IMAGES NUMÉRIQUES ET PROGRAMMATION PYTHON

Introduction

Quelques notions sur les images numériques

Exemples de modification d'images

Conclusion

Introduction

IREM d'Aquitaine

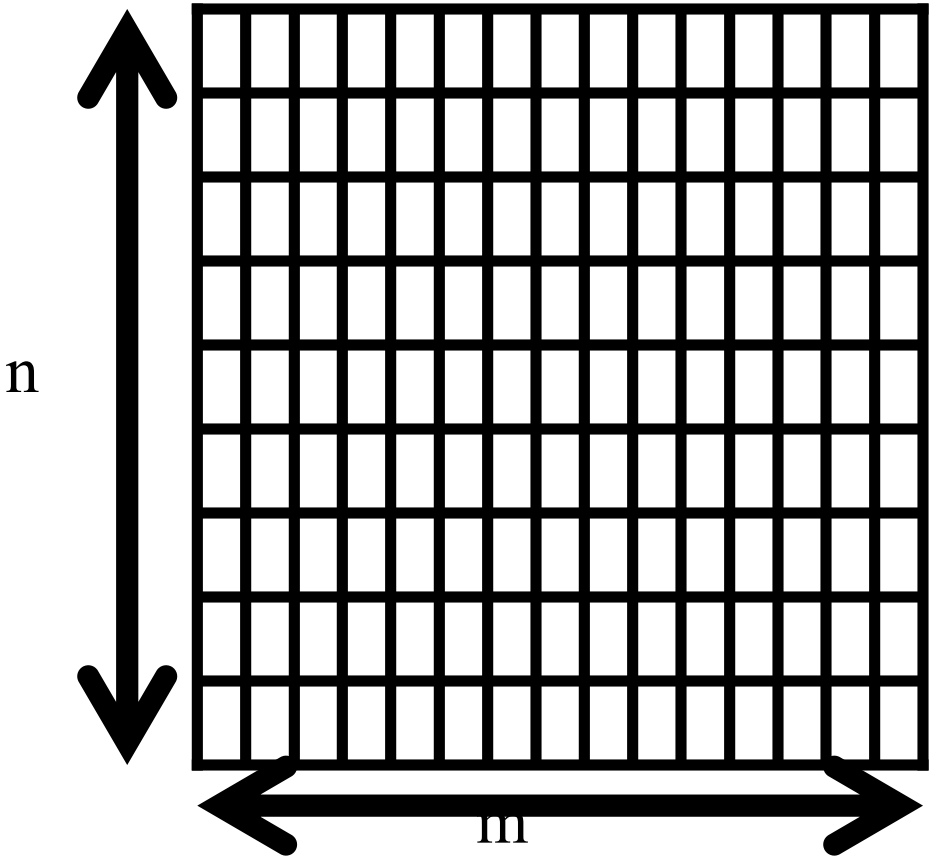
Groupe algorithmique et programmation

Présenté par : Chloé UBERA
Alexandre LOUSTAUNAU

Quelques notions

IMAGES NUMÉRIQUES

Qu'est-ce qu'une image numérique ?



