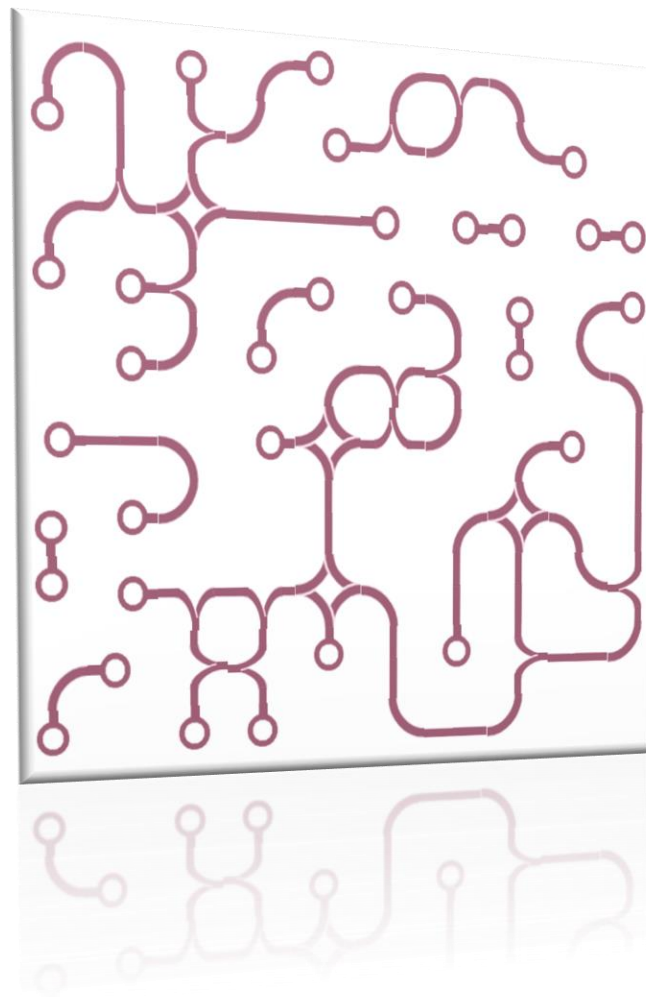




GUIDE UTILISATEUR

Phine Loops



Abderraouf HADDAD
Julien SAUSSIER



SOMMAIRE

SOMMAIRE.....	1
0. Historique du document	2
1. Préambule.....	3
1.1 Contexte	3
1.2 Objectifs du document.....	3
2. Interface Graphique.....	4
3. Générateur de niveau.....	5
4. Vérificateur de solution	6
5. Solveur de niveau	7
6. À vous de jouer.....	9
7. Annexes	10
Format fichier texte.....	10

0. Historique du document


Informations générales

Domaine :	Programmation Java Avancée
Rédacteur(s) :	Abderraouf HADDAD – Julien SAUSSIER
Valideur(s) :	Florian SIKORA

Historique des mises à jour

Version	Date	Description
0.1	24/01/18	Création du document
1.0	27/01/18	Modification suite changement code
1.1	28/01/18	Version finale

Documents de référence

Nom	Lien
Projet Phine Loops	MyCourse
Fichier test interface	 Solution.txt



1. Préambule

1.1 Contexte

Dans le cadre d'un projet universitaire en rapport avec la Programmation Avancée en Java, il nous a été demandé de développer une version amateur du fameux jeu « [Infinity Loop](#) » .

Le jeu se joue en plusieurs niveaux, un niveau consiste en une grille où chaque case comporte une pièce (la case vide est une pièce qu'on appellera « void »). Une pièce peut se connecter à 1 jusqu'à 4 de ses voisins. Une pièce peut subir des rotations de 90 degrés. Un niveau est résolu si les extrémités de chaque pièce sont reliées à une extrémité d'une autre pièce sur la case voisine correspondante.

Voilà ce qui était attendu à l'issu de ce projet :

1. Un générateur de niveaux.
2. Un vérificateur de solution.
3. Un solveur de niveau.
4. Une visualisation d'un niveau.

1.2 Objectifs du document

L'objectif du document est de présenter à l'utilisateur les différentes options/utilisations du jeu que nous avons codé et de définir précisément les manières dont un utilisateur pourra se servir de notre projet etc.

2. Interface Graphique

L'interface graphique est un intermédiaire auquel nous ferons systématiquement appel afin de vérifier les autres fonctions et éviter la vérification via fichiers *.txt*, nous avons jugé plus judicieux de commencer par présenter cette dernière.

