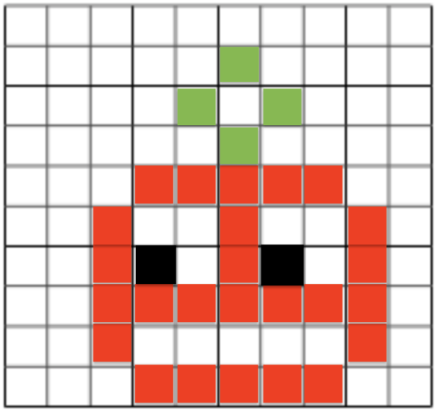
TD1 - Images numériques

Questions:

1. Je veux avoir une idée de la couleur que produira un mélange de couleurs sur mon écran d'ordinateur, est ce que je peux essayer la même combinaison de couleurs sur papier avec de feutres pour voir ce que cela donnera?
2. Vrai/faux: L'œil humain est plus sensible aux nuances de vert car il possède plus de cônes vert dans la rétine.
3. L'échantillonnage d'une image, c'est:
   (a) extraire le nombre de couleurs qui apparaissent dans l'image
   (b) extraire un système discret de valeurs pour chaque position de l'image
   (c) déterminer le nombre de bits nécessaires pour représenter chaque couleur
4. La quantification des pixels dans une image numérique, c'est :
   (a) déterminer les dimensions hauteur / largeur de l'image
   (b) fixer la valeur qui leur est attribuable parmi des valeurs possibles
   (c) le choix du format de l'image (Gif, Jpeg, PNG...)
5. L'image numérique suivante a une définition de 10×10 pixels. Sachant que la dimension de l'image analogique est 5cm×5cm calculez la resolution.



6. On travaille encore avec l'image numérique ci-dessus, sachant que les pixels vertes et rouges qui apparaissent correspondent au vert pure et rouge pure, écrivez la matrice associée à cette image en format RGB.

2 Traitement d'images avec Python

2.1 Premiers pas

Pour commencer, choisissez une photo. Moi, par exemple, j'ai téléchargé cette photo sur Pixabay (banque d'images gratuites)

`https://pixabay.com/fr/photos/jeune-fille-visage-coloré-couleurs-2696947/`
(Image par Alexandr Ivanov de Pixabay)

et je l'ai enregistré dans le fichier `Lola.jpg` que j'ai mis dans un repertoire appelé `CS3` . Ouvrez ce repertoire sur la console de Pyzo.

Avant tout traitement, il faut charger l'image: cela se fait facilement avec la fonction `Image.open` . Pour vérifier que l'image a bien été chargée nous pouvons ensuite l'afficher avec la méthode `(` .show):

```
from PIL import Image

img = Image.open("Lola.jpg") #charge l'image
img.show() #affiche l'image chargée
```

Une petite fenêtre doit apparaître et afficher l'image du fichier `Lola.jpg` .

Maintenant, nous allons afficher la dimension de notre image:

```
w, h = img.size
print("Largeur : {} px, hauteur : {} px".format(w,h))
```

Dans mon cas j'obtiens: `Largeur : 640 px, hauteur : 640 px`

Ce résultat signifie que l'image analogique a été découpée en 640 pixels sur la largeur et 640 pixels sur la hauteur pendant l'échantillonnage. On peut modifier les dimensions d'une image avec la fonction `resize`

```
from PIL import Image

img= Image.open("Lola.jpg")
print(img.size)
img2= img.resize((600, 600))
print(img2.size)
```

Exercice: écrivez un programme qui prend en entrée une image et réduit à moitié ses dimensions.

