Programmation avancée
Projet : Plumber

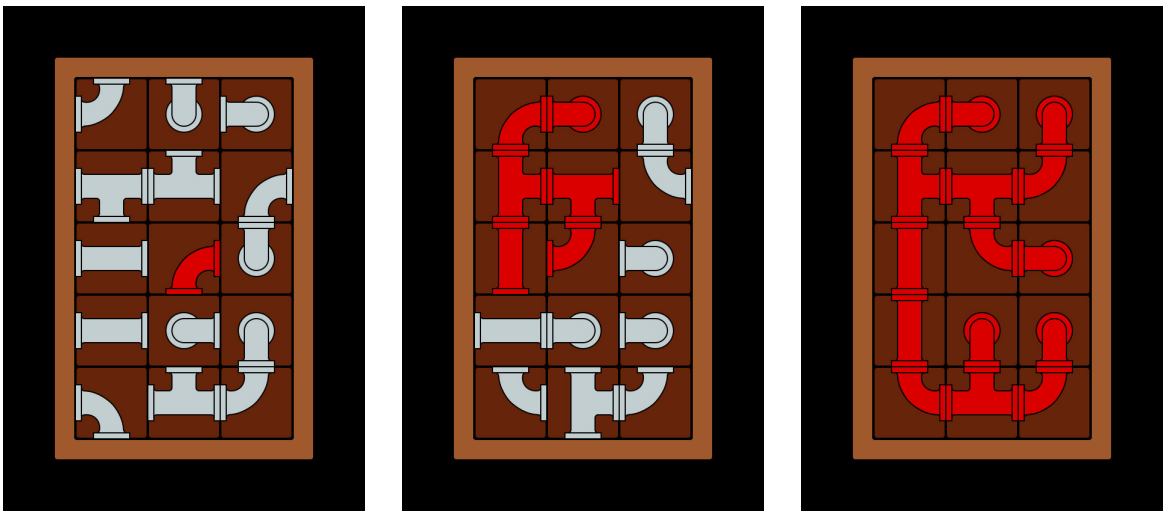
Le but de ce projet est d'écrire en `java`, à l'aide de la librairie `swing`, une application permettant de jouer au jeu vidéo "Plumber".

1 Plumber

Le jeu Plumber se joue sur un damier rectangulaire de taille quelconque. Sur les cases du plateau sont disposés des tuyaux de différentes sortes, que le joueur peut librement faire tourner d'un quart de tour. Le tuyau au centre du plateau est une source d'eau : l'eau se propage à partir de cette source vers tous les tuyaux connectés à celle-ci, directement ou par l'intermédiaire de tuyaux intermédiaires. Le but du jeu est de tourner les tuyaux de manière à ce que chaque tuyau soit connecté à la source, sans qu'aucune extrémité de tuyau ne soit face à du vide.

Au début du jeu, le plateau est dans une configuration gagnante, mais chaque tuyau subit une rotation aléatoire avant que le plateau ne soit présenté au joueur.

Voici par exemple l'état du jeu au début, au milieu et en fin de partie. Les tuyaux connectés à la source centrale sont ceux colorés en rouge :



2 Implémentation minimale du jeu Plumber en Java

Votre projet devra permettre de jouer au jeu Plumber sur un ensemble de configurations du plateau – des *niveaux* encodés par des fichiers textes, chaque fichier décrivant la configuration finale que devra retrouver le joueur. Les images de cet énoncé ne font qu'illustrer le principe du jeu et de son interface et ne sont données qu'à titre indicatif : il ne vous est pas demandé de les reproduire de manière exacte dans votre implémentation, mais seulement de respecter la spécification détaillée dans cette section.

Le design de l'interface, les graphismes, le placement des différents éléments, etc., sont libres, mais les contraintes de comportement ci-dessous devront toutes être respectées, à commencer par la première d'entre elles : le jeu doit être intégralement jouable à la souris, sans jamais avoir besoin de recourir au clavier.

2.1 L'écran de jeu

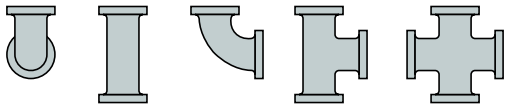
L'écran d'accueil du programme devra permettre de sélectionner un niveau parmi un ensemble de niveaux numérotés par de simples entiers. Une fois un niveau choisi, cet écran sera remplacé par un écran de jeu, affichant le plateau de jeu après une rotation aléatoire de chaque tuyau. L'écran de jeu devra également contenir un bouton permettant de revenir à l'ecran d'accueil, ainsi que tout autre él'ement que vous jugerez utile (aide, solution, score, etc).

