Séance 4 :
La technique du fond vert

Objectif :

- Traiter par programme une image pour la transformer en agissant sur les trois composantes de ses pixels

Beaucoup de tournage sont réalisés devant un fond vert qui sera remplacé par un autre décor en post-production. En réalité, ce sont les pixels verts de ce fond qui vont être remplacés par les pixels d'une autre vidéo (ou d'une autre image).

Personne sur fond vert
fichier `homme.png`

Décor de fond souhaité
fichier `montagne.jpg`

Résultat : incrustation de la personne dans le décor

Vous allez programmer dans le langage Python ce détourage et l'incrustation dans le décor de fond.

Les deux images de départ ont la même taille : 600*400 pixels.

Germain BECKER & Sébastien POINT, Lycée Mounier, ANGERS

Partie A : analyse de la première image

Récupérez les deux images sur le site info-mounier.fr (onglet Seconde > Thème 2 > Séance 4) et sauvegardez-les dans un même dossier sur votre clé USB.

Lancez ensuite le logiciel Spyder (sur le bureau : Dossier Informatique > Dossier Anaconda > Spyder).

1. Créez puis enregistrez dans le **même dossier que les images** un fichier `incrustationA.py` qui contiendra les lignes suivantes :

```
1 from PIL import Image
2
3 img_homme = Image.open("homme.png")
4
5 p = img_homme.getpixel((0,0))
6 print(p)
```

Si vous obtenez une erreur '`no such file or directory`', c'est que Python n'est pas parvenu à trouver le fichier `homme.png`. Ce fichier doit se situer dans le même dossier que votre fichier `incrustationA.py`.

Explications :

- Ligne 1 : on importe le module `Image` de la bibliothèque PIL qui va permettre la manipulation d'images.
- Ligne 3 : on ouvre l'image `homme.png`. Dans toute la suite du programme, elle sera désignée par le nom `img_homme`.
- Ligne 5 : on récupère et stocke dans une variable `p` la valeur du pixel en haut à gauche (le pixel de coordonnées (0,0)).
- Ligne 6 : on affiche la valeur de `p`.

2. Exécutez ce programme (icône ) et observez la valeur du pixel affichée. Ce résultat vous paraît-il correct ?

3. Modifiez votre code afin d'observer la valeur du pixel en bas à droite de l'image, puis essayez de trouver les coordonnées d'un pixel qui ne soit pas vert.

4. Complétez le code ci-dessous afin qu'il affiche le message `"le pixel est vert"` si `p` est un pixel vert et `"le pixel n'est pas vert"` sinon.

```
if p == (0, 255, 0):
    print("..............................")
else:
    print("..............................")
```

5. Rajoutez ces 4 lignes à votre code et testez-le avec différents pixels verts et non verts.

Germain BECKER & Sébastien POINT, Lycée Mounier, ANGERS

Partie B : parcours de tous les pixels de l'image

On rappelle que l'image `homme.png` a pour taille 600*400 pixels.

6. Le programme suivant a pour but :

- de parcourir tous les pixels de l'image `homme.png` à l'aide d'une double boucle `for` (voir Séance 2 si besoin) ;
- de récupérer et stocker dans la variable p la valeur de chaque pixel de coordonnées `(x,y)` rencontré ;
- d'afficher si chaque pixel rencontré est vert ou non.

Complétez les pointillés afin que le programme effectue ce qui est indiqué.

```
1 from PIL import Image
2
3 img_homme = Image.open("homme.png")
4
5 for y in range(...):
6     for x in range(...):
7         p = ...
8         ...
9         ...
10        ...
11        ...
```

7. Recopiez puis enregistrez ce fichier en le nommant `incrustationB.py`.

Exécutez ensuite le code (icône ) et observez les affichages dans la console. Vous pouvez interrompre l'exécution de votre code si celle-ci prend trop de temps.

8. On aimerait maintenant remplacer tous les pixels verts par des pixels bleus. Compléter avec deux phrases l'algorithme suivant qui permet d'effectuer cette transformation :

```
On parcourt tous les pixels de l'image
On récupère la valeur de chaque pixel
Si ....................................
    alors ..................................................
```

Rappel : l'instruction `img_homme.putpixel((x, y), (255, 0, 0))` permet de colorier le pixel de coordonnées (x,y) en rouge.

9. Modifiez votre code précédent afin que tous les pixels verts deviennent bleus. Pensez à enlever le `print("le pixel est vert")` et le `print("le pixel n'est vert")` de votre code car il le ralentit énormément.
Pour vérifier si votre code fonctionne, sauvegardez l'image transformée en rajoutant la ligne `img_homme.save("homme_bleu.png")` en fin de code (après la double boucle) et ouvrez le fichier ainsi créé dans votre dossier de travail.

Germain BECKER & Sébastien POINT, Lycée Mounier, ANGERS

Partie C : incrustation dans le décor

10. En vous inspirant des parties précédentes, créez un programme appelé `incrustationC.py` qui remplace chaque pixel vert de l'image `homme.png` par le pixel situé au même endroit dans l'image `montagne.jpg` afin d'obtenir l'image suivante.

N'oubliez pas de sauvegarder l'image sous un autre nom que celui d'origine et de vérifier que le programme fonctionne en affichant l'image.

Pour aller plus loin

Comment pouvez-vous expliquer que le détourage ne soit pas parfait ?
Il est possible de donner une plus grande tolérance au test `if p == (0, 255, 0):` en testant séparément chacune des composantes :

```
if p[0] < 100 and p[1] > 150 and p[2] < 100:
```

Cela donne un meilleur résultat même s'il n'est pas encore parfait.
En réalité, des techniques supplémentaires de détections de contours peuvent être utilisées par les logiciels pour gommer ses effets indésirables.

Libre à vous de tester l'algorithme du site remove.bg qui donne des résultats impressionnants pour détourer des personnes, des objets ou des voitures.

Source : Gilles Lassus, La photographie numérique, bit.ly/SNT_BDX