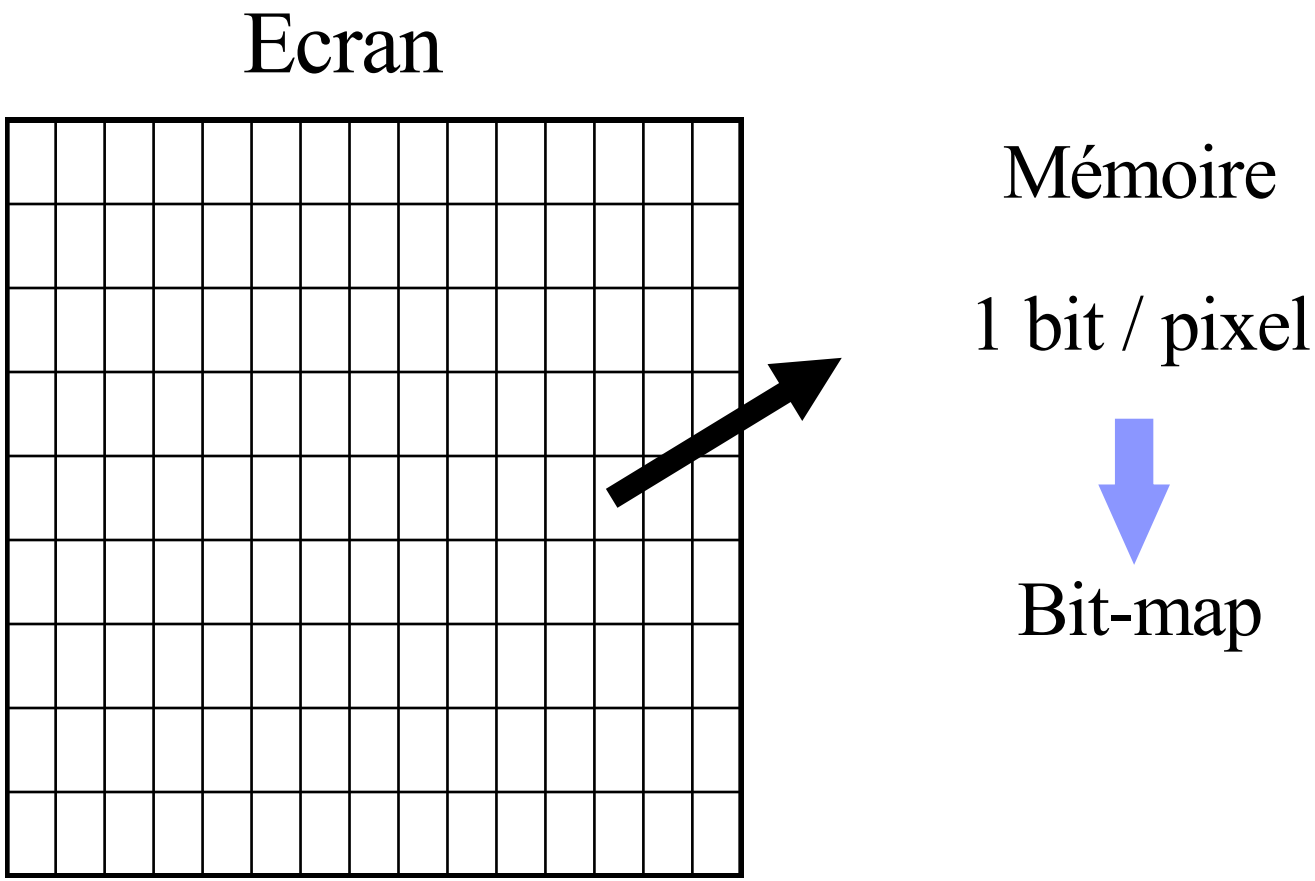
Une matrice
de
n x m
Pixels
(picture element)

C'est-à-dire…

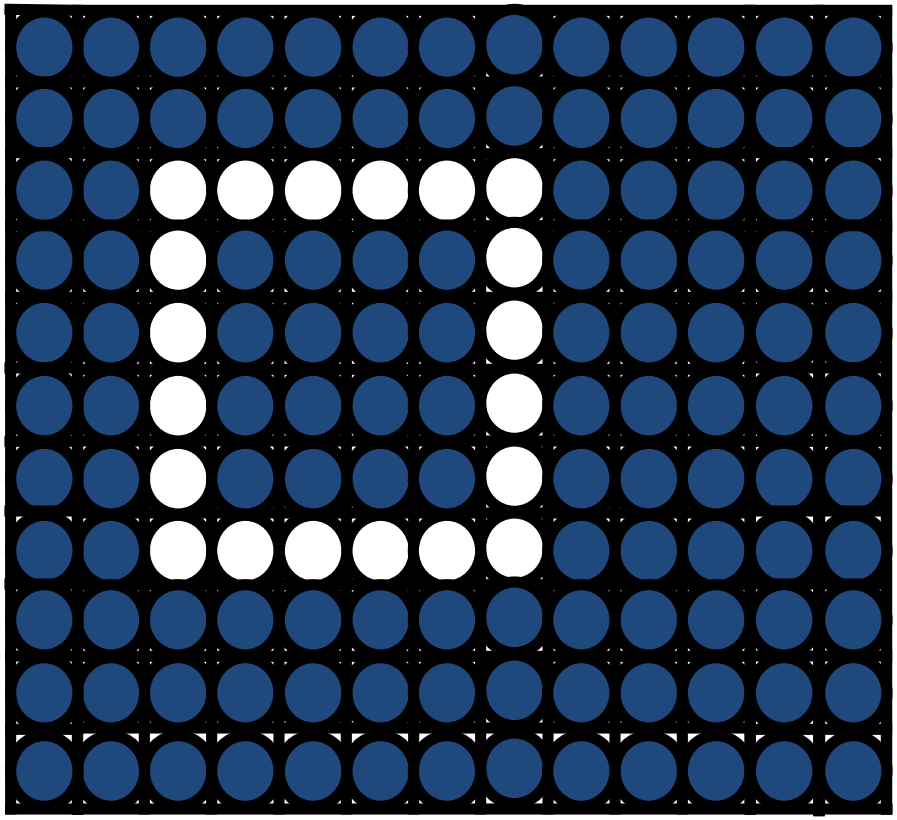
Une image en informatique est un tableau à 2 dimensions de points associés à une couleur.

