Programme principal

Il s'agit maintenant d'articuler toutes les «briques» de notre programme.

Étapes-clés

1. Chaque groupe doit disposer:
   - d'une image de base (disponibles sur Moodle);
   - d'un enregistrement sonore (disponibles sur Moodle);
   - d'une vidéo et de l'audio extrait de cette vidéo (disponibles sur https://methotapes.com/BILANS_CLIMATIQUES/uploads/video/);
2. On commencera par créer 4 variables contenant **les noms de ces fichiers**.
3. Le protocole de création du GIF comporte 25 itérations du même procédé:
   - extraire une image de la vidéo, puis sa couleur dominante;
   - extraire les données du son de l'enregistrement et du son de la vidéo (qui en fait se font au préalable);
   - extraire la zone de l'image de base;
   - lui appliquer le filtre
   - enregistrer l'image
4. Une fois les 25 images créées, lancer la création du GIF:
   - redimensionner les 25 images à la bonne taille 500x500;
   - créer le GIF !

Redimensionnement des images

Le logiciel (Open-source) ImageMagick permet de faire à peu près tout ce qu'on veut comme retouche d'images... en ligne de commande dans un terminal.

Par exemple, pour redimensionner une image `monimage.png` en créant une **nouvelle** image:

Bash

```
convert -resize 500x500 monimage.png monimageredimensionnee.png
```

Ou bien en **écrasant** l'image:

Bash

```
mogrify -resize 500x500 monimage.png
```

Shell et Python

En Python, on peut exécuter une ligne de commande du shell depuis un programme. Pour cela il faut utiliser le module `os` et la fonction `system`:

Script Python

```
1 import os
2 os.system('la commande en chaine de caractères')
```

On s'en servira pour:

- extraire l'audio de la vidéo avec le logiciel `ffmpeg`
- redimensionner avec `ImageMagick` les images générées en 500x500 pixels pour ensuite les ajouter au GIF.

Module code élèves

Voici le fichier/module composé de vos fonctions qu'il faudra importer dans le programme principal.

moduleprojet.py

```python
import scipy.io.wavfile as wave
import math
import numpy
import imageio

### 1. Traitement du son ###########################################################

def spectre(data: list, rate: int, debut: float, duree: float) -> list:
    '''
    Renvoie le spectre correspondant à un intervalle du signal.

    data: le signal d'un canal
    rate: la fréquence d'échantillonnage
    debut: le début de l'intervalle à étudier (en secondes)
    duree: la durée de l'intervalle à étudier (en secondes)
    '''
    start = int(debut * rate)
    stop = int((debut+duree) * rate)
    s = numpy.absolute(numpy.fft.fft(data[start:stop]))
    s = s / s.max()
    return [math.log10(i) for i in s if i != 0]


def volumes_min(son)->list: #Mélodie
    '''
    Prend en paramètre un son qui correspond
    et renvoie une liste de 25 valeurs qui
    correspondent aux volume minimaux de
    chaque 1/25eme de la durée totale du son.
    '''
    rate, echantillon = wave.read(son)
    cd = [elt[1] for elt in echantillon]
    duree = len(echantillon)/rate
    duree_intervalle = duree//25
    depart = 0.0
    volumes_mini = []
    for k in range (25):
        s = spectre(cd, rate, depart, duree_intervalle)
        volumes_mini.append(min(s))
        depart += duree_intervalle
    return volumes_mini

def pourcentage (lst:list)-> list: #Mélodie
    '''
    Prends en paramètre la liste des volumes minimaux
    et renvoie la liste contenant les pourcentages de
    chaque point dans l'intervalle minimal-maximal de
    la liste de volumes minimaux.
    '''
    lst_pourcentage = []
    maxi = max(lst)
    mini = min(lst)
    for elt in lst :
        p = ((elt-mini)/(maxi-mini))*100
        lst_pourcentage.append(int(p))
    return lst_pourcentage

