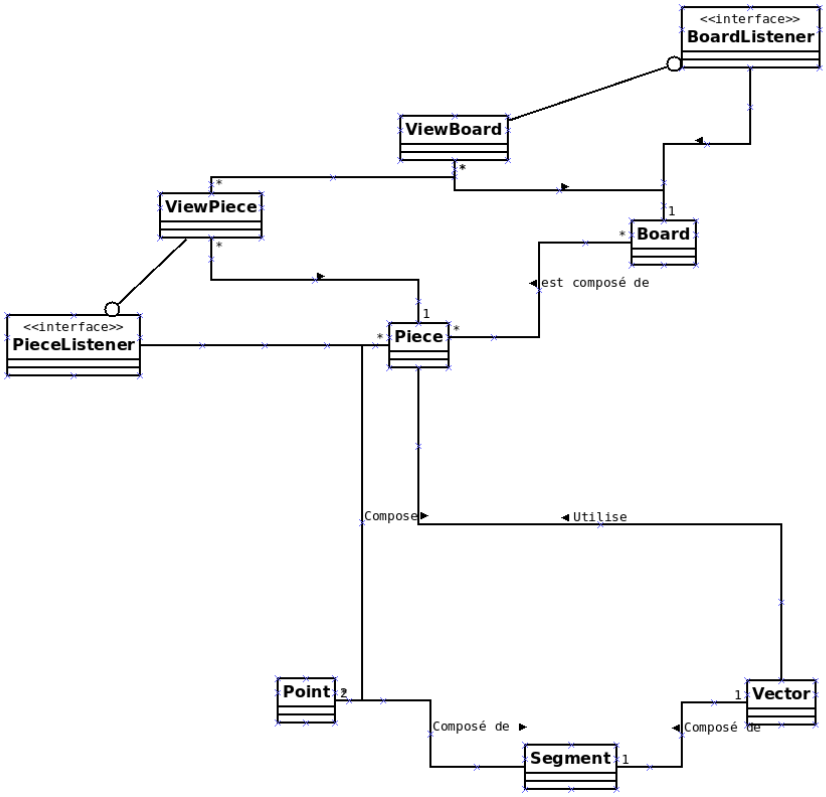

FIGURE 1 – Diagramme de classe du projet

3.2 Interface graphique

Pour concevoir l'interface graphique nous avons utilisé le principe Modèle-Vue-Contrôleur étudié en cours.
Les packages utilisés ont notamment été **javax.swing** pour les éléments graphiques et **java.awt.event** pour la gestion des événements sur l'interface.

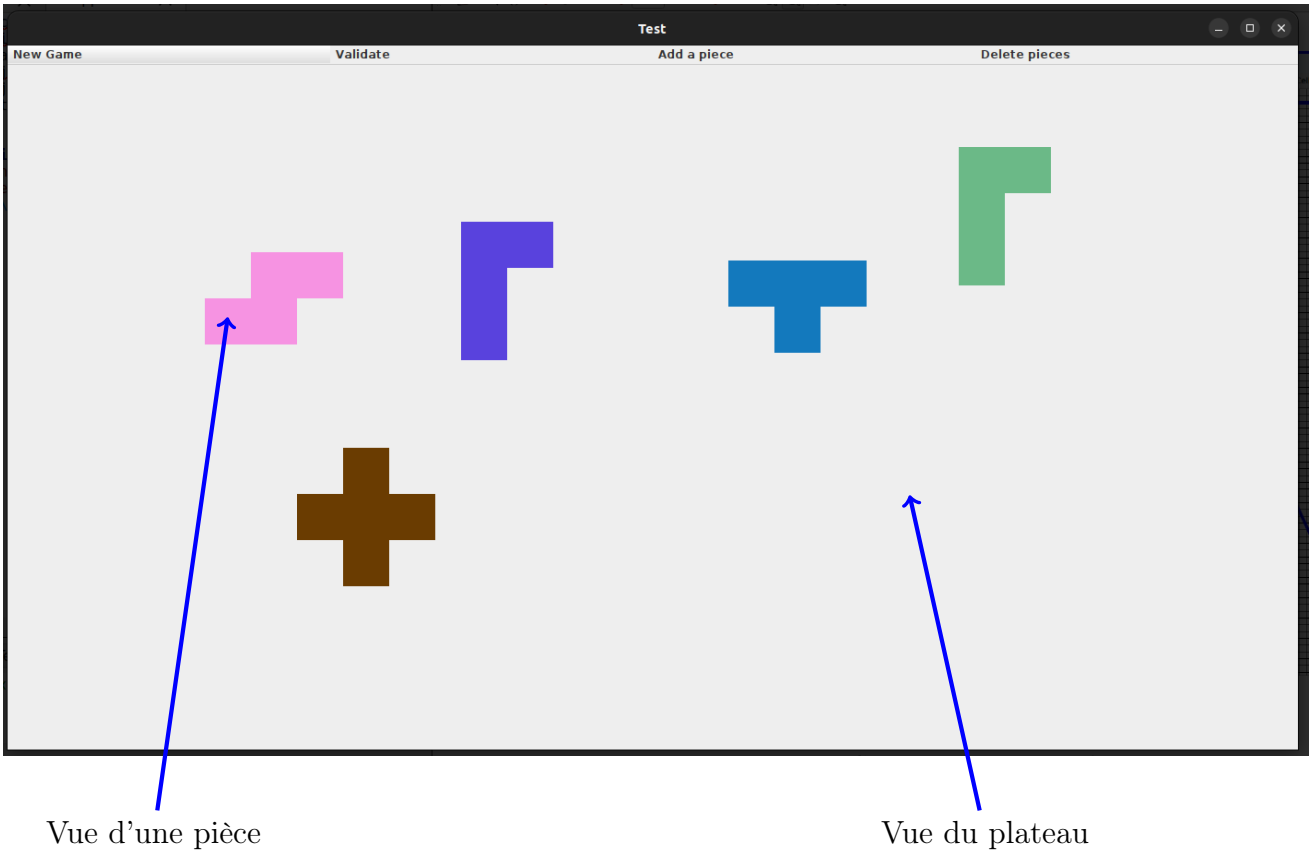

FIGURE 2 – Capture d'écran de l'interface graphique

L'interface est donc composée de trois éléments principaux à savoir le menu, la vue du plateau, et les vues des pièces qui composent le plateau.
Le menu contient 4 onglets :

New Game : Pour remplir aléatoirement le plateau d'une liste de pièce.