Les coordonnées `(x,y)` d'un pixel expriment sa position dans le repère de l'image.

Mais encore…



Codage…

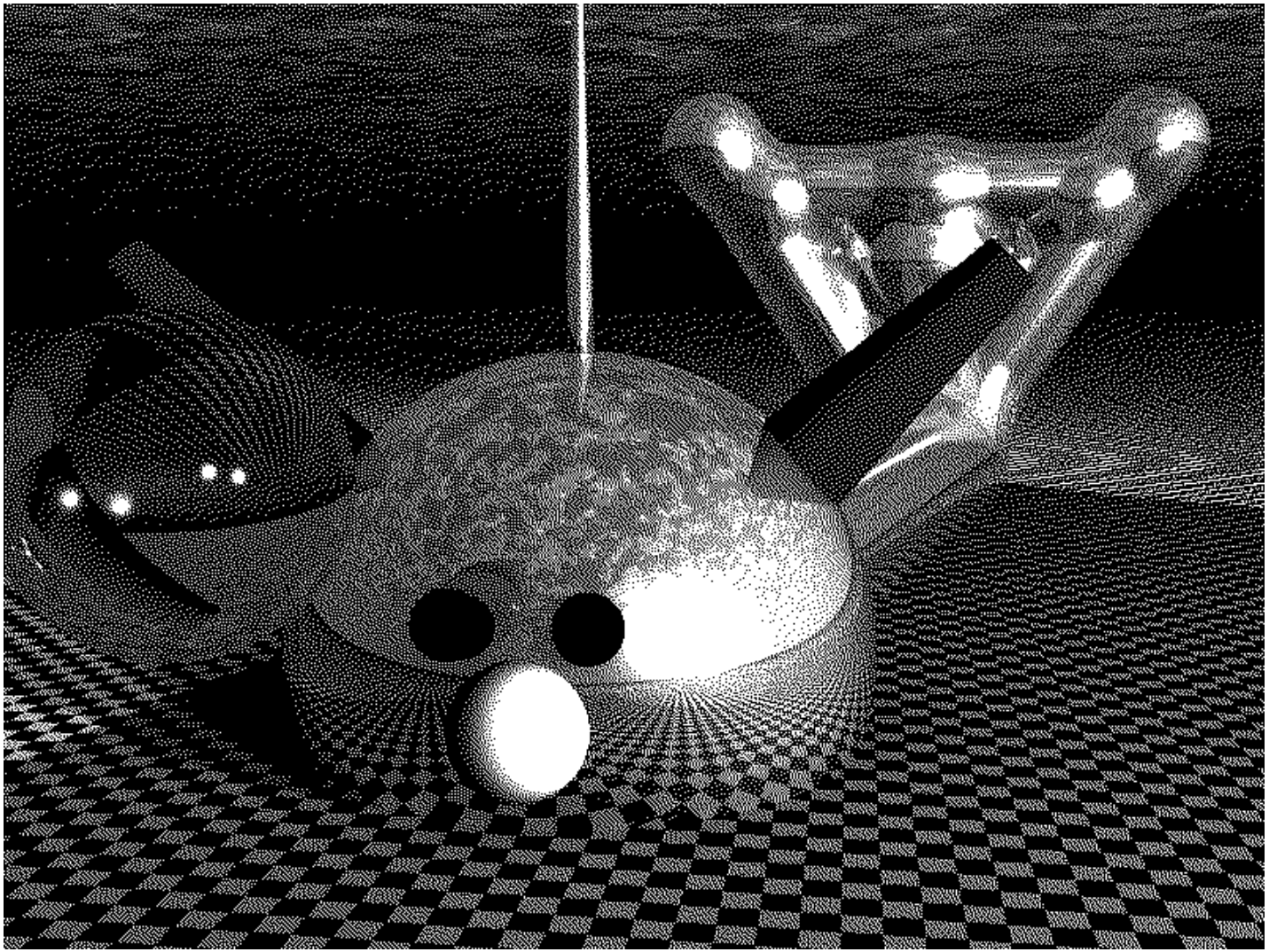


0111 111000
0100001000
0100001000
0100001000
0100001000
0111 111000

