Les images : traitement de l'information

Contenus	Capacités attendues
Traitement d'image	Traiter par programme une image pour la transformer en agissant sur les trois composantes de ses pixels.
Rôle des algorithmes dans les appareils photo numériques	Expliciter des algorithmes associés à la prise de vue. Identifier les étapes de la construction de l'image finale.

Webographie : **Dominique Lefebvre - www.tangenteX.com janvier**

Activité n°1 : Rappel sur le codage RVB

Le code le plus utilisé est le codage Rouge, Vert, Bleu (RVB).
Chaque couleur est codée sur 1 octet = 8 bits.
Chaque pixel est donc codé sur 3 octets, c'est à dire 24 bits :
le rouge de 0 à 255 , le vert de 0 à 255, le Bleu de 0 à 255.

- Rappeler en quelques lignes le principe de la synthèse additive des couleurs.
- Calculer le nombre de couleurs différentes obtenues
- Quelle est la taille minimale (en octets) occupée en mémoire par l'image de 100 000 pixels ?
- Avec le même logiciel de traitement d'image ouvrir le fichier « lena_std.tif » et convertir l'image en noir et blanc (binaire) puis en niveau de gris (256 niveaux). Enregistrer à chaque fois le fichier convertit et comparer sa taille en mémoire avec le fichier original. Conclure.

Exemple de codage hexadécimal d'un pixel en couleur 24 bits :

Couleur	R	V	B
Hexadécimal	F5	28	E7
Décimal			

- Compléter le tableau
- Quelle est la couleur dominante du pixel décrit dans ce tableau?

Activité n°2 : traitement de l'image : butterfly.jpg

Pré-requis

Dans l'écriture d'un nombre, les chiffres ont un poids croissant de la droite vers la gauche : par exemple en binaire, le premier 1 (à gauche) de 11000101 représente $128=1x2^7$ alors que celui de droite représente $1=1x2^0$. Connaitre les bases de la programmation en Python et de l'utilisation d'une bibliothèque ici PIL.

Petit lexique PIL (Python Imaging Library)

- **Ouverture d'un fichier image :** *nom_image* = Image.open("*nom_fichier*")
- **Sauvegarde d'une image dans un fichier :** *nom_image*.save("*nom_fichier*")
- **Créer une image de largeur L et de hauteur H (en pixels) :**

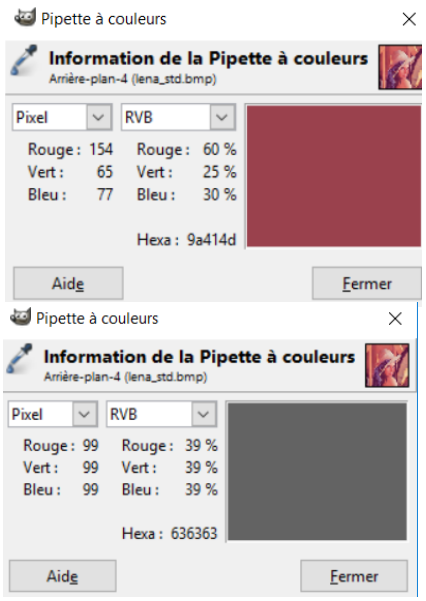
nom_image = Image.new("RGB",(L,H))

ou éventuellement *nom_image* = Image.new("RGB",(L,H),(R,V,B)) si l'on souhaite une image avec un fond coloré de couleur (R,V,B).

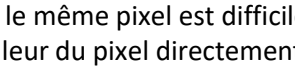
- **Lecture d'un pixel de l'image :***nom_pixel* = *nom_image*.getpixel((x,y))
  (x,y) sont les coordonnées du pixel; (0,0) = coin supérieur gauche de l'image
  nom_pixel[0] donne la composante rouge du pixel; *nom_pixel*[1] la verte, et *nom_pixel*[2] la bleue.
- **Écriture d'un pixel :***nom_image*.putpixel((x,y),(r,v,b))

r, v, et b représentent les composantes rouge, verte, et bleue

Accès aux composantes RVB d'un pixel dans GIMP : niveaux de gris



Pour agrandir l'image : touche Ctrl maintenue enfoncée tout en tournant la molette de la souris, ou bien menu Affichage > Zoom > 1600 %

Dans la boîte à outils, sélectionner l'outil pipette : et cocher 

À chaque clic sur un pixel, les composantes rouge, verte, et bleue du pixel apparaissent dans une fenêtre.
Dans l'exemple ci-contre, on a R = 154, V = 65, B = 77.
On peut recommencer l'opération sur le même pixel après avoir ouvert l'image en niveaux de gris :
Une seule valeur est indiquée : R = V = B = 99

PB : Retomber exactement sur le même pixel est difficile.
On va donc aller chercher la valeur du pixel directement avec un programme

Accès aux composantes RVB d'un pixel avec Python :

- ***Ouvrir edupython ou spyder et taper le programme suivant :***

```
from PIL import Image                    # Importation de la bibliothèque PIL
img = Image.open("butterfly.jpg")        #A la variable img on associe l'image « lena_std.bmp » que l'on a ouvert
r,v,b=img.getpixel((150,310))            # On va chercher les composantes RVB du pixel (150,300) que l'on place dans les variables, r, v, b.
print("canal rouge : ",r,"canal vert : ",v,"canal bleu : ",b)# on écrit les variables r, v, b.
```

- ***Modifier le programme de façon à ouvrir le même pixel dans l'image butterfly-gris.jpg et vérifier que l'on a bien R = V = B et que l'on est pas très loin de (R+V+B)/3 de l'image couleur.***

Récupérer tous les pixels rouges puis les bleus puis les verts.

