Annexe 1 – Exemples d'activités sur les images numériques

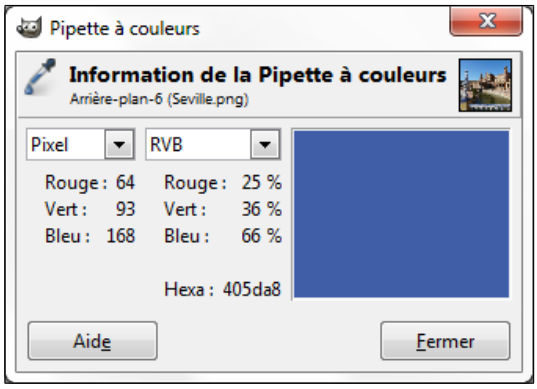
1 Accès aux composantes RVB d'un pixel dans GIMP

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 : 

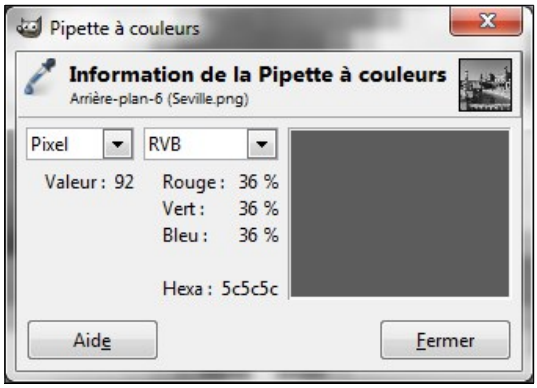
et cocher 

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 = 64, V = 93, B = 168.



On peut recommencer l'opération sur le même pixel après avoir passé l'image en niveaux de gris : menu Image > Mode > Niveaux de gris :

Une seule valeur est indiquée : R = V = B = 92 ici.



2 Exemple de programme Python pour passer d'une image couleur à une image en niveaux de gris

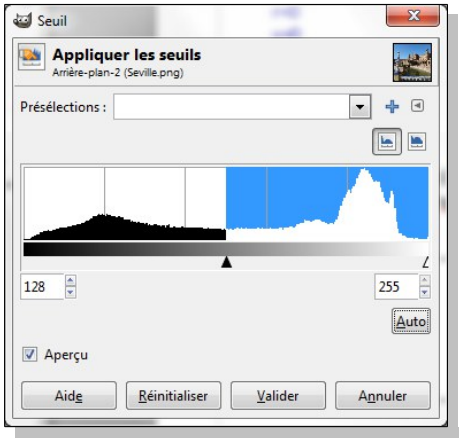
```
from PIL import Image                    # la bibliothèque PIL doit avoir été installée
im1 = Image.open("T:\Seville.png")       # ouverture du fichier image source
L,H = im1.size                           # L = largeur de l'image, H = sa hauteur, en pixels
im2 = Image.new("RGB",(L,H))             # création de l'image destination (niveaux de gris)
                                         # « vide » pour l'instant
for y in range(H):                       # on balaie toutes les lignes de l'image source, de 0 à H-1
 print y                                 # on affiche la ligne traitée pour patienter
                                         # mais le traitement est assez rapide
 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
                                         # p[0] est la composante rouge
                                         # p[1] la composante verte, et p[2] la composante bleue
      gris = (p[0]+p[1]+p[2])/3          # on calcule le niveau de gris (gris « moyenne » ici)
      r = gris                           # en niveaux de gris, les composantes RVB
      v = gris                           # sont toutes égales
      b = gris
      im2.putpixel((x,y),(r,v,b))        # on écrit le pixel modifié sur l'image destination
im2.save("T:\Seville_niv_gris.png")      # on stocke l'image résultante, ici sous T:\
im2.show()                               # on montre l'image
```

