Géomatique Appliquée
Licence Professionnelle
Université Chouaib Doukkali

PROJET FIN MODULE

PYTHON , Programmation SIG

Outils de chainage des points

PLUGIN QGIS

Encadré par

Mr Houd Mohammed

Réalisé par

Bahia Zada
Larhnimi Omar
Marsaoui Ayoub
Msahal Yasser

Remerciement

Au terme de ce travail nous tenant à exprimer notre gratitude et nos remerciements dans le premier lieu à Monsieur Fekkak qui nous a donné la chance d'être parmi cette promotion d'LPGA

Nous remercions également monsieur Houd Mohammed pour son aide précieuse, ses conseils, son encouragement et sa disponibilité ainsi que l'opportunité qui nous a donné d'avoir utilisé un outil moderne du développement sig qui va être très utile dans notre carrière professionnelle

Introduction

Les projets sont les meilleurs occasions pour que les étudiants puissent élargir leur cercle de connaissance, de partager leur savoir-faire entre eux et d'appliquer les méthodologies et les notions enseignées jusqu'en Maîtrise du module concerné. Dans ce sens, notre projet a porté sur la réalisation d'une application qui fait la liaison entre les points avec des lignes par luis attribuer des identifiants. Celui-ci nous a initié à la recherche, à appliquer les connaissances acquises durant notre formation et de favoriser le travail en groupe encadré par notre professeur MR Houd Mohammed.

Le sujet du projet :

Le thème qui a été choisi à partir d'une proposition du monsieur Houd nous donne la possibilité par exemple de générer un trajet

à travers un ensemble de prise de position avec GPS. Donc Il s'agit de réaliser un outil qui recevra en entrée une couche de points projetés et qui sur la base de leurs coordonnées va créer une classe d'entités de lignes liant les points les uns aux autres. Chaque ligne sera unique et contiendra un champ ID

Pourquoi ce sujet ?

Avant d'aborder le vif de ce travail il est nécessaire de préciser le motif du choix de celui-ci. Le sujet traité est réalisé pour générer un ensemble de lignes facilement avec des identifiants pour lier des points .Le choix de ce sujet et fait pour d'autre raison tels que : QGIS est un logiciel gratuit qui nous offre La possibilité de partager notre plug-in facilement contrairement à arcgis qui est payant

Premier pas premier problème

Dans les premières étapes de travail et de nombreux problèmes rencontrés, notamment que nous avons essayé d'utiliser PyQgis avec Windows Mais il y avait un seul problème vient de la compatibilité et la compilation entre le système d'exploitation et le logiciel, même si en a essayé de travailler avec les dernières versions, ça donnera toujours le même résultat

Solution :

Donc pour sortir de ce problème On a pensé à changer le Système d'Exploitation donc on a essayé de travailler avec Linux pour certaines raisons parmi lesquelles on trouve :
La 1ere raison : c'est parce que il est open source et gratuit
La 2eme raison : c'est parce que Qgis est plus puissant dans Linux
La 3eme raison : la puissance de linux au côté ligne de commande ainsi qu'il facilite l'installation des outils dédiée au développement python à partir des dépôts

Outil de chainage des points

Qgis :

- QGIS est un logiciel SIG (système d'information géographique) libre multi plate-forme qui dispose d'une interface graphique accessible Il était également appelé Quantum GIS jusqu'à la version 1.9. La version 2.6 est sortie en octobre 2014.
- Au départ QGIS était destiné à la seule visualisation de données SIG Il supporte de nombreux formats aussi bien pour les données et les vecteurs (Arc/Info, ESRI Shapefile, Mapinfo File, ODBC, PostgreSQL, ...) que pour les rasters (Arc/Info ASCII Grid, GRASS Rasters, TIFF/GeoTIFF, USGS SDTS DEM, ...), ainsi que les webservices (WMS/WFS).

Pourquoi utiliser QGIS ?

Plusieurs raisons expliquent aujourd'hui que bon nombre de structures (publiques ou privées) se dotent de QGIS dans le cadre de leurs activités. Beaucoup d'entre elles délaissent même les produits des principaux éditeurs commerciaux de logiciels du marché pour se mettre à Quantum GIS.

