1 Le contenu du dossier projet

Vous allez conduire votre projet par groupes de 3.

Vous devez réaliser une application informatique à l'aide de scripts écrits en Python sur le thème du module « *Traitement, transformation, création d'images* ».

Les fichiers sources des codes Python pourront être demandés par le professeur pendant la phase de développement. Ils devront respecter les conditions suivantes :

- être bien documentés
- présenter un découpage du code en fonctions ou modules facilitant sa lisibilité et sa maintenance

Votre projet doit être rendu sous la forme d'un dossier projet qui prendra la forme d'un diaporama réalisé avec Impress ou Powerpoint, qui comportera cinq parties courtes (pas plus de deux diapositives à l'exception de la partie 3) :

☞ Partie 1 : **motivation générale** du projet et inscription dans le parcours.

☞ Partie 2 : le **cahier des charges** c'est-à-dire la liste des objectifs avec une description précise, qui doivent être atteints au terme du projet.

☞ Partie 3 : la **documentation** du projet avec l'explication de sa structure et la présentation de chaque fonction : paramètres, algorithme, valeur de retour.

☞ Partie 4 : **questionnement** d'un enjeu sociétal, juridique ou éthique connexe au projet.

☞ Partie 5 : **conclusion** avec ouverture.

☞ Annexe : **codes sources** du projet.

2 Propositions de sujets

- **Sujet 1** : Tracés de fractales avec un *L-système* et le module `turtle` : flocon de Von Koch, courbe de Peano, triangle de Sierpinski, fougère de Barnsley, arbre de Pythagore ...
- **Sujet 2** : Le bruit d'une image est constitué de pixels altérés répartis aléatoirement dans une image. Écrire une fonction `bruitage(image, p)` qui retourne une image obtenue par bruitage d'une proportion `p` de pixels dans l'image source. Écrire une fonction `filtre_moyen(image, n)` qui réduit le bruit d'une image en remplaçant chaque pixel qui peut se trouver au centre d'un carré de n^2 pixels voisins, par la moyenne de ses n^2 voisins (lui compris). Écrire une fonction `filtre_mediane(image, n)` qui réduit le bruit d'une image en remplaçant chaque pixel qui peut se trouver au centre d'un carré de n^2 pixels voisins, par la médiane de ses n^2 voisins (lui compris).
- **Sujet 3** : Faire une recherche sur le filtrage d'une image par masque de convolution (voir https://docs.gimp.org/fr/plug-in-convmatrix.html) ou demander une explication au professeur. Écrire une fonction `filtre_masque(image, masque)` qui retourne l'image obtenue par application d'un masque de convolution aux pixels de l'image source passée en paramètre.

  Tester cette fonction avec des masques permettant la détection de contour comme :

  – les masques de Kirsch, vertical $\begin{pmatrix} \frac{-3}{15} & \frac{-3}{15} & \frac{-3}{15} \\ \frac{-3}{15} & 0 & \frac{-3}{15} \\ \frac{5}{15} & \frac{5}{15} & \frac{5}{15} \end{pmatrix}$ ou horizontal $\begin{pmatrix} \frac{-3}{15} & \frac{-3}{15} & \frac{5}{15} \\ \frac{-3}{15} & 0 & \frac{-3}{15} \\ \frac{-3}{15} & \frac{-3}{15} & \frac{5}{15} \end{pmatrix}$;

  – les masques de Sobel, vertical $\begin{pmatrix} \frac{-1}{4} & \frac{-2}{4} & \frac{-1}{4} \\ 0 & 0 & 0 \\ \frac{1}{4} & \frac{2}{4} & \frac{1}{4} \end{pmatrix}$ ou horizontal $\begin{pmatrix} \frac{-1}{4} & 0 & \frac{1}{4} \\ \frac{-2}{4} & 0 & \frac{2}{4} \\ \frac{-1}{4} & 0 & \frac{1}{4} \end{pmatrix}$.
- **Sujet 4** : Écrire un programme permettant de cacher puis de dévoiler un texte dans une image, on pourra ainsi illustrer la théorie des correspondances de Baudelaire en dissimulant un poème de Baudelaire dans l'image d'un tableau de Delacroix :-)

Comme de longs échos qui de loin se confondent
Dans une ténébreuse et profonde unité,
Vaste comme la nuit et comme la clarté,
Les parfums, les couleurs et les sons se répondent

Charles Baudelaire extrait du sonnet Correspondances dans *les Fleurs du mal*

Les caractères comme les pixels sont codés par des entiers représentés en écriture binaire dans l'ordinateur. Ce codage universel (pour assurer l'interopérabilité des fichiers textes entre les différentes machines) est un standard appelé Unicode qui étend à toutes les langues le code ASCII développé par les Américains dans les années 60.

La fonction `Python ord` s'applique à un caractère et retourne son codage Unicode, sa réciproque est la fonction `chr` qui s'applique à un entier et retourne le caractère associé.

```
In [7]: ord('a'), ord('z')
Out[7]: (97, 122)

In [8]: chr(97), chr(122)
Out[8]: ('a', 'z')

In [9]: ord('A'), ord('Z')
Out[9]: (65, 90)

In [10]: chr(65), chr(90)
Out[10]: ('A', 'Z')

In [11]: ''.join([chr(k) for k in range(97, 123)])
Out[11]: 'abcdefghijklmnopqrstuvwxyz'

In [12]: ''.join([chr(k) for k in range(65, 91)])
Out[12]: 'ABCDEFGHIJKLMNOPQRSTUVWXYZ'
```

- **Sujet 5** : Écrire une fonction `flou_mosaique(im, cbloc)` qui découpe une image en blocs carrés de côté `cbloc` et qui retourne une image où dans chaque bloc, les pixels ont été mélangés aléatoirement. Écrire une fonction `melange_mosaique(im, cbloc)` qui découpe une image en blocs carrés de côté `cbloc` et qui retourne une image où les blocs ont été mélangés aléatoirement.

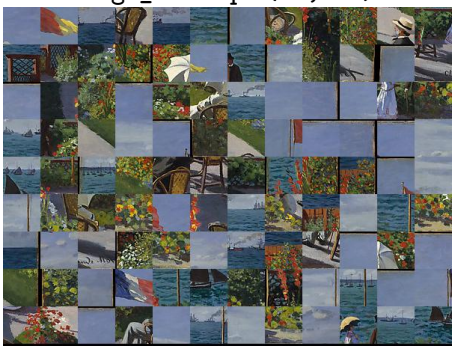
  La fonction `shuffle` du module `random` permet de mélanger une liste.

```
In [34]: from random import shuffle
In [35]: L = [(255,0,0), (0,255,0), (0, 0,255)]
In [36]: shuffle(L)
In [37]: L
Out[37]: [(0, 0, 255), (255, 0, 0), (0, 255, 0)]
```

On donne ci-dessous un exemple d'application sur l'image [1] du tableau « *Jardin à Sainte-Adresse* » peint par *Claude Monet*.

Original `im` | `flou_mosaique(im, 4)` | `melange_mosaique(im, 50)`





- **Sujet 6** : Sujet libre sur la création ou le traitement d'image.

1. Licence CC0 1.0 téléchargeable sur https://ccsearch.creativecommons.org