3 Exemple de programme pour passer d'une image couleur à une image en noir et blanc

Les lignes modifiées sont en bleu.

```
from PIL import Image
im1 = Image.open("T:\Seville.png")
L,H = im1.size
im2 = Image.new("RGB",(L,H))
seuil = 128                 # seuil de gris en-dessous duquel un pixel sera noir
                            # à ajuster en fonction de ce qu'on souhaite mettre en valeur dans l'image
for y in range(H):
    print y
    for x in range(L):
        p = im1.getpixel((x,y))
        gris = (p[0]+p[1]+p[2])/3
        if gris < seuil:    # si le niveau de gris calculé est inférieur au seuil, le pixel est noir
            gris = 0        # (R=V=B=0)
        else:               # sinon il est blanc
            gris = 255      # (R=V=B=255)
        r = gris
        v = gris
        b = gris
        im2.putpixel((x,y),(r,v,b))
im2.save("T:\Seville_N&B.png")
im2.show()
```

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 :



4 Petit lexique PIL (Python Imaging Library)

Ce lexique peut être donné aux élèves pour concevoir une partie d'un programme par exemple.

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

5 Superposition de deux images en une seule

On ne conserve que la composante rouge de la première (filtre rouge), et les composantes verte et bleue de la seconde (filtre cyan). R = R(image1), V = V(image2), B = B(image2). Si l'on utilise deux images de la même scène prises avec un léger décalage pour reproduire la vision droite et la vision gauche, on obtient un anaglyphe ! Le tout est à regarder avec des lunettes « 3D » munies d'un filtre rouge pour un œil et cyan pour l'autre. Dans le premier cas (image de gauche ci-dessous), on ouvre l'œil droit ou l'œil gauche; dans le second (anaglyphe), les deux ...

Au hasard de discussions avec les élèves, l'analogie peut être faite avec la répartition d'une bande de fréquences entre plusieurs émetteurs radiophoniques, qui est indissociable du filtrage.

Exemple de résultats; le programme ne dépasse pas 15 lignes … (2 images sources, 1 image destination ...)

6 Les images transparentes

Le format d'image PNG permet de gérer la transparence des images, ce qui est appréciable pour la composition de pages web par exemple. Le programme ci-dessous permet d'obtenir une image dont la transparence est ajustable :

```
from PIL import Image
im1 = Image.open("T:\Seville.png")
L,H = im1.size
                                # conversion de l'image RGB(1) en RGBA (A pour alpha, la transparence)
im2 = im1.convert("RGBA")
im3 = Image.new("RGBA",(L,H))      # image destination (« vide » pour l'instant)
for y in range(H):
 for x in range(L):
      p = im2.getpixel((x,y))
      r = p[0]
      v = p[1]
      b = p[2]
      a = p[3]-100                 # p[3] correspond à la transparence du pixel p
                                   # 255 = transparence minimale et 0 = transparence totale
                                   # ici, la transparence de tous les pixels vaudra 255-100 = 155
                                   # écriture du pixel sur l'image destination; on rajoute le paramètre a
```

(1) RGB pour Red Green Blue (RVB donc)

```
      im3.putpixel((x,y),(r,v,b,a))
im3.save("T:\Seville_transparente.png")
im3.show()
```

7 Quelques éléments sur la rotation des images

Il s'agit bien ici de recalculer à chaque étape les coordonnées de chaque pixel et non d'utiliser la méthode « rotate » intégrée à la bibliothèque PIL, auquel cas le programme se résumerait à ces trois lignes pour la rotation d'un quart de tour :

```
from PIL import Image
im = Image.open("T:\Seville.png")
im.rotate(90).show()
```

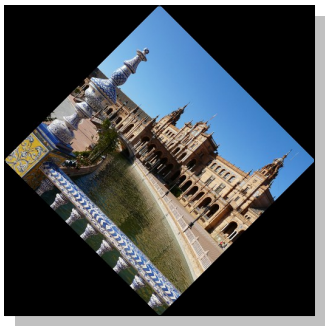
Si l'on se contente de permuter les coordonnées x et y de chaque pixel, on obtient une rotation d'un quart de tour accompagnée d'un retournement. Il faut donc approfondir cette idée.

