

Documentation développeur

Date Avril 2024 – SIMV IGN

Lexique

Vintage : millésime

Commandes récurrentes

`Self.getMasterWindow()` : récupérer l'objet `TraiNminaTor2Dialog_Master` qui correspond à la fenêtre maîtresse. Cette méthode est définie dans un peu toutes les classes

`Self.getMasterWindow().associatedWindow` : récupérer l'objet `TraiNminaTor2Dialog_Differential` qui correspond à la fenêtre secondaire

`self.getMasterWindow().projectManager.isDifferential` : True pour le mode bidate, False pour le mode monodate

`Self.getMainWindow()` : récupère l'objet `TraiNminaTor2Dialog_Base`. Plus exactement récupère `TraiNminaTor2Dialog_Master` si la classe appartient à la fenêtre principale et `TraiNminaTor2Dialog_Differential` si la classe appartient à la fenêtre secondaire

Dans `TnTWidgetsGroup.py` : appel récurrent de `self.sender()` qui permet de d'obtenir l'objet qui a déclenché l'événement, et donc le bouton sur lequel l'utilisateur vient de cliquer.

Utilisation des fonctions du type `self.findChild(QPushButton, "show_Context")` pour trouver l'élément qui nous intéresse.

Structure des données

Lorsqu'on lance pour la première fois un chantier, pour chaque niveau de segmentation, un nouveau shapefile est créé dans le répertoire `LABELED_DATA`. Dans chacun de ces nouveaux shapefiles, on trouve les mêmes attributs que dans la couche originelle, mais aussi pour chaque millésime les attributs `"color_{vintage}"`, `"code_{vintage}"` et `"class_{vintage}"`. En termes de visualisation, sur la fenêtre principale, on verra ces couches regroupées sous le nom `LABELED_DATA_{vintage1}` et dans la fenêtre secondaire sous le nom `LABELED_DATA_{vintage2}`. Malgré le fait que le nom du groupe soit différent, il s'agit en réalité du même fichier shapefile. Selon la fenêtre où les modifications seront saisies, on modifiera seulement les attributs correspondant au millésime de cette fenêtre.

Structure du code

Il y a pas mal de dossiers et scripts générés lors de la création d'un plugin QGIS que l'on ne détaillera pas ici. Les scripts du plugin sont :

- *Css/Default.css*
- *trainminator2.py* : lancement du plugin et ouverture de la fenêtre principale Master
- *trainminator2_dialog.py* : définition des fenêtres Master et Differential. Initialisation du style avec le css. Initialisation de la gestion des couches : *TnTLayerTree_DockWidget*. Gestion ouverture et changement de nomenclature : *TnTnomenclatureWidget*. Création des outils de sélection : *selectingToolsGroup*. Création du slider de niveau de segmentation : *sliderGroup*
- *trainminator2_Widget.py* : définition du canvas et création des widgets situés autour du canvas.
- *TnT_WidgetsGroup.py* : définition des widgets composants le plugin.
- *TnT_MapCanvas* : définition des actions sur le canvas : mise à jour du pointeur, de l'affichage des labels, ...
- *TnT_DockWidget.py* : gestion des couches affichées et de la nomenclature
- *TnT_CaptureManager.py* : définition des outils de sélection : point, ligne, polygone.
- *TnT_Features.py* : gestion des attributs de couche
- *TnT_ProjectManager.py* : gestion du projet, vérification de l'architecture du dossier du projet chargé. Création des shapefile Labeled.
- *TnT_SavingLabeledData.py* : gestion de la sauvegarde des données.

Organisation du plugin

- La **nomenclature** est gérée par :
 - o Les DockWidget définis dans les classes *TnTNomenclature_DockWidget* et *TnTNomenclature_DockWidget_Master* dans le fichier *TnT_DockWidget.py*
 - o Les Widget définis dans les classes *TnTnomenclatureWidget* et *TnTnomenclatureWidget_Master* dans le fichier *TnT_WidgetsGroup.py*
- Les **couches** sont gérées par :
 - o Les DockWidget définis dans les classes *TnTLayerTree_DockWidget* et *TnTLayerTree_DockWidget_Master* dans le fichier *TnT_DockWidget.py*
 - o Les Widget définis dans les classes *TnTLayerTreeWidget* et *TnTLayerTreeWidget_Master* dans le fichier *TnT_WidgetsGroup.py*
- Toute la **partie centrale** du plugin est dans les conteneurs *TraiNminaTor2Widget_Differential* et *TraiNminaTor2Widget_Master* dans le fichier *trainminator2_Widget.py*
 - o Les widgets de la partie centrale suivants sont tous codés dans le fichier *TnT_WidgetsGroup.py* :
 - Les outils de **visualisation** : classe *viewsManagerGroup_Master* (Fit All, Synchro Views, Synchro Levels, Add View)
 - Le **slider** : classe *sliderGroup*
 - Les **informations** sur la sélection en cours : classe *infoSelectionGroup*
 - Les **outils d'annotation** : classe *toolsGroup_Master*
 - Pour **démarrer** et arrêter l'annotation : classe *startStopToolsGroup*

- Pour **remplir** la pyramide à la fin de l'annotation : classe *mergeToolsGroup*
- Choisir le **type** de sélection (annoter, supprimer) : *taskToolsGroup*
- Choisir l'**outil** de sélection (point, ligne) : *selectingToolsGroup*
- Choisir les **attributs** à afficher (tous, non labellisés) :
attributSelectingToolsGroup
- Choisir les **informations** à afficher (code, contexte) : *displayToolsGroup*
- Afficher les **labels** au passage du curseur après activation avec la touche 'i' :
displayLabelsGroup