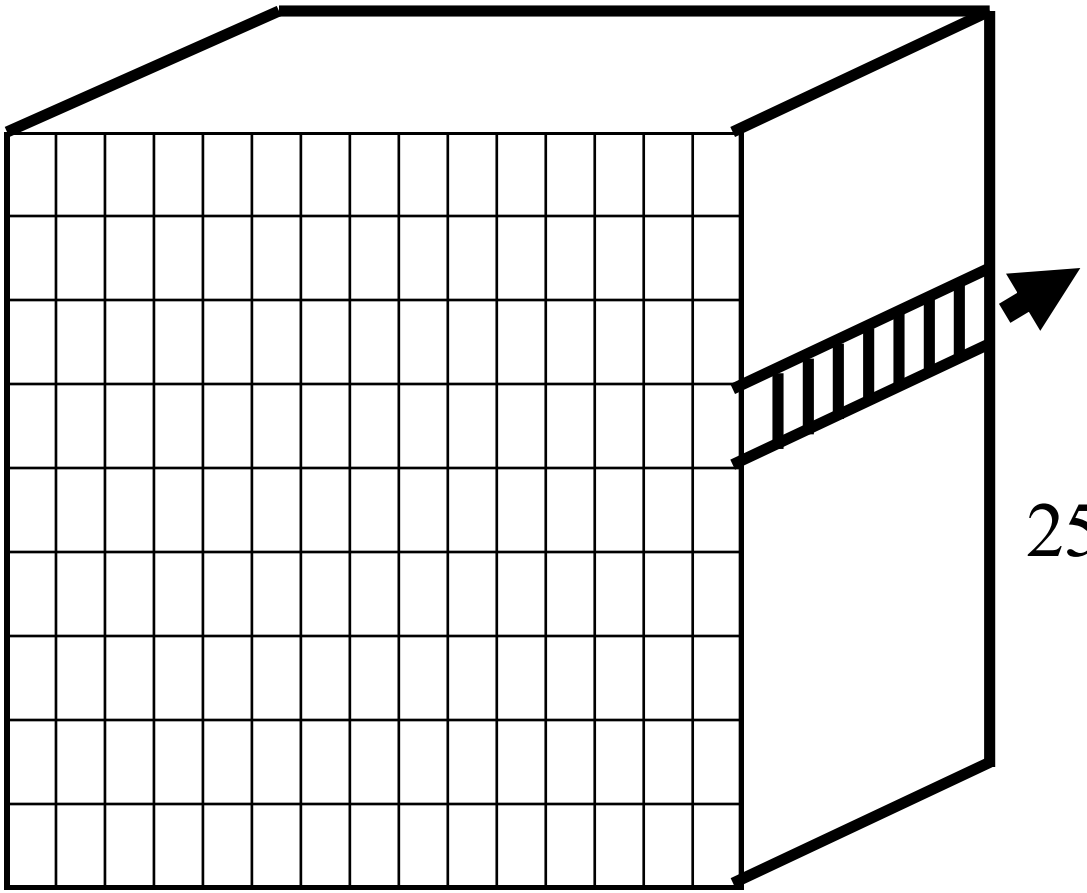
Ecran

Mémoire

Par exemple…



En mémoire…

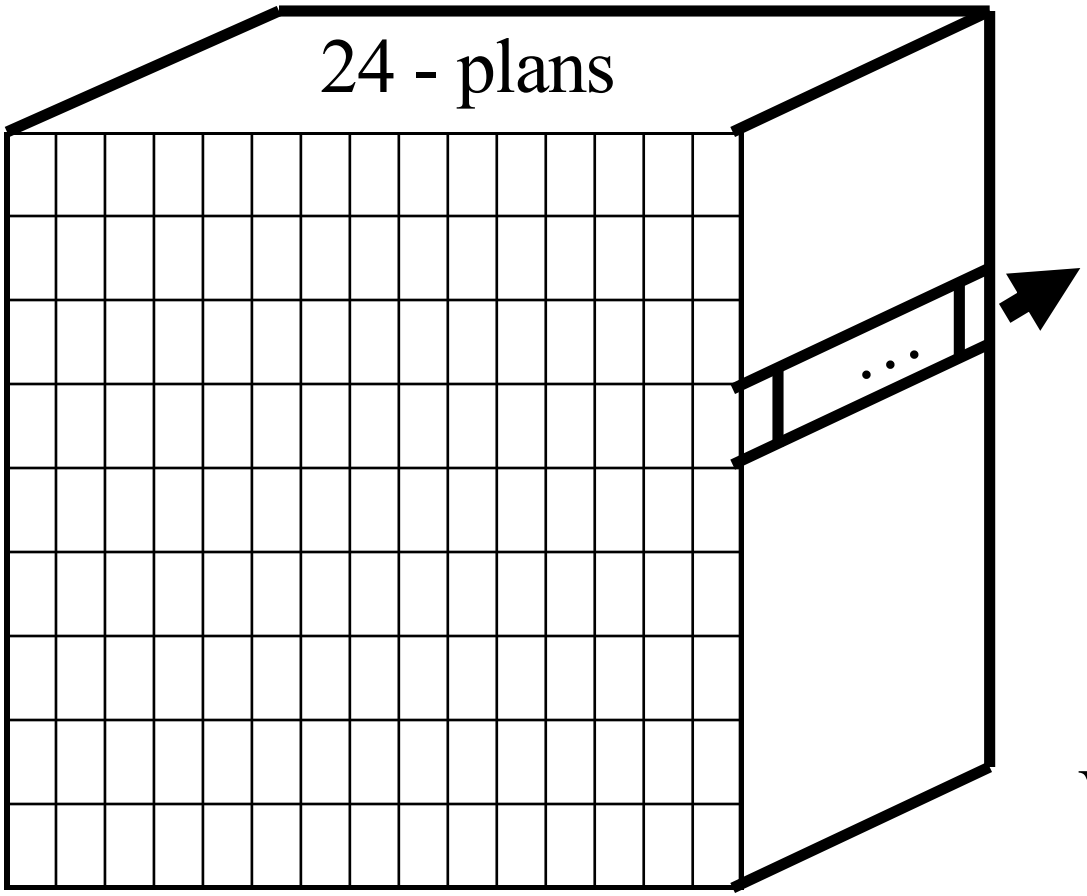


Mémoire

1 octet / pixel

256 niveaux de gris

Ou bien…



Mémoire

3 octets / pixel

16 Millions de Couleurs.

