Première partie

1. Différents types de labyrinthes

Reconnaître parmi les labyrinthes précédents, ceux qui sont parfaits ou non.

2. Une première idée pour générer un labyrinthe

Cette partie a pour but de vous familiariser avec les manipulations utiles par la suite du projet. Prenez du temps pour tester les programmes, les modifier ....

a) Entreprise de démolition

répertoire `demolition/`

Pour générer un labyrinthe, on peut partir d'un labyrinthe rempli de murs et les casser :

```
import numpy as np
import matplotlib.pyplot as plt
"""
Nombre de cases du labyrinthe en ligne  et en colonne
"""
n = 4
p = 7
"""
Un labyrinthe est initialement rempli de murs
"""
labyrinthe = np.zeros((n,p),int)
"""
On casse un mur
"""
labyrinthe[2,1] = 1
"""
On affiche le resultat
"""
fig = plt.figure()
im = plt.imshow(labyrinthe, interpolation='None', cmap='gist_gray')
plt.show()
```

Quels murs seront cassés si j'écris cela ?

```
for j in range(p):
  labyrinthe[2,j] = 1
```

Remarque :
Vous remarquerez que la fenêtre graphique possède une petite barre d'outils qui permettent quelques actions sur le dessin :  . En particulier la sauvegarde sous forme d'un fichier images (.png ...)

b) Vers un premier algorithme ...

répertoire `premierAlgorithme/`

Récupérer le fichier `premierAlgorithme.py`.
Vous pouvez le lancer, changer les paramètre `n`, `p`, `murs_a_casser`.
Vous devez savoir expliquer son fonctionnement.

Remarque :
La ligne `np.savetxt("premierAlgorithme.txt",labyrinthe,fmt="%d")` permet de sauvegarder dans un fichier le labyrinthe.
Ceci permettra de sauvegarder vos labyrinthes sous forme d'un fichier texte.

c) Animons la démolition

Pour effectuer une animation, il faut sauvegarder tous les labyrinthes intermédiaires. Ensuite on les affiche les uns après les autres pour réaliser l'animation voulue. Nous allons modifier le script python précédent en ajoutant les lignes suivantes (au bon endroit) :

- `import matplotlib.animation as animation` dans l'entête pour bénéficier des fonctions d'animation,
- On stockera les images dans une liste (initialement vide).
  On ajoute donc avant la boucle `images = []`
- À la fin de chaque tour de boucle, on sauvegarde l'image du labyrinthe en cours :

```
im = plt.imshow(np.copy(labyrinthe), interpolation='None', cmap='gist_gray')
images.append([im])
```

- On dessine l'animation

```
ani = animation.ArtistAnimation(fig, images, interval=1, blit=True,repeat_delay=1000)
plt.show()
```

Remarque :
– Le script python est disponible sous le nom `[anim]premierAlgorithme.py`.
– Vous remarquerez qu'une ligne permet de sauvegarder le fichier sous forme d'une vidéo (attention cela peut ralentir la génération du labyrinthe - désactiver la sauvegarde en mettant un # devant la ligne pour les gros labyrinthes)

3. Une autre idée

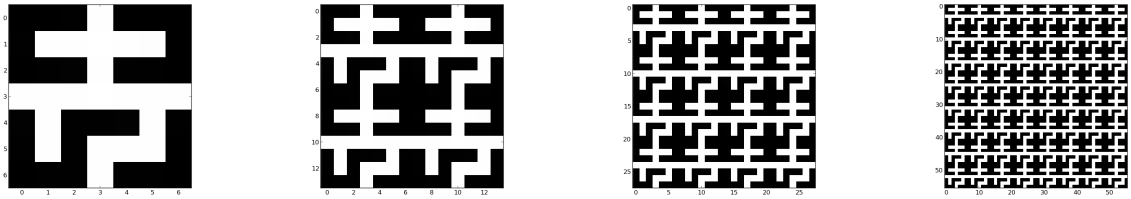
répertoire `fractal/`

... c'est ici que commencent les choses sérieuses.

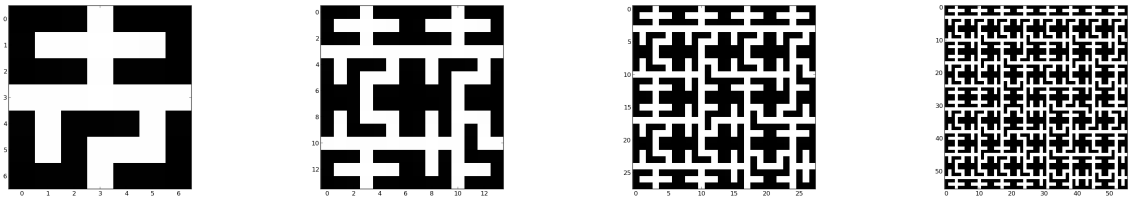
Comparer les dessins suivants et en déduire un méthode de génération de labyrinthes.

Le but est donc d'écrire un algorithme permettant la génération de ce type de labyrinthe.

- Série 1 :



- Série 2 :



```
import numpy as np
import matplotlib.pyplot as plt
fig = plt.figure()
labyrinthe = np.loadtxt("base7x7.txt")
#
#
#Votre algorithme ici
#
#
im = plt.imshow(labyrinthe,
    interpolation='None',
    cmap='gist_gray')
plt.show()
```

La structure du fichier est la suivante :

On commence par charger un fichier de base,

on effectue des opérations pour construire un labyrinthe plus grand,

on affiche le résultat.

Ressources :

- Sur les labyrinthes fractals (attention la représentation des labyrinthes n'est pas tout à fait la même que nous) :
  - `http://www.astrolog.org/labyrnth/algrithm.htm#perfect`
  - `http://www.mathcurve.com/fractals/hilbert/hilbert.shtml`
  - `http://www.mathcurve.com/fractals/peano/peano.shtml`
- Opération sur les matrices :
  - `http://docs.scipy.org/doc/numpy/reference/generated/numpy.vstack.html`
  - `http://docs.scipy.org/doc/numpy/reference/generated/numpy.hstack.html`
  - `http://docs.scipy.org/doc/numpy/reference/generated/numpy.rot90.html`
  - `http://docs.scipy.org/doc/numpy/reference/generated/numpy.fliplr.html`
  - `http://docs.scipy.org/doc/numpy/reference/generated/numpy.flipud.html`
- Vous trouverez des labyrinthes de base dans le répertoire `fractal/bases/`
  Ce sont des fichiers textes, vous pouvez en créer d'autres vous même.

```
# base7x7.txt
0 0 0 1 0 0 0
0 1 1 1 1 1 0
0 0 0 1 0 0 0
1 1 1 1 1 1 1
0 1 0 0 0 1 0
0 1 0 1 1 1 0
0 0 0 1 0 0 0
```

```
  # hilbert.txt
0 1 0 0 0
0 1 1 1 0
0 0 0 1 0
1 1 1 1 0
0 0 0 0 0
```

```
#base9X9.txt
0 0 0 0 0 0 0 0 0
0 1 1 1 1 1 0 1 0
0 1 0 1 0 1 0 1 0
0 1 0 1 0 1 1 1 0
0 1 0 0 0 0 0 1 0
0 1 1 1 0 1 1 1 0
0 1 0 1 0 0 0 1 0
0 1 0 1 1 1 0 1 0
0 0 0 0 0 0 0 0 0
```