Université Claude Bernard – Lyon1
L3 – Licence Informatique

Casse-tête - Lignes

Projet LIFAP7 - POO

ABDRABO Khaled p1713323
MESSOUD Mohamed Salem p1714033

Encadré par Frédéric ARMETTA
(Automne 2019)

TABLEAU DES MATIERES

TABLEAU DES LLUSTRATIONS

INTRODUCTION

Dans le cadre de l'UE "LIFAP7 - Algorithmique et Programmation Orientée Objet", il nous est proposé de réaliser un projet de programmation orientée objet, pour se familiariser avec le langage Java et la bibliothèque graphique Java FX. Dans ce sens, nous avons réalisé un rapport de projet ayant pour finalité d'expliquer nos étapes de conception et le processus du développement de l'application "Casse-tête Lignes", tout en mettant en avant les éventuelles extensions ajoutées fur et à mesure du l'implémentation, et ajoutant un diagramme de classes permettre de présentant les différentes relations entre les classes et les interfaces de notre jeu. Nous avons aussi inclus diverses copies d'écran de notre programme pour mieux illustrer l'aperçu de l'application.

Le but du jeu

Le jeu "Casse-tête - Lignes" présente des différents casse-têtes. Chaque casse-tête est une grille carrée de taille NxN avec des boules colorées occupant certaines cases. L'objectif du jeu est de relier la paire de boules de la même couleur en dessinant un chemin entre elles. La partie du jeu est gagnée dans le cas où toutes les paires de boules sont reliées et que toutes les cases de la grille sont parcourues. Une configuration résolue a donc toutes les paires de symboles sont connectées, et que toutes les cases de la grille sont parcourues.

Exemple explicatif

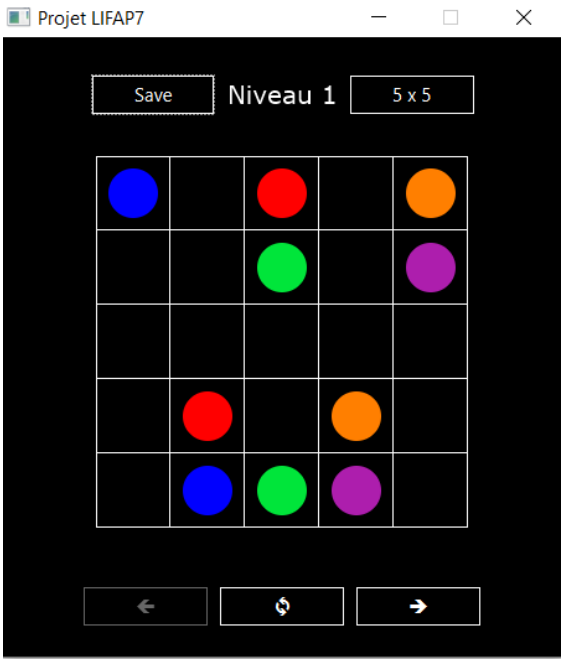

Figure 1 : Capture d'écran de la configuration initiale

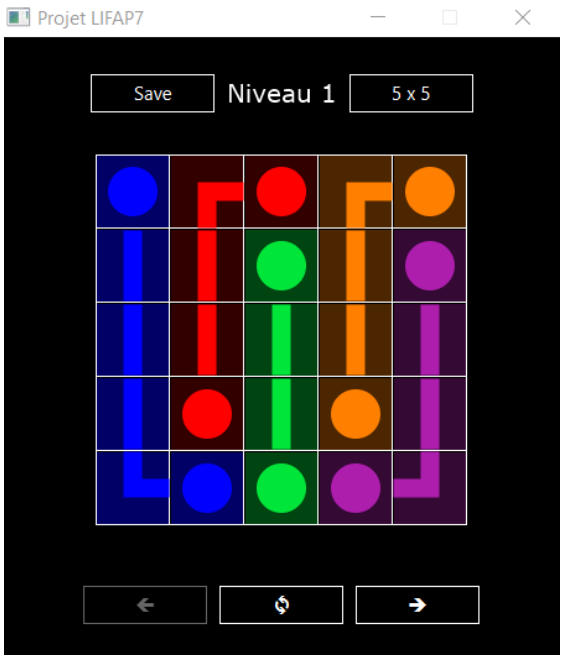

Figure 2 : Capture d'écran de la configuration résolue

1. DEMARCHES DE CONCEPTION

Dans cette partie, nous avons étudié la faisabilité du projet de son aspect fonctionnel et technique. Tout en se basant sur ce que nous avons appris en cours et sur nos recherches individuelles sur internet.

