gui_unifiée/class_Graphique3D.py

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


"""
Classe Graphique3D, hérite de FigureCanvasQTAgg
Cette classe permet de gérer un graphique 3D Matplotlib pouvant être tourné et inséré dans un environnement Qt
Basé sur ce script : https://stackoverflow.com/questions/52379426/pyqta-matplotlib-how-to-redraw-update-a-3d-surface-plot-in-a-window
@author Amaury
"""
class Graphique3D(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

        self.aEteInit = False

    """
    Dessine ou actualise avec un nouveau graphique
    Le paramètre "liste" est une liste de listes
    Chaque sous-liste représente une courbe
    Ces sous-listes doivent comprendre 3 sous-sous-listes étant les coordonnées X, Y et Z à tracer
    @param "courbeAfficher" : La courbe à afficher dans "liste" + 1, 0 si il faut les afficher toutes
    @param "tempsAfficher" : L'instant à afficher dans "liste" + 1, 0 si il faut afficher tous les instants
    @param "conserverLimites" : Conserver les limites du précédent appel de cette fonction (Ne fait rien si c'est la première fois)
    @param "limites" : Limites de 3 éléments avec les limites X, Y et Z à imposer (Outrepasse le paramètre précédent)
    """
    def dessinerGraphique3D(self, liste, courbeAfficher, tempsAfficher, conserverLimites = True, limites = None) : # Procédure qui dessine le graphique
        if limites != None :
            self.axes.clear() # Nettoie les axes et leur contenu

            self.axes.set_xlim(limites[0])
            self.axes.set_ylim(limites[1])
            self.axes.set_zlim(limites[2])

        elif conserverLimites :
            if self.aEteInit :
                # Sauvegarde taille des axes
                saveXLim = self.axes.get_xlim()
                saveYLim = self.axes.get_ylim()
```

```
                saveZLim = self.axes.get_zlim()

            self.axes.clear()

            if self.aEteInit :
                # Remet la sauvegarde
                self.axes.set_xlim(saveXLim)
                self.axes.set_ylim(saveYLim)
                self.axes.set_zlim(saveZLim)

            self.aEteInit = True

        else :
            self.axes.clear()

        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

        couleurs = ["r", "b", "g", "c", "m", "y"] # Liste des couleurs de
base de Matplotlib

        if courbeAfficher != 0 :
            couleur = couleurs[(courbeAfficher - 1) % len(couleurs)] #
Couleurs périodiques
            if tempsAfficher != 0 :
                try :
                    self.axes.plot( [liste[courbeAfficher -
1][0][tempsAfficher - 1]],
                                    [liste[courbeAfficher -
1][1][tempsAfficher - 1]],
                                    [liste[courbeAfficher -
1][2][tempsAfficher - 1]],
                                   couleur + 'o-' ) # Dessine le graphique
3D à partir de 3 listes dans les axes
                except IndexError :
                    print( "[Erreur] Les courbes n'ont pas la même longueur
!" )
            else :
                self.axes.plot( liste[courbeAfficher - 1][0],
                                liste[courbeAfficher - 1][1],
                                liste[courbeAfficher - 1][2],
                                couleur + 'o-' ) # Dessine le graphique 3D
à partir de 3 listes dans les axes
                try :
                    self.axes.plot( [liste[courbeAfficher - 1][0][0]],
                                    [liste[courbeAfficher - 1][1][0]],
                                    [liste[courbeAfficher - 1][2][0]],
                                    'ko-' ) # Affiche le début de la courbe
en noir
                except IndexError :
                    pass

        else :
            for numeroCourbe in range(len(liste)) :
                couleur = couleurs[numeroCourbe % len(couleurs)] # Couleurs
périodiques
                if tempsAfficher != 0 :
                    try :
```

```
                        self.axes.plot(
[liste[numeroCourbe][0][tempsAfficher - 1]],

[liste[numeroCourbe][1][tempsAfficher - 1]],

[liste[numeroCourbe][2][tempsAfficher - 1]],
                                       couleur + 'o-' ) # Dessine le
graphique 3D à partir de 3 listes dans les axes
                    except IndexError :
                        print( "[Erreur] Les courbes n'ont pas la même
longueur !" )
                else :
                    self.axes.plot( liste[numeroCourbe][0],
                                    liste[numeroCourbe][1],
                                    liste[numeroCourbe][2],
                                    couleur + 'o-' ) # Dessine le graphique
3D à partir de 3 listes dans les axes
        self.draw() # Dessine le graphique 3D avec les axes

    """
    @return Liste de 3 éléments avec les limites X, Y, et Z du graphe
    """
    def getLimitesGraphe() :
        return [self.axes.get_xlim(), self.axes.get_ylim(),
self.axes.get_zlim()]
```