Pour faire appel à l'interface graphique, il suffit d'exécuter la commande suivante sur la ligne de commandes¹ :

java -jar projetHaddadSaussier.jar --interface file

Où *file* est le fichier contenant la grille à afficher graphiquement (sans double quote)

Le message suivant devrait s'afficher :

```
$ java -jar projetHaddadSaussier.jar --interface "Level0.txt"
Launching the interface...
Level0.txt
```

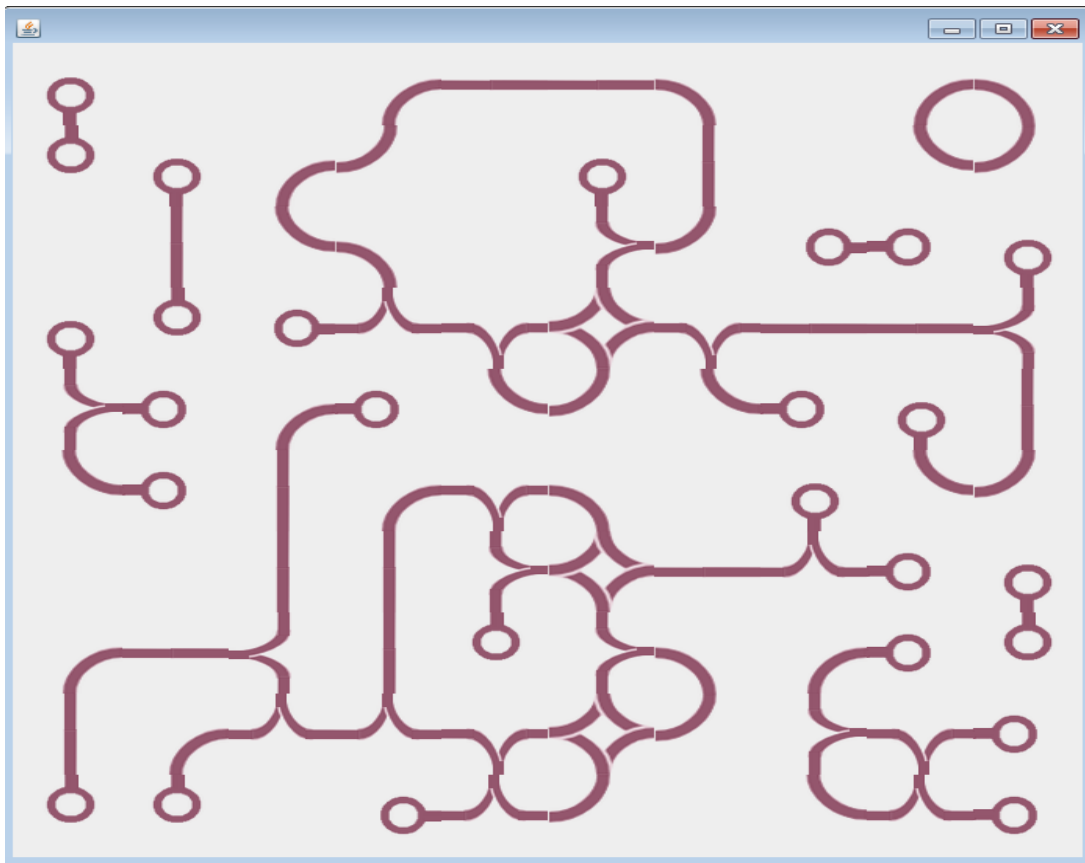
En cas d'erreur de saisie dans les commandes, un message d'erreur s'affiche ainsi que l'aide du programme :

```
Running phineloop generator.
Error parsing commandline : Missing mandatory --output argument.
usage: phineloopgen
-a,--algorithm <arg>   Algorithm of choice of piece
-c,--check <arg>       Check whether the grid in <arg> is solved.
-g,--generate <arg>    Generate a grid of size height x width.
-h,--help              Display this help
-i,--interface <arg>   Launching the interface of the <arg> grid
-o,--output <arg>       Store the generated or solved grid in <arg>. (Use
                        only with --generate and --solve.)
-s,--solve <arg>        Solve the grid stored in <arg>.
-t,--threads <arg>     Maximum number of solver threads. (Use only with
                        --solve.)
-x,--nbcc <arg>         Maximum number of connected components. (Use only
                        with --generate.)
```

Nous n'allons pas trop nous attarder sur cette partie, puisque nous l'occasion de tester un affichage sur l'interface graphique lors de la présentation des autres modules.