Phases de conceptions et distribution des tâches

- Phase d'étude [P0] : qui a eu lieu lors de la première séance du TP Projet : est une phase de réflexion : Réfléchir ensemble sur la conception du jeu, créer une page du projet sur GitHub, et faire le module cases.
    - Chacun chez lui : se familiariser avec Java FX
    - Khaled : Le module Chemin
    - Salem : Le module JeuLignes
- Jalon J0 : Lancement de la phase de réalisation
- Phase de réalisation [P1] : qui a eu séance TP projet autonome ensemble mettre en place le code du drag & drop donné par le prof faire les premiers affichages
    - Chacun chez lui : se familiariser avec Java FX :
    - Khaled : mettre en place l'affichage des différents éléments de la grille et une autocomplétion du chemin
    - Salem : trouver une manière propre pour dessiner les chemins
- Jalon 1 : Etat d'avancement. Qui a eu lieu en 2ème séance du TP Projet : établir l'état de l'avancement du projet, savoir ce qu'il faut améliorer fixer certains bugs.
    - Khaled : mettre en place les boutons, les différentes scènes du Stage, le menu, résoudre certains bugs ;
    - Salem : Gérer les Input Output « IO » (la sauvegarde de la grille, la lecture de la grille, les différents niveaux, les chemins etc.)
- Phase Test [P3] : Test final de l'application réalisée. Permettre de charger une partie sauvegardée, nettoyer le code et le commenter, résoudre certains bugs. Présentation de la démo de l'application.

