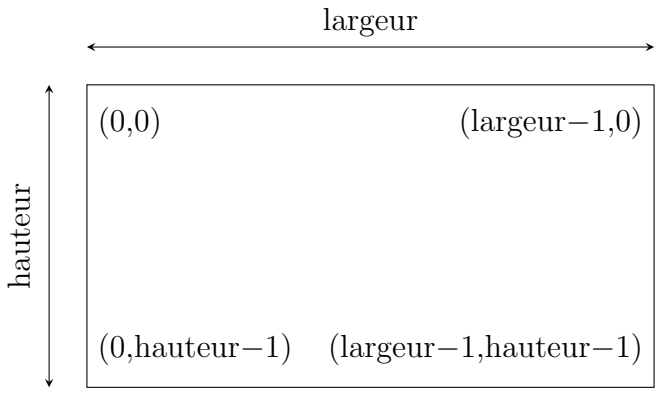
Informatique 1ere année, CPBX, TD3

Carole Blanc, Paul Dorbec

Les coordonnées (x, y) d'un pixel expriment sa position au sein de l'image : x est son abscisse, en partant de la gauche de l'image, et y est son ordonnée, en partant du haut de l'image (à l'inverse de l'ordonnée mathématique, donc). Elles partent de 0. Le schéma ci-dessous indique en exemple les coordonnées des pixels situés aux coins d'une image.



1 Mise en jambes

python permet bien sûr de manipuler des images, à l'aide de la *python Imaging Library* (PIL), qui est préinstallée sur vos machines. Il faudra donc, pour ce chapitre, appeler

```
from PIL.Image import *
```

pour importer la PIL. On dispose alors d'un ensemble de fonctions pour manipuler des images. Voici un résumé des fonctions que nous utiliserons dans cette feuille :

`open(nom)` (ou `Image.open` dans Python Tutor)	Ouvre le fichier *nom* et retourne l'image contenue dedans (par exemple `open("teapot.png")`.
`Image.save(img, nom)`	Sauvegarde l'image *img* dans le fichier *nom*.
`{new("RGB",(large,haut))` (ou `Image.new` dans Python Tutor)	Retourne une image de taille $large \times haut$, initialement noire.
`Image.show(img)` (seulement dans Idle3)	Affiche l'image *img*.
`Image.putpixel(img, (x,y), (r,g,b))`	Peint le pixel (x, y) dans l'image *img* de la couleur (r, g, b)
`Image.getpixel(img, (x,y))`	Retourne la couleur du pixel (x, y) dans l'image *img*

Question 1.1 *Exécuter les instructions suivantes :*

```
image1 = new("RGB",(300,200))
Image.putpixel(image1, (50,50),(255,255,255))
Image.show(image1)
```

1. *Expliquez ce qu'effectue chaque instruction, et observez l'image produite. Confirmez ainsi le sens dans lequel fonctionnent les coordonnées.*
2. *Peignez le pixel de coordonnées* $(250, 150)$ *en vert (n'oubliez pas d'appeler de nouveau* `Image.show(img)` *pour afficher le résultat).*
3. *Peignez le pixel de coordonnées* $(50, 150)$ *en violet.*
4. *Exécutez l'instruction suivante :*

   ```
   (l,h) = image1.size
   ```

   Et observez le contenu de `l` *et de* `h`.
5. *Récupérez une image de la célèbre théière de l'Utah sur le moodle et sauvegardez-la dans le répertoire de vos fichiers* `.py`*. Exécutez les instructions suivantes :*

   ```
   image2 = open("teapot.png")
   Image.show(image2)
   ```

On pourra donc facilement travailler sur cette photo.

Question 1.2 *Qu'effectue la fonction suivante ?*

```
def remplit(img):
    (l,h) = img.size
    for x in range(l):
        for y in range(h):
            Image.putpixel(img, (x,y),(255,255,255))
```

1. *Décrivez précisément ce qu'effectue chaque ligne, testez-la.*
2. *En vous inspirant très largement de cette fonction* `remplit`*, écrivez une fonction* `rectanglePlein (img,x1,x2,y1,y2)` *qui dessine un rectangle plein dont les coins sont les pixels de coordonnées* $(x1, y1)$*,* $(x2, y1)$*,* $(x1, y2)$*,* $(x2, y2)$*. Vérifiez que vous avez bien affecté la couleur du point de coordonnées* $(x2, y2)$*.*

2 À vous de jouer

Question 2.1 *Écrivez une fonction* `f(img)` *qui dessine une ligne horizontale blanche au milieu de l'image (séparant donc l'image en deux parties). Testez-la sur une image noire (générée par un appel à* `new`*).*

Ajoutez un paramètre `c` *à votre fonction* `f`*, et remplacez, dans le corps de* `f`*,* `(255,255,255)` *par* `c`*, ce qui permet ainsi de désormais choisir la couleur de la ligne lors de l'appel. Testez :*

```
f(img,(255,0,0))
Image.show(img)
f(img,(0,255,0))
Image.show(img)
```

`f` *récupère donc l'ensemble des trois valeur* r, g, *et* b *dans le seul paramètre* `c`, *qu'elle peut passer tel quel à* `Image.putpixel`.

Question 2.2 *Écrivez une fonction* `rectangleCreux(img,x1,x2,y1,y2,c)` *qui dessine les côtés d'un rectangle de couleur* `c`, *les coins du rectangles étant les pixels de coordonnées* $(x1, y1)$, $(x2, y1)$, $(x1, y2)$, $(x2, y2)$. *Il s'agit bien sûr de dessiner quatre lignes formant les côtés du rectangle. Testez-la plusieurs fois avec des coordonnées différentes et des couleurs différentes, sur une image noire et sur la photo de théière.*

Que se passe-t-il si l'un des paramètres dépasse de la taille de l'image ? Est-il facile de corriger le problème ?