```

```python

### 2. Traitement de l'image ######################################################

# Extraction de la zone

def extraction_zone(img: list, S: int, Vson: float, Vvideo: float) -> list: #Manon
    '''
    Renvoie une image constituée des pixels de l'image img, de taille W avec le décalage x, y
    donnés par les paramètres S, Vson et Vvideo.
    '''
    W = int(500 * S/100)
    x = int((500-W) * Vson)
    y = int((500-W) * Vvideo)
    #On créait une image vide pour pouvoir y placer la zone extraite
    zone = numpy.zeros((W, W, img.shape[2]), dtype=numpy.uint8)
    for i in range(W):
        for j in range(W):
            zone[i][j] = img[i+y][j+x]
    return zone

def extraction(image_AP: list, S: int, Vson: float, Vvideo: float) -> list: #Jules

    '''
    Créer une image vide, sélectionne une zone dans l'image
    en paramètre et copie les pixels de la zone dans l'image vide.
    '''

    W = int(500 * S/100)
    x = int((500-W) * Vson)
    y = int((500-W) * Vvideo)

    zone = numpy.zeros([W,W,3], dtype=numpy.uint8)

    for i in range(W):
        for j in range(W):
            zone[i][j] = image_AP[y+i][x+j]

    imageio.imsave("monimage.jpg", zone)
    return zone

# Recherche de la couleur dominante

def color_dom(image): #Yanis

    # Cherche le pixel de l'image img qui revient le plus grand nombre de fois dans img.
    hauteur = image.shape[0]
    largeur = image.shape[1]
    pixels = {}
    for i in range(hauteur):
        for j in range(largeur):
            if tuple(image[i][j]) in pixels:
                pixels[tuple(image[i][j])] += 1
            else:
                pixels[tuple(image[i][j])]  = 1

    # recherche du pixel ayant la plus grande occurence
    grand = ''
    nmax = 0
    for k, v in pixels.items():
        if v > nmax:
            grand = k
            nmax = v
    return grand

def couleur_dominante(image) -> tuple: # Manon
```

```
    # On parcous l'image pour savoir combien de fois les couleurs apparaîssent dans le tableu.
    dico_RGB = {}
    for i in range(img_test.shape[0]):
        for j in range(img_test.shape[1]):
    # On utilise un tuple pour qu'il soit possible de parcourir l'image.
            t = tuple(img_test[i][j])
            if t in dico_RGB:
                dico_RGB[t] += 1
            else:
                dico_RGB[t] = 1

    # On recherche la couleur qui revient le plus.
    clr_max_apparition = 0
    for clr, n in dico_RGB.items():
        if n > clr_max_apparition:
            clr_max_apparition = n
            couleur = clr

    return couleur

# Application du filtre

'''
la fonction f permet de superposer une teinte sur une couche
'''
def f(a,b): #Florette
    a=a/255
    b=b/255
    if a<0.5 :
        r= 2*a*b
    else:
        r= 1-2*(1-a)*(1-b)
    return int(r*255)

    '''
    Renvoie une nouvelle image, donnée par l'application d'une filtre coloré à l'image de départ
img.
    '''
def filtre(img : list, couleur: tuple): #Florette
    img_filtree = numpy.zeros((img.shape[0],img.shape[1],3), dtype=numpy.uint8)
    for i in range(img.shape[0]):
        for j in range(img.shape[1]):
            r=img[i][j][0]
            g=img[i][j][1]
            b=img[i][j][2]
            img_filtree[i][j] = (f(r,couleur[0]), f(g,couleur[1]), f(b,couleur[2]))
    imageio.imsave("imageavecfiltre.jpg", img_filtree)
    return img_filtree

def f(a, b): #Dawson
    a = a / 255
    b = b / 255
    if   a < 0.5:
        r = 2 * a * b
    else:
        r = 1-2*(1-a)*(1-b)
    return int(r*255)


def filtre(img: list, couleur: tuple): #Dawson
    '''
    Renvoie une nouvelle image, donnée par l'application d'une filtre coloré à l'image de départ
img.
    '''
    img_filtree = numpy.zeros((img.shape[0], img.shape[1],3), dtype=numpy.uint8)
```

```
    for i in range(img.shape[0]):
        for j in range(img.shape[1]):
            r = img[i][j][0]
            g = img[i][j][1]
            b = img [i][j][2]
            img_filtree[i][j]=[f(r,couleur[0]), f(g,couleur[1]), f(b,couleur[2])]
    return img_filtree


# Conversion RGB -> HSL

def convertisseur (r,g,b :int)-> int: #Safia
    '''
    Convertit la couleur RGB en HSL
    '''
    r, g, b = r/255, g/255, b/255
    Cmax=max(r,g,b)
    Cmin=min(r,g,b)
    Cdiff=Cmax-Cmin
    if Cmax==0:
        T=0
    elif Cmax == r:
        T=(((g-b)/Cdiff)) % 6
    elif Cmax == g:
        T = (((b-r)/Cdiff) + 2) % 6
    elif Cmax == b:
        T = (((r-g)/Cdiff) + 4) % 6
    if Cmax == 0:
        T = 0
    t=60*T
    L=1/2*(Cmax+Cmin)
    if L==1:
        S=0
    else:
        S=Cdiff/(1-abs(2*L-1))
    return t,S*100,L*100
```

Squelette du code

Script Python

```python
# Import des modules


# Chargement des fichiers de données

video = 'data/video_station1_groupe5.mp4'
son = 'son_station1_groupe5.wav'
image = imageio.imread('images/Gr5_Sp1_Florette2.jpg')
audio = 'audio.wav'

# Extraction du son de la video

os.system('ffmpeg -i ' +  + ' ' + )

# Extraction des volumes du son et de l'audio

v_son =
v_video =

# Création du reader de la vidéo

reader = imageio.get_reader(video)

# Boucle principale

for k in range(25):
    # extraction de l'image de la video

    # extraction de la couleur dominante, convertie en hsl

    # extraction de la zone

    # application du filtre

    # enregistrement de l'image


# Création du GIF
with imageio.get_writer('GIF_ultime.gif', mode='I') as writer:
    for k in range(25):
        #redimensionnement de l'image en ligne de commande

        #lecture de la k-ième image du gif

        # ajout de l'image au writer
```