Les avantages :

- Quantum GIS est open-source
- Bénéficie d'une ergonomie proche de logiciels du marché payants, tels qu'ArcGIS ou SuperGIS.
- Permet de visualiser, d'éditer et de créer une grande variété de formats vecteurs
- Logiciel qui évolue et s'améliore en permanence grâce à la mise à disposition de plugins personnalisés
- Permet une mise en page aisée de cartes grâce à un composeur d'impression.
- Permet de mettre à jour ses données géographiques en un clin d'œil grâce à son mode d'édition évolué.
- Propose des modes d'analyse thématique évolués
- Permet une interopérabilité à des bases de données externes

Tableau récapitulatif

QGIS	ARCGIS
open source	Payant
Multiplateformes	Windows
Python	Python

pyqt	Arcpy

Tableau : Récapitulatif de la comparaison entre ArcView et QGIS - En vert, les points positifs ; et rouge, les points négatifs

Caractéristiques \ Logiciels		ArcView	QGIS
Coût		Payant	Gratuit
Performance		Bonne	Bonne
ArcCatalog/QGIS Browser	Arborescence de l'ordinateur	Oui	Oui
	Manipulation des données	Oui	Non
Traitement des données		Payant	Gratuit
Formats compatibles	Shape	Natif	Natif
	MapInfo (TAB, MID/MIF)	Non	Oui
	KML	Non	Oui
	Autocad (DXG, DWG)	Oui	Oui
Mise en page	Plusieurs cartes par projet	Non	Oui
Divers	Lecture des fichiers ZIP (vecteur)	Non	Oui
	Définitions d'actions	Non	Oui
	Raccourci dans la barre de menus	Non	Oui

Python :

Python est un langage de programmation interprété, à ne pas confondre avec un langage compilé.
Il permet de créer toutes sortes de programmes, comme des jeux, des logiciels, des progiciels, etc.
Il est possible d'associer des bibliothèques à Python afin d'étendre ses possibilités.
Il est portable, c'est à dire qu'il peut fonctionner sou différents systèmes d'exploitation (Windows, Linux, Mac OS X,…).

Caractéristiques

- C'est un langage interprété, c'est-à-dire que le code ne nécessite pas d'être compilé pour être exécuté
- Il est multiplateforme et il est d'ailleurs natif sur tous les systèmes Unix (Mac OS, Linux) et ne nécessite pas d'y être installé !
- Il peut être utilisé en tant que langage de script pour exécuter une suite simple de commande mais c'est aussi un langage objet qui permet de développer des applications solidement construites !
- Sa syntaxe est très simple et aérée et permet de se libérer de tous les caractères de démarcation des blocs de code.
- Python est portable, non seulement sur les différentes variantes d'Unix, mais aussi sur les OS propriétaires

Domaines d'application :

Les domaines d'application naturels de Python incluent entre autres : L'apprentissage de la programmation objet.

- Les scripts d'administration système ou d'analyse de fichiers textuels.
- Tout le développement lié à l'Internet et en particulier au Web: scripts CGI, navigateurs Web, moteurs de recherche, agents intelligents, objets distribués...
- L'accès aux bases de données (relationnelles).
- La réalisation d'interfaces graphiques utilisateurs.
- Le calcul scientifique et l'imagerie. Python ne sert alors pas à écrire les algorithmes, mais à combiner et mettre en œuvre rapidement des librairies de calcul écrites en langage compilé (C, C++, Fortran, Ada,...).

PyQGIS :

Dès la version 0.9, QGIS intégrait un support optionnel pour le langage Python. Nous avons choisi Python car c'est un des langages les plus adaptés pour la création de scripts. Les dépendances PyQGIS proviennent de SIP et PyQt4. Le choix de l'utilisation de SIP plutôt que de SWIG plus généralement répandu est dû au fait que le noyau de QGIS dépend des librairies Qt. Les dépendances Python pour Qt (PyQt) sont opérées via SIP, ce qui permet une intégration parfaite de PyQGIS avec PyQt.

Utiliser PyQGIS dans une application personnalisée

Ne pas utiliser qgis.py comme nom de script test — Python ne sera pas en mesure d'importer les dépendances étant donné qu'elles sont occultées par le nom du script.