Question 2.3 *Écrivez une fonction* `diagonale(img)` *qui dessine en blanc la ligne diagonale entre le coin en haut à gauche et le coin en bas à droite de l'image (en vous rappelant le théorème de Thalès). Testez avec différentes tailles d'image. En particulier, testez sur une image* 50×400. *Corrigez l'erreur que vous avez commise.*

3 Manipulation de couleurs

Il est très courant d'appliquer un *filtre* sur une image, c'est-à-dire un calcul sur chacun des pixels de l'image.

Question 3.1 *Séparation des couleurs :*

L'instruction `(r,g,b) = Image.getpixel(img, (10,10))` *récupère les valeurs* r, g *et* b *de la couleur du pixel de coordonnées* $(10, 10)$. *On peut alors par exemple utiliser* `Image.putpixel(img, (10,10),(r,0,0))` *pour ne conserver que la composante rouge du pixel* $(10, 10)$.

Écrivez une fonction `filtreRouge(img)` *qui, pour chaque pixel de l'image, ne conserve que la composante rouge. Testez-la sur la photo de théière. Faites de même pour le vert et le bleu et affichez les trois résultats ainsi que l'image d'origine côte à côte.*

Question 3.2 *Écrivez une fonction* `luminosite(img,facteur)` *qui pour chaque pixel multiplie par* $facteur$ *les valeurs des trois composantes* r, g, *et* b. *Remarquez que la fonction* `Image.putpixel` *n'apprécie pas que l'on donne des valeurs non entières. Utilisez donc la fonction* `int(f)` *qui convertit une valeur avec virgule* `f` *en valeur entière.*

Testez les facteur `1.2`, `2`, `0.8`, `0.5` *sur la photo de théière (n'oubliez pas de recharger la photo à chaque fois pour éviter de cumuler les effets).*

Question 3.3 *Il paraît facile de convertir une photo couleur en photo en niveaux de gris. Écrivez une fonction* `monochrome(img)` *qui pour chaque pixel, calcule la moyenne* $l = \frac{r+g+b}{3}$ *des composantes* r, g, b, *et peint le pixel de la couleur* (l, l, l).

Les éléments verts semblent cependant plus foncés que sur la photo d'origine. L'œil humain est effectivement peu sensible au bleu et beaucoup plus au vert. Une conversion plus ressemblante est donc d'utiliser plutôt $l = 0.3 * r + 0.59 * g + 0.11 * b$. *Essayez, et constatez que les éléments verts ont une luminosité plus fidèle à la version couleur.*

Question 3.4 *Une manière simple de rendre une image floue est d'utiliser l'opération moyenne. Écrivez une fonction* `flou(img)` *qui pour chaque pixel n'étant pas sur les bords, le peint de la couleur moyenne des pixels voisins. Comparez le résultat à l'original. Pourquoi faudrait-il plutôt passer une deuxième image en paramètre à la fonction, et ne pas toucher à la première image ?*

4 Agrandissement d'une image

Question 4.1 *Agrandissement d'un facteur deux :*

Écrivez une fonction `agrandirFacteur2(img)` *qui retourne une image correspondant à l'agrandissement d'un facteur deux de l'image* `img`*. Pour réaliser cet agrandissement, chaque pixel de l'image source donnera un carré* 2×2 *pixels dans l'image destination.*

Testez cette fonction sur l'image `teapot.png`*. Appelez plusieurs fois la fonction pour produire un agrandissement d'un facteur 8 de cette même image. L'image apparaît quelque peu* pixelisée*, comment atténuer facilement ce phénomène ?*

5 Pour les plus rapides

5.1 Conversion d'une image en noir et blanc

Question 5.1 *Écrivez une fonction* `noirEtBlanc(img)` *qui convertit une image monochrome (telle que produite par la fonction* `monochrome` *de l'exercice 3.3) en une image noir et blanc : chaque pixel peut valoir (0,0,0) ou (255,255,255)) selon que la luminosité est plus petite ou plus grande que 127.*

Question 5.2 *Comme vous l'avez constaté, la qualité des images produites par la fonction précédente laisse à désirer. L'algorithme proposé par Floyd et Steinberg, permet de limiter la perte d'information due à la quantification des pixels en blanc ou noir. Écrivez une fonction* `floydSteinberg(img)` *qui convertit une image monochrome en noir et blanc à l'aide de l'algorithme de Floyd et Steinberg (cf. page Wikipedia :* http://fr.wikipedia.org/wiki/Algorithme_de_Floyd-Steinberg *).*

5.2 Cercle

Question 5.3 *En vous souvenant du schéma du cercle trigonométrique pour écrire l'équation paramétrique de la courbe du cercle, écrivez une fonction* `cercle(img,x,y,r)` *qui dessine un cercle de rayon* r *dont le centre a pour coordonnées* (x, y)*.*

Note : Les outils trigonométriques (`cos`, `sin`, `pi`, ...) sont disponibles en tapant :

```
from math import *
```

De plus, vous pouvez forcer le type d'une valeur x à être entière en utilisant `int(x)`

Question 5.4 *En pratique, ce n'est pas de cette manière que l'on dessine un cercle. Vous pouvez consulter cette page d'écrivant l'algorithme de Bresenham et essayer de reproduire l'algorithme.*