Vraies couleurs

Le pixel…

A chaque pixel est associée une couleur RGB (par exemple) :

- R : Red
- G : Green
- B : Blue

Chaque canal est codé par une valeur entière entre 0 et 255.

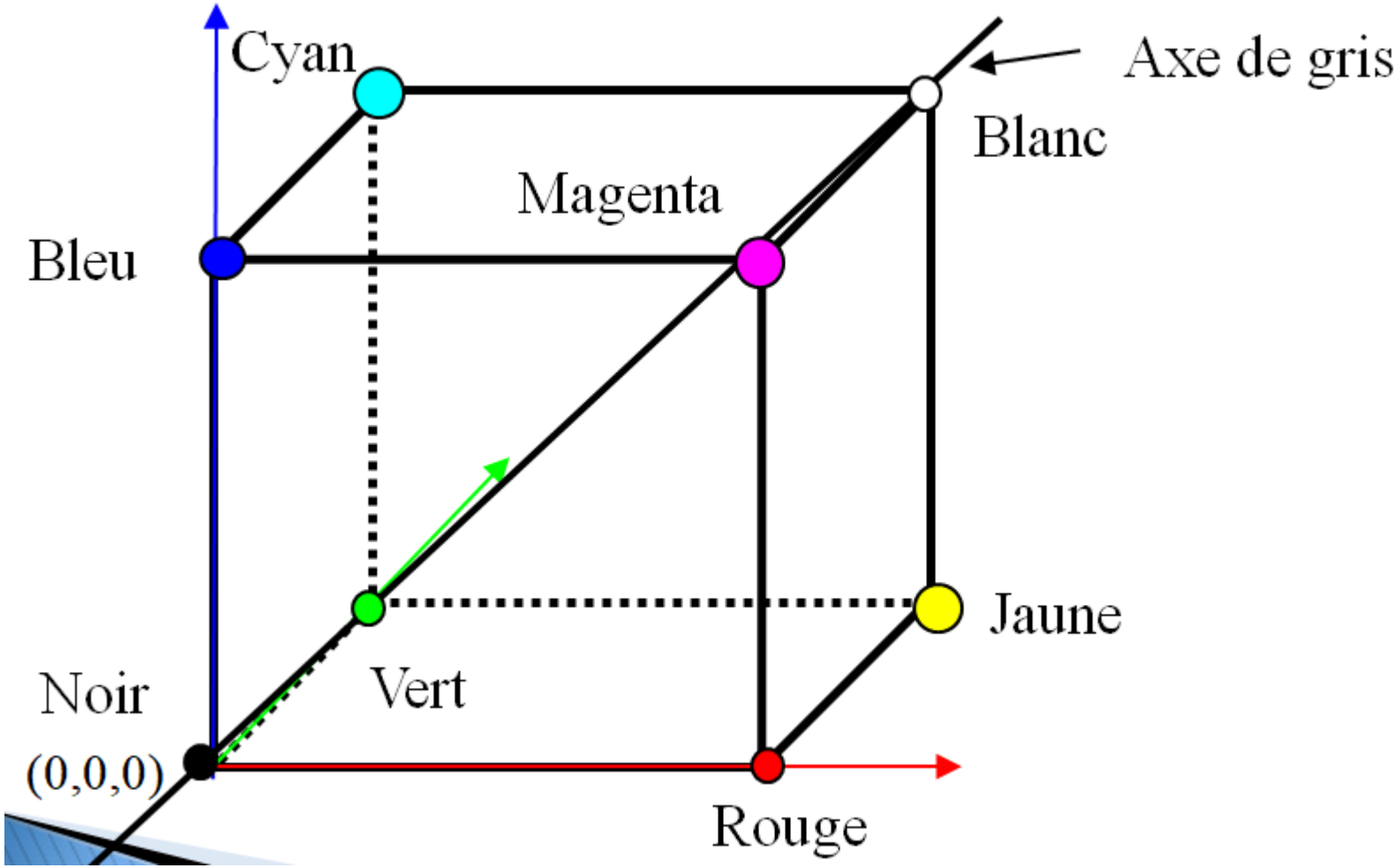
Exemple :

