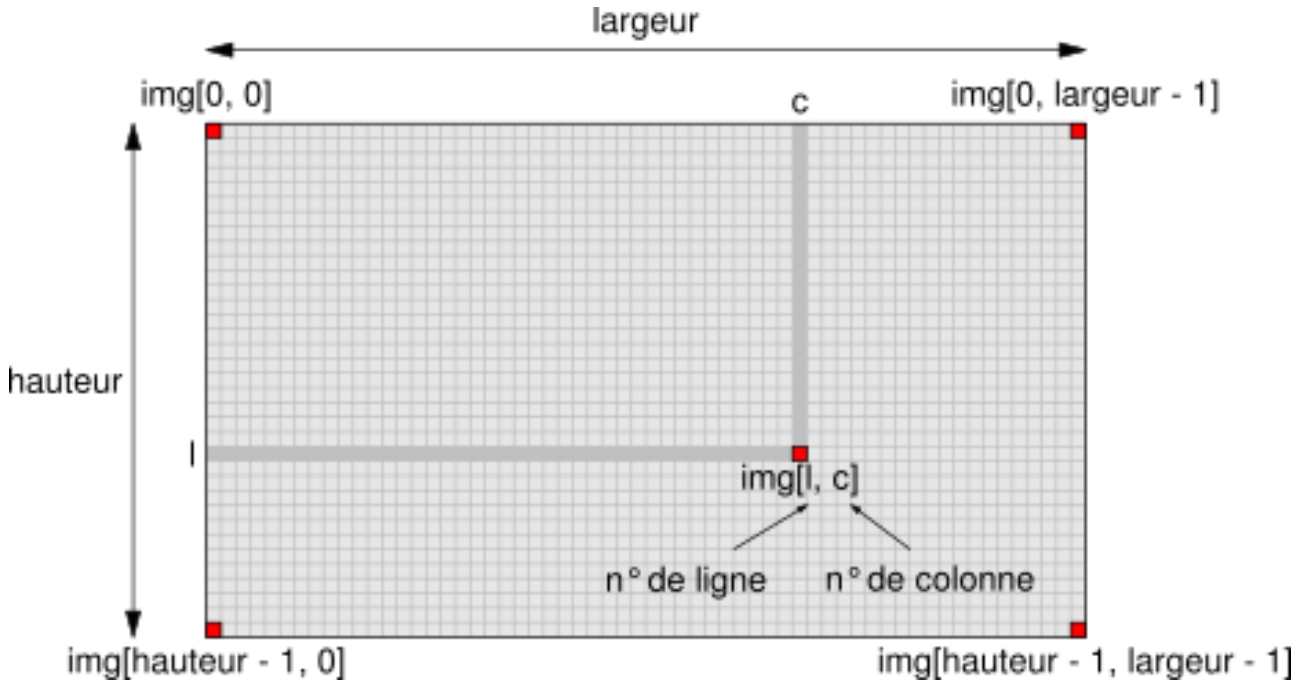
Image numérique

1. 1. Tableau de pixels

Les caractéristiques d'une image

Quadrillage

Une image numérique se présente sous la forme d'un quadrillage - ou d'un tableau - dont chaque case est un pixel d'une couleur donnée. On peut donc repérer chaque pixel par sa ligne et sa colonne dans ce tableau (ou à l'aide de coordonnées en partant du coin en haut à gauche[1]).



Définition

La définition de l'image est le nombre total de pixels qui la composent. Celle-ci n'est pas forcément égale à la définition du capteur.

On l'obtient donc en multipliant sa largeur par sa hauteur. Par exemple, une image de 1920 pixels de largeur sur 1080 pixels de hauteur a une définition de 1920 x 1080 = 2073600 pixels soit à peu près 2 millions de pixels.