Tout d'abord vous devez importer module de QGIS, réglez QGIS chemin où rechercher des ressources - base de données des projections, des fournisseurs, etc. Lorsque vous définissez chemin de préfix second argument défini comme vrai, QGIS permet d'initialiser tous les chemins avec dir standard sous le répertoire préfix. Appel initQgis () fonction est important de laisser QGIS recherche des fournisseurs disponibles.

```
from qgis.core import *

# supply path to where is your qgis installed
QgsApplication.setPrefixPath("/path/to/qgis/installation", True)

# load providers
QgsApplication.initQgis()
```

QtDesigner :

Qt Designer est un outil de Qt pour la conception et la création d'interfaces utilisateur graphiques (GUI) de composants Qt. Vous pouvez composer et personnaliser vos widgets ou des dialogues dans un-what-you-get what-you-see-est (WYSIWYG) de manière, et de les tester en utilisant différents styles et résolutions.

Widgets :

Widgets et les formes créées avec Qt Designer intégrés de façon transparente avec le code programmé, en utilisant des signaux et slots mécanisme de Qt, qui vous permet de facilement assigner le comportement des éléments graphiques. Toutes les propriétés définies dans Qt Designer peuvent être modifiés dynamiquement dans le code. En outre, des fonctionnalités telles que la promotion des widgets et plugins personnalisés vous permettent d'utiliser vos propres composants avec Qt Designer

Qu’est-ce qu’un plugin :

En informatique, un plugin ou plug-in, aussi nommé module d'extension, module externe, greffon, plugiciel, ainsi que add-in ou

add-on en France, est un paquet qui complète un logiciel hôte pour lui apporter de nouvelles fonctionnalités.

Objectifs :

Tous les logiciels ne sont pas capables de fonctionner à l'aide de plugin, le logiciel en question doit être conçu pour pouvoir communiquer avec des programmes extérieurs selon certaines règles que ces derniers doivent respecter pour qu'ils puissent échanger des informations. Les objectifs des auteurs choisissant de concevoir ce genre de logiciel est de pouvoir ajouter des fonctionnalités sans avoir à tout reprogrammer et également de permettre aux utilisateurs d'ajouter leurs propres fonctionnalités de manière indépendante. Idéalement, cette indépendance inclut la possibilité pour le logiciel principal d'évoluer tout en restant compatible avec les plugins existants ; cette condition est cependant loin d'être toujours remplie.

QGIS Plugin Builder :

Builder Plugin vous fournit un modèle de travail à partir De laquelle vous pouvez créer votre propre plugin.

