# gui_unifiée/class_MilleFeuilleIRM.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 MilleFeuilleIRM, 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 de 3 plans perpidenculaires,
similaire à un IRM
Basé sur MilleFeuille3D dans class_MilleFeuille3D.py
@author Amaury
"""
"""
PROBLEME INSOLVABLE :
En fait, les 3 coupes s'affichent les unes sur les autres, il n'est pas
possible qu'elles s'emelent proprement
En fait, MatPlotLib dessine chaque couche une par une... Donc c'est mort !
SOURCE : https://stackoverflow.com/questions/13932150/matplotlib-wrong-
overlapping-when-plotting-two-3d-surfaces-on-the-same-axes
"""
class MilleFeuilleIRM(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 "imageX" : Liste contenant une image au format PGM (Base 8) à
afficher dans un plan YZ, ainsi que sa position en X.
    @param "imageY" : Liste contenant une image au format PGM (Base 8) à
afficher dans un plan XZ, ainsi que sa position en Y.
    @param "imageZ" : Liste contenant une image au format PGM (Base 8) à
afficher dans un plan XY, ainsi que sa position en Z.
    """
    def dessinerMilleFeuilleIRM(self, imageX, imageY, imageZ) : # 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

        """ Plan en YZ (Pour imageX) """
        if os.path.isfile( imageX[0] ) : # Si le chemin d'accès à l'image
existe
            # Traitement de l'image
            image = readPGM(imageX[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

            # Create a vertex mesh
            Y, Z = numpy.meshgrid(numpy.linspace(0, len(image[0])-2,
len(image[0])-1 ), numpy.linspace(0, len(image)-2, len(image)-1 ))
            X = numpy.zeros(Y.shape) + imageX[1]

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

            print( "[Info MilleFeuilleIRM] Ajout : " + imageX[0] )

        else :
            print( "[Erreur MilleFeuilleIRM] " + imageX[0] + " n'existe pas
!" )

        """ Plan en XZ (Pour imageY) """
        if os.path.isfile( imageY[0] ) : # Si le chemin d'accès à l'image
existe
            # Traitement de l'image
            image = readPGM(imageY[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

            # Create a vertex mesh
            X, Z = numpy.meshgrid(numpy.linspace(0, len(image[0])-2,
len(image[0])-1 ), numpy.linspace(0, len(image)-2, len(image)-1 ))
            Y = numpy.zeros(X.shape) + imageY[1]

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

            print( "[Info MilleFeuilleIRM] Ajout : " + imageY[0] )

        else :
            print( "[Erreur MilleFeuilleIRM] " + imageY[0] + " n'existe pas
!" )

        """ Plan en XY (Pour imageZ) """
        if os.path.isfile( imageZ[0] ) : # Si le chemin d'accès à l'image
existe
            # Traitement de l'image
            image = readPGM(imageZ[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
```

```
            # 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) + imageZ[1]

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

            print( "[Info MilleFeuilleIRM] Ajout : " + imageZ[0] )

        else :
            print( "[Erreur MilleFeuilleIRM] " + imageZ[0] + " n'existe pas
!" )

        self.draw()
```