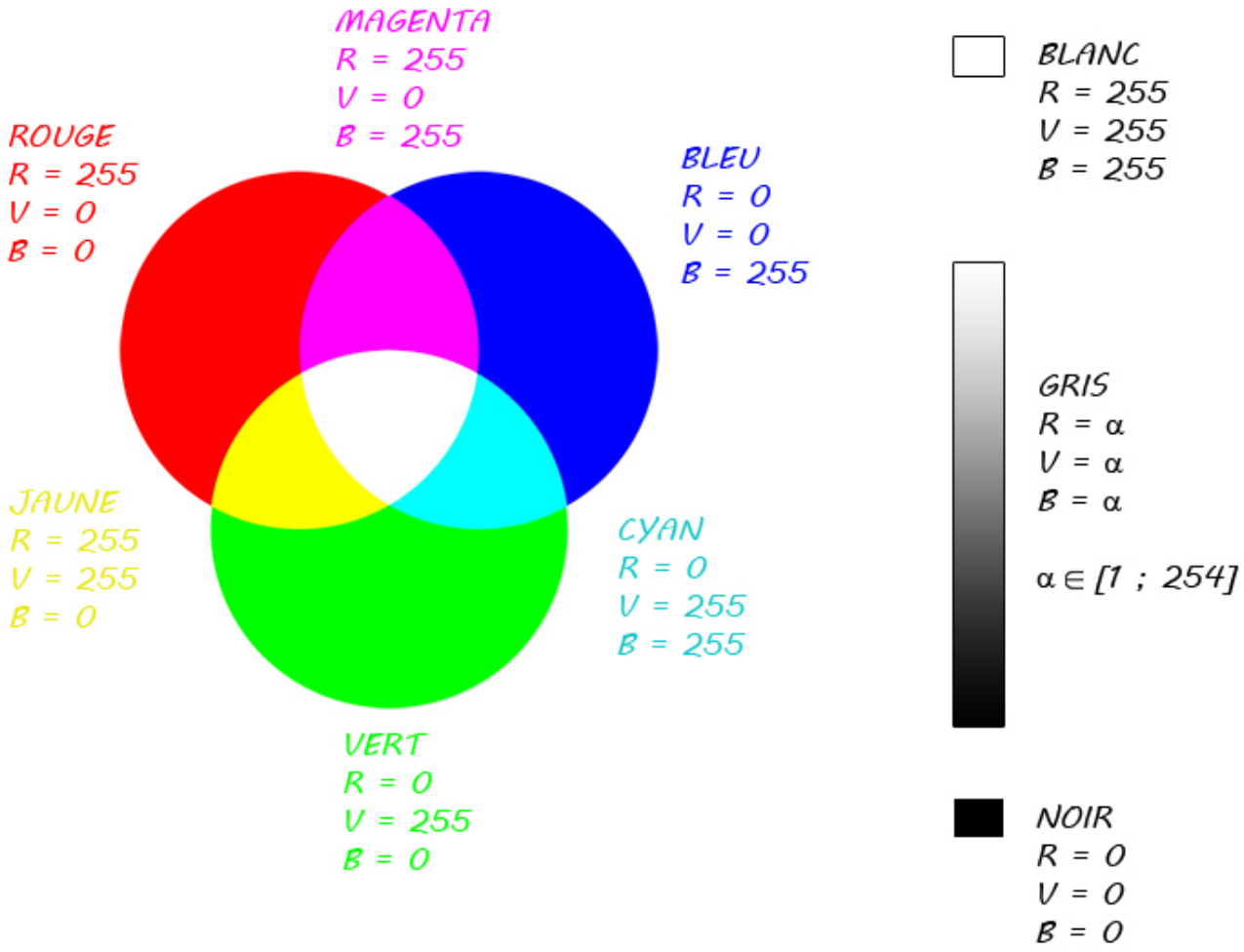
Résolution

La résolution de l'image, c'est-à-dire le nombre de pixels par unité de longueur, détermine sa qualité à l'impression ou sur un écran.

Par exemple, la résolution standard pour affichage sur le web est de 72 ppp (pixels par pouce) alors qu'une résolution de 300 ppp est recommandée pour l'impression.

Le codage des pixels (couleurs)

Chaque pixel correspond à un triplet de trois nombres entiers, soit les valeurs de rouge (Red), de vert (Green) et de bleu (Blue) afin de reconstituer la couleur. Chaque valeur est codée sur un octet, soit entre 0 et 255 (ou en pourcentages, ou en hexadécimal, voir ici). On parle de code RGB (RVB in french).



À noter:

- une couleur dont les 3 composantes sont identiques correspond à un niveau de gris;
- selon les formats, une quatrième composante peut s'ajouter: le **canal alpha**. Cette valeur (sur un octet également) indique le niveau de transparence du pixel.