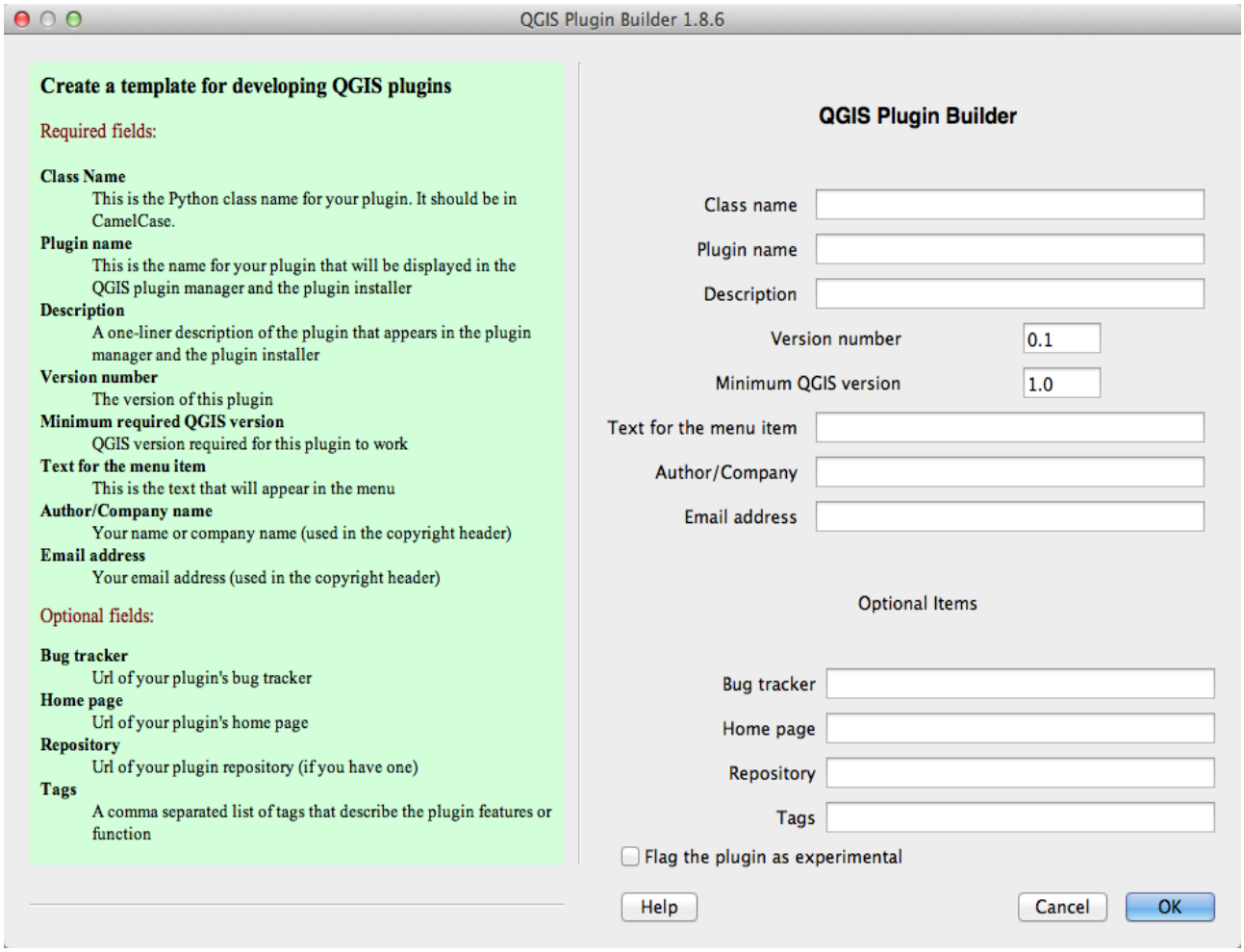
Les étapes à l'aide Plugin Builder sont assez simples:

- ❑ Ouvrez le Plugin Builder à partir de QGIS
- ❑ Remplissez les informations requises
- ❑ Cliquez sur OK
- ❑ Désigner où stocker votre nouveau plugin
- ❑ Compiler vos fichiers d'interface utilisateur et de ressources
- ❑ Installez le plugin

❑ Testez

Plugin Builder :

Lorsque vous exécutez Plugin Builder, vous verrez une fenêtre avec les champs de texte sur le droit et les descriptions sur la gauche :



Paramètres requis :

Nom de la classe	C’est le nom qui sera utilisé pour créer la classe de Python pour votre plugin
Nom du plugin	C’est un titre à votre plugin et sera affiché dans le gestionnaire de plugin QGIS
Description	C’est une description d'une ligne de la fonction du plug-in
Numéro de version	C’est le numéro de version de votre plugin. Builder Plugin suggère 0,1
Version minimale de QGIS	C’est la version minimale de QGIS requis pour votre plugin fonctionne
Texte pour l'élément de menu	C’est le texte qui apparaîtra dans le menu
Auteur / Société	Mettez votre nom ou le nom de l'entreprise
Adresse e-mail	Mettez une adresse où les utilisateurs de votre plugin peuvent vous contacter

Paramètres facultatifs :

Il Ya plusieurs champs facultatifs, mais fortement recommandé que vous devriez considérer lors de la génération de remplir un nouveau plugin.

