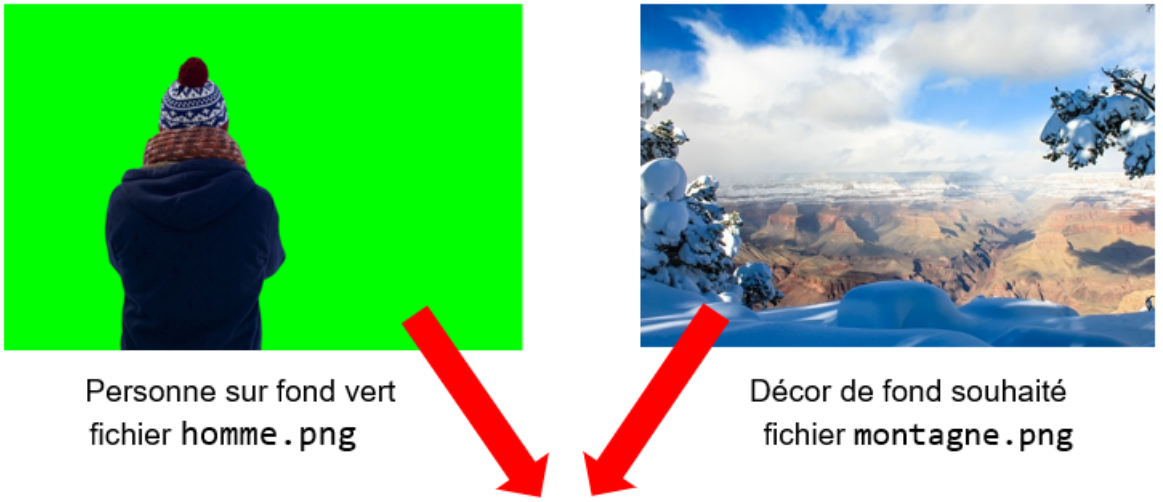
La technique du fond vert

Introduction ¶

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).

Dans cette activité, vous allez travailler avec deux images `homme.png` et `montagne.png` avec pour objectif de détourer le personnage et l'incruster dans le décor de fond :



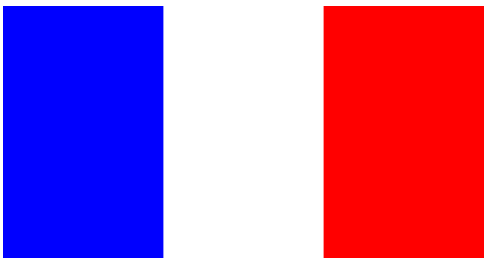
Résultat : incrustation de la personne dans le décor

Travail préliminaire

Avant de programmer l'incrustation, nous allons commencer par un travail préliminaire permettant de réactiver des connnaissances.

Création du drapeau de la France

On souhaite dans ce paragraphe créer une image de définition 300*200 pixels correspondant au drapeau de la France.



```
Les pixels blancs ont une abscisse qui varie entre ... et ... ; et une ordonnée qui varie entre
... et ...
Les pixels rouges ont une abscisse qui varie entre ... et ... ; et une ordonnée qui varie entre
... et ...
```

Pour créer cette image avec Python, on rappelle que l'on peut utiliser la bibliothèque `PIL` .

Question 2 : Dans le programme ci-dessous, on a créé une image, appelée `img_drapeau` , aux bonnes dimensions (ligne 4). Par défaut, tous les pixels sont noirs. Il faut donc colorier les différentes parties en bleu, blanc et rouge. Terminez l'écriture du programme en complétant les parties pour colorier les pixels blancs et rouges.

Entrée[ ]:
```
from PIL import Image

# Création de l'image aux bonnes dimensions (noire par défaut)
img_drapeau = Image.new("RGB", (300, 200))

# Coloriage des pixels bleus
for x in range(0, 100):
    for y in range(0, 200):
        img_drapeau.putpixel((x, y), (0, 0, 255))

# Coloriage des pixels blancs
for x in range(..., ...):
    for y in range(..., ...):
        img_drapeau.putpixel((x, y), (..., ..., ...))

# Coloriage des pixels rouges
...


img_drapeau.show()  # pour afficher l'image
```

> Même si cette partie n'a pas été bien réussie, vous pouvez poursuivre le notebook car on utilise une image du drapeau français existante.

Importer une image existante

Plusieurs images ont été "attachées" à ce notebook, en particulier une image appelée `drapeau_france.png` .

On peut ouvrir avec Python une image existante en utilisant la fonction `open` à laquelle on précise juste le nom du fichier en question. Par exemple, on peut ouvrir l'image `drapeau_france.png` avec la ligne 3 du programme suivant :

Entrée[ ]:
```
from PIL import Image

img_drapeau = Image.open("drapeau_france.png")  # on ouvre l'image du drapeau français

img_drapeau.show()  # pour afficher l'image
```

> Pour Python, cette image a été importée dans la variable `img_drapeau` (ligne 3) et on peut utiliser cette variable pour manipuler l'image en question (ou l'afficher, comme à la ligne 5).

