5 décembre 2020

Jeu d'assemblage de formes

Méthode de conception logiciel

Marguerite Bauchez 21803320 Olivier Cocquerez 21803239
Paul Lebranchu 21403460 Raphaelle Lemaire 21802756

L3 informatique
Groupe 1B
2020-2021

Table des matières

Introduction

Le jeu d'assemblage de formes est un jeu de puzzle où l'on doit constituer une forme à l'aide de différentes pièces de forme et de taille aléatoire de façon à ce que la forme puisse rentrer dans un rectangle dont la taille devra être la plus petite possible. Afin de réaliser ce jeu, nous avons utilisé le langage de programmation objet Java et fait un dépôt sur SVN pour pouvoir travailler en équipe (même à distance). Ce jeu sera aussi jouable dans le terminale de commande.

1 Organisation du projet

1.1 Organisation des fichiers

Lorsque nous récupérons le projet depuis svn, les répertoires sont organisés de la façon suivante :

- un répertoire livraison contenant :
  - un répertoire dist contenant toute les ressources nécessaire au lancement de l'application (.jar + contenu du répertoire lib)
  - un répertoire doc qui contiendra la javadoc des classes présentes dans le répertoire src
  - un répertoire lib qui contient les images du jeu ainsi qu'une archive .jar permettant de générer les pièces de jeu
  - un répertoire rapport qui contient ce rapport au format pdf ainsi qu'un répertoire LaTeX contenant l'ensemble des ressources nécessaire à la création de ce pdf
  - un répertoire src qui contient l'ensemble des classes nécessaires pour faire tourner le jeu d'assemblage, ce répertoire est divisé de la façon suivante :
    - un répertoire Controlleur qui contient les interfaces/classes abstraites nécessaires pour que la Vue et le Modele communiquent
    - un répertoire Modele qui contient les classes permettant de générer un Plateau de jeu et les méthodes nécessaires pour agir sur ce plateau
    - un répertoire Vue qui contient les classes nécessaire pour donner un aspect visuel et interactif au Modele
    - un fichier Main.java qui permet de lancer l'application
  - un ficher build.xml qui permet de compiler le code, qui créé un répertoire build contenant tout les fichier .class, qui génère la javadoc des classes présentes dans src et les stock dans le répertoire doc, qui recréer le contenu du répertoire dist et qui lance le jeu.

  - un fichier README.txt qui explique à l'utilisateur comment lancer l'application et qui lui explique les deux modes de jeu disponible.
  - un fichier score.txt vide qui contiendra les scores des différentes parties jouées.
- un répertoire piecesPuzzle contenant tout les classes qui se rapportent au pièce de jeu et un fichier ant qui génère la javadoc et une archive .jar qui sera placé automatiquement dans le répertoire lib de livraison.

1.2 Gestion du projet

Pour gérer ce projet, nous avons fait appel à plusieurs ressources :
- SVN pour mettre en commun nos fichiers
- Discord/messenger pour pouvoir communiquer entre nous et nous tenir au courant des différentes avancées effectuées
- Les cours (e-campus/discord/cours en présentiels)
- Atom/NetBeans/Geany pour rédiger notre code
- Oracle (https ://docs.oracle.com/javase/8/docs/api/)
- Trello pour nous répartir le travail

2 Architecture du projet

2.1 Diagramme de la librairie piece

La librairie pieces nous permet de construire les différentes pièces de jeu. Elle utilise le design pattern strategy : elle est composée d'une interface (InterfacePiece) qui nous donne les propriétés à implémenter pour les différentes pièces (rotation et déplacement), une classe abstraite qui définit les spécificités des pièces de notre jeu en particulier : cette classe défini les paramètres des pièces :leur position x et y, leur taille minimale et maximale, leur largeur, leur hauteur, le tableau définissant la pièce (rempli avec des 1 et des 0) et leur identifiant. C'est cette classe qui définit la rotation des pièces(rotation du tableau définissant la pièce) et leur déplacement : la fonction prend en paramètre un entier x et un entier y qui seront les nouvelles coordonnées de la pièce. Ensuite, chaque sous-classe définit les spécificités de sa pièce (par exemple, pour une pièce de type O, la largeur et la hauteur doivent être identique) et créé la pièce en utilisant sa version du createPiece().Nous trouvons également dans ce package une classe PieceFactory (pattern Factory) qui nous permet de créer des instances d'un objet en fonction du type qu'on lui a attribué en paramètre.