Bug Tracker	Une URL pointant vers le bug / Issue Tracker pour votre plugin
Page d'accueil	L'URL de la page d'accueil de votre plugin
Repository	L'URL du référentiel de code source pour votre plugin
Tags	Les tags sont une liste de mots-clés décrivant la fonction (s) de votre plugin séparé par des virgules
Expérimental	Cochez cette case si votre plugin est considérée comme expérimental, ce qui signifie qu'il est incomplet ou peut provoquer des conséquences inattendues

Environnement linux :

Linux est le nom couramment donné à tout système d'exploitation libre fonctionnant avec le noyau Linux. C'est une implémentation libre du système UNIX respectant les spécifications POSIX. Ce système est né de la rencontre entre le mouvement du logiciel libre et le modèle de développement collaboratif et décentralisé via Internet

Caractéristiques :

La sécurité :

La sécurité est nécessaire pour protéger les données des utilisateurs mais aussi pour éviter de servir de point de départ à une attaque sur internet. Linux offre de bonnes garanties de sécurité. D'abord il est conçu dès le départ pour fonctionner en réseau. Ensuite il profite de son mode de développement.

Fiabilité et stabilité :

Un effort particulier a été fait lors du développement de linux pour s'assurer que le système soit fiable et stable :

- Les applications sont cloisonnées et ne peuvent faire planter le système entier

- Linux est conçu pour des serveurs devant fonctionner des mois durant ;

- les bogues découverts sont corrigés très rapidement.

Simplicité :

Avec une bonne configuration, Linux se révèle plus simple que la plupart des autres SE dans le cadre de la maintenance quotidienne. D'où un gain de temps appréciable. Sa conception multi-utilisateur rend possible et même facilite la mise en place d'un environnement distinct pour chacune des personnes amenées à utiliser la machine

A part ses caractéristiques on a préféré de travailler dans notre projet avec linux pour d'autres raisons parmi lesquelles on trouve :

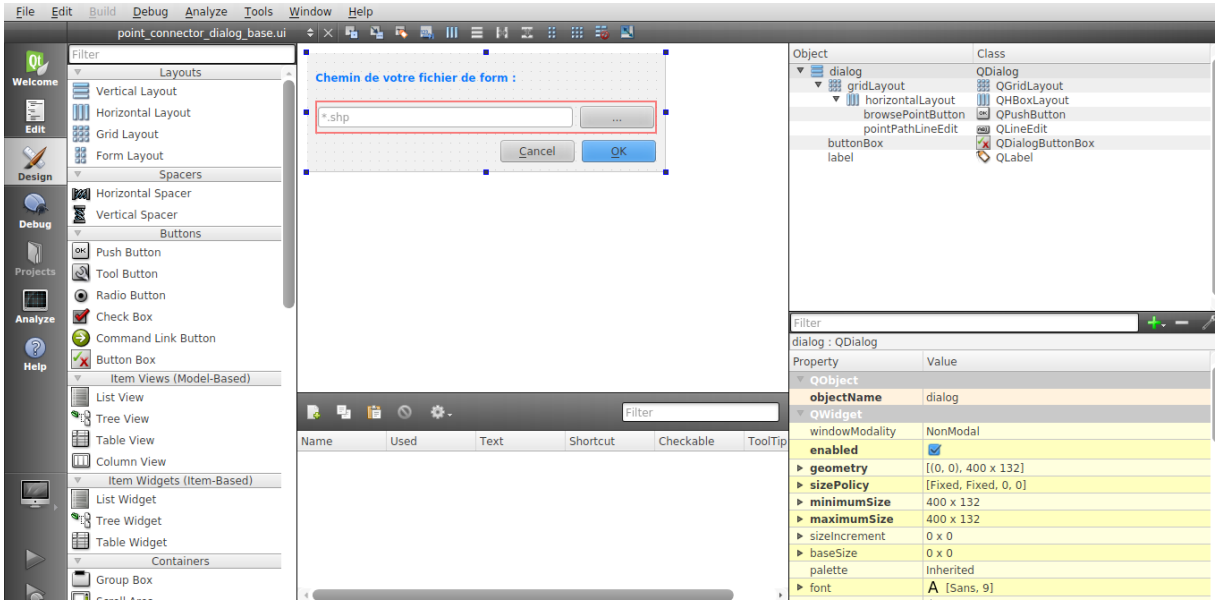
- Il est open source
- QGIS est très puissant dans linux par rapport à d'autres systèmes d'exploitation
- Ainsi qu'on a trouvé des problèmes dans la compilation de PyQgis dans Windows par contre dans linux

