gui_unifiée/class_TabGraphique3D.py

```
import sys

#from PyQt5.QtCore import *
#from PyQt5.QtGui import *
#from PyQt5.QtWidgets import *
from PyQt5.QtCore import *
from PyQt5.QtWidgets import QApplication, QWidget, QGridLayout, QScrollBar,
QHBoxLayout, QVBoxLayout, QLabel, QGroupBox

from class_Graphique3D import Graphique3D

from class_Parametres import Parametres # Ne sert que si est exécuté
séparemment
from parametres_graph3D_pour_demo import grapheDeDemonstration # Ne sert
que si est exécuté séparemment

from numpy import load


"""
Classe TabGraphique3D, hérite de la classe QGridLayout, c'est donc une
grille
Cette classe représente le contenu d'une fenêtre PyQt
Elle peut donc aussi être utilisée comme un onglet dans une fenêtre
@author Amaury
"""
class TabGraphique3D(QGridLayout) :
    """
    Constructeur, crée le contenu de l'onglet
    """
    def __init__(self, objParams, parent=None) :
        super(TabGraphique3D, self).__init__(parent) # Appel du
constructeur de QGridLayout

        self.objParams = objParams


        # Ajout Maylis
        vertical_layout=QVBoxLayout()

        # Valeurs actuelles des barres de scroll
        group_box=QGroupBox("Positions courantes des barres de scroll")
        horizontal_layout = QHBoxLayout()
        group_box.setLayout(horizontal_layout)

        self.valeur_temps = QLabel("Temps : 0")
        self.valeur_courbe = QLabel("X : 0")

        horizontal_layout.addWidget(self.valeur_courbe)
        horizontal_layout.addWidget(self.valeur_temps)


        vertical_layout.addWidget(group_box)
        # Fin Ajout Maylis


        # Graphe à afficher
        if __name__ != '__main__' : # Si on n'est pas le script principal
```

```
            print( "[Info TabGraphique3D] Fichier NPY utilisé : " +
self.objParams.genererURLGraph3D() )
            try :
                self.graphe = load( self.objParams.genererURLGraph3D() )
            except FileNotFoundError :
                print( "[Info TabGraphique3D] " +
self.objParams.genererURLGraph3D() + " n'existe pas !" )
                self.graphe = [[[],[],[]]]
        else :
            self.graphe = grapheDeDemonstration

        self.graphique3D = Graphique3D()
#         self.graphique3D.setMinimumSize(QSize(400, 400)) # Définit la
taille minimum en pixels de ce Widget
#         # Cela permet de bloquer le trop retrécissement de la fenêtre
#         # On peut remplacer "QSize(400, 400)" par
"self.graphique3D.sizeHint()" pour que la taille par défaut soit la taille
minimum
#       Devenu inutile car on définit la taille minimale de la fenêtre

        # Défilement coupes
        self.barreDeScrollCourbes = QScrollBar() # C'est une barre de
défilement
        self.barreDeScrollCourbes.setMaximum( len(self.graphe) ) # Défini
le nombre de valeurs qu'on peut y parcourir
        # len(self.graphe) est le nombre de courbes
        self.barreDeScrollCourbes.valueChanged.connect(
self.dessinerGraphique3D ) # La procédure à appeler lorsque l'utilisateur y
touche

        # Défilement temporel
        self.barreDeScrollTemps = QScrollBar(Qt.Horizontal)
        self.barreDeScrollTemps.setMaximum( len(self.graphe[0][0]) ) #
Temps à 0 signifie tous les temps
        # len(self.graphe[0][0]) est le nombre d'échantillons temporels
dont on dispose
        self.barreDeScrollTemps.valueChanged.connect(
self.dessinerGraphique3D )

        vertical_layout.addWidget( self.graphique3D , stretch=2) # Ajoute
le graphique 3D en position ligne 2 colonne 1
        self.addLayout(vertical_layout,1,1)
        self.addWidget( self.barreDeScrollCourbes,1,2 ) # Ajoute la barre
de défilement 1 en position ligne 2 colonne 2
        self.addWidget( self.barreDeScrollTemps,2,1 ) # Ajoute la barre de
défilement 2 en position ligne 2 colonne 2

        self.dessinerGraphique3D(0) # Afficher graphique de base

    """
    Gère le dessin et les changements par l'utilisateur dans les barres de
défilement
    """
    def dessinerGraphique3D(self, value) :
         self.graphique3D.dessinerGraphique3D( self.graphe,
self.barreDeScrollCourbes.value(), self.barreDeScrollTemps.value() )

         # Ajout Maylis
         if self.barreDeScrollTemps.value() == 0 :
             self.valeur_temps.setText("Temps : Tous les temps")
         else :
```

```
            self.valeur_temps.setText("Temps : " +
str(self.barreDeScrollTemps.value() - 1))

        if self.barreDeScrollCourbes.value() == 0 :
            self.valeur_courbe.setText("Courbe : Toutes les courbes")
        else :
            self.valeur_courbe.setText("Courbe : " +
str(self.barreDeScrollCourbes.value() - 1))
        # Fin Ajout Maylis

        print( "[Info TabGraphique3D] Temps : " + str(
self.barreDeScrollTemps.value() ) + ", Courbe : " + str(
self.barreDeScrollCourbes.value() ) )

    """
    Modifier position barres de scrolls
    """
    def setScrollBarsValues( self, courbe = None, temps = None ) :
        if courbe != None :
            print( "[Info TabGraphique3D] Valeurs forcées : Courbe = " +
str(courbe) )
            self.barreDeScrollTemps.setValue(courbe)
        if temps != None :
            print( "[Info TabGraphique3D] Valeur forcée : Temps = " +
str(temps) )
            self.barreDeScrollTemps.setValue(temps)


"""
Code principal pour démonstration
"""
# Si on est le script principal
# Cela permet de ne pas exécuter ce bloc de codes lorsque ce script est
importé par un autre
# Source : https://stackoverflow.com/questions/419163/what-does-if-name-
main-do
if __name__ == '__main__' :
    application = QApplication(sys.argv) # Crée un objet de type
QApplication (Doit être fait avant la fenêtre)
    fenetre = QWidget() # Crée un objet de type QWidget
    fenetre.setWindowTitle("MODE DÉMONSTRATION") # Définit le nom de la
fenêtre
    fenetre.setLayout( TabGraphique3D( Parametres() ) )
    fenetre.show() # Affiche la fenêtre
    application.exec_() # Attendre que tout ce qui est en cours soit
exécuté
```