UPMC/Licence/Info/2I013
Flowdroid

Janvier 2015

Le jeu consiste à relier deux-à-deux certaines cases d'une grille. Pour relier deux cases, partant de l'une, on se déplace de case en case, horizontalement ou verticalement, jusqu'à atteindre l'autre. Le déplacement en diagonale est proscrit. Le jeu propose plusieurs paires de cases à relier (les cases à relier sont marquées par des couleurs). Les chemins reliant les cases de ces paires ne doivent pas se croiser. À la fin du jeu, les chemins doivent recouvrir l'ensemble de la grille.

1 Modèle du jeu

Le plateau de jeu est une grille sur laquelle figurent des cases préremplies. Nous utiliserons ici des couleurs, mais tout autre type de valeur peut convenir (des entiers, des caractères, etc.). Le nombre de couleurs n'est pas fixé (il est toutefois limité par la taille de la grille).

On pose qu'il existe exactement une seule paire de cases préremplies pour chaque couleur.

1.1 Les actions de jeu et leurs règles

On pose les trois *actions de jeu* suivante :

1. Choisir le départ.
2. Prolonger.
3. Arrêter.

1.1.1 «Choisir le départ»

Les cases préremplies ne peuvent être prolongées que dans un seul sens. Donc une case préremplie ne peut être choisie comme départ d'une prologation de chemin que si aucun chemin n'en part déjà.

Lorsqu'un chemin a déjà été entamé (voir ci-dessous «Arrêter»), l'extrémité d'une de ses portions qui n'est pas une case préremplie peut être choisie comme départ d'une prolongation de chemin.

Aucune autre case ne peut être choisie comme départ.

1.1.2 «Prolonger»

Cette action présuppose qu'un départ a été choisi. La case choisie comme chemin constitue une *position courante*. On donne au chemin une couleur identique à celle de sa case de départ.

D'une position courante, le chemin peut être prolongé vers toute case vide contigüe horizontalement (même ligne de la grille) ou verticalement (même colonne de la grille). Cette case constitue alors la nouvelle position courante d'où le chemin peut être itérativement prolongé.

Il existe un cas où un chemin peut être prolongé vers une case non vide : c'est le cas où cette case est l'autre extrémité d'une portion du même chemin (même couleur). Ce cas inclut celui où la «portion» est réduite à une seule case (préremplie).

Lorsqu'une case a été intégrée à un chemin, on considère qu'elle est remplie avec la couleur du chemin.

Remarque : ne pouvoir prolonger un chemin que sur une case vide ou sur la case à l'extrémité de l'autre portion du même chemin garantit qu'aucun chemin n'en croise un autre ni ne comporte de cycle.

1.1.3 «Arrêter»

On peut arrêter un chemin sur n'importe quelle position courante valide (case vide avant prolongation ou extrémité de l'autre portion du même chemin).

Remarques : il est ainsi possible de relier deux cases en construisant le chemin entre elles en plusieurs étapes. Par exemple, on part d'une case pour s'arrêter sur une case vide et part de l'autre case pour rejoindre celle sur laquelle on s'était arrêté complétant ainsi le chemin.

1.2 Fin du jeu

Le jeu s'arrête lorsque toutes les cases préremplies ont été reliées par un chemin et qu'il n'existe plus aucune case vide.

2 Une *API* pour le modèle du jeu

L'API du modèle doit fournir ce qui est nécessaire pour jouer une partie : *i.e.* résoudre une grille. Elle doit donc fournir des méthodes pour les actions de jeu en contrôlant le respect des règles. Elle fournit également une description de l'état du jeu (c'est-à-dire, ici, de l'ensemble des portions de chemins construites) ainsi qu'une indication de fin de partie. Enfin, il est demandé qu'elle fournisse aussi une représentation de l'état du jeu sous forme d'une chaîne de caractères.

L'API est donnée par la classe `FlowModel`.

2.1 Actions de jeu

Chaque action de jeu donne un statut d'exécution sous forme d'un booléen qui sera `true` si l'action a réussi et `false` sinon.

≪Choisir le départ≫

```
public boolean start(int x, int y)
```

Les arguments `x` et `y` donnent les coordonnées de la case choisie entamer ou reprendre la construction d'un chemin.

≪Prolonger≫

```
public boolean extend(int x, int y)
```

Les arguments `x` et `y` donnent les coordonnées de la case à ajouter au chemin que l'on est en train de construire.

≪Arrêter≫

```
public boolean stop(int x, int y)
```

Les arguments `x` et `y` donnent les coordonnées de la case choisie pour s'arrêter.

Supprimer

```
public void delPath(int x, int y)
```

Supprime la portion de chemin passant par (`x`,`y`).

2.2 État du jeu

Chemins en construction

```
public ArrayList<PathUnderBuilding> getPathesUnderBuilding()
```

Donne la liste les chemins en construction ou achevés.

La classe `PathUnderBuilding` contient la description d'un chemin sachant que celui-ci peut être

- soit à l'état initial et on ne connaît que les deux cases des extrémités qu'il aura ;
- soit partiellement construit et on connaît l'une ou (inclusif) l'autre de ses portions ;
- soit complètement construit et on en connaît toutes les cases.

Fin de partie

```
public boolean endOfGame()
```

Donne `true` si la partie est finie et `false` sinon.

Représentation textuelle

```
public String toString()
```

Donne une représentation «*ASCI art*» de la grille du jeu.

2.3 Constructeurs

On n'exige rien quant aux constructeurs que doit fournir l'API.

3 Données XML

Les puzzles à résoudre sont enregistrés dans des fichiers au format XML. L'élément racine du fichier est `flowpuzzles`. Il renferme une suite de descriptions de grilles à résoudre (éléments `flowpuzzle`). La taille de la grille est donnée par les attributs `width` et `height`. Un élément `flowpuzzle` contient une suite d'éléments `line`. Ces éléments décrivent les chemins à construire. Ils contiennent eux-mêmes trois éléments :

- `color` dont l'attibut `value` donne la couleur du chemin ;
- `src` dont les attributs `x` et `y` donnent les coordonnées d'une extrémité du chemin ;
- `dst` dont les attributs `x` et `y` donnent les coordonnées de l'autre extrémité du chemin ;

Remarque : les noms `src` et `dst` n'obligent pas le début ou la fin de la construction des chemins.

Exemple : fichier contenant la description d'une grille 5x5 et d'une grille 7x7

```
<flowpuzzles>
<flowpuzzle width="5" height="5">
 <line> <color value="red"/> <src x="0" y="4"/> <dst x="1" y="0"/> </line>
 <line> <color value="green"/> <src x="2" y="4"/> <dst x="1" y="1"/> </line>
 <line> <color value="bblue"/> <src x="2" y="3"/> <dst x="2" y="0"/> </line>
 <line> <color value="yellow"/> <src x="3" y="1"/> <dst x="4" y="4"/> </line>
 <line> <color value="orange"/> <src x="3" y="0"/> <dst x="4" y="3"/> </line>
</flowpuzzle>
<flowpuzzle width="7" height="7">
 <line> <color value="blue"/> <src x="0" y="0"/> <dst x="2" y="6"/> </line>
 <line> <color value="yellow"/> <src x="0" y="1"/> <dst x="5" y="1"/> </line>
 <line> <color value="red"/> <src x="3" y="1"/> <dst x="2" y="4"/> </line>
 <line> <color value="green"/> <src x="1" y="2"/> <dst x="1" y="6"/> </line>
```

```
 <line> <color value="orange"/> <src x="5" y="2"/> <dst x="5" y="5"/> </line>
</flowpuzzle>
</flowpuzzles>
```