Validate : Permet de valider la disposition actuel et d'afficher le score correspondant.

Add Piece : Pour ajouter une pièce (configurable grâce à un sous menu) au plateau de jeu.

Delete pieces Pour supprimer toutes les pieces du plateau

3.3 Éléments techniques

Une pièce est définis comme étant un polygone non nécessairement convexe mais nécessairement non croisé que l'on représente par une suite de Point

3.3.1 Algorithme détection point dans une Piece

Pour savoir si une pièce en chevauche une autre ou encore savoir si la souris est positionné dans une pièce il a fallu développé un algorithme permettant de savoir si un Point donné se situe dans une Pièce ou non
L'algorithme de lancer de rayonlancer de rayon a donc été utilisé pour résoudre ce problème. Voici son fonctionnement tel qu'il a été implementé :

Algorithm 1: Ray cast

Input: *piece* : Piece *la Piece a tester*
point : Point *le point a testé*,
edge : bool *indique si point peut être sur les coté ou non*
Result: True si point est dans Piece(coté compris si edge est a True
sinon coté non compris) sinon false
points ← La liste des points qui compose la *piece*;
min_x ←Le point avec l'axe x minimum parmis *points*;
max_x ←Le point avec l'axe x maximum parmis *points*;
ray_end ← Le point symétrique a min_x par rapport a max_x;
ray ←Le segment allant de *point* a *ray_end* ;
ray_cross ← 0 ;
forall *Segment s de piece* **do**
  if *point est sur s* **then**
    return valeur de *edge*
  end
  if *Il y a un unique point d'intersection entre ray et s* **then**
    ray_cross ← *ray_cross* + 1
  end
end
if *ray_cross est pair* **then**
  return False
end
return True

3.3.2 Ant

Pour nous aider dans la réalisation du projet, nous avons utilisé apache Ant, un utilitaire de scripts (principalement pour les projets en Java) qui permet de faire de manière simple des tâches répétitives qui deviennent compliquées au fur et à mesure que le projet grandit. Nous avons créé 8 scripts permettant de simplifier les manipulations sur le projet. Pouvoir compiler, exécuter et nettoyer le projet à partir d'une simple commande nous a permis de gagner du temps et nous a évité bien des erreurs dans les commandes ;

Nom Commande	**Description commande**
clean	supprime le dossier bin
init	crée les repertoires bin, doc et dist
compile	compile le projet
run	lance l'application
test	lance les tests
doc	génère la documentation
dist	construit le projet (compile + doc + .jar + clean)

TABLE 1 – Commandes ant

4 Tests

Pour s'assurer que les fonctions écrites et utilisées fonctionnent, nous avons mis en place une partie de tests dans laquelle nous testons les fonctions ayant un impact sur les données du model ou sur les fonctions du controlleur. Cela permet de s'assurer que les fonctions ont le comportement attendu face à des cas particuliers

Les tests ont permis d'identifier et de corriger rapidement les erreurs qui peuvent survenir lors de l'exécution de certaines fonctions, mais ils ont également permis de modifier certaines fonctions pour les rendre plus génériques.

Ci-dessous le résultat obtenu avec la commande ant test, qui exécute tous les tests

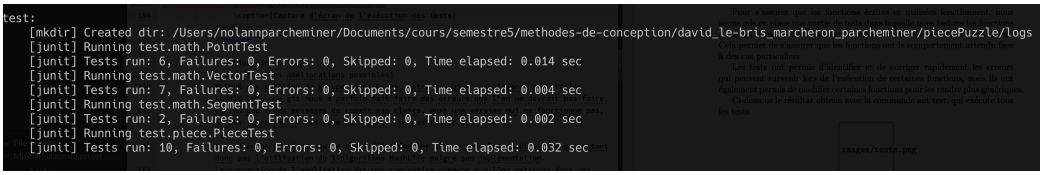

FIGURE 3 – Capture d'écran de l'exécution des tests

5 Conclusion

5.1 Les améliorations possibles

L'application que nous rendons est fonctionnelle, or nous sommes conscients que les points suivants peuvent être améliorés voir implémentés :

— Algorithme de résolution automatique

— Meilleure gestion des collisions

— Une interface graphique plus simple à utiliser

5.2 Ce que le projet nous a apporté

Nous avons réalisé ce projet en suivant le principe du MVC, nous avons réalisé des test afin de nous assurer du bon fonctionnement des différentes parties. Nous avons aussi séparé les pièces du plateau de jeu, permettant ainsi de réutiliser les pièces dans un autre projet. Certaines fonctions telles que la vérification de collisions entre les pièces du plateau peuvent être améliorées, cependant les performances sont correctes lorsque de l'on utilise le plateau avec les valeurs par défaut. Le code fournit est maintenable et peut facilement évoluer dans le temps. Nous avons de même utilisé différents desgin patterns vus en cours.