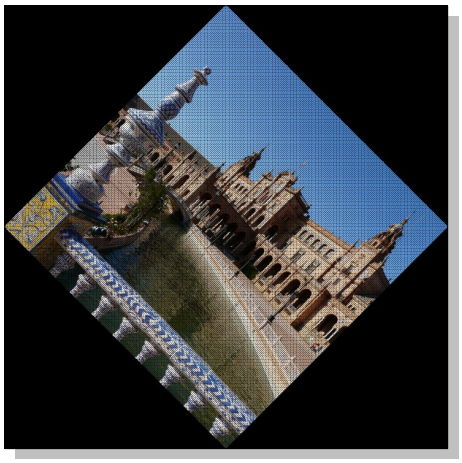
Pour une rotation d'un huitième de tour, on peut essayer l'algorithme suivant :

```
x1 = x-L/2              # translation dans le coin supérieur gauche avant rotation
y1 = y-H/2              # L = largeur de l'image, H = sa hauteur, en pixels
x2 = k*(x1-y1)          # rotation de 45°
y2 = k*(x1+y1)
x3 = int(x2+L/2)        # recentrage du pixel sur l'image destination
y3 = int(y2+H/2)
```

En supposant qu'on veuille une image de mêmes dimensions que l'image source, quelle valeur donner à k ? Sans cette contrainte, si on utilise le facteur $\sqrt{2}/2$, on obtient une image présentant des trous ... pourquoi ? À noter que les calculs de racines carrées nécessitent de rajouter en tête de programme la ligne suivante :

```
from math import *      ou au minimum      from math import sqrt
```





8 Extraction d'une sous-image

L'exemple proposé ici permet d'envisager ultérieurement une segmentation de la sous-image, c'est à dire l'extraction d'une image pour chaque caractère, en vue d'une reconnaissance automatique par exemple.

Image source, et zone à extraire encadrée en rouge

Image destination

En notant (x_0,y_0) les coordonnées du pixel supérieur gauche de la zone à extraire, L_0 et H_0, la largeur et la hauteur de cette zone, l'algorithme consiste à créer une image destination de largeur L_0 et de hauteur H_0 puis à copier chaque pixel de coordonnées (x,y) en $(x-x_0,y-y_0)$ sur l'image destination.

9 Application d'un masque à une image

Le masque est une forme quelconque, créée à l'aide de GIMP par exemple; il est plus simple de lui donner une couleur facilement identifiable, par exemple bleu (0000ff c'est à dire R = 0, V = 0, et B = 255) :

Créer une image RVB, de mêmes dimensions que l'image à traiter :

Menu Fichier > Nouvelle image … (cliquer sur Options avancées dans la fenêtre qui s'ouvre)

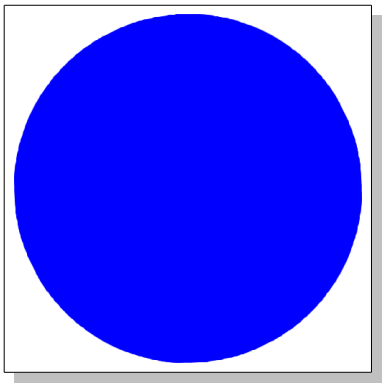
Créer une forme géométrique :

Menu Filtres > Rendu > Figures géométriques … (pointeur de la souris au centre, puis étirer)

Enregistrement de l'image au format PNG :

Menu Fichier > Enregistrer sous … Choisir le nom du fichier. Sélectionner le type de fichier PNG. Puis cliquer sur Exporter et sur Enregistrer.

image source



masque

image « masquée »

L'algorithme consiste à créer une image destination de couleur blanche, ou mieux, transparente, de même

dimensions que l'image source et le masque. On parcourt tous les pixels du masque à l'aide d'une double boucle : si le pixel (x,y) n'est pas blanc, alors on copie le pixel (x,y) de l'image source en (x,y) sur l'image destination.

