Rapport de projet - Interfaces Graphiques

Rémi Lévy, Guillermo Morón Usón

21 mai 2023

Pour lancer le programme, voir le fichier README à la racine du projet.

1 Fonctionnement

Au premier lancement du programme le répertoire `$HOME/.energy/` est créé, ainsi que les sous-répertoire `editable/` et `playable/`. Ces dossiers serviront à stocker les niveaux jouables et éditables. Ainsi, pour supprimer complètement cette implémentation du jeu Energy, il suffit de supprimer le répertoire `$HOME/.energy/` et l'archive décompressée contenant le code source et executable. Pour lancer le jeu, il faut se placer à la racine de l'archive décompressée et exécuter la commande `./gradlew run`.

Après l'installation, une fenêtre s'affiche à l'écran vous proposant deux banques de niveaux, les niveaux *jouables* et les niveaux *éditables*. Les niveaux prédéfinis sont jouables mais non éditables, cependant tout nouveau niveau créé est jouable et éditable. Selon la banque sélectionnée, l'utilisateur peut tenter de finir le niveau ou éditer entièrement le niveau, les modifications étant effectives si le circuit est dans une configuration gagnante.

2 Packages

Le *package* principal est `energy`, dans lequel se trouve le fichier `App.java` et les *sous-packages* `model`, `view` et `controller`. Le fichier `App.java` est le point d'entrée du programme, celui qui construit l'application.

2.1 Model

Le *package* `model` contient les données de l'application, dont les objets les plus importants suivants :

- `Level` : encapsule un `Circuit` et son identifiant.
- `Circuit` : encapsule une collection de `Tile` connectées entre elles.
- `Tile` : encapsule une forme, une collection de côtés connectés, une `Position`, formant collectivement une matrice dans `Circuit`, ainsi qu'un `Component`, représentant les éléments posés sur les tuiles tels les lampes ou les sources d'énergie.

— `ReadOnlyCircuit` : représente un `Circuit` accessible uniquement en lecture. Est utilisé notamment pour mettre à jour la Vue.
— `EditableLevel` : définit ce qu'est un niveau éditable. Est utilisé notamment en tant que Modèle lorsque l'on édite un niveau. C'est le patron de conception Adapteur.
— `PlayableLevel` : définit ce qu'est un niveau jouable. Est utilisé en tant que Modèle lorsque l'on joue à un niveau.

La forme des tuiles et leurs composant sont encapsulés dans des énumérations et sont des attributs des tuiles. Ainsi, on évite la duplication de code dûe aux multiples combinaisons de forme/composant possibles.

2.2 View

Le *package* `view` s'occupe de l'affichage des composants sur l'interface graphique. Les objets les plus importants sont les suivants :

— `ScreenSwitch` : interface qui permet aux écrans de sélection de niveau, edition de niveau et jeu de passer d'un écran à l'autre selon l'état de l'application. La fonction `next(int, bool)` permet de montrer à l'écran la représentation du niveau d'identifiant donné en mode édition ou jeu selon la valeur du drapeau.
— `CircuitView` : affiche sous forme de plateau rectangulaire les tuiles d'un niveau. Les tuiles et la disposition sont récupérées à partir d'un `Circuit` accessible en lecture seule, un `ReadOnlyCircuit`, on limite ainsi la connaissance des données au strict nécessaire.
— `TileView` : permet de charger, placer et orienter les images constituant la représentation d'une `Tile` à l'écran. Les images ne sont chargées qu'au besoin et qu'une fois.
— `LevelView` : encapsule la représentation graphique d'un niveau, ajoutant au besoin les boutons nécessaires pour l'édition de niveau ainsi que les boutons de retour à l'écran principal. Observe le Modèle et met à jour la représentation du circuit lorsqu'il est notifié d'un changement d'état.

2.3 Controller

Le *package* `controller` contient les classes qui gèrent les entrées de l'utilisateur et agissent sur le modèle en conséquence. Lorsque l'une des ces actions se produit, les contrôleurs transmettent au Modèle les données et si son état change, notifie la Vue. Il y a deux contrôleurs, l'un pour l'édition de niveau et l'autre pour le jeu. Leur généralisation sert uniquement à déterminer la tuile sur laquelle l'utilisateur a agit.

— `LevelController` : un MouseInputAdapter qui permet de déterminer la position de la tuile sur laquelle l'utilisateur a cliqué. C'est la classe de base pour les contrôleurs de programme elle n'est pas utilisé directement.
— `EditorController` : c'est le contrôleur à utiliser lorsque l'utilisateur sélectionne un niveau à éditer. Il encapsule un `EditableLevel` qui n'autorise que des actions de modification du nombre de ligne ou colonnes, des

composants et des connexions des tuiles dans le circuit.

— `PlayableLevel` : c'est le contrôleur à utiliser lorsque l'utilisateur selectionne un niveau pour jouer. Il limite l'action de l'utilisateur sur le circuit à la rotation des tuiles en utilisant un `PlayableLevel` comme Modèle.