(0,0,0) : noir

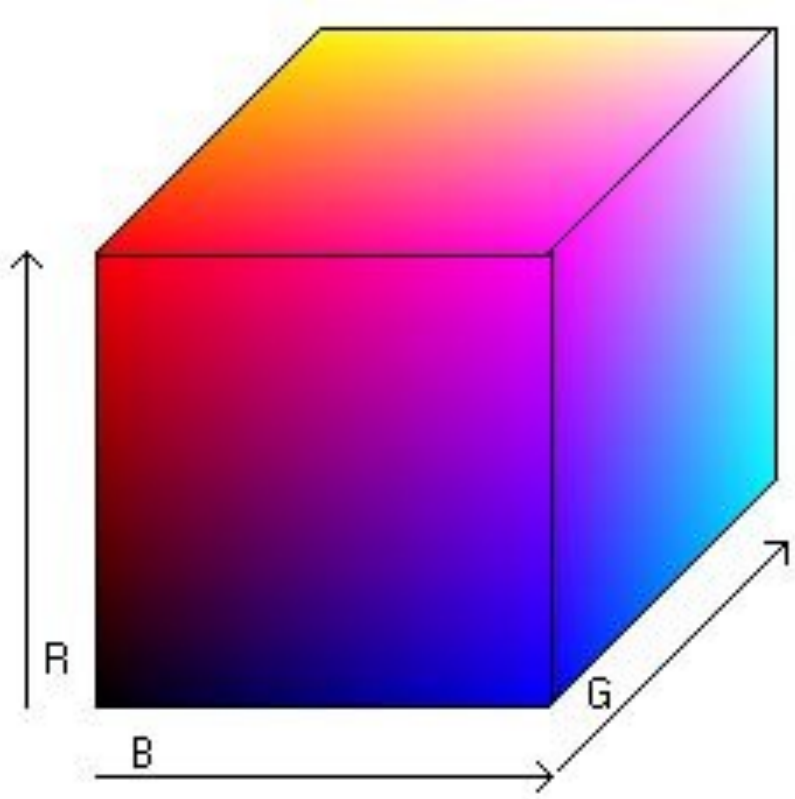
(255,255,255) : blanc

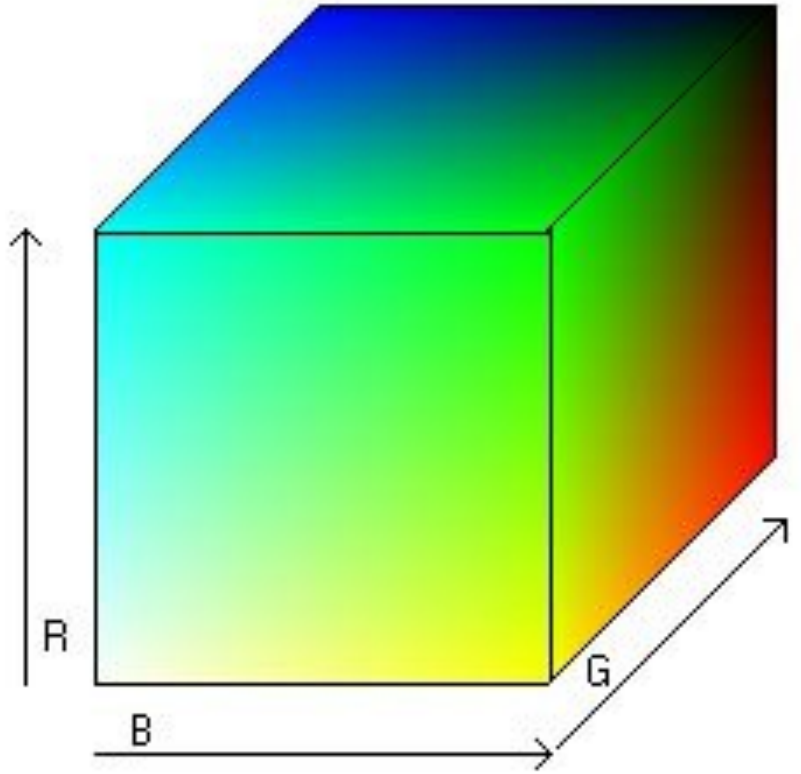
(255,0,0) : rouge

On compose nos couleurs…



D’un côté ou de l’autre…





Quantification des couleurs : RGB vs NGris

$$NGris = 0.299 \times R + 0.587 \times G + 0.114 \times B$$

Les coefficients sont **issus de la sensibilité de l'œil humain**

Les niveaux de gris (NGris) sont invariants par cette transformation.