On peut aussi envisager un masque aux contours flous, ou encore un masque en forme de lettres :

Dans le cas d'un masque de forme simple, comme le disque ci-dessus, on peut envisager un algorithme sans masque :

On se fixe une valeur pour le rayon du disque, par exemple, rayon_disque = 0.9*L/2 (L est la largeur de l'image). On parcourt tous les pixels de l'image source : si la distance d du pixel (x,y) par rapport au centre de l'image est inférieure ou égale à rayon_disque, alors on copie ce pixel sur l'image destination.

$d = \sqrt{(x - L/2)^2 + (y - H/2)^2}$ d'après le théorème de Pythagore.

Rappel : l'utilisation de la fonction racine carrée (sqrt) nécessite de placer la ligne from math import * en début du programme Python.

10 Correction des couleurs d'une image à l'aide d'une courbe tonale

Dans GIMP, le menu Couleurs > Courbes … fait apparaître l'histogramme de l'image et une courbe initialement du type y = x. Voir l'illustration ci-dessous.

Si l'on saisit un point au milieu de la droite à l'aide de la souris et que l'on tire vers la bas, l'image devient plus foncée. En effet, en abscisse figurent les niveaux de gris initiaux, et en ordonnée les niveaux de gris résultants. La courbe étant sous la droite y = x, un niveau de gris donné (en abscisse) est remplacé par un niveau de gris plus petit donc plus foncé (en ordonnée). Le Canal étant initialisé par défaut à « Valeur » (voir ci-dessous), la correction est appliquée à l'identique sur les 3 canaux R,V,B, mais on peut sélectionner successivement chaque canal et appliquer une correction différente, pourquoi pas.

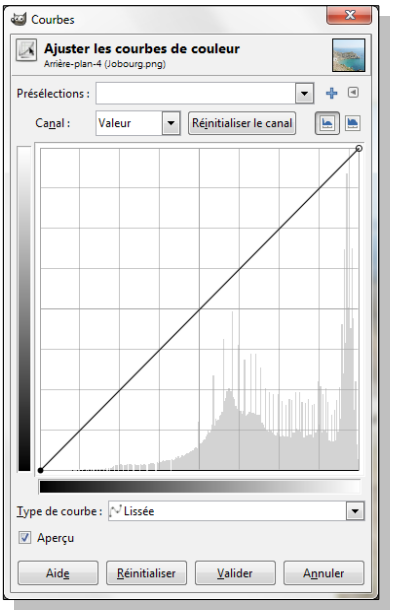
Dans le cas d'une correction gamma, un niveau n ($0 \leq n \leq 255$) est remplacé par $255\left(\frac{n}{255}\right)^{(1/\text{gamma})}$

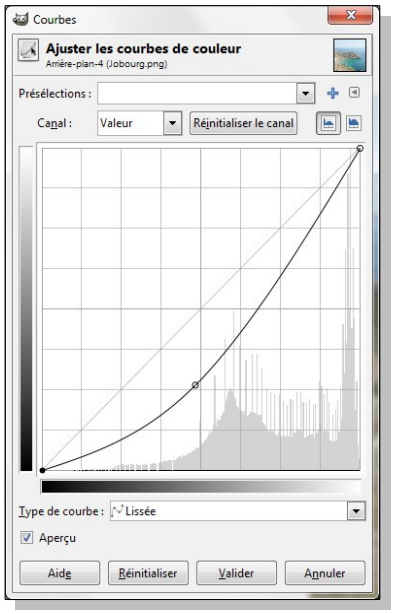
gamma = 1 ne change rien; gamma > 1 éclaircit l'image, tandis que gamma < 1 la fonce.

En langage Python, x**y ou pow(x,y) permet de calculer x à la puissance y.