Néanmoins, pour le fichier texte disponible sur les documents de référence, qui contient la description d'une grille 10x10 déjà résolue, voici ce qui devrait être affiché :

¹ Il faudra se positionner sur le répertoire contenant le JAR, grâce à la commande « cd CHEMIN »



3. Générateur de niveau

Le générateur est la base du jeu, il permet de générer une grille, avec une largeur & hauteur qu'on peut spécifier.

La commande pour exécuter le générateur sera (sur ligne de commande) :

java -jar projetHaddadSaussier.jar --generate wxh --output file

Où *file* est le nom du fichier de sortie, *w* et *h* les dimensions.

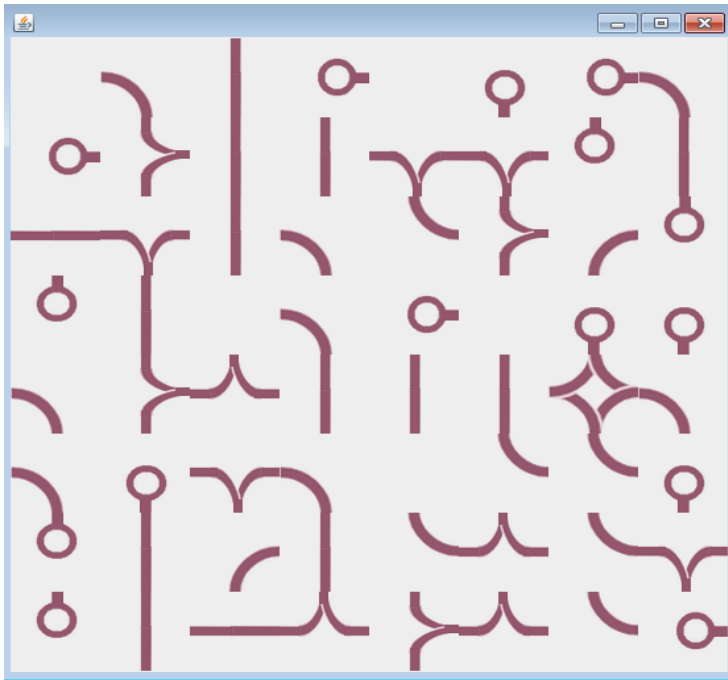
À la fin de l'exécution, le générateur crée un fichier texte « file » contenant une description de la grille.

Exemple sur la figure suivante (les doubles quotes ne sont pas nécessaires):

```
Raouf@Raouf-Z50 MINGW64 /d/projetHaddadSaussier/projetHaddadSaussier/target (master)
$ java -jar projetHaddadSaussier.jar --generate 20x20 --output "Level0.txt"
Running phineloop generator.
```

Le fichier ***Level0*** sera créé sur le même répertoire que le JAR.

On peut voir la grille générée en faisant appel à l'interface en donnant « Level0 » comme fichier d'entrée :



On peut aussi choisir un nombre de composante qui soit de 1,2 ou 4.

Pour cela, il suffit de changer la commande en rajoutant l'option nbcc :

```
java -jar projetHaddadSaussier.jar --generate wxh --output file [ --nbcc c ]
```

où c = 1, 2 ou 4.

4. Vérificateur de solution

Le Vérificateur de solution, qu'on appellera « Checker », permet de vérifier si une grille décrite dans un fichier est solution ou pas (toutes les pièces sont connectées).

La commande pour exécuter le vérificateur sera (sur ligne de commande) :

```
java -jar projetHaddadSaussier.jar --check file
```

Où *file* est le nom du fichier d'entrée.

Si nous testons le Checker avec le fichier « Level0 » créé ci-dessous, nous devrions avoir une réponse négative (false) :

```
$ java -jar projetHaddadSaussier.jar --check "Level0.txt"  
Running phineloop checker.  
SOLVED: false
```

Par contre, si nous essayons le checker avec le fichier « Solution.txt » qui se trouve dans les documents de référence, la réponse devrait être positive (true) :

```
$ java -jar projetHaddadSaussier.jar --check "Solution.txt"
Running phineloop checker.
SOLVED: true
```

5. Solveur de niveau

Le Solveur de niveau est le module le plus important de ce projet, il permet de résoudre une grille donnée en entrée sous forme de fichier texte.

