Mini projet : traitement d'images

Pour ce mini-projet, la pièce finale sera un fichier « mosaique-prenom-nom.py » contenant votre programme.

Le seul fait que votre programme fonctionne ne suffit pas. Il devra également :

1. être lisible (choix pertinent pour les noms de variables etc.),
2. être commenté (placer des commentaires aux endroits appropriés).

Comme point de départ, nous vous fournissons :

- Le document que vous avez entre les mains, intitulé « Mini-projet : traitement d'images ». Ce document contient **les éléments indispensables de cours** et les **exercices à faire impérativement** pour aborder la phase de réalisation dans de bonnes conditions.
- Un document intitulé « Le système binaire », qui contient du cours et quelques exercices pour aider à la compréhension. **A lire impérativement** avant, pendant et après le projet.
- Un fichier « boite_a_outils.py » spécialement créé pour ce mini-projet et qui contient les déclarations de fonctions dont vous aurez besoin pour créer, ouvrir, enregistrer, modifier des images.

Sujet : voir page 7

Liens utiles :

- Le site : http://www.info-isn.fr/ sur lequel vous retrouverez les liens, les documents à télécharger.
- La documentation des bibliothèques standards : http://docs.python.org/3.2/library/index.html
- Le site de téléchargement du visualiseur d'image xnview (pour visualiser les images .ppm sous Windows) : http://www.xnview.com/fr/download.html

1. Les images

Préliminaire : les formats Netpbm

Dans ce projet, nous n'utiliserons que des images dans des formats très simples : les formats Netpbm. Ces images sont stockées en ASCII ou en BINAIRE, et sont reconnues nativement par Linux mais pas par Windows. Ces images sont lisibles sous Windows avec des visualiseurs très simples et gratuits (Xnview, IrfanView).

Les formats Netpbm sont :

- PBM pour les images noir et blanc codée en caractères ASCII (P1),
- PGM pour les images en niveau de gris codée en caractères ASCII (P2),
- PPM pour les images en couleurs RGB codée en caractères ASCII (P3).
- PBM pour les images noir et blanc codée en caractères BINAIRE (P4),
- PGM pour les images en niveau de gris codée en caractères BINAIRE (P5),
- PPM pour les images en couleurs RGB codée en caractères BINAIRE (P6).

Remarque : certaines applications utilisent l'extension **.pnm**

Par exemple, voici deux images en noir et blanc et en niveaux de gris :

```
P1
10 10
1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1
1 1 1 0 0 0 0 1 1 1
1 1 1 0 0 0 0 1 1 1
1 1 1 0 0 0 0 1 1 1
1 1 1 0 0 0 0 1 1 1
1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1
```

```
P2
10 10
10
1 1 1 1 1 1 1 9 9 9
2 2 2 2 2 2 2 2 9 9
3 3 3 3 3 3 3 9 2 9
4 4 4 4 4 4 9 3 2 1
5 5 5 5 5 9 4 3 2 1
6 6 6 6 9 5 4 3 2 1
7 7 7 9 6 5 4 3 2 1
8 8 9 7 6 5 4 3 2 1
9 9 8 7 6 5 4 3 2 1
9 9 8 7 6 5 4 3 2 1
```

exercice 1.1 images PBM et PGM

1. Créez les images ci-dessus à l'aide d'un éditeur de texte et affichez les avec un visualiseur pour faire le lien entre le contenu du fichier et l'image affichée.
2. Modifiez le contenu du fichier avant de le réafficher.
3. Créez quelques images en gris ou en noir et blanc avec un éditeur de texte et visualisez les.

exercice 1.2 images PPM

1. Créez une image en couleur de format 10×10 pixel et visualisez-la avec l'éditeur de texte.
2. Quelles sont les différences entre les formats ASCII et BINAIRE (brut) ?

exercice 1.3 que contiennent vraiment les images ?

Téléchargez l'image « image01 » sur le site http://www.info-isn.fr/ , ouvrez-la avec un éditeur hexadécimal et retirez le plus d'informations possibles.

exercice 1.4 quelle image pour quelle utilisation ?