Récupérer les composantes RVB d'un pixel avec la fonction `getpixel`

On peut utiliser la fonction `getpixel` (littéralement "*obtenir le pixel*") pour récupérer la valeur d'un pixel, c'est-à-dire ses composantes RVB. Cette fonction s'utilise comme ci-dessous :

Entrée[ ]:
```
from PIL import Image

img_drapeau = Image.open("drapeau_france.png")  # on ouvre l'image du drapeau français
pix_bleu = img_drapeau.getpixel((50, 50))  # pix_bleu contient les composantes (R,V,B) du pixel de

print(pix_bleu)  # pour afficher la variable pix_bleu
```

Explications :

- on applique la fonction `getpixel` à notre image `img_drapeau` avec l'instruction `img_drapeau.getpixel()`
- il suffit ensuite de lui donner le couple de coordonnées du pixel souhaité (ici `(50, 50)`).
- on a stocké les valeurs de ce pixel dans la variable `pix_bleu` , que l'on affiche pour terminer avec la fonction `print`

✍ **Question 3** : En observant correctement l'image du drapeau français, donnez les coordonnées d'un pixel blanc et d'un pixel rouge de l'image.

```
Votre réponse (à compléter) :
- on peut trouver un pixel blanc aux coordonnées : ...
- on peut trouver un pixel rouge aux coordonnées : ...
```

🐍 **Question 4** : Complétez le programme suivant pour récupérer dans deux variables `pix_blanc` et `pix_rouge` des pixels respectivement blanc et rouge du drapeau français. Affichez aussi la valeur de ces deux variables pour vérifier !

Voir l'aide

Entrée[ ]:
```
from PIL import Image

img_drapeau = Image.open("drapeau_france.png")

pix_bleu = img_drapeau.getpixel((50, 50))  # un pixel bleu
pix_blanc = ...  # un pixel blanc
...  # un pixel rouge

# affichages :
print(pix_bleu)
...
```

Fonctions pour tester la couleur d'un pixel

Pour remplacer les pixels verts du fond de notre image de départ, il faut être capable de déterminer si un pixel est vert (ou s'il ne l'est pas). On va écrire dans ce paragraphe une fonction qui va effectuer ce travail.

> Pour vous aider à écrire cette fonction, on va commencer par écrire une fonction presque identique, qui détermine si un pixel est rouge.

Tester si un pixel est rouge

On veut écrire une fonction `est_rouge` qui prend en paramètre un `pixel` et qui renvoie Vrai si le pixel est rouge, et Faux sinon.

🧩 **Question 5** : Exécutez la cellule suivante puis remettez les instructions dans l'ordre pour que la fonction `est_rouge` soit correcte.

Entrée[ ]:
```
from IPython.display import IFrame
IFrame('https://www.codepuzzle.io/iframe/937C', width='100%', height=600)
```

🐍 **Question 6** : Recopier dans la cellule ci-dessous le code de la fonction `est_rouge` puis exécutez les 3 cellules qui suivent pour tester la fonction sur les pixels `pix_bleu` , `pix_blanc` et `pix_rouge` définis à la question 4.

Entrée[ ]:
```
# copier le code de la fonction est_rouge en-dessous :

```

Entrée[ ]:
```
# cellule à exécuter
est_rouge(pix_bleu)
```

Entrée[ ]:
```
# cellule à exécuter
est_rouge(pix_blanc)
```

Entrée[ ]:
```
# cellule à exécuter
est_rouge(pix_rouge)
```

Tester si un pixel est vert

🐍 **Question 7** : Inspirez-vous de la question précédente pour écrire une fonction `est_vert` qui prend en paramètre un pixel `pixel` et qui renvoie `True` si le pixel est vert, et `False` sinon. *N'hésitez pas à faire un copier-coller et modifier ce qui doit l'être.*

Entrée[ ]:
```
# à vous de jouer !

```

Importation de l'image sur fond vert

On peut ouvrir l'image de l'homme sur fond vert (fichier `homme.png` associé au notebook) avec la fonction `open` avec le code ci-dessous. Pour Python cette image s'appellera `img_homme` :

Entrée[ ]:
```
from PIL import Image

img_homme = Image.open("homme.png")

img_homme.show()  # pour vérifier que l'image est bien importée dans Python
```

Remarque importante pour la suite : Cette image est de dimension 600*400 pixels comme on peut le voir en exécutant l'instruction suivante :

Entrée[ ]:
```
img_homme.size
```

🐍 **Question 8** : Utilisez la fonction `getpixel` (voir les questions 3 et 4) pour récupérer :

- dans une variable `pix_vert` un pixel vert de l'image `img_homme`

- dans une variable `pix_pas_vert` un pixel de l'image `img_homme` qui n'est pas vert

Vous vérifierez en affichant les valeurs de ces deux pixels.