A chaque instant du jeu, les tuyaux connectés à la source seront affichées en une couleur singulière, les autres étant affichés d'une couleur neutre. Le joueur devra pouvoir faire subir une rotation d'un quart de tour à un tuyau en cliquant sur celui-ci : l'affichage devra être mis à jour après chaque rotation.

Le programme devra bien sûr pouvoir détecter si la configuration courante est gagnante selon la règle du jeu présentée dans l'introduction, et si c'est le cas, en informer le joueur, passer s'il existe au niveau suivant, ou revenir à l'écran d'accueil si le niveau joué est le dernier.

2.2 Fichiers de niveaux

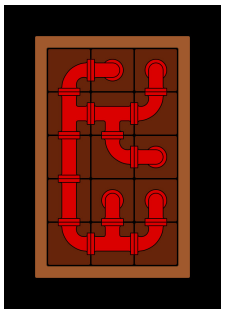
A rotation près, il n'y a que quatre sortes de tuyaux que nous appellerons `END`, `LINE`, `FORK`, et `CROSS`. La figure ci-dessous montre ces éléments dans ce que nous considérerons comme leur orientation par défaut :



Un *fichier de niveau* est un fichier texte dont le nom est de la forme `pipe`*num*`.p` où *num* est le numéro du niveau. Il est structuré de la manière suivante :

- Le fichier commence par deux entiers h w, suivis de $h \times w$ chaînes séparées par des espacements. Ces chaînes spécifient le contenu de chaque case du plateau dans sa configuration gagnante.
- Une chaîne commençant par `E`, `L`, `T`, `F` ou `C`, spécifie sur une case du plateau la présence d'un tuyau ayant ce caractère comme première lettre. Ce caractère est immédiatement suivi de `0` (zéro), `1`, `2` ou `3`. Ce nombre indique le nombre de rotations d'un quart de tour vers la droite que doit subir cet élément dans son orientation par défaut pour atteindre son orientation dans la solution du niveau [1].

Voici par exemple le contenu du fichier encodant le niveau présenté dans l'introduction :



```
5 3
T1 E3 E2
F0 F1 T3
L0 T0 E3
L0 E2 E2
T0 F3 T3
```

Le choix d'un niveau dans l'écran d'accueil sera suivi de la lecture du fichier de niveau associé [2], afin de reconstruire en mémoire la configuration gagnante du niveau, avant de mélanger celle-ci par rotation et de soumettre le résultat au joueur.

1. Cette information est inutile pour `CROSS`, et `0` ou `1` suffisent pour `LINE`, mais il est plus simple d'avoir un encodage uniforme pour toutes les sortes de tuyaux.
2. La classe prédéfinie `Scanner` permet de lire le contenu d'un fichier mot par mot, j'en reparlerai en cours.

3 Extensions possibles

Afin de consolider votre note, de nombreuses extensions de la partie minimale ci-dessus sont envisageables. La liste ci-dessous est loin d'être exhaustive, et vous pouvez me soumettre en séance toute autre idée.

3.1 Cases vides

Cette extension est très simple : certaines cases du plateau pourront être vides. Dans un fichier de niveau, un simple point (`.`) non suivi d'un chiffre permet de représenter cette absence de contenu.

3.2 Mise à l'échelle de l'affichage

Les dimensions de l'écran de jeu ne devront pas excéder des valeurs maximales, choisies de manière à ce qu'il ne recouvre pas une trop grande portion de l'écran de la machine. L'échelle d'affichage sera adaptée de manière à respecter cette contrainte.

3.3 Éditeur de niveau

L'écran d'accueil pourra donner la possibilité de jouer à l'un des niveaux de manière ordinaire, mais aussi de passer en mode édition. Les choix d'interfaçage pour effectuer les actions décrites ci-dessous sont libres, mais toutes devront pouvoir être effectuées à la souris de façon fluide. Toute action irréversible – modification effective d'un fichier de niveau, abandon de modifications – devra être confirmée par l'utilisateur.

En mode édition, l'utilisateur doit pouvoir depuis l'écran d'accueil vider un niveau de tous ses tuyaux sans modifier sa géométrie, altérer la géométrie d'un niveau en le vidant implicitement de tous ses éléments – les deux actions mettrons bien sûr à jour le fichier du niveau – ou éditer un des niveaux sans encore modifier son fichier.

Une demande d'édition de niveau entraînera le passage à un écran de jeu affichant, si le niveau est non vide, sa solution. L'utilisateur devra pouvoir librement modifier le contenu du plateau. Lors d'une demande de retour à l'écran d'accueil, si le plateau est dans un état gagnant, le fichier du niveau devra être mis à jour. Sinon, les modifications faites sur le niveau seront abandonnées.

