gui_unifiée/class_MilleFeuille3D.py

```
import os

from matplotlib.pyplot import figure as Figure
from matplotlib.backends.backend_qt5agg import FigureCanvasQTAgg
from mpl_toolkits.mplot3d import Axes3D
import matplotlib.cm as cm

import numpy

from function_readPGM import readPGM


"""
Classe MilleFeuille3D, hérite de FigureCanvasQTAgg
Cette classe permet de gérer un graphique 3D d'images pouvant être tourné
et inséré dans un environnement Qt
Ces images sont affichées sous la forme d'un mille-feuilles
Basé sur Graphique3D dans class_Graphique3D.py
@author Amaury
"""
class MilleFeuille3D(FigureCanvasQTAgg) :
    """
    Constructeur, initialise le graphique
    """
    def __init__(self) :
        self.figure = Figure()
        self.figure.subplots_adjust(bottom=0, top=1, left=0, right=1) #
Supprime les marges
        FigureCanvasQTAgg.__init__( self, self.figure ) # Objet de type
FigureCanvas
        self.axes = self.figure.gca( projection = '3d' ) # On lui dit qu'on
veut des axes 3D, et on les stockes dans un attribut

    """
    Dessine ou actualise avec un nouveau graphique
    @param "listeImages" : Liste d'images à afficher, au format PGM (Base
8), associées à leur hauteur à afficher
    """
    def dessinerMilleFeuille3D(self, listeImages) : # Procédure qui dessine
le graphique
        self.axes.clear() # Nettoie les axes et leur contenu
        self.axes.set_xlabel( 'Axe X' ) # Label sur l'axe X
        self.axes.set_ylabel( 'Axe Y' ) # Label sur l'axe Y
        self.axes.set_zlabel( 'Axe Z' ) # Label sur l'axe Z
#         self.axes.set_aspect( 'equal' ) # Permet d'avoir un repère
orthonormal

        for I in range( len( listeImages ) ) :
            if os.path.isfile( listeImages[I][0] ) : # Si le chemin d'accès
à l'image existe
                # Source :
https://stackoverflow.com/questions/45663597/plotting-3d-image-form-a-data-
in-numpy-array
                # Traitement de l'image
                image = readPGM(listeImages[I][0], byteorder='<') # Matrix
au format uint8
                imageConvertie = image.astype(numpy.float64) / 255 #
Convertie en float64
```

```
                T = cm.gist_gray(imageConvertie) # Matrix float64 que
facecolors peut prendre
                # Liste des colormaps disponibles sur matplotlib.cm :
                #
https://matplotlib.org/3.1.0/gallery/color/colormap_reference.html

                # Source :
https://stackoverflow.com/questions/25287861/creating-intersecting-images-
in-matplotlib-with-imshow-or-other-function/25295272#25295272
                # Create a vertex mesh
                X, Y = numpy.meshgrid(numpy.linspace(0, len(image)-2,
len(image)-1 ), numpy.linspace(0, len(image[0])-2, len(image[0])-1 ))
                Z = numpy.zeros(X.shape) + listeImages[I][1]

                self.axes.plot_surface(X, Y, Z, facecolors=T)

                print( "[Info MilleFeuille3D] Ajout : " + listeImages[I][0]
)

            else :
                print( "[Erreur MilleFeuille3D] " + listeImages[I][0] + "
n'existe pas !" )

        self.draw()
```