gui_unifiée/class_TabMilleFeuille3D.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,
QLabel, QHBoxLayout, QVBoxLayout, QGroupBox

from class_MilleFeuille3D import MilleFeuille3D

from parametres import ENABLE_ANTI_LAG
from class_Parametres import Parametres # Ne sert que si est exécuté
séparemment


"""
Classe TabMilleFeuille3D, 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 TabMilleFeuille3D(QGridLayout) :
    """
    Constructeur, crée le contenu de l'onglet
    """
    def __init__(self, objParams, parent=None) :
        super(TabMilleFeuille3D, self).__init__(parent) # Appel du
constructeur de QGridLayout

        self.objParams = objParams

        # Graphe à afficher
        self.milleFeuille3D = MilleFeuille3D()

        # Défilement de couches inférieures (Valeur de la couche minimum à
afficher)
        self.barreDeScrollMFCoucheMin = QScrollBar()
        self.barreDeScrollMFCoucheMin.setMaximum(
self.objParams.nombreImagesPlanXY() )
        self.barreDeScrollMFCoucheMin.valueChanged.connect(
self.dessinerMilleFeuille3D )

        # Défilement de couches supérieures (Valeur de la couche maximum à
afficher)
        self.barreDeScrollMFCoucheMax = QScrollBar()
        self.barreDeScrollMFCoucheMax.setMaximum(
self.objParams.nombreImagesPlanXY() )
        self.barreDeScrollMFCoucheMax.valueChanged.connect(
self.dessinerMilleFeuille3D )

        # Défilement temporel
        self.barreDeScrollMFTemps = QScrollBar(Qt.Horizontal)
        self.barreDeScrollMFTemps.setMaximum(
self.objParams.nombreInstantsTemporels() )
        self.barreDeScrollMFTemps.valueChanged.connect(
self.dessinerMilleFeuille3D )
```

```
        # Ajout des Widgets
#         self.addWidget( self.milleFeuille3D, 2, 1 )
        self.addWidget( self.barreDeScrollMFCoucheMin, 1, 2 )
        if not ENABLE_ANTI_LAG : self.addWidget(
self.barreDeScrollMFCoucheMax, 1, 3 )
        # Ne pas l'afficher quand l'ANTI_LAG est activé, donc inutilisable,
donc une seule couche affichée
        self.addWidget( self.barreDeScrollMFTemps, 2, 1 )


        #  Ajout Maylis
        self.valeur_temps = QLabel("Temps : 0")
        self.valeur_Z = QLabel("Z : 0")

        group_box=QGroupBox("Positions courantes des barres de scroll")
        horizontal_layout = QHBoxLayout()
        group_box.setLayout(horizontal_layout)
        vertical_layout = QVBoxLayout()

        horizontal_layout.addWidget(self.valeur_temps)
        horizontal_layout.addWidget(self.valeur_Z)
        vertical_layout.addWidget(group_box)
        vertical_layout.addWidget(self.milleFeuille3D,stretch=2)

        self.addLayout(vertical_layout,1,1)
        # Fin Ajout Maylis


        self.dessinerMilleFeuille3D(0)

    """
    Gère le dessin et les changements du mille-feuilles 3D
    """
    def dessinerMilleFeuille3D(self, value) :
        # Si ENABLE_ANTI_LAG est activé, ET/OU que barreDeScrollMFCoucheMax
est à 0 (Forcément si ENABLE_ANTI_LAG), on ne commande qu'avec
barreDeScrollMFCoucheMin
        listeImages = [] # Liste des images que on veut afficher dans le
mille-feuilles
        if self.barreDeScrollMFCoucheMax.value() != 0 : # Commander le
défilement avec les deux barres
            for i in range(self.barreDeScrollMFCoucheMin.value(),
self.barreDeScrollMFCoucheMax.value(), 1) :
                urlImage = self.objParams.genererURLdesPGM3D( 'XY',
self.barreDeScrollMFTemps.value(), i ) # Chemin de l'image
                hauteurImage = self.barreDeScrollMFCoucheMin.value() + i #
Hauteur de l'image dans le mille-feuille
                listeImages.append( [urlImage, hauteurImage] )
        else : # Permet de ne commander qu'avec le défilement de la valeur
minimum, forcément si ENABLE_ANTI_LAG activé
            urlImage = self.objParams.genererURLdesPGM3D( 'XY',
self.barreDeScrollMFTemps.value(), self.barreDeScrollMFCoucheMin.value() )
            listeImages.append( [urlImage,
self.barreDeScrollMFCoucheMin.value()] )

        self.milleFeuille3D.dessinerMilleFeuille3D( listeImages )

        # Ajout Maylis
        self.valeur_temps.setText("Temps : " +
str(self.barreDeScrollMFTemps.value()))
```

```
        self.valeur_Z.setText("Z : " +
str(self.barreDeScrollMFCoucheMin.value()))
        # Fin Ajout Maylis

        print( "[Info TabMilleFeuille3D] Min : " + str(
self.barreDeScrollMFCoucheMin.value() ) + ", Max : " + str(
self.barreDeScrollMFCoucheMax.value() ) + ", Temps : " + str(
self.barreDeScrollMFTemps.value() ) )
        if ENABLE_ANTI_LAG : print( "[Info TabMilleFeuille3D] Affichage : "
+ urlImage )


"""
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( TabMilleFeuille3D( Parametres() ) )
    fenetre.show() # Affiche la fenêtre
    application.exec_() # Attendre que tout ce qui est en cours soit
exécuté
```