Algorithmique des tableaux

Compétences

- Manipuler des images avec le module PIL
- Concevoir des algorithmes simples pour modifier le contenu d'une image

Contexte : Les images avec Python

Notions de base

Les bouts de code suivants pourront être testés avec l'image `lion.jpeg` contenue dans le dossier `TD_machine/images` du github.

Lire une image

```
1 from PIL import Image
2
3 img = Image.open("path/mon_image.png")
4 print(img.format, img.size, img.mode)
5 img.show()
```

Fabriquer une image

```
1 # création d'une image de même taille
2 #que mon_image.png
3 img1 = Image.new("RGB", img.size)
4 img1.show()
```

Changer le contenu d'un pixel avec `putpixel`

```
1 YELLOW = (255, 255, 0)
2 img1 = Image.new("RGB", (300, 200))
3 j = 100
4 for i in range(img1.size[1]):
5     img1.putpixel((i, j), YELLOW)
6 img1.show()
```

Que remarquez-vous? Modifiez le code pour que la ligne jaune traverse la fenêtre.

Récupérer le contenu d'un pixel avec `getpixel`

```
1 pixel = img.getpixel((i, j))
2 print(pixel)
```

Pour s'exercer

1. créer une image "RGB" aux dimensions (300, 300)
2. Écrire une fonction `divide(image, n)` qui divise l'image en `n` bandes verticales de couleurs différentes.

Histoire de placard

Pour cet exercice, vous utiliserez l'image `placard.png`

1. On souhaite ouvrir la porte gauche du placard. Écrire une fonction
   `def translatePixelToLeft(img, xi, yi, xf, yf, value)`
   qui prend en paramètre une image, le cadre à décaler avec le pixel en haut à gauche (xi, yi) et le pixel en bas à droite (xf, yf) du cadre ainsi que la valeur de décalage. Pour la porte de gauche les coordonnées de ces deux pixels sont (71, 4) à (131, 186)) et le décalage est de 64 pixels. Vous pourrez remplir l'espace libéré avec de la couleur, ou avec l'image `fantome.png`.
2. Même question en entrouvrant la porte de droite (des pixels (135, 4) à (196, 186)) en la décalant de seulement 32 pixels vers la droite (attention il peut y avoir un piège!).

Flouter une image

Le filtre moyenneur est une opération de traitement d'images utilisée pour réduire le bruit dans une image et/ou flouter une image. Le principe est très simple : un pixel est remplacé par la moyenne de lui-même et de ses voisins. C'est dans la définition du voisinage que tout réside. Concrètement, avec un filtre moyenneur de largeur 3, pour calculer la nouvelle valeur du pixel rouge de l'image originale de gauche, on calcule la valeur moyenne (pixel traité compris) des pixels situés dans un carré de dimension 3x3 centré sur ce pixel. Cela donne la nouvelle valeur du pixel sur l'image transformée :

24	32	234	255	123
44	122	34	200	12
5	167	121	221	202
240	232	128	155	98
124	132	58	167	107

		153		

$$\frac{122+34+200+167+121+221+232+128+155}{9} = 153$$

1. Écrire une fonction `blur(img)` qui pour chaque pixel n'étant pas sur les bords, le peint de la couleur moyenne des pixels voisins.
2. Que pensez-vous de votre flou?
3. Proposer une idée d'amélioration et l'implémenter