Site incontournable

Un site pour visualiser les couleurs au formet RGB, et convertir en héxadécimal :
http://www.proftnj.com/RGB3.htm

2. 2. Les modules

Pour manipuler les images, nous allons avoir besoin du module `imageio`. Ce module nécessite d'utiliser également le module `numpy` pour créer des tableaux d'entiers non signés sur 8 bits (un octet).

`imageio`

- Ouvrir et charger une image existante (`ada.png` par exemple) dans une variable (`img` par exemple):

Script Python

```
img = imageio.imread("ada.png")
```

Accès à l'image

L'image doit être dans le dossier courant de travail, a fortiori le même que le fichier `.py`. Si ce n'est pas le cas, il faudra le modifier.

- La taille de l'image est accessible dans le triplet (hauteur, largeur, nombre de composantes) donné par:

Script Python

```
img.shape
```

- Lire/modifier un pixel: il s'agit tout simplement de travailler sur le tableau, par indices et par réaffectation.

Script Python

```
print(img[2][10])       # pour afficher le pixel ligne 2, colonne 10
img[2][10] = [0, 0, 0]  # pour le mettre en noir
```

- Sauvegarder une image contenue dans une variable `img`:

Script Python

```
imageio.imsave("monimage.png", img)
```

numpy

Le module `numpy` est un module de calcul scientifique orienté vers les matrices, qui sont des objets mathématiques bien pratiques... En gros ce sont des tableaux. On se servira uniquement de ce module pour créer des tableaux vides, au format que le module `imageio` exige pour pouvoir ensuite sauvegarder l'image (et donc la visualiser).

On utilise la fonction `zeros` du module `numpy` qui prend en paramètres un triplet (hauteur, largeur, nombre de composantes) et le type des valeurs, ici donc des entiers non signés sur 8 bits.

Par exemple pour une image de 100 pixels (de haut) sur 256 pixels (de large), avec 3 composantes (pas de canal alpha):

Script Python

```
img_vide = numpy.zeros([100,256,3], dtype=np.uint8)
```

3. 3. Exercices

Exercice 1

Énoncé

1. Télécharger l'image `ada.png` ci-dessus (simple clic-gauche), puis la charger dans un programme avec le module `imageio`.
2. Trouver ses dimensions et son nombre de composantes.
3. Faire un crime de lèse-majesté et tracer une ligne horizontale rouge au nivau du front.

Solution

Script Python

```python
import imageio

# on charge l'image dans une variable img
img = imageio.imread('ada.png')

# on affiche les dimensions et le nombre de composatnes contenues dans le tuple img.shape
print('Hauteur:', img.shape[0], 'Largeur:', img.shape[1], 'Nombre de composantes:', img.shape[2])

# on parcourt la ligne 100 et on remplace tous les pixels par du rouge
for j in range(img.shape[1]):
    img[100][j] = (255, 0, 0, 255)

# on enregistre l'image modifiée
imageio.imsave('ada_modifie.png', img)
```

Exercice 2

Énoncé

Cette image est-elle vraiment composée de pixels tous noirs?

Solution

- on parcourt tous les pixels de l'image avec deux boucles `for` imbriquées: `i` sur la hauteur (les lignes) et `j` sur la largeur (les colonnes);
- on regarde si le pixel `img[i][j]` est noir, c'est-à-dire que ses 3 composantes sont égales à 0;
- si c'est le cas, on le remplace par un pixel blanc;
- sinon on ne fait rien: le pixel restera sur une teinte proche du noir.

Script Python

```
import imageio
img = imageio.imread('message.png')

```

```
for i in range(img.shape[0]):
    for j in range(img.shape[1]):
        if img[i][j][0] == 0 and img[i][j][1] == 0 and img[i][j][2] == 0:
            img[i][j] = (255, 255, 255)

imageio.imsave('message_decrypte.png', img)
```

Exercice 3

Énoncé

Incruster John Travolta devant le lycée

Solution

- on charge les deux images dans deux variables;
- on parcourt l'image sur fond vert, et si le pixel est vert, on le remplace par le pixel (aux mêmes coordonnées) de l'autre image.
- on peut bien entendu faire le contraire...