L'outil Réalisé :

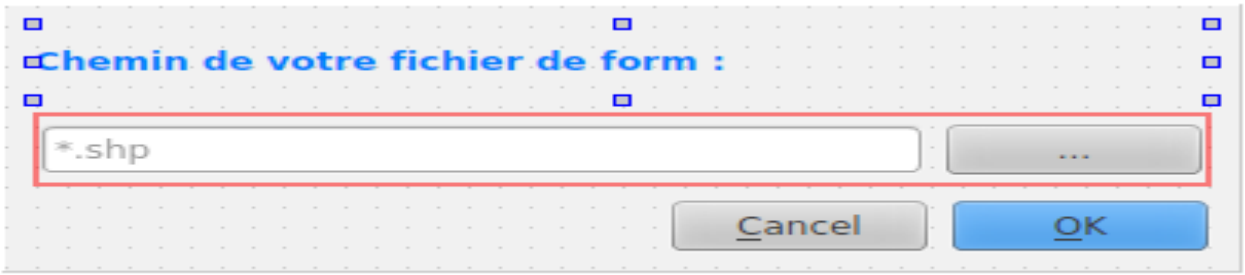
Un outil qui recevra en entrée une couche de points projetés et qui sur la base de leurs coordonnées va créer une classe d'entités de lignes liant les points les uns aux autres. Chaque ligne sera unique et contiendra un champ ID avec comme clé les identifiants combinés deux points liés.

Réalisation :

Nous avons utilisé le logiciel Qt Designer qui permet de créer des interfaces très riches d'une façon simple et interactive.



Notre formulaire se compose de 2 parties principales :

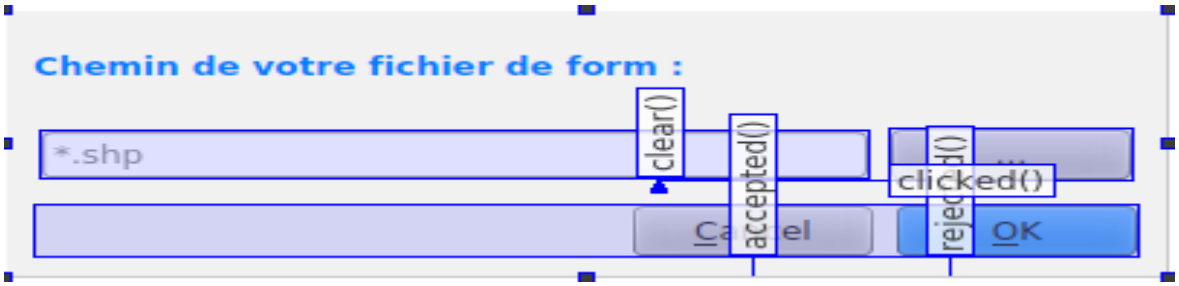


La partie qui est considérer pour les données d'entrer « Input » qui va être dans notre cas un fichier de forme, donc nous avons décidé de laisser à l'utilisateur le choix soit de saisir le chemin directement

dans le champ de texte si il connait le chemin sinon il peut utiliser la fenêtre de dialogue de l'explorateur des fichiers de son système d'exploitation.

La deuxième parti est celle considérer a la validation de l'action en cliquant sur « ok » sinon annuler en cliquant sur « Cancel »