1. Téléchargez les image « image02A » et « image02B » sur le site http://www.info-isn.fr/.
2. Laquelle de ces deux image vous semble la mieux adaptée à l'affichage sur un site internet ? *argumentez*

2. La « boite_a_outils »

Pour les besoins de ce mini-projet, nous avons créé une petite bibliothèque de fonctions nommée « boite_a_outils » qui permet de créer, ouvrir, et sauvegarder des images simples aux format « PBM » (noir et blanc), « PGM » (niveaux de gris) ou « PPM » (couleurs RVB) codées en « ASCII ». Pour utiliser ces fonctions vous devez télécharger le fichier « boite_a_outils.py » et le placer dans le répertoire du fichier source.

La « boite à outils » contient deux fonctions : **lire_fichier_image()** et **creer_fichier_image()**

fonction **lire_fichier_image (f)** :

exemple : `matrice_image = lire_fichier_image(fichier_image)`
Cette commande crée la matrice appelée « matrice_ image » aux dimensions de l'image contenue dans le fichier « fichier_image », qui se trouve dans le répertoire du programme.

fonction **creer_fichier_image_P1(m,f)** :

exemple : `creer_fichier_image_P1(matrice_image,nouveau_fichier)`
Cette commande crée, dans le répertoire courant, un fichier PBM image appelé « nouveau_fichier ». Ce fichier contient l'image contenue par la matrice « matrice_image ». Chaque pixel est codée sur 2 niveaux (0 : blanc ; 1 : noir).

fonction **creer_fichier_image_P2(m,f,valmax)** :

exemple : `creer_fichier_image_P2(matrice_image,nouveau_fichier,255)`
Cette commande crée, dans le répertoire courant, un fichier PGM image appelé « nouveau_fichier ». Ce fichier contient l'image contenue par la matrice « matrice_image ». Chaque pixel est codé sur 256 niveaux de gris allant de 0 à 255 (0 : noir ; 255 : blanc)

fonction **creer_fichier_image_P3(m,f,valmax)** :

exemple : `creer_fichier_image_P3(matrice_image,nouveau_fichier,255)`
Cette commande crée, dans le répertoire courant, un fichier PPM image appelé « nouveau_fichier ». Ce fichier contient l'image contenue par la matrice « matrice_image ». Chaque couleur est codée sur 256 niveaux allant de 0 à 255.

Exemples d'utilisations de la « boite_a_outils » :

Exemple 1 : Création d'un fichier image PPM ascii (de type P3)

```
#-*-coding:Utf-8-*-
from boite_a_outils import *

nom_fic=input('Entrer le nom de fichier : ')
NbX=int(input('Entrer le nombre de colonnes : '))
NbY=int(input('Entrer le nombre de lignes : '))
rouge=int(input('Valeur du ROUGE : '))
vert=int(input('Valeur du VERT : '))
bleu=int(input('Valeur du BLEU : '))
val=[rouge,vert,bleu]

m3=creer_matrice(NbY,NbX,val)
fic_P3='P3_'+nom_fic
creer_fichier_image_P3(m3,fic_P3,255)
```

Exemple 2 : Programme de création du négatif d'une image couleur

```
-*-coding:Utf-8-*-
from boite_a_outils import *
from string import *

def lire_entete(nom_du_fichier):
 global l, h,val_max
 fichier=open(nom_du_fichier,'r') #ouverture du fichier
###########################################################
# Lire le nombre magique qui indique le mode de l'image :
# P1 pour un fichier noir et blanc
# P2 pour un fichier en Niveax de gris
# P3 pour un fichier en couleurs RVB
###########################################################
 mode=fichier.readline()
###########################################################
# Lire les dimension de l'image :
# nombre de pixels sur une ligne : l
# nombre de pixels sur une hauteur : h
###########################################################
 dim=fichier.readline()
 while dim[0]=='#': # les lignes du commentaire sont ignorees
 dim=fichier.readline()
 t_dim=dim.split()
 l=int(t_dim[0])
 h=int(t_dim[1])
 val_max=fichier.readline() # valeur max de codage

### debut du programme principal ###
global l,h,val_max # definition des variables globales
nom_image='p3-joconde' # nom du fichier image a ouvrir
extension='.ppm' # nom de l'extension
image = nom_image + extension
lire_entete (image) # lecture dans l'entete de l, h et val_max
###################################################
# matrice_1 stocke les pixels en memoire vive
# chaque pixel est code par un triplet (r,v,b)
###################################################
matrice_1 = lire_fichier_image(image)
matrice_2 = matrice_1      # matrice_1 est dupliquee dans matrice_2
###################################################
# pour chaque pixel inverse la couleur
# et on stocke le resultat dans matrice_2
###################################################
for i in range (h):
     for j in range (l):
          for k in range (3):
               matrice_2[i][j][k] = 255 - matrice_1[i][j][k]
###################################################
# creation d'un nouveau fichier image P3
# a partir des donnees stockees dans matrice_2
###################################################
image_negatif = nom_image + '_negatif'
creer_fichier_image_P3(matrice_2,image_negatif,int(val_max))
```