Maintenant, analysons cette image numérique. L'attribut `image.mode` nous donne le format de pixel utilisé, autrement dit la façon dont la quantification a été faite. La méthode `image.getpixel` permet de récupérer l'intensité associée au pixel à une position donnée.

```
print("Format des pixels : {}".format(img.mode)) #affiche son mode de quantification

#Récupérer et afficher la valeur du pixel à une position précise
px_value=img.getpixel((200, 400))
print("Valeur du pixel situé en (200, 400) : {}".format(px_value))
```

J'obtiens:

```
Format des pixels : RGB
Valeur du pixel situé en (200, 400) : (236, 43, 122)
```

`Image.getpixel` prend en argument un tuple, dont la première valeur indique l'abscisse du pixel et la deuxième, son ordonnée. Par convention dans Pillow, le repère cartésien a pour origine le pixel le plus en haut à gauche de l'image, et l'axe des ordonnées est orienté vers le bas. De plus, n'oubliez pas que l'indexation en Python commence à 0 (et non pas à 1) !

Le mode `RGB` signifie que les intensités sont codées sur 3 octets car c'est une image en couleur, donc à chaque pixel est associé un triplet de valeurs correspondants aux inténsités de Rouge, Vert et Bleu . Voici quelques autre mode de quantification reconnus par Pillow.

- 1 (1-bit par pixel, noir et blanc)
- L (1 octet par pixel, niveaux de gris)
- RGB (3 octets par pixel, true color)
- RGBA (4 octets par pixel, true color avec une composante pour la transparence)
- CMYK (4 octets par pixel)
- YCbCr (3 octets par pixel, pour les images vidéos couleurs)
- LAB (3 octets par pixel, modèle L*a*b)
- HSV (3 octets par pixel, modèle TSV, teinte, saturation, valeur de l'intensité)

Vous pouvez modifier le mode de votre photo. Par exemple, si votre photo est en format `RGB` et vous voulez la convertir en une photo en niveaux de gris, tapez le code suivant:

```
from PIL import Image

img = Image.open("Lola.jpg").convert('L')
img.show()
```

Enregistrez la nouvelle image en gris avec `img.save` , par exemple: `img.save("Desktop/CS3/LolaGray.png")`

Nous pouvons récupérer la matrice des pixels à l'aide de la librairie `numpy` . Essayez ce code avec votre image en gris.

```
from PIL import Image
import numpy as np

img = Image.open("LolaGrey.png")
mat = np.array(img)
print(mat)

print("Taille de la matrice de pixels : {}".format(mat.shape))
```

Une matrice de ce type est affichée

```
[[35 15 15 ...  4  5  5]
 [42  4 18 ...  4  4  4]
 [22 15 39 ...  4  3  3]
 ...
 [38 19 49 ...  4  4  5]
 [44 19 19 ...  4  4  5]
 [39 29 29 ...  4  4  5]]

Taille de la matrice de pixels : (640, 640)
```

Attention à l'inversion ligne/colonne et abscisse/ordonnée : l'intensité du pixel d'abscisse x et d'ordonnée y correspond à l'élément de la matrice situé à la y-ème ligne et x-ème colonne !

`np.array()` retourne la matrice de pixels, conversement si vous voulez afficher l'image correspondante à une matrice de pixels, vous utiliserez la fonction `Image.fromarray(mat)` .

Si vous répétez la procedure avec une image `RGB` , vous obtenez une matrice de triplets. Vous pouvez séparer les composantes rouges, vert et bleu d'une image `RGB` avec la fonction `Image.split()` . Par exemple:

```
path=...
img = Image.open(path)
R, G, B = img.split()
R.show()
G.show()
B.show()
```

`Image.split()` retourne un tuple contenant toutes les bandes de l'image, i.e. dans le cas d'une image RGB, ca donne 3 images R, G, B en niveaux de gris correspondantes aux composantes rouge, vert et bleu resp. de l'image.

Conversement, `Image.merge()` fusionne un ensemble d'images monobandes en une image multibande:

```
merged = Image.merge("RGB", (img_rouge, img_vert, img_bleu))
```

Exercices:

1. Remplacez la composante verte de l'image de Lola par sa composante rouge. Que remarquez vous?
2. Remplacez la composante verte de l'image de Lola par une image complètement noire de même dimension que l'image de Lola.
3. téléchargez les images carL.jpg et roadL.jpg et fusionnez-les

[photo ref. Larisa Koshkina et Lisa Johnson sur Pixabay]