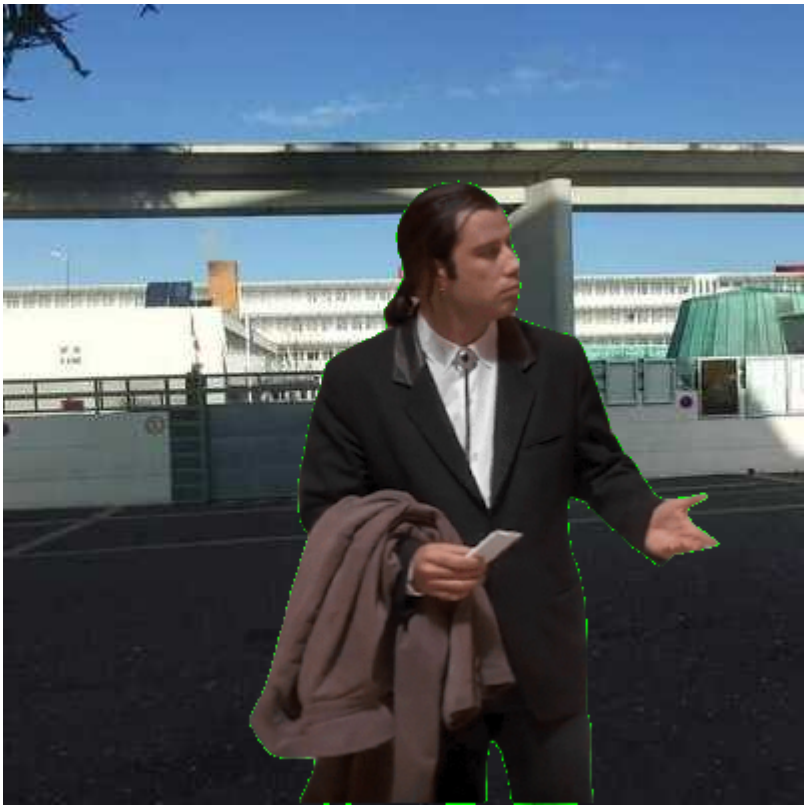
Script Python

```
import imageio
img_john = imageio.imread('john.bmp')
img_lycee = imageio.imread('lycmdv_crop.jpg')

for i in range(img_john.shape[0]):
    for j in range(img_john.shape[1]):
        if img_john[i][j][0] == 0 and img_john[i][j][1] == 255 and img_john[i][j]
[2] == 0:
            img_john[i][j] = img_lycee[i][j]

imageio.imsave('john_devant_lycee.bmp', img_john)
```

On obtient:



4. 4. Création d'effets

Dans cette dernière partie, on va recréer des effets que des logiciels de retouche d'image (GIMP, Photoshop, ...) proposent.

On travaillera (par exemple) sur l'image ci-dessous:

Effets

Filtre rouge

Pour créer un filtre rouge il suffit de conserver la composante rouge et de remplacer les autres composantes par 0.

Si vous n'aimez pas le rouge, faites un filtre vert. Ou bleu.

Correction

Script Python

```python
import imageio
img = imageio.imread('VanGogh_Arles.png')

for i in range(img.shape[0]):
    for j in range(img.shape[1]):
        r = img[i][j][0]
        img[i][j] = (r, 0, 0)

imageio.imsave('img_filtre.png', img)
```

Négatif

Pour obtenir le négatif d'une image, il faut remplacer chaque composante RGB par son complémentaire à 255.

Par exemple, si une composante vaut 42, il faut la remplacer par 213 (= 255 - 42).

Correction

Script Python

```
import imageio
img = imageio.imread('VanGogh_Arles.png')

for i in range(img.shape[0]):
    for j in range(img.shape[1]):
        r = 255 - img[i][j][0]
        g = 255 - img[i][j][1]
        b = 255 - img[i][j][2]
        img[i][j] = (r, g, b)

imageio.imsave('img_negatif.png', img)
```

Niveaux de gris

Dans sa norme 709, la Commission Internationale de l'Éclairage propose de remplacer les 3 composantes d'un pixel (r, g, b) par la valeur suivante :

Correction

Script Python

```
import imageio
img = imageio.imread('VanGogh_Arles.png')

for i in range(img.shape[0]):
    for j in range(img.shape[1]):
        r = img[i][j][0]
        g = img[i][j][1]
        b = img[i][j][2]
        m = int(0.2126*r + 0.7152*g + 0.0722*b)
        img[i][j] = (m, m, m)

imageio.imsave('img_gris.png', img)
```

Flip

On retourne l'image horizontalement.