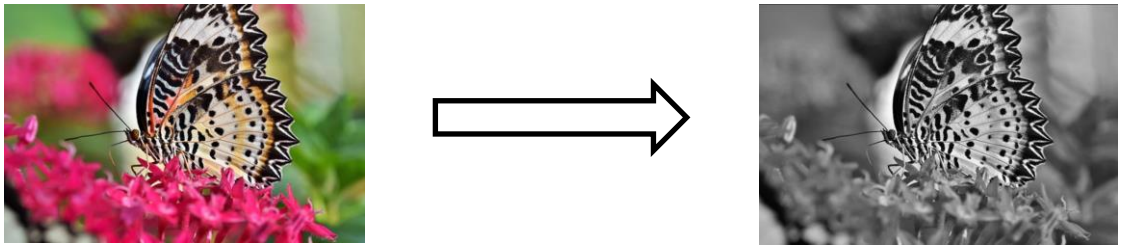
Exemple détaillé en Python : rouge.py

```
from PIL import Image                    # la bibliothèque PIL doit avoir été installée (pas de pb avec edupython)

im1 = Image.open("butterfly.jpg")        # ouverture du fichier image source.
L,H = im1.size                           # L = largeur de l'image, H = sa hauteur, en pixels
print (L,H)
im2 = Image.new("RGB",(L,H))             # création de l'image destination (niveaux de gris) « vide »
for y in range(H):                       # on balaie toutes les lignes de l'image source, de 0 à H-1
    for x in range(L):                   # pour chaque ligne on balaie toutes les colonnes, de 0 à L-1
    p = im1.getpixel((x,y))              # en stockant le pixel (x,y) dans p
    r = p[0]                             # p[0] est la composante rouge, p[1] la composante verte, et p[2] la
    v = 0                                #composante bleue
    b = 0
    im2.putpixel((x,y),(r,v,b))          # on écrit le pixel modifié sur l'image destination
im2.save("lena_rouge.bmp")               # on stocke l'image résultante,
im2.show()                               # on affiche l'image.
```

- ***Modifier le programme pour faire apparaitre les composantes vertes et bleu (butterfly_bleu.py et butterfly.verte.py)***

- ***Modifier le programme précédent pour transformer l'image lena_std.bmp en niveau de gris (lena_gris.bmp). On se servira d'une nouvelle variable : gris, et de la division entière (c'est-à-dire dont le résultat - tronqué - ne peut être qu'un entier), en utilisant l'opérateur // au lieu de /. Le programme s'appellera butterfly-gris.py***



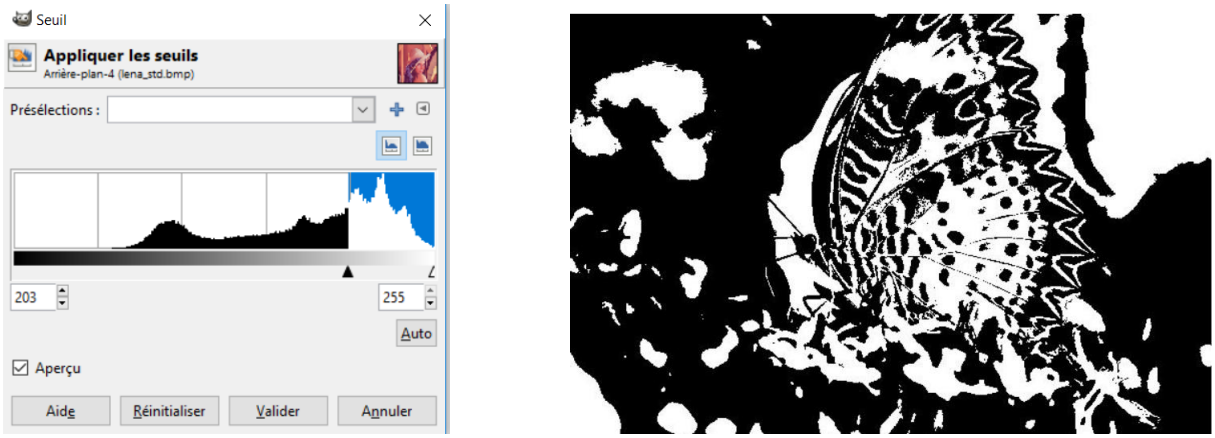
Même si le résultat est convenable, ce n'est pas encore cela.

En réalité les proportions recherchées sont normalisées : c'est la CIE (Commission Internationale de l'Eclairage) qui normalise ce genre de chose, histoire de savoir de quoi l'on parle. Et dans sa norme 709, elle dit que pour les images naturelles les poids respectifs doivent être **0.2125 * R + 0.7154 * V + 0.0721 * B**. Nous allons donc utiliser cette répartition dans notre code.

- ***Remodifier le programme pour satisfaire à la norme de la CIE (butterfly_gris_cie.py). On comparera le résultat (butterfly_gris_cie.bmp) avec l'image butterfly_gris.bmp obtenue précédemment.***

- ***Modifier le programme précédent pour passer d'une image couleur, à une image N&B (NB.py). On pourra par exemple choisir un seuil au-dessous duquel on sera noir (p= 0 ou 255 ?) et au-dessus blanc.***

- ***On peut confronter l'image obtenue avec ce que donne GIMP (menu Couleurs > Seuil...), qui permet par ailleurs de faire varier le seuil de 0 à 255 à l'aide d'un curseur et d'avoir un aperçu en direct du résultat :***



http://www.tsi.enst.fr/pages/enseignement/ressources/mti/egal-histo/rapport.htm