La commande pour exécuter le vérificateur sera (sur ligne de commande) :

java -jar projetHaddadSaussier.jar --solve file --output filesolved

On peut essayer de résoudre la grille « Level0.txt » donné en exemple ci-dessus :

```
$ java -jar projetHaddadSaussier.jar --solve "Level0.txt" --output "Solved0.txt"
Running phineloop solver.
SOLVED: true
```

Maintenant, si on appelle le vérificateur sur « Solved0.txt », cela devrait donner une réponse positive :

```
$ java -jar projetHaddadSaussier.jar --check "Solved0.txt"
Running phineloop checker.
SOLVED: true
```

Maintenant, on peut s'amuser et essayer de voir si on peut générer une grille 10x10 à 4 composantes connexes, la résoudre et l'afficher :

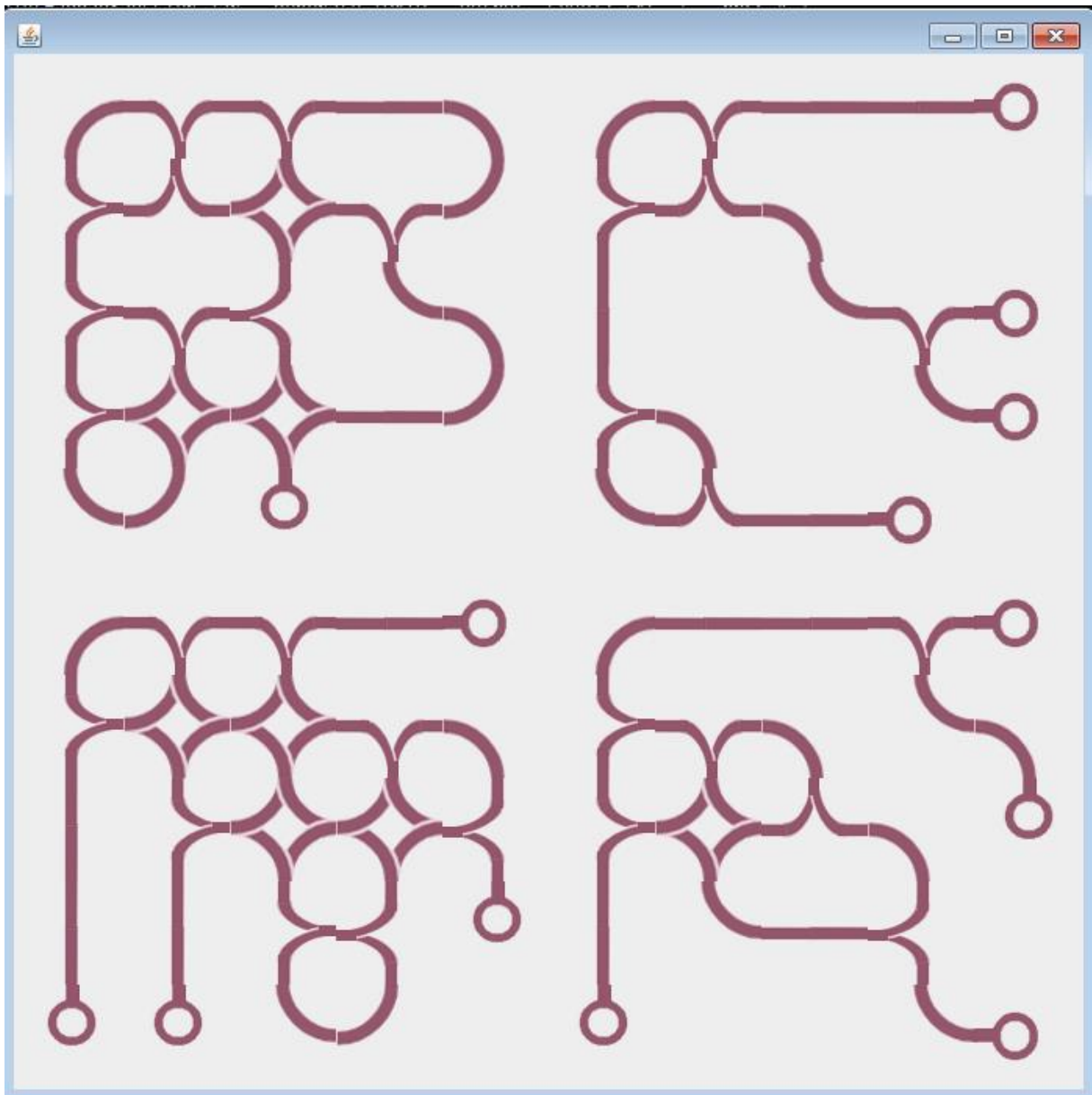
```
Raouf@Raouf-Z50 MINGW64 /d/projetHaddadSaussier/projetHaddadSaussier/target (master)
$ java -jar projetHaddadSaussier.jar --generate 10x10 --output "Level2.txt" [ --nbcc 4 ]
Running phineloop generator.
```

```
$ java -jar projetHaddadSaussier.jar --solve "Level2.txt" --output "Solved2.txt"
Running phineloop solver.
SOLVED: true
```

On s'accroche et ...

```
$ java -jar projetHaddadSaussier.jar --interface "Solved2.txt"
Launching the interface...
Solved2.txt
```

BIM ! Résultat next page.