Diagramme des classes

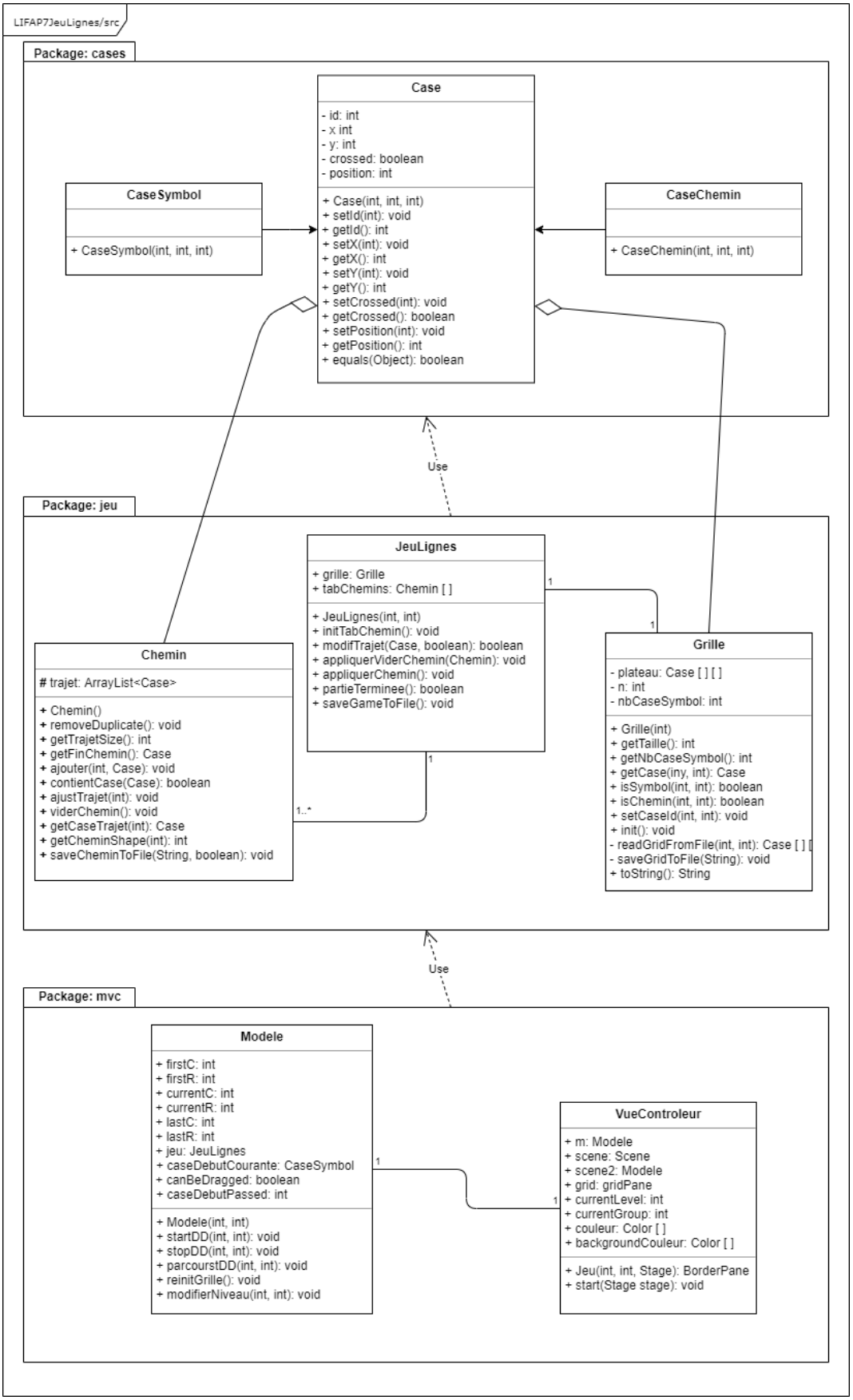

Figure 3 : Digramme des classes

2. GESTION DU PROJET - CHOIX ET ENJEUX TECHNIQUES

Nous avons organisé notre travail, en partageant nos tâches sous forme de modules. Tout en pratiquant les modules MVC qui nous a permis d'avoir une claire visibilité sur la manière d'organiser nos modules de développement. Nous avons partagé un module par personne, cela nous a permis de travailler en autonomie. En cas de blocage, de la ressource responsable d'une tâche, il demande de l'aide à l'autre ressource. Cette méthode nous a permis d'être plus efficace et d'avoir esprit d'équipe.

MVC

Il nous est proposé d'utiliser le design pattern MVC. Nous avons donc partagé notre programme en trois packages qui sont les suivants :

- cases : contient toutes les classes liées à la gestion des différentes types de cases (CaseSymbol et CaseChemin)
- jeu : contient les classes Grille, Chemin, JeuLignes.
    - Grille : une agrégation de cases qui permet de gérer les données des cases
    - Chemin : une agrégation de caseChemin qui permet de stocker les différents chemins parcourus par l'utilisateur.
    - JeuLignes : Elle est le chef d'orchestre permet de gérer une grille et de
- mvc : contient les classes Modele et VueControleur
    - Modele : c'est la partie qui implémente les données de l'application en instanciant un objet JeuLignes. Dans notre application elle représente une séparation conceptuelle. Elle gère en aucun cas l'affichage
    - VueControlleur : c'est la partie du code qui permet l'affichage des composants et gère l'interface utilisateur (UI) de l'application. Elle prend en compte les données du modèle et sélectionne finalement une vue à rendre qui affiche l'interface utilisateur

Outils utilisés

Nous présentons ci-dessus toutes les technologies/outils utilisées et brièvement la raison des choix

- NetBeans : IDE Environnement de développement permettant de développer en Java
- GitHub : permettant de gérer les différentes versions/modifications du code. Travailler en parallèle et de fusionner nos avancements
- Java avec JDK 8 : langage orienté objet permettant une exécution indépendante de la plate-forme hardware
- Java FX : permettant de construire l'aspect graphique du Jeu et de gérer les actions de l'utilisateur (clique bouton, drag & drop, …)
- CSS : permet de changer l'aspect style de l'application

3. LIVRAISON

Les extensions

- **Autocomplétion du chemin**
  Nous avons remarqué que lors du parcours de la grille, l'utilisateur est tenté de sortir de la grille. Pour lui faciliter la tâche de relier deux cases, nous avons décidé de mettre une autocomplétion horizontale et verticale selon les actions de l'utilisateur

- **Charger la grille à partir d'un fichier**
  Pour simplifier la gestion des grilles, et pour ne pas stocker inutilement plusieurs grilles dans la mémoire, nous avons choisi d'avoir un fichier ".txt" correspondant à chaque grille dans lequel on peut puiser pour la grille initiale.

- **Création d'un menu pour gérer les niveaux**
  Le menu du démarrage permet à l'utilisateur de choisir la taille de la grille avec laquelle il veut jouer, et un autre menu permettant de revenir en arrière de sauvegarder la partie courante, des buttons permettant la navigabilité entre les différentes grilles de la même taille. Boutons du haut et boutons du bas

- **Gestion dynamique de la grille**
  Comme la difficulté est principalement délimitée par la taille de la grille et le nombre de paires symboles sur la grille. Nous avons choisi d'avoir des grilles allant 3x3 à 5x5. Nous avons aussi bien entendu différences configurations de grilles.