Exemple :

Si $R = G = B = x$ dans [0, 255]

alors

$$NGris = (0.299 + 0.587 + 0.114)\, x = x$$

Utilisation de la PIL

```
>>PIL.open(nom)
```

Ouvre le fichier « nom » et retourne l'image contenue dedans.
Par exemple :

```
>>monImage1 = PIL.open("teapot.png")
```

l'image ne s'affiche pas

Utilisation de la PIL

```
>>PIL.new("RGB",(large,haut))
```

Retourne une image de taille large × haut , initialement noire

```
>>monImage2 = PIL.new(``RGB'',(300,200))
```

Pour choisir une autre couleur :

```
>>monImage3 =
PIL.new(``RGB'',(300,200),(255,0,0))
```

l'image ne s'affiche pas

Utilisation de la PIL

```
>>PIL.Image.getpixel(img,(x,y))
```

Retourne la couleur du pixel (x,y) de l'image img.

Exemple :

```
>>(r,g,b)= PIL.Image.getpixel(Image2,(50,50))

>>> r
0
>>> g
0
>>> b
0
```

Utilisation de la PIL

```
>>PIL.Image.show(img)
```

Affiche l'image img

Exemple :

```
>>monImage3 = PIL.new(''RGB'',300,200),(255,0,0))
>>PIL.Image.show(monImage3)
```

Utilisation de la PIL

```
>>PIL.Image.putpixel(img,(x,y),(r,g,b))
```

Peint le pixel (x, y) dans l'image img de la couleur (r, g, b) : modification de la couleur initiale.

Exemple :

```
>>PIL.Image.putpixel (Image3,(50,50),(255,255,255))
```

```
>>PIL.Image.show(Image3)
```

Utilisation de la PIL

Le principe :

Toutes les fonctions qui modifient les images seront écrites par répétitions de cette action sur tout ou partie des pixels de l'image.

Fin

ATELIER CORFEM

Images numériques et programmation Python

Une image numérique matricielle (bitmap en anglais) est une matrice d'éléments appelés pixels. Chaque pixel possède des coordonnées (x,y) au sein de l'image, ainsi qu'une couleur codée sous un certain format, par exemple RGB (pour red-green-blue, ou RVB en français).

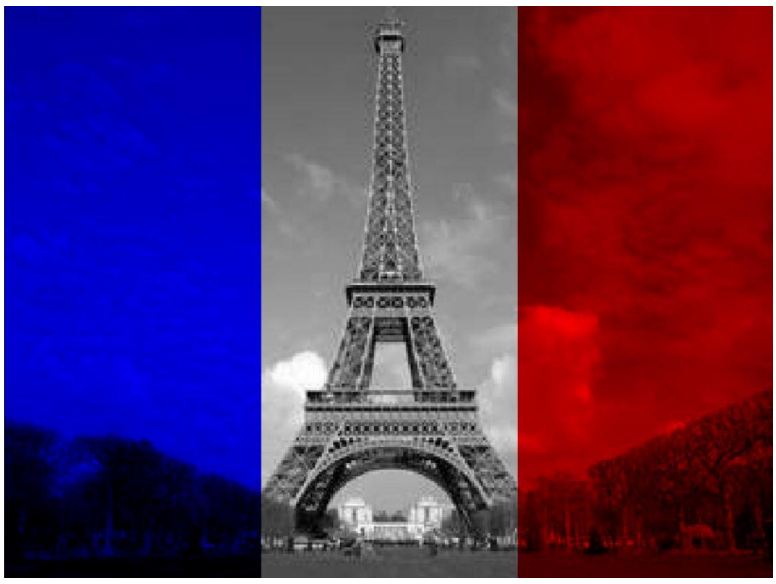
Il est possible de manipuler de telles images numériques en Python, grâce à la bibliothèque de fonctions PIL (pour Python Image Library). Au cours de cet atelier ludique, nous verrons comment, à partir de nos connaissances en mathématiques et en algorithmique, réaliser certaines opérations sur une image : flouter une image, appliquer un filtre de couleur, passer en niveaux de gris, postériser une image, insérer une image dans une autre, restaurer une image dégradée, effectuer des symétries, ou encore des rotations...

FICHIER 1

L'objectif est d'écrire une fonction *filtreBleuBlancRouge* qui prend pour argument une image et qui retourne cette image filtrée comme ci-dessous.

