Titanic
- Rapport -

Tristan Savaria et Jonathan Forget

4 novembre 2013

Table des matières

1 Description et utilisation des fonctions

1.1 Utilisateur

Le programme est une réplique du jeu de table Titanic de la compagnie Smart Games. Le programme charge un fichier texte au lancement et génère une grille de jeu. Le programme est capable d'importer un fichier .txt de format UTF-8 contenant une grille de jeu de dimension 5 par 5. Une ligne vide ou une ligne commencant par # est ignorée. Un tiret (-) représente une case vide, une lettre est un naufragé et un chiffre complété d'une flèche représente un bateau. Les flèches représentes la direction du bateau et les symboles possibles sont $\leftarrow$ $\rightarrow$ $\uparrow$ et $\downarrow$. Le joueur peut sélectionner un bateau avec un clique gauche de la sourit. Une fois sélectionné, les flèches du clavier peuvent être utilisés pour déplacer la bateau dans la même direction sous condition que la destination n'est pas bloqué ou hors grille. Lorsqu'un bateau ce déplace à côté d'un naufragé, le naufragé rentre dans le bateau et le navire est désormais impossible d'être déplacé. Le but du jeu consiste à sauver tout les naufragés.

1.2 Programmeur

La classe *ImporteurGrille* prend un int qui deviendra une constante privé pour forcer la dimension de la grille lors de la création de l'objet. La méthode *Chargement* prend un String représentant le chemin vers un fichier .txt. La méthode lis le flus dans le format UTF-8, ignore les lignes vides ou commencant par le symbole # , rajoute chaque caractère lut dans un tableau de char de deux dimensions et retourne le tableau.

La classe *EcranJeu* est un JFrame responsable de la fenêtre du programme, ainsi des menus disponibles au dessus de la fenêtre. EcranJeu instantie la classe Plateau, Plateau est une composante fille de EcranJeu.

Plateau est la grille de jeu de taille fixe. Plateau utilise un GridLayout et a des objets de type Case comme composantes filles. À partir d'un tableau de char de 2 dimensions, la méthode *GenerateGrid* instantie les objets Cases, les positionnes dans un tableau de Case de deux dimensions et finalise par rajouter les Cases comme composante de Plateau. La méthode *SetKeybind* lie un dictionnaire de touche d'entré du clavier à un dictionnaire d'Action, utilisé pour déplacé les bateaux selon les touches du clavier. Finalement, la classe vérifie si les mouvements des bateaux sont légaux.

Dans la classe *Case*, classe fille de JPanel, plusieurs constructeur sont disponibles. Pour les case vide sur la grille, le constructeur prend comme paramètre une référence du plateau, la coordonnée x et la coordonnée y de l'emplacement de la case sur la grille. Une case avec un Flottant prend les mêmes paramètres en plus d'une référence sur un objet de type Flottant. Une Case avec un flottant a également un MouseListener qui sert à la

sélection avec le clique gauche de la souris. La case teste le type de son attribut flot pour lancer l'appel d'affichage approprié selon si c'est un naufragé, un bateau ou une proue. La classe a un attribut de type Color qui est une couleur gris-argenté avec un alpha pour le carré de sélection.

La classe *Bateau* a comme attribut une direction, un naufragé et une référence vers une case contenant sa proue. La classe instantie sa propre case et sa propre proue. *Proue* n'est que la tête du bateau. Selon la direction son orientation change.

Le programme utilise un enum appelé *Direction*. Direction est utilise à plusieurs endroit, comme la classe Bateau, Proue et pour envoyer une commande de direction. L'enum a comme champs : "Haut", "BAS", "GAUCHE" et "DROITE" et associe chacun à un char symbolisant une flèche de la même direction en UTF-8. L'enum permet de bien représenter une direction, d'où le nom choisis.

2 Structures de données employées et justification

Aucune structure de données n'a été utilisé.

3 Discussion

Le projet était simple à complété, nous avons structuré le programme en plusieurs classes séparant la logique. Nous n'avons pas eu de problème avec la programmation orienté objet puisque nous avons tout les deux plus de 2 ans d'expérience en programmation orienté objet avec C# et C++. Le projet nous as permis de comprendre le fonctionnement particulier des composantes de Java Swing et de comment les dessiner. Quelques problèmes ont été rencontré, tel qu'inversé les composantes d'accès des tableau de deux dimensions. Également, le fichier d'exemple d'une grille de jeu était illisible au début de notre travail, mais nous avons réalisé que le format était en UTF-8. Lors de la lecture du fichier, chaque ligne débutait avec un point d'interrogation. Une petite recherche a dévoilé que c'était un symbole de Byte Order rajouté par l'infâme éditeur de texte : notepad.

4 Améliorations possibles

Il serait bien de pouvoir charger plusieurs fichiers de grille de jeu à partir d'un menu. Également, il serait d'une nette amélioration de pouvoir chargé plusieurs grilles avec un seul fichier. Aucune vérification est fait sur les dimensions de la grille dans le fichier de grille, donc une erreur ce produit et le programme sera terminé. Plusieurs portions du code manipule des tableaus de deux dimensions et ce n'est pas tout à fait clair, nous pourrions passer davantage de temps pour améliorer la cohérence de la classe Plateau.

Nous avons intégré des menus dans le jeu, mais ils ne sont pas encore fonctionnelles, une deuxième phase du projet implémenterait des menus fonctionnelles.