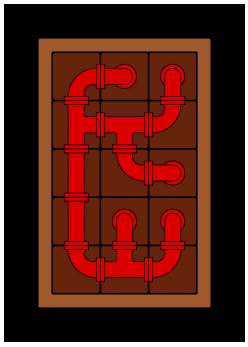
Cette extension ne suppose pas d'écrire un second programme, mais de réutiliser le mécanisme d'affichage du jeu en interprétant différemment les événements de la souris.

3.4 Historique

Deux boutons permettront de défaire ou de rétablir les dernières rotations de tuyaux d'une partie en cours, ou encore d'annuler ou rétablir une suite de modifications d'un niveau en cours d'édition, voir ci-dessus.

3.5 Niveaux aléatoires

Vous pouvez ajouter la possibilité pour l'utilisateur de tirer à tout moment dans l'éditeur de niveau une configuration aléatoire de sources et de tuyaux qui est aussi une solution. Plusieurs algorithmes de génération sont envisageables. Les tuyaux pourront par exemple tracer un labyrinthe parfait (deux points quelconques du labyrinthe sont reliés par un et un seul chemin) :



```
+---+---+---+
|       |   |
+   +---+   +
|           |
+   +   +---+
|   |       |
+   +---+---+
|   |   |   |
+   +   +   +
|           |
+---+---+---+
```

4 Critères d'évaluation

4.1 Qualité de la conception objet et du code

Ce cours est un cours de programmation objet. Servez-vous de l'héritage, des interfaces de la liaison dynamique et des énumérations partout où ces éléments améliorent la factorisation et la clarté du code. Prenez le temps de réfléchir à la hiérarchie nécessaire, et à la répartition des données et des traitements, et ne programmez pas (jamais) par copier-coller.

4.2 MVC

Je vous parlerai en cours de l'architecture MVC. Ce type de projet se prête idéalement à son usage : votre code sera à la fois mieux organisé, plus facile à développer, plus facile à débugger, et valorisé par le choix de cette architecture si elle est réalisée dans sa forme la plus académique.

4.3 Ergonomie de l'interface

Votre interface doit être souple, intuitive, naturellement utilisable par un utilisateur même s'il la découvre pour la première fois.

Une image `png`, sa version `svg` et quelques fichiers de niveau vous sont fournis. L'image contient tous les éléments nécessaires pour afficher les éléments de l'écran de jeu sans perdre de temps à les dessiner – l'esthétique des graphismes est sans importance pour ce projet (elle est toujours améliorable) et ne sera pas pris en compte dans l'évaluation.

5 Modalités

5.1 Groupes

Le projet sera de préférence réalisé en binôme. La répartition des tâches au sein d'un groupe doit être raisonnablement équilibrée. En cas de déséquilibre avéré, les notes finales pourront être individualisées.

5.2 Individualité de chaque projet

De manière évidente, votre code doit être strictement personnel. L'adaptation d'exemples repris sur le Swing Tutorial d'Oracle ou dans les introductions des API des classes est tolérée, mais pas la reprise de code trouvé sur le web ou venant d'une quelconque "aide" trouvée sur un forum, encore moins le partage de code entres groupes[3]. Il relève de votre responsabilité de faire en

3. La soutenance de projet est un examen comme un autre. Le plagiat à un projet constitue une fraude aux examens.

sorte que votre code reste inaccessible aux autres groupes : par exemple, si vous vous servez d'un dépôt `git`, ce dépôt doit être privé.

Pour certaines des extensions, vous pouvez reprendre des algorithmes connus [4], à condition que leur implémentation soit la vôtre et que vous citiez clairement vos sources.

5.3 Forme du rendu

Les dates de rendu et de soutenance seront précisées ultérieurement. Votre rendu consistera en :

- un code-source écrit en `java`, compilable et utilisable tel quel sous Linux et/ou sur les ordinateurs de l'UFR,
- un fichier texte nommé README contenant vos noms, prénoms et numéros d'étudiant,
- un rapport de quelques pages au format PDF décrivant votre projet, et expliquant et justifiant les choix de conception ou d'implémentation,
- tout autre fichier nécessaire à la compilation et à l'exécution.

Tous ces éléments seront placés dans une unique archive compressée en `.tar.gz`.

L'archive devra s'appeler *`nom1-nom2`*`.tar.gz`, et s'extraire dans un répertoire *`nom1-nom2/`*, où *`nom1`* et *`nom2`* sont les noms des deux personnes constituant le groupe.

Par exemple, si vous vous appelez Denis Diderot et René Descartes, votre archive devra s'appeler `diderot-descartes.tar.gz` et s'extraire dans un répertoire `diderot-descartes/`.

4. *e.g.* `https://fr.wikipedia.org/wiki/Modélisation_mathématique_d'un_labyrinthe` pour la génération aléatoire de niveau par labyrinthe parfait.