A noter que l'on peut choisir un algorithme de résolution précis à l'aide de l'option : -a (--algorithm), les arguments possibles sont 0,1 ou 2 avec:

- 0 : solveur gauche-droite / haut-bas en récursif simulé
- 1 : solveur droite-gauche / bas-haut en récursif
- 2 : solveur droite-gauche / bas-haut en récursif simulé

6. À vous de jouer

Afin de jouer une partie, il faut commencer par générer un niveau en choisissant la largeur et hauteur de votre grille grâce au générateur.

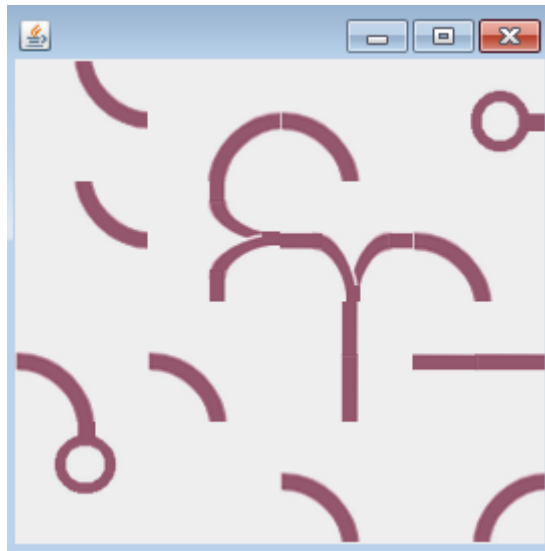
Ensuite, il faudra appeler l'interface sur le fichier généré par le générateur.

Enfin, vous avez juste à cliquer sur les pièces pour les faire pivoter, et quand vous résolvez la partie, un message, écrit et vocal, s'affichera 😊

Exemple :

```
Raouf@Raouf-Z50 MINGW64 /d/projetHaddadSaussier/projetHaddadSaussier/target (master)
$ java -jar projetHaddadSaussier.jar --generate 4x4 --output "Level3.txt"
Running phineloop generator.
Raouf@Raouf-Z50 MINGW64 /d/projetHaddadSaussier/projetHaddadSaussier/target (master)
$ java -jar projetHaddadSaussier.jar --interface "Level3.txt"
Launching the interface...
Level3.txt
```

Cela nous donne :



En jouant la partie et en résolvant la grille, voici ce qu'on obtient :



7. Annexes

Format fichier texte²

Les fichiers générés par le générateur et par le solveur doivent respecter le format suivant.

- première ligne : largeur de la grille w
- deuxième ligne : hauteur de la grille h
- Ligne 3 à ligne w · h : numéro-pièce-(i, j) orientation-pièce-(i, j)

Où « numéro-pièce-(i, j) » correspond à l'identifiant numérique du type de la pièce positionné sur la ligne i et la colonne j et « orientation-pièce-(i, j) » correspond à l'orientation de la même pièce (voir tableau ci-dessous).

pièce		numéro-pièce	orientation-pièce
emplacement vide		0	0
pièce 1 connection nord	┆	1	0
pièce 1 connection est	-	1	1
pièce 1 connection sud	┆	1	2
pièce 1 connection ouest	-	1	3
pièce 2 connections nord-sud	┆	2	0
pièce 2 connections est-ouest	-	2	1
pièce 3 connections (T) nord-est-ouest	┆	3	0
pièce 3 connections (T) nord-sud-est	┆	3	1
pièce 3 connections (T) est-sud-ouest	┆	3	2
pièce 3 connections (T) nord-sud-ouest	┆	3	3
pièce 4 connections nord-sud-est-ouest	┆	4	0
pièce L nord-est	┆	5	0
pièce L est-sud	┆	5	1
pièce L sud-ouest	┆	5	2
pièce L ouest-nord	┆	5	3

² Source : Lien du projet disponible sur les documents de référence