Correction

Script Python

```
import imageio
img = imageio.imread('VanGogh_Arles.png')

# on doit commencer par faire une copie de l'image
img_miroir = img.copy()

for i in range(img.shape[0]):
    for j in range(img.shape[1]):
        img_miroir[i,img.shape[1]-j-1] = img[i,j]

imageio.imsave('img_miroir.png', img_miroir)
```

Photomaton

C'est une transformation réversible, puisqu'on envoie un pixel sur quatre dans chaque carré...

Correction

L'idée est d'«envoyer» chaque pixel dans l'un des 4 carrés, en considérant la parité des indices de ligne `i` et de colonnes `j` :

- les pixels sur une ligne paire sur les carrés du haut;
- les pixels sur une ligne impaire sur les carrés du bas;
- les pixels sur une colonne paire sur les carrés de gauche;
- les pixels sur une colonne impaire sur les carrés de droite;

Script Python

```
import imageio
img = imageio.imread('VanGogh_Arles.png')

img_photomaton = img.copy()

for i in range(img.shape[0]):
    for j in range(img.shape[1]):
        img_photomaton[i//2 + 128*(i%2)][j//2 + 128*(j%2)] = img[i][j]

imageio.imsave('img_photomaton.png', img_photomaton)
```

Pop-art

Le principe est, pour chaque pixel, d'appuyer sur la composante majoritaire: on récupère la composante maximale et on l'augmente d'une certaine valeur (par exemple 50 sur l'image ci-dessous) sans dépasser 255 bien entendu.

Correction

Avec une fonction...

Script Python

```python
import imageio
img = imageio.imread('VanGogh_Arles.png')

def popart(pix: list, val: int) -> list:
    '''
    renvoie la liste des composantes de pix, en ayant augmenté la valeur maximale
    de la valeur val, sans dépasser 255 bien évidemment
    '''
    m = max(pix)
    t = []
    for composante in pix:
        if composante == m:
            t.append(min(255, composante+val))
        else:
            t.append(composante)
    return t

for i in range(img.shape[0]):
    for j in range(img.shape[1]):
        img[i][j] = popart(img[i][j], 50)

imageio.imsave('img_popart.png', img)
```

Pixellisation

Je vous laisse deviner...

Correction

Le principe est de décider tout d'abord d'une taille de «carrés» qui vont composer l'image pixellisée. Bien entendu, cette taille doit être un diviseur commun de la hauteur et de la largeur de l'image. Ici comme l'image fait 256x256, on peut choisir n'importe quelle puissance de 2. Par exemple, prenons 8 pixels. Il y aura donc 256//8 = 32 carrés en hauteur et en largeur.

Ensuite on va définir la couleur «moyenne» qu'on va mettre dans chaque carré: on fait la moyenne des composantes dans le carré (moyenne des rouges, moyenne des verts et moyenne des bleus).

On affecte enfin cette couleur à chaque pixel du carré.

Script Python

```
import imageio
img = imageio.imread('VanGogh_Arles.png')

cote = 8

def couleur_moyenne(ligne, colonne):
    moy = 3 * [0]
    for i in range(cote):
        for j in range(cote):
            for k in range(3):
                moy[k] += img[cote*ligne + i,cote*colonne + j][k] / (cote**2)
    return moy


for ligne in range(img.shape[0]//cote):
    for colonne in range(img.shape[1]//cote):
        for i in range(cote):
            for j in range(cote):
                img[cote*ligne + i,cote*colonne + j] = couleur_moyenne(ligne, colonne)

imageio.imsave("img_pixellisee.png", img)
```

Floutage

Je vous laisse deviner...

Correction

Script Python

```
import imageio
img = imageio.imread('VanGogh_Arles.png')
```

1. en fait cela dépend de l'outil (module) utilisé pour lire et écrire des images. ↩