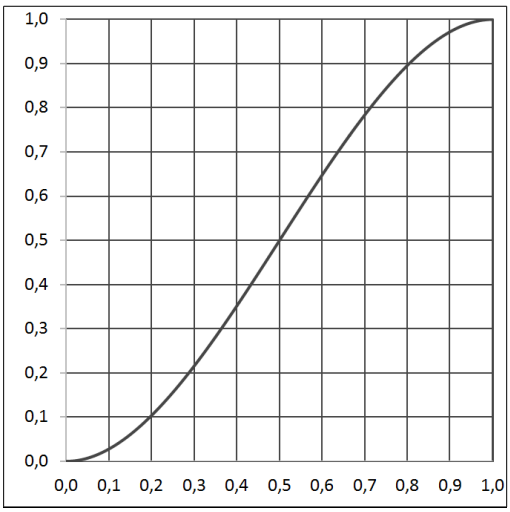
Il faut penser à prendre la partie entière du résultat, int(...), puisque c'est un entier compris entre 0 et 255.

Pour réaliser une correction de type gamma dans GIMP, il faut passer par le menu Couleurs > Niveaux … La valeur qui s'affiche sous le curseur central en forme de triangle est la valeur de gamma, que l'on peut modifier directement, ou en déplaçant le curseur. Pour visualiser la correction sous forme de courbe tonale, il suffit de cliquer sur le bouton : 





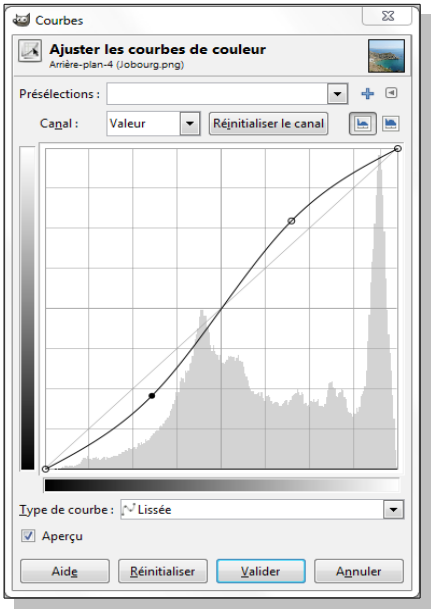
On peut aussi envisager une correction à l'aide d'une courbe « en S » ayant pour équation $y=S(x)=3x^2-2x^3$ (pour x compris entre 0 et 1). L'image de l'intervalle [0;1] par la fonction S étant ce même intervalle, les niveaux RVB associés restent compris entre 0 et 255. Les niveaux 0 et 255 ne sont pas modifiés car la courbe passe par les points (0,0) et (1,1). Quant aux tons intermédiaires, le contraste est amélioré dans toute la zone où la pente de la courbe est accrue, en laissant cependant le niveau moyen inchangé.
Concrètement, on ramène chaque niveau n ($0 \leq n \leq 255$) R, V, ou B, dans l'intervalle [0;1] par : $x \leftarrow n/255$. Puis on applique la correction ci-dessus : $x \leftarrow 3x^2-2x^3$. Enfin, on fait l'opération inverse de la précédente : $n \leftarrow x*255$, dont il faut bien sûr arrondir le résultat à l'entier le plus proche.
D'autres courbes peuvent être envisagées évidemment.



Courbe d'équation $y = 3x^2 - 2x^3$

Photo précédente à laquelle une correction par la courbe tonale ci-contre a été appliquée.

Dans GIMP (menu Couleurs > Courbes ...), pour obtenir une correction de ce type, on peut placer des points de contrôle, soit en cliquant, soit en étirant la droite vers le bas dans les tons foncés et vers le haut dans les tons clairs. On peut aussi choisir « Main levée » comme type de courbe et, après avoir dessiné cette courbe, revenir au choix « Lissée ». L'effet obtenu est sensiblement le même que celui obtenu par programmation; il peut être affiné :



Crédits photographiques : les photographies sont la propriété de l'auteur.