Objectif : mettre en œuvre les outils de base de modification des images `.png`.

1 Vérifications préliminaires

Avant de travailler sur un fichier image, il faut :

- Importer les images ;
- Afficher les images ;
- Vérifier les caractéristiques des images
  - Format ;
  - Taille ;
  - Codage des couleurs ;
  - Modification du codage couleurs (facultatif).

1.1 Importer les images

Une image est un type de fichier particulier.

Pour importer les images, il faut importer la bibliothèque **`matplotlib.image as mping`**.

Il faut ensuite utiliser la commande **`imageOrigine=mping.imread('chemin_d_acces_a_l_image.png')`**.

1.2 Afficher les images

Pour afficher les images, il faut importer la bibliothèque **`import matplotlib.pyplot as plt`**.

Il faut ensuite utiliser la commande

`plt.imshow(imageOrigine)`

`plt.show()`

1.3 Vérifier les caractéristiques de l'image

1.3.1 Vérification du format de l'image

Le format de l'image est un tableau numpy. Il peut parfois être utile de le vérifier.

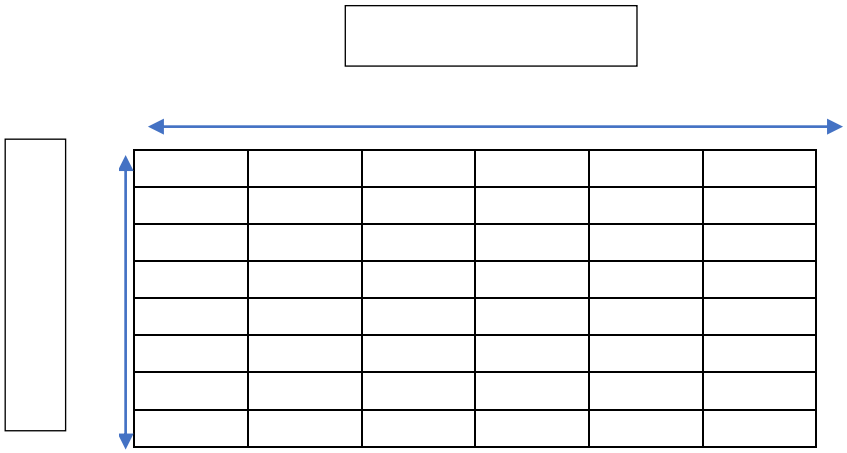
Pour vérifier le format de l'image on utilise la commande

`print("l'image est au format", type(imageOrigine)).`

1.3.2 Vérification de la taille de l'image

Pour vérifier la taille de l'image on utilise la commande

```
print("la hauteur de l'image vaut", imageOrigine.shape[0])
print("la largeur de l'image vaut", imageOrigine.shape[1])
print("la taille de l'image vaut", imageOrigine.shape).
```



1.3.3 Vérification du codage des couleurs

Les 3 couleurs RVB sont en général codées :

- Par un entier sur un octet (0 à 255) ;
- Ou par un flottant (0 à 1).

Si r = v = b = 0, le pixel est noir. S'il vaut sa valeur maximale, le pixel est blanc.

Avant de travailler sur une image, il faut vérifier si les couleurs sont codées par un entier sur un octet ou par un flottant de 0 à 1. Pour le savoir, il suffit de regarder la valeur d'un octet au hasard :

```
print("le pixel situé à la coordonnée (50,50) vaut", imageOrigine[50,50])
```

1.3.4 Modification du codage des couleurs (facultatif)

Les couleurs sont le plus souvent codées sur un octet (0 à 255). Pour travailler sur plusieurs images en même temps, il faut que le codage des couleurs soit le même. Si les couleurs d'une image sont codées sur un flottant de 0 à 1, il est possible de les convertir en octet par cette commande :

```
#CONVERSION
imageConvertie=imageOrigine             #on garde l'image d'origine en l'état
if imageConvertie.dtype==np.float32:
    imageConvertie=(imageConvertie*255).astype(np.uint8)
#SAUVEGARDE DE L'IMAGE
mping.imsave('but_converti.png',imageConvertie)
#VERIFICATION DE LA CONVERSION
print("après conversion, le pixel situé à la coordonnée (50,50) vaut",
imageConvertie[50,50])
```

2 Modifications d'images les plus courantes

Les modifications d'images les plus courantes sont :

- La mise en place d'un filtre coloré sur l'image ;
- La création d'un effet négatif ;
- La conversion de l'image en niveau de gris ;
- La conversion de l'image en noir et blanc.

2.1 Mise en place d'un filtre rouge

On met en place un filtre rouge en mettant à 0 les couleurs verte et bleue sur tous les pixels.

- Créer une fonction `filtreRouge` qui prend en argument le chemin de l'image sur laquelle on travaille et qui renvoie une image passée au filtre rouge.
- Afficher cette image.
- Enregistrer cette image sous le nom « filtre_rouge.png » et le programme sous le nom filtre_rouge.py.

2.2 Créer un effet négatif

Un effet négatif est créé en faisant le complément de chaque pixel (255 – valeur couleur pixel ou 1 – valeur couleur pixel).

- Créer une fonction `effetNegatif` qui prend en argument le chemin de l'image sur laquelle on travaille et qui renvoie une image passée au négatif.
- Afficher cette image.
- Enregistrer cette image sous le nom « negatif.png » et le programme sous le nom negatif.py.

2.3 Convertir l'image en niveau de gris

Une des manières de passer une image en niveau de gris est d'attribuer aux 3 valeurs de couleur la moyenne des 3.

- Créer une fonction `niveauGris` qui prend en argument le chemin de l'image sur laquelle on travaille et qui renvoie une image en niveau de gris.
- Afficher cette image.
- Enregistrer cette image sous le nom « niveau_gris.png » et le programme sous le nom niveau_gris.py.

2.4 Convertir une image en noir et blanc

Pour convertir une image en noir et blanc, il faut au préalable la passer en niveau de gris. Il faut ensuite fixer un seuil. Toutes les valeurs au-dessus de ce seuil vaudront 1, toutes les valeurs en-deçà vaudront 0.

Remarque : pour sauvegarder les images, préférer **`mping.imsave('niveau_gris.png',imageGrisee)`** à **`plt.savefig('niveau_gris.png')`**. Cette dernière amoindrit la qualité des images est plus adaptée à la sauvegarde de graphiques.

3 Pour s'amuser :

- Créer un effet miroir
- Inverser l'image (droite à la place de la gauche)
- Doubler la taille de l'image
- Diviser par 2 la taille de l'image
- Construire un mur dans les buts
- Faire un ballon carré
- Dessiner un pied carré à l'un des joueurs