gui_unifiée/class_Parametres.py

```
from os.path import join
from os.path import abspath

import parametres_par_defaut


"""
Objet de centralisation des paramètres
@author Amaury
"""
# Note : Les getters et les setters, en Python, ça sert à rien, tout est
public
class Parametres() :
    def __init__(self) :
        """ Valeurs par défaut """
        self.contientVariablesImportees = False

        self.NB_IMGS = parametres_par_defaut.NB_IMGS # Nombre d'images au
format PGM
        self.INTERVALLE_XY = parametres_par_defaut.INTERVALLE_XY #
Intervalle temporel dans cette liste d'images, c'est à dire le pas dans le
plan XY
        self.INTERVALLE_XZ = parametres_par_defaut.INTERVALLE_XZ # Pas dans
le plan XZ
        self.INTERVALLE_YZ = parametres_par_defaut.INTERVALLE_YZ # Pas dans
le plan YZ
        self.URL_PGM = parametres_par_defaut.URL_PGM # URL vers les
fichiers PGM
        self.URL_VTK = parametres_par_defaut.URL_VTK # URL vers les
fichiers VTK
        self.URL_GRAPHIQUE_3D = parametres_par_defaut.URL_GRAPHIQUE_3D #
URL vers le fichier NPY
        self.URL_VITESSE_MOY_GRAINS =
parametres_par_defaut.URL_VITESSE_MOY_GRAINS # URL vers le fichier NPY
        self.CHEMIN_ABSOLU_FICHIER_IMPORTE = None # Chemin absolu du
fichier contenant les params importés

        self.tabGraphique3D = None # Objet TabGraphique3D

    """
    @return True si il y a une logique, False sinon
    """
    def verifierParams(self) :
        if self.NB_IMGS % self.INTERVALLE_XY != 0 :
                return False
        return True


    """
    @param plan : Le plan de la caméra, 'YZ', 'XZ' ou 'XY'
    @param instantTemporel : L'instant temporel du PGM
    @param couche : Le numéro de la couche du PGM
    @param typeDeTraitement : Optionnel, le type de traitement. Peut être :
                              - "originales", images originales dans le
fichier traité,
                              - "borders, images avec détection des bords,
                              - "carte_dist", carte de distances,
```

```
                                - "contours_blancs", images avec contours
blancs,
                                - "contours_rouges", images avec contours
rouges,
                                - "water", watershed
    """
    def genererURLdesPGM3D( self, plan, instantTemporel, couche,
typeDeTraitement = "originales" ) :
        coucheFormate = format(couche, '04d') # String sur 4 digits
        tempsFormate = format(instantTemporel, '02d') # String sur 2 digits

        if typeDeTraitement == "originales" :
            extension = "_org.pgm"
        elif typeDeTraitement == "borders" :
            extension = "_border.pgm"
        elif typeDeTraitement == "carte_dist" :
            extension = "_distcolor.ppm"
        elif typeDeTraitement == "contours_blancs" :
            extension = "_contblanc.pgm"
        elif typeDeTraitement == "contours_rouges" :
            extension = "_controuge.ppm"
        elif typeDeTraitement == "water" :
            extension = "_wa.pgm"
        else :
            raise Exception("Type de traitement inconnu !")

        if plan == 'YZ' :
            fichierPGM = self.URL_PGM + typeDeTraitement + "/" + "y_z/" +
tempsFormate + "/t_" + tempsFormate + "coupe_yz_" + coucheFormate +
extension
        elif plan == 'XZ' :
            fichierPGM = self.URL_PGM + typeDeTraitement + "/" + "x_z/" +
tempsFormate + "/t_" + tempsFormate + "coupe_xz_" + coucheFormate +
extension
        elif plan == 'XY' :
            fichierPGM = self.URL_PGM + typeDeTraitement + "/" + "x_y/" +
tempsFormate + "/t_" + tempsFormate + "coupe_xy_" + coucheFormate +
extension
        else :
            raise Exception("Plan inconnu !")

        if self.contientVariablesImportees :
            # Les URL doivent être relatives au fichier d'importation
            return abspath( join(
self.CHEMIN_ABSOLU_FICHIER_IMPORTE.replace("\\", "/"),
                                 fichierPGM.replace("\\", "/")
                                ).replace("\\","/") )
        else :
            return fichierPGM

    """
    @param instantTemporel : L'instant temporel du VTK
    """
    def genererURLdesVTK( self, instantTemporel ) :
        tempsFormate = format(instantTemporel, '02d') # String sur 2 digits
        fichierVTK = self.URL_VTK + "vtk_t_" + tempsFormate + ".vtk"

        if self.contientVariablesImportees :
            # Les URL doivent être relatives au fichier d'importation
            return abspath(join(
self.CHEMIN_ABSOLU_FICHIER_IMPORTE.replace("\\", "/"),
```

```
                                    fichierVTK.replace("\\", "/")
                                   ).replace("\\","/"))
        else :
            return fichierVTK

    """
    @return URL du fichier NPY à utiliser
    """
    def genererURLGraph3D(self) :
        if self.contientVariablesImportees :
            # Les URL doivent être relatives au fichier d'importation
            return abspath( join(
self.CHEMIN_ABSOLU_FICHIER_IMPORTE.replace("\\", "/"),
                                    self.URL_GRAPHIQUE_3D.replace("\\", "/")
                                   ).replace("\\","/") )
        else :
            return self.URL_GRAPHIQUE_3D

    """
    @return URL du fichier NPY à utiliser
    """
    def genererURLInfos(self) :
        if self.contientVariablesImportees :
            # Les URL doivent être relatives au fichier d'importation
            return abspath( join(
self.CHEMIN_ABSOLU_FICHIER_IMPORTE.replace("\\", "/"),
                                    self.URL_VITESSE_MOY_GRAINS.replace("\\",
"/")
                                   ).replace("\\","/") )
        else :
            return self.URL_VITESSE_MOY_GRAINS


    """
    @return Nombre d'images dans le plan de la caméra YZ, moins 1
    """
    def nombreImagesPlanYZ(self) :
        return self.INTERVALLE_YZ - 1

    """
    @return Nombre d'images dans le plan de la caméra YZ, moins 1
    """
    def nombreImagesPlanXZ(self) :
        return self.INTERVALLE_XZ - 1

    """
    @return Nombre d'images dans le plan de la caméra YZ, moins 1
    """
    def nombreImagesPlanXY(self) :
        return self.INTERVALLE_XY - 1

    """
    @return Nombre d'instants temporels, moins 1
    """
    def nombreInstantsTemporels(self) :
        return self.NB_IMGS / self.INTERVALLE_XY - 1
```