« toureiffel.jpg » :

filtreBleuBlancRouge(toureiffel)



Aide :

```
############## import ###################
import PIL.Image as PIL
import math
########################################
toureiffel = PIL.open("toureiffel.jpg")
```

```
def filtreBleu(img):
    # ne conserve que la composante bleue de l'image

    l, h = img.size
    for x in range(l):
        for y in range(h):
            (r, g, b) = PIL.Image.getpixel(img, (x, y))
            PIL.Image.putpixel(img, (x, y), (0, 0, b))
```

FICHIER 2

L'objectif est d'écrire une fonction *NoirEtBlanc* qui prend pour argument une image et qui retourne cette image en noir et blanc comme ci-dessous.

« bus.jpg » :

NoirEtBlanc(bus)

monochrome1(bus)

monochrome2(bus)

Aide :

```
############## import ####################
import PIL.Image as PIL
import math
##########################################
bus = PIL.open("bus.jpg")
```

```
def filtreBleu(img):
    # ne conserve que la composante bleue de l'image

    l, h = img.size
    for x in range(l):
        for y in range(h):
            (r, g, b) = PIL.Image.getpixel(img, (x, y))
            PIL.Image.putpixel(img, (x, y), (0, 0, b))
```

FICHIER 3

L'objectif est d'écrire une fonction *InsererImage* qui prend pour arguments 2 images img1 et img2 et qui retourne une image insérant img1 dans img2 comme ci-dessous.

« bus.jpg »

« route.jpg »

InsererImage(bus,route)

Aide :	A vous de réfléchir !
############ import ################### import PIL.Image as PIL import math ## bus = PIL.open("bus.jpg") route = PIL.open("route.jpg")	

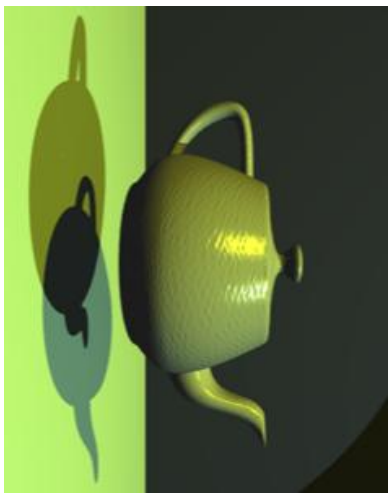
FICHIER 4

L'objectif est d'écrire une fonction *MiroirHorizontal* qui prend pour argument une image et qui retourne le miroir horizontal de cette image.

« teapot.png »

MiroirHorizontal (teapot)

rotation90 (teapot)



Aide :

```
############## import ###################
import PIL.Image as PIL
import math
########################################
teapot = PIL.open("teapot.png")
```

```
def miroirVertical(img):
    # renvoie le miroir vertical de l'image img

    l, h = img.size
    imgMiroirV = PIL.new("RGB", (l, h))
    for x in range(l):
        for y in range(h):
            c = PIL.Image.getpixel(img, (x, y))
            PIL.Image.putpixel(imgMiroirV, (x, h - 1 - y), c)
    return imgMiroirV
```

Prolongement :

Ecrire une fonction *rotationImage(img, angle)* qui renvoie l'image img tournée de angle radians dans le sens horaire.

FICHIER 5

L'objectif est d'écrire une fonction *flou* qui prend pour argument une image et qui retourne cette même image floue.

« portrait.jpg »

flou(portrait) ou flou(flou(flou(portrait)))

Aide :	On renvoie une image floue en peignant chaque pixel non au bord par la couleur moyenne des pixels voisins.
############ import ################### import PIL.Image as PIL import math ## portrait = PIL.open("portrait.jpg")	

FICHIER 6

L'objectif est d'écrire une fonction *croix* qui prend pour argument une image et qui retourne cette même image barrée par le tracé de ses 2 diagonales.

« mickey.jpg »

croix(mickey)

croix_epaisse (mickey,15)

Aide :	def ligneHorizontaleBlancheAuMilieu(img):
############ import ################### import PIL.Image as PIL import math ## mickey = PIL.open("mickey.jpg")	#dessine une ligne horizontale #blanche au milieu de l'image img l,h=img.size for x in range(l): PIL.Image.putpixel(img,(x,h//2),(255,255,255))

FICHIER 7

L'objectif est d'écrire une fonction *debruitage* qui prend pour argument une image bruitée et qui retourne cette même image débruitée.

« femme_bruitee.jpg »

debruitage(femme_bruitee)

Aide : Le statisticien J.W Tuckey qualifiait d'aberrantes les valeurs d'une série statistique se trouvant en dehors de l'intervalle :

$$[\ Q_1 - \frac{3}{2}(Q_3 - Q_1)\ ;\ Q_3 + \frac{3}{2}(Q_3 - Q_1)\ ]$$

Par exemple, on considère :

100	102	99
101	25	103
103	100	103