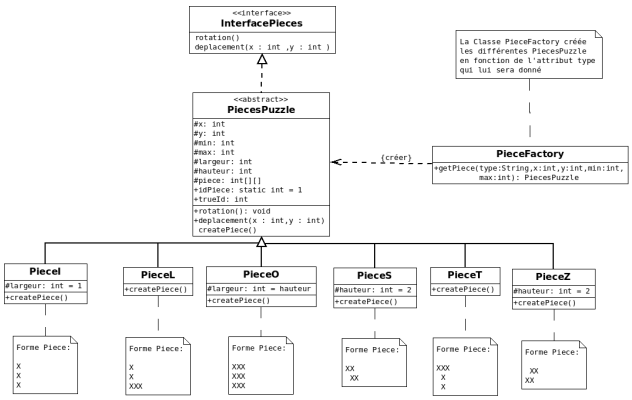


FIGURE 1 – Diagramme de classe : librairie pieces

2.2 Diagramme de livraison

Sur le diagramme ci-dessous, nous pouvons les interactions entre les différents packages qui composent notre projet. Nous constatons qu'il y a trois gros packages : le package Controlleur qui permet de faire le lien entre la vue et le modèle, le package Model qui gère la création du plateau et les méthodes pour interagir dessus et le package Vue qui gère l'interface graphique et les actions sur cette dernière.

Nous pouvons noter que trois classes ont recours à la librairie pieces : la classe CreationPlateau (Model) et les classe Menu et AffPlateau (Vue).

Nous voyons que le package Model est constitué de trois classes : Plateau qui s'occupe des paramètres du plateau de jeu (taille), CreationPlateau qui appelle Plateau et qui étend la classe AbstractModeleEcoutable pour faire en sorte que la vue puisse écouter le modèle et la classe Main qui est là pour rendre le jeu jouable en version console.

Le package Vue est composé de quatre classes : une classe AffPlateau qui créera et affichera le plateau de jeu, une classe Menu qui gère le menu est les contrôles des pièces, cette classe possède une instance de CreationPlateau et une instance d'AffPlateau et fais le lien entre les deux grâce à sa méthode modeleMiseAJour (héritée de l'interface EcouteurModel). La classe Fenêtre est un JPanel qui prend en paramètre un objet de type Menu et qui l'affiche. Cette Fenetre est ensuite placée dans une JFrame dans la classe Affichage (classe permettant de lancé le jeu en mode interface graphique).

La classe Main appelle une instance de Modele.Main et une instance d'Af-

fichage pour faire en sorte que le jeu soit jouable en mode console ou en mode interface graphique.

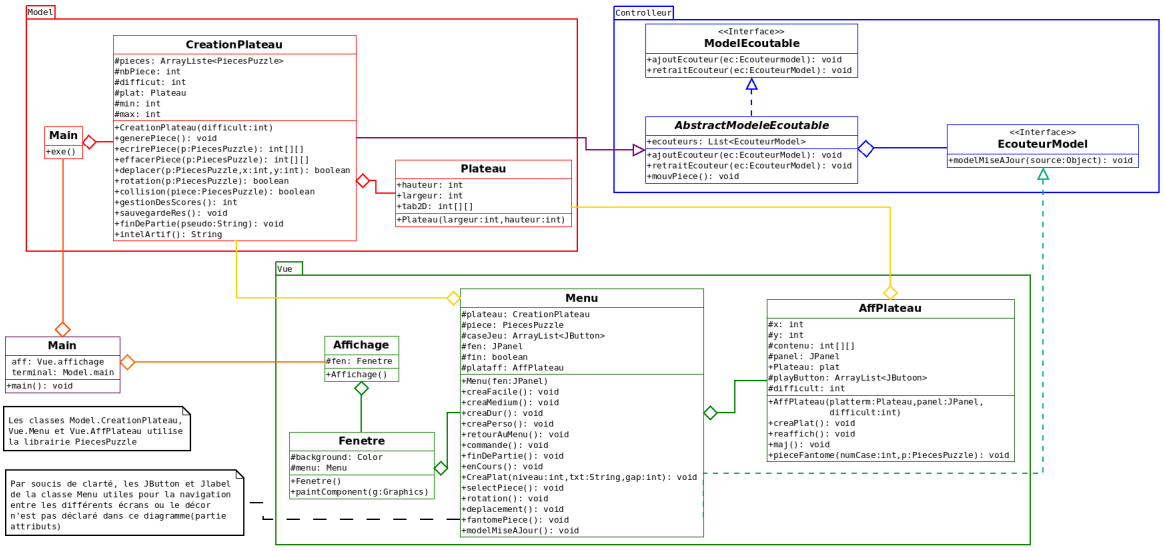
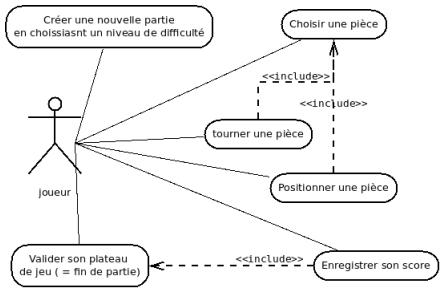