Entrée[ ]:
```
from PIL import Image

img_homme = Image.open("homme.png")

# à vous de jouer !
```

Question 9 : Appelez votre fonction `est_vert` sur ces deux pixels `pix_vert` et `pix_pas_vert` pour vérifier qu'elle renvoie bien ce qui est attendu (voir question 6 si nécessaire)

Entrée[ ]:
```
# à vous de jouer !
```

Entrée[ ]:
```
# à vous de jouer !
```

Remplacer le fond vert

Remplacement par un fond bleu

Avant d'écrire le programme permettant d'incruster le personnage dans le décor, on va écrire un autre programme pour vous aider à comprendre la logique.

Question 10 : On veut remplacer tous les pixels verts de l'image `img_homme` par des pixels bleus. Pour cela, il est nécessaire de parcourir tous les pixels `(x, y)` de l'image `img_homme` et si on tombe sur un pixel vert on le remplace par un pixel bleu. Pour tester si un pixel est vert on pourra utiliser la fonction `est_vert` définie à la question 7. Compléter le programme suivant en conséquence :

Entrée[ ]:
```
from PIL import Image

img_homme = Image.open("homme.png")

for x in range(..., ...):
    for y in range(..., ...):
        pixel = img_homme.getpixel((x,y))  # pixel est le pixel de coordonnées (x, y)
        if ...:  # si pixel est vert
            img_homme.putpixel((x, y), (..., ..., ...))  # on colore pixel en bleu

img_homme.show()
```

Remplacement par les pixels du décor de fond

Pour incruster le personnage dans le décor montagneux, on procède comme dans le programme précédent, mais cette fois il faut remplacer chaque pixel vert par le pixel situé *au même endroit* dans l'image `montagne.png` .

Question 11 : Complétez le programme suivant pour incruster le personnage dans le décor :

Voir l'aide

Entrée[ ]:
```
from PIL import Image

img_homme = Image.open("homme.png")  # on ouvre l'image du personnage
img_montagne = Image.open("montagne.png")  # et aussi l'image du décor

# à compléter ici



img_homme.show()
img_homme.save("homme_incrustation.png")  # pour sauvegarder le résultat dans un nouveau fichier
```

Amélioration

Le programme de la question 11 devrait donner l'image suivante :

Vous pouvez constater que le détourage n'est pas parfait, tout simplement car les pixels verts restant visibles ne sont pas du vert "pur" (de composantes (0, 255, 0)).

Il est possible de donner une plus grande tolérance à notre fonction `est_vert`. En effet, on peut considérer qu'un pixel est vert :

- si sa composante rouge est strictement inférieure à 100
- et que sa composante verte est strictement supérieure à 150
- et que sa composante bleue est strictement inférieure à 100

Pour récupérer les composantes rouge, verte et bleu d'un pixel on peut procéder ainsi :

Entrée[ ]:
```
pixel = (50, 198, 81)
r = pixel[0]  # composante rouge
v = pixel[1]  # composante verte
b = pixel[2]  # composante bleue
```

On vient de stocker dans les variables `r`, `v` et `b` les composantes respectives du pixel `(50, 198, 81)`, comme on peut le voir en évaluant les valeurs de ces trois variables :

Entrée[ ]:
```
r
```

Entrée[ ]:
```
v
```

Entrée[ ]:
```
b
```

Question 12 : Modifiez la fonction `est_vert` (de la question 7) pour tenir compte de cette plus grande tolérance, et exécutez à nouveau le code de la question 11 pour constater que le détourage est meilleur !

Vous devez désormais obtenir une image similaire à celle ci-dessous :

Entrée[ ]:
```
# à vous de jouer !
```

📣 En réalité, des techniques supplémentaires de détections de contours peuvent être utilisées par les logiciels pour gommer ces effets indésirables. Par exemple, le site remove.bg (https://www.remove.bg/fr) utilise des algorithmes d'intelligence artificielle pour détourer des personnes et donne des résultats impressionnants !

Mémento sur la bibliothèque PIL

- Créer une image couleur de dimension L*H :

  ```
  nom_image = Image.new("RGB", (L, H))
  ```

- Ouvrir une image existante à partir de son nom de fichier :

  ```
  nom_image = Image.open("nom_du_fichier.png")  # remplacer png par le bon format s'il est d
  ifférent
  ```

- Afficher une image :

  ```
  nom_image.show()
  ```

- Récupérer dans une variable `pixel` la valeur (= les composantes (R, V, B)) du pixel de coordonnées `(x, y)` d'une image :

  ```
  pixel = nom_image.getpixel((x, y))
  ```

- Colorier le pixel de coordonnées `(x, y)` d'une image avec la couleur `(r, v, b)` :

  ```
  nom_image.putpixel((x, y), (r, v, b))
  ```

- Connaître la définition d'une image :

  ```
  nom_image.size
  ```

- Sauvegarder une image créée/modifiée avec Python :

  ```
  nom_image.save("nom_du_fichier_a_sauvegarder.png")  # remplacer png par le format souhaité
  ```

Référence : D'après une idée proposée par Gilles Lassus : bit.ly/SNT_BDX (https://bit.ly/SNT_BDX)

Germain Becker, Lycée Emmanuel Mounier, Angers.