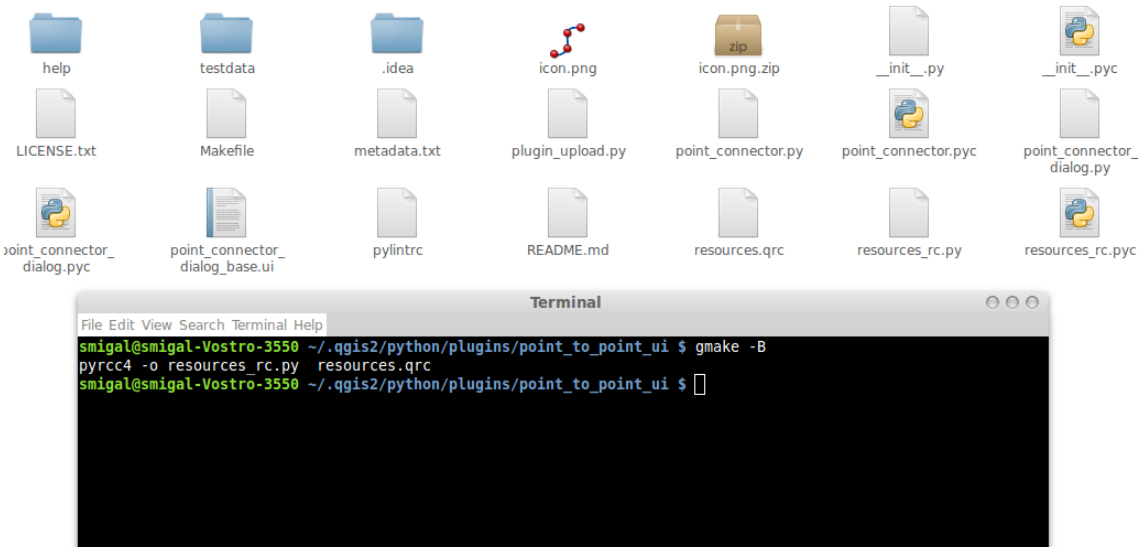
Voici une image qui représente les slot d'évènement de l'interface graphique



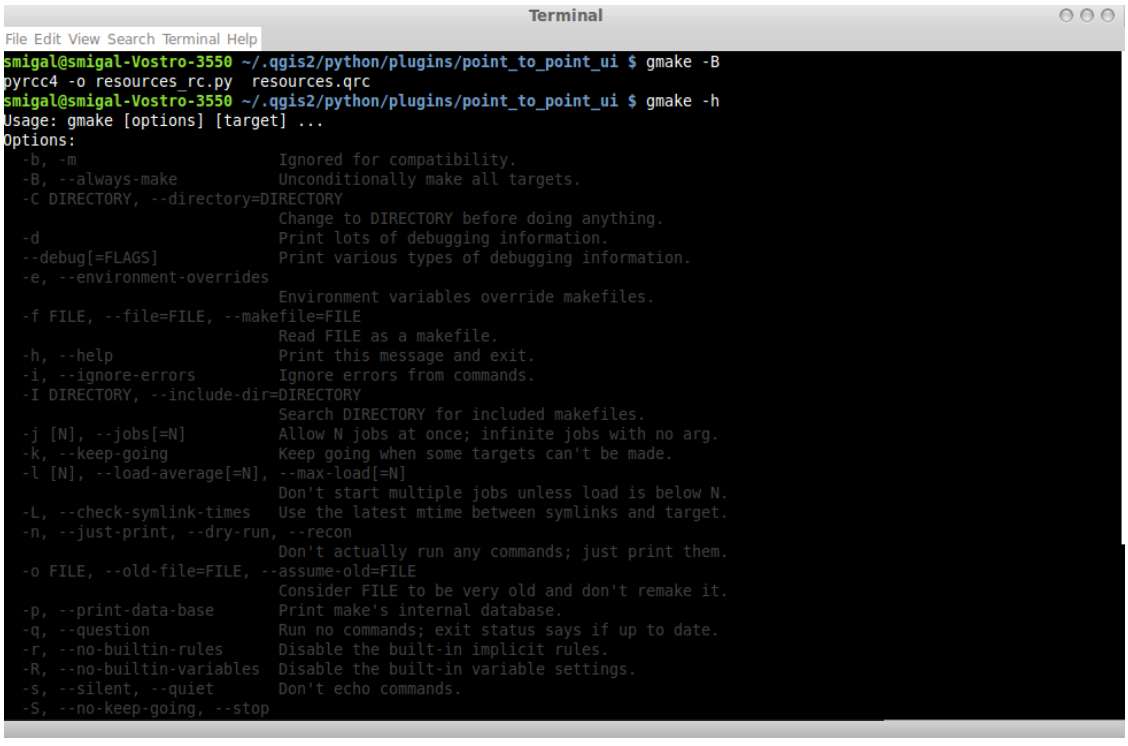
Nous avons aussi nommé chaque élément pour pouvoir y accéder à travers le code python par exemple le champ de texte on l'a nommé pointPathLineEdit

