Projet 2 : Paradoxes Graphiques et Stéganographie

Ce projet comporte trois parties. Les deux premières sont indépendantes, et la dernière dépend des précédentes.

Dans le projet les images seront à choisir au format png.

1 Paradoxe du photomaton

Dans cette partie, on partira d'une image que vous choisirez, qui doit avoir un nombre pair de pixels en hauteur et en largeur. L'opération à implémenter sur cette image est la suivante :

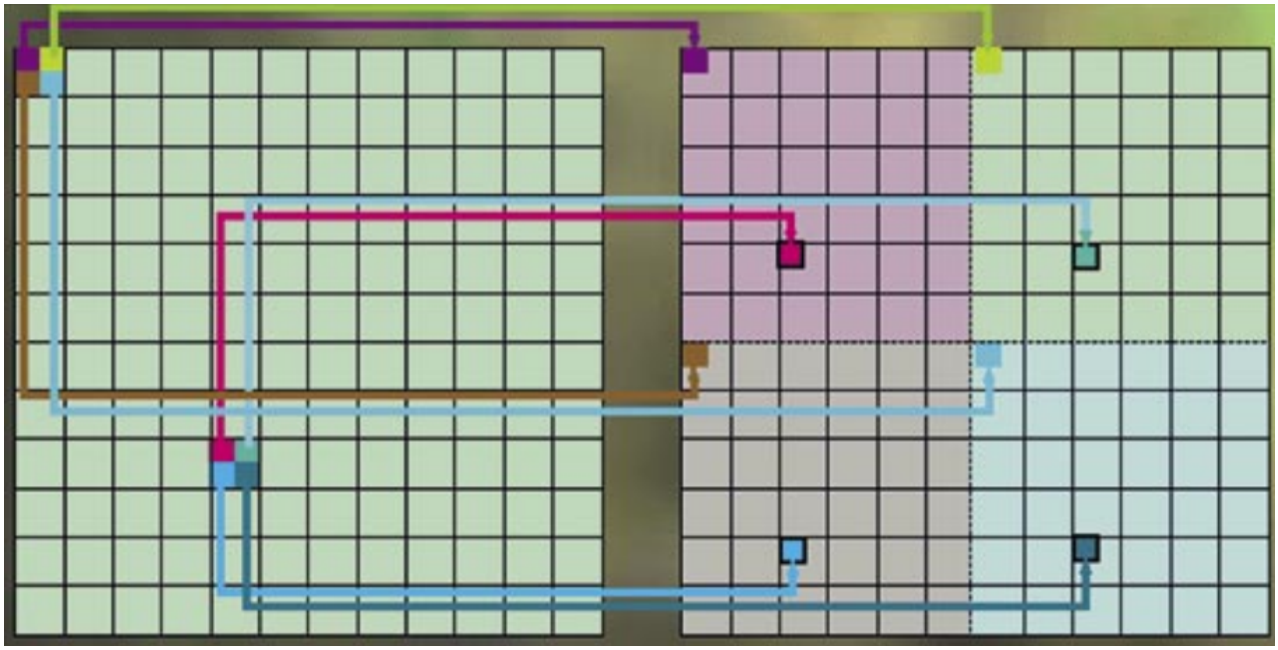

Image issue de l'article de J.P. Delahaye et P. Mathieu :
http://www.lifl.fr/~jdelahay/dnalor/ImagesBrouillees.pdf

Le but est d'écrire une fonction Python prenant en paramètre une image, et qui applique cette opération à l'image jusqu'à retrouver l'image de départ. On affichera toutes les étapes intermédiaires.

On notera qu'il peut y avoir *beaucoup* d'étapes ; ce nombre est fortement diminué pour des images carrées, avec un nombre puissance de deux de pixels en largeur et hauteur.

2 Paradoxe du boulanger

De la même façon, implémenter la transformation du boulanger :

- on étire l'image : chaque pixel d'une ligne paire de coordonnées (i, j) va en $(i/2, 2j)$, et chaque pixel d'une ligne impaire de coordonnées (i, j) va en $((i-1)/2, 2j+1)$.
- on replie l'image : la moitié droite de l'image est inversée (comme dans un miroir), et placée sous la moitié de gauche.

On affichera toutes les étapes, jusqu'à retrouver l'image de départ.

Là encore, on utilisera des images carrées dont le nombre de pixel par côté est une puissance de deux.

3 Application à la stéganographie

La stéganographie est l'art de cacher des messages dans des images.

On utilisera une méthode très simple pour cacher notre message :

- on traduit notre message en binaire (on pourra utiliser les fonctions proposées), en ajoutant le caractère nul `'\0'` à la fin
- chaque composante du i-ième pixel de la première colonne est remplacée par le i-ième bit du message.

Écrire des fonctions permettant de cacher un message dans une image, et permettant de retrouver un message caché.

Cette méthode n'est pas très sûre : on a modifié une bonne partie des pixels de la première colonne, ce qui peut se voir. On va utiliser la transformation du photomaton pour améliorer notre méthode.

Écrire une fonction qui calcule la période d'une image pour cette transformation, c'est-à-dire qui compte combien d'étapes sont nécessaires pour revenir à l'image.

Appelons N ce nombre. On va alors appliquer $N/2$ fois la transformation, cacher le message, et appliquer à nouveau la transformation jusqu'à retrouver l'image de départ. Les pixels modifiés sont ainsi "mélangés" dans l'image.

Écrire une fonction permettant de cacher et retrouver un message avec cette méthode.