FIGURE 2 – Diagramme de classe : répertoire src dans livraison

3 Cas d'utilisation



L'utilisateur aura plusieurs actions à faire. Dans un premier temps, il devra créer une partie : Pour cela, il devra choisir son niveau de difficulté (cliqué sur un bouton dans le mode interface graphique, rentrer un int dans le mode console). Une fois le plateau de jeu créé, le joueur devra sélectionner une pièce et effectuer une action (rotation ou déplacement), dans le mode graphique cela se fait avec le clavier et la souris et dans le mode console, le joueur devra rentrer des commandes dans le terminal. Quand l'utilisateur estimera qu'il ne pourra pas faire mieux, il terminera la partie et aura la possibilité d'enregistrer son score. Une fois que cela sera fait, le joueur verra son score et celui de l'IA qu'il aura affronté et pourra commencer une nouvelle partie.

3.1 Mode Console

Dans un premier temps, on présente les différents niveaux de difficulté au joueur et on lui demande de sélectionner le niveau de son choix, une fois le niveau choisis, on montre la création du plateau pièce par pièce au joueur puis le joueur aura la possibilité de jouer. Lorsque l'utilisateur aura fini de jouer, on lui demandera de rentrer son pseudo pour la sauvegarde des scores, on affiche ensuite son score puis l'IA fait son calcul et nous affiche le sien, on demande ensuite à l'utilisateur s'il souhaite rejouer ou s'arrêter là, s'il rejoue, nous retournons à la création de partie, sinon le programme s'arrête.

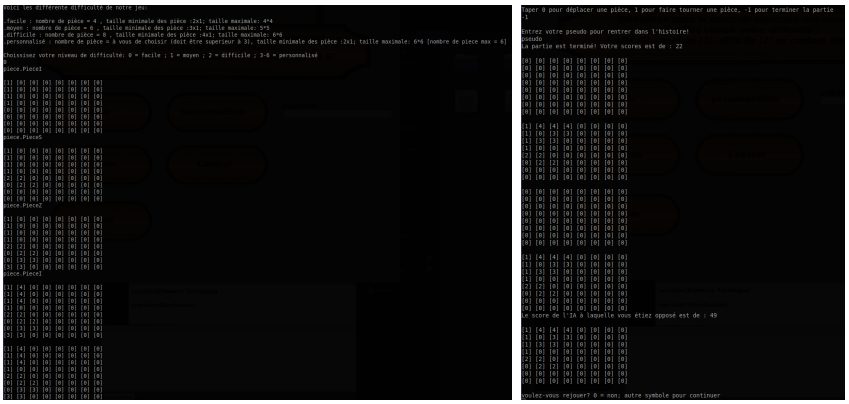


FIGURE 3 – Début et fin de partie en mode console

Nous pouvons voir sur ces images de la figure 4 le déroulement des différent actions de jeu. Dans un premier temps, nous devons dire si nous souhaitons effectué une rotation ou un déplacement, si nous effectuons une rotation, nous avons juste à renseigner la pièce que nous souhaitons faire tourner, vous pouvez voir sur l'image de gauche une rotation réussie(cadre vert) et une rotation qui a échoué à cause d'une collision(cadre rouge).Si l'on souhaite effectuer un déplacement, nous devrons en plus renseigner un numéro de ligne et de colonne, vous pouvez voir dans le cadre vert un exemple de déplacement réussi et dans le cadre rouge, un exemple de déplacement raté. Le cadre jaune dans l'image de gauche nous montre que tant qu'on aura pas saisie un numéro de pièce valide pour une action, le programme nous redemandera de saisir un numéro de pièce.

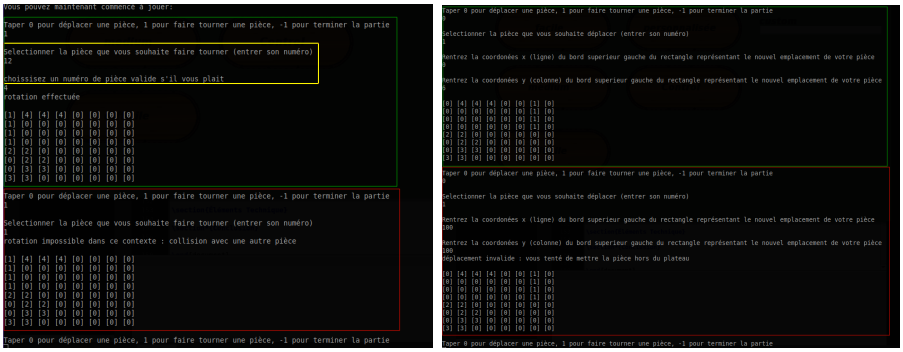


FIGURE 4 – jeu en mode console

3.2 Mode Interface Graphique

La figure 5 représente les différentes versions du menu qui existe, la première version correspond à ce qui s'affiche lorsqu'on lance l'application, la deuxième nous montre le menu lorsque nous tentons de créer une partie personnalisée en ne donnant pas les bonnes informations et la troisième version nous montre le menu lorsque nous retournons sur le menu lors d'une partie en cours.