- **Sauvegarde de la partie courante**
  Pour permettre à l'utilisateur de reprendre sa partie là où il s'est arrêté, nous avons décidé de mettre en place l'option de sauvegarde. Il est de même important de pouvoir aussi charger la partie déjà sauvegardée, mais notre algorithme de chargement n'était pas assez optimal, et provoquer certains bugs d'affichages, on a décidé de le mettre à côté.

- **Au-dessus de la grille**
  Vous retrouvez un bouton permettant de sauvegarder la grille, un titre indiquant le niveau courant de la partie, un bouton permet de savoir la taille de la grille et de changer la taille de la grille.

- **En-dessous de la grille**
  Un bouton gauche permettant de passer au niveau précédant, un bouton au milieu permettant de réinitialiser la grille, et un dernier bouton permettant de passer si au niveau suivant. Le bouton permettant le passage au niveau précédant est activable uniquement s'il existe un niveau précédant. Cela est fait pour diminuer les erreurs de l'utilisateur.

Difficulté rencontrée

- **Définir le style de grille avec CSS**
  Il nous a fallu utiliser notre propre feuille de styles pour remplacer les styles de la feuille de style par défaut et ajouter vos propres styles. Cela nous a permis de mieux styliser les différents composants de notre grille.

- **Adaptation du code du glisser-déposer**
  Le code fourni enregistré le passage par une plusieurs fois dans le chemin. Ce qui présentait un principal désavantage lors du stockage de cette case dans le chemin courant. Il nous a fallu donc de l'adapter pour mieux l'utiliser.

- **Affichage du chemin (sens : verticale, horizontale et les 4 coins)**
  Dessiner le chemin courant était très difficile à utiliser, surtout que nous n'avons pas se limiter à charger des images qui contient un dessin. Donc il nous a fallu à se familiariser avec javafx.scene.shape est de trouver le meilleur de dessiner le chemin. La classe Polyline permet une meilleure flexibilité que la classe Line. Décider de la forme selon le parcours de la grille de l'utilisateur a nécessité une étude des relations entre les différentes cases du jeu surtout pour pouvoir comment dessiner les coins.

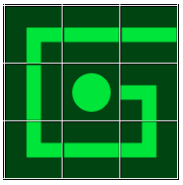

Figure 4 : Capture d'écran qui montre les différents sens que le chemin peu faire

- **Gestion d'erreur**
  Pour réduire et éviter les erreurs de l'utilisateurs, il nous a fallu faire plusieurs tests, régler plusieurs bugs, et même faire un crash test avoir d'autres groupes pour mieux anticiper les différentes erreurs qui peuvent se produire.

- **L'organisation de la fenêtre**
  Organiser la fenêtre en trois sections nous a fallu beaucoup de travail. Le BorderPane contient tous les composants. Nous avons donc implémenté à son centre notre grille qui est un GridPane. Dans la partie du haut, nous implémenté un HBox qui contient les boutons du "au-dessus" de la grille. Dans le bas, nous avons implémenté aussi un autre HBox qui contient les boutons du "en-dessous" de la grille.

- **Passer d'une grille de taille donnée à une autre**
  Comme ce passage a nécessité de changer la scène, et donc le modèle correspondant. Il nous a fallu donc avoir une deuxième scène qui permet de dissimuler à l'utilisateur ce passage. On a donc choisi d'avoir une scène qui correspond au menu du démarrage et la scène principale de la grille. Pour se rendre compte de ce fait, on a notamment beaucoup cherché sur la toile pour trouver cette solution adéquate.

Figure 5 : Capture d'écran de la page d'accueil qui permet de choisir la taille de la grille

CONCLUSION

Ce projet nous a permis de bien maîtriser le concept MVC et de mettre en pratique les bases du langage orienté objet Java. En aspect gestion de projet, ce projet nous a permis de s'organiser d'une manière à aboutir un projet qui réponds au besoin demander qui est de créer un jeu qui plaît à l'utilisateur. Ainsi qu'il nous a appris l'esprit d'équipe et en quoi il est efficace pour un résultat acceptable. Dans cette partie, nous avons étudié la faisabilité du projet de son aspect fonctionnel et technique. Tout en se basant sur ce que nous avons appris en cours et sur nos recherches individuelles sur internet.