exercice 2.1 créer des images et comparer leurs caractéristiques

1. Créer, à l'aide de la boîte à outils, un fichier image ASCII, de couleur jaune, nommée « simpson.ppm »,
   de dimension 100 × 100 pixels.
2. Quel est le poids de ce fichier (en ko) ?
3. Visualiser cette image avec un éditeur d'image (geany, photoshop, xnview,...).
4. Comparez les poids (en octets) des fichiers « simpson.ppm » et « simpson.jpg ».
5. Enregistrer cette image au format jpeg : « simpson.jpg ».
6. Comparez les rendus des images « simpson.ppm » et « simpson.jpg »
7. Ouvrir ces fichiers avec un éditeur hexadecimal et comparez.
8. Quel serait le poids d'un fichier 100 × 100 pixels en niveau de gris.
9. Vérifiez...

3. Modification d'une image par un programme

Le but est maintenant de transformer une image par un programme qui modifie les données stockées dans une matrice (la matrice est un tableau bidimensionnel).

exercice 3.1 ***créer un dégradé***

> Réaliser un programme qui créée un dégradé vertical dont les dimensions et les valeurs extrêmes sont à définir par l'opérateur.

Le cas le plus simple est un dégradé de gris qui va du noir au blanc dans un rectangle de hauteur 256 : le rectangle ressemble à

```
... 0 0 0 0 0 0 0 0 0 0 ...
... 1 1 1 1 1 1 1 1 1 1 ...
... 2 2 2 2 2 2 2 2 2 2 ...
... 3 3 3 3 3 3 3 3 3 3 ...
... ...
... ...
... 254 254 254 254 254 254 254 254 254 254 ...
... 255 255 255 255 255 255 255 255 255 255 …
```

Lorsque la hauteur est plus petite, il faut « sauter des lignes ». Par exemple, dans le cas d'une hauteur de 128 :

```
... 0 0 0 0 0 0 0 0 0 0 ...
... 2 2 2 2 2 2 2 2 2 2 ...
... 4 4 4 4 4 4 4 4 4 4 ...
... ...
... ...
... 252 252 252 252 252 252 252 252 252 252 ...
... 254 254 254 254 254 254 254 254 254 254 …
```

Autrement dit, la ligne *i* du rectangle à la couleur *2i*.

Lorsque la hauteur est plus grande, il faut répéter des lignes :

```
... 0 0 0 0 0 0 0 0 0 0 ...
... 0 0 0 0 0 0 0 0 0 0 ...
... 1 1 1 1 1 1 1 1 1 1 ...
... 1 1 1 1 1 1 1 1 1 1 ...
... ...
... ...
... 255 255 255 255 255 255 255 255 255 255 ...
... 255 255 255 255 255 255 255 255 255 255 …
```

Lorsqu'on créée un dégradé entre deux couleurs arbitraire, il y a des difficultés supplémentaires :

- les couleurs de première et dernière lignes ne sont pas 0 et 255,
- il faut gérer les composantes rouge, verte et bleue indépendamment.

exercice 3.2 ***retourner une image***

Réaliser un programme qui fait pivoter une image de 180°

exercice 3.3 ***produire un négatif***

Réaliser un programme qui créée le négatif d'une image

exercice 3.4 ***convertir en niveaux de gris***

Réaliser un programme qui convertit une image couleur en image en niveau de gris

exercice 3.5 ***convertir en noir et blanc***

Réaliser un programme qui convertit une image couleur en image en noir et blanc

exercice 3.6 ... ***à vous de jouer***

Imaginez un ou plusieurs programmes qui modifient des images ...






Le sujet du mini-projet : programme de création de mosaïques

Réaliser un programme qui remplace une image par une mosaïque dont la dimension des carreaux est à définir par l'opérateur.

Exemple :

image originale : 300 × 200 pixels

mosaïque 300 × 200 pixels avec des carreaux de 10 pixels