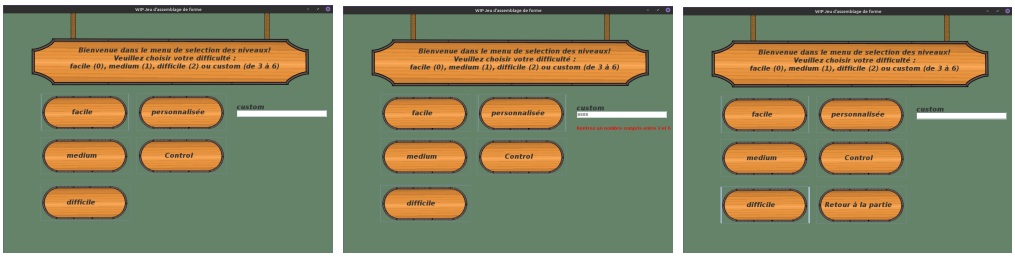


FIGURE 5 – Les trois versions du menu

La figure 6 nous montre une partie en cours, la première image représente le plateau de jeu tel qu'il est créé, la seconde image nous montre une pièce fantôme de la pièce jaune qui est la pièce sélectionnée et les deux dernières images correspondent au fenêtre qui s'ouvre lorsque nous commettons une erreur lors du déplacement ou de la rotation d'une pièce.

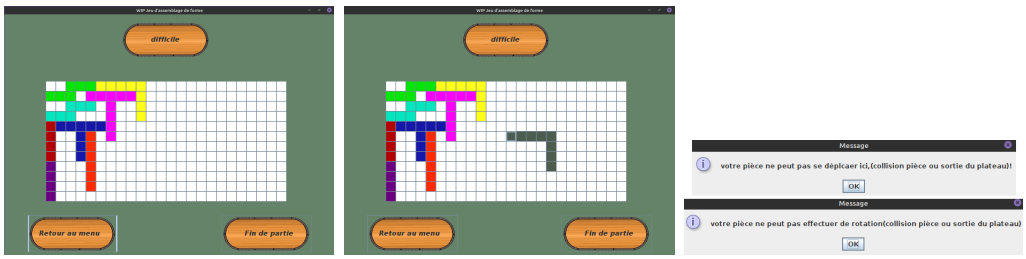


FIGURE 6 – Jeu

La figure 7 nous montre comment se déroule une fin de partie, dans un premier temps, nous appuyons sur le bouton fin de partie puis nous voyons une fenêtre apparaître et nous demander de rentrer notre pseudo. Ensuite, on voit apparaître notre score ainsi que le score de l'IA. Puis nous pouvons retourner au menu et créer une nouvelle partie. La dernière image nous montre le panneau affichant les controls du jeu, cette écran est accessible depuis le menu.

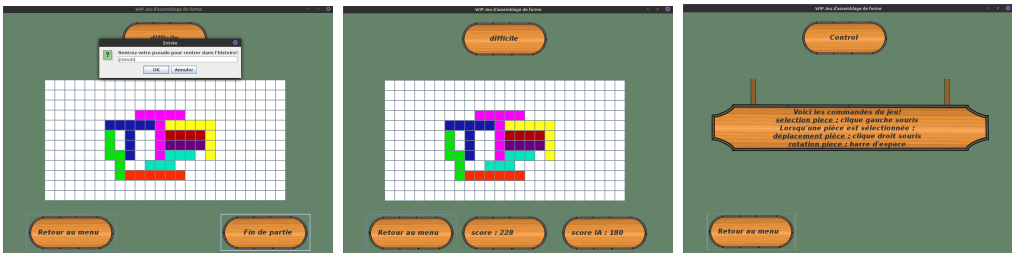


FIGURE 7 – fin de partie et contrôles

Conclusion

Notre application est donc jouable en mode console comme en mode interface graphique.Cependant, quelques points de notre application pourraient être améliorer : notre IA se retrouve parfois bloquer dans des boucles infinies et nous renvoie une erreur, nous pourrions également ajouter des ascenseurs verticaux et horizontaux sur notre jeu pour les utilisateurs qui voudraient réduire la taille de leur fenêtre en mode interface graphique.