Object	Class
dialog	QDialog
gridLayout	QGridLayout
horizontalLayout	QHBoxLayout
browsePointButton	QPushButton
pointPathLineEdit	QLineEdit
buttonBox	QDialogButtonBox
label	QLabel

Une fois nous avons terminé de créer l'interface et la sauvegarder il nous reste que la compiler et la mettre en référence avec le fichier python, nous avons d'utiliser la commande gmake qui permet d'utiliser le fichier intelligent « MakeFile » elle permet de lancer à la fois la commande pyrcc4 et pyui4 selon la modification faite sur le fichier d'où on nom le fichier « Makefile » intelligent par ce qu'il permet de tracer le changement faite sur les fichiers



Le paramétré de la commande –B il permet tout simplement de forcer la compilation



Le code

La méthode la plus importante du fichier python c'est la méthode run qui représente le déroulement de l'exécution, qui prend un

paramètre nommé self qui représente la class actuel, et qui a une propriété dlg qui représente l'interface graphique cette interface graphique a une méthode nommé show qui permet l'affichage de l'interface graphique, pour maintenir l'afficher on est obligé de le mettre dans une boucle infini qui se trouve dans la exec_().

```
def run(self):
    # afficher le formulaire
    self.dlg.show()
    # et mettre son execution dans une boucle
    result = self.dlg.exec_()
```

Une fois l'utilisateur va clicker sur OK donc la variable result va etre égale à vrai ou 1, nous allonger récupérer le chemin saisie dans la zone de text pointPathLineEdit dans une variable appelé pointPath puis créer un object couche vectoriel depuis le chemin saisie, de type qu'on va nommer points.

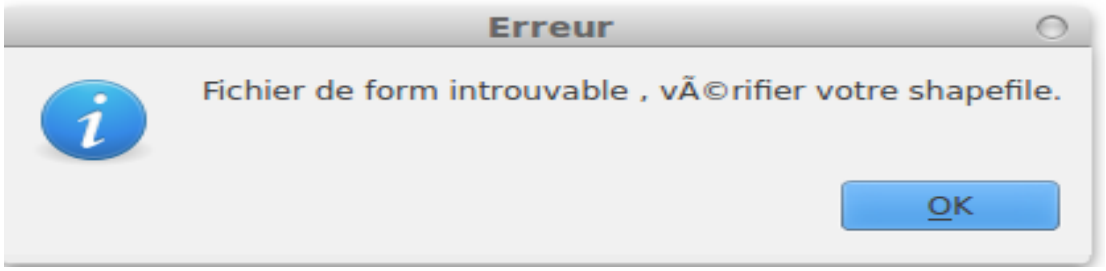
Créer un deuxiéme object vecorial cette fois de type line qui aura la même projection récupéré dans la variable point_layer_crs avec la méthode crs() en récupérant son identifiant avec la méthod authid().

Puis affecter le référence de données du couche vectorial line crée dans une variable nommé pr.

```
# si l'utilisateur appui sur ok
if result == 1:
    pointPath = self.dlg.pointPathLineEdit.text()
    point_layer = QgsVectorLayer(pointPath, 'points', 'ogr') #créer instance du shapefi
    point_layer_crs = point_layer.crs().authid() # récuper le code de sa projection
    lines_layer = QgsVectorLayer('LineString?crs='+point_layer_crs, 'lines', 'memory')
    point_name_index = 0 # l'index du champ id est par defaut a 0
    pr = lines_layer.dataProvider() # récuper réference du shapefile crée
```

Pour vérifier si le fichier sélectionner par l'utilisateur peut être ouvert nous avons utilisé la méthode suivant qui lors d'une exception affiche un message d'erreur

```
#verifier si on réussi a ouvrir le shapefile
try:
    p = open(pointPath, 'r')
    p.close()
except IOError:
    QMessageBox.information(None, "Erreur", "Fichier de form introuvable , vérifier votre shapefile.")
    return
```



Nous allons ouvrir une session d'Edition de couche vectoriel ligne, Après nous allons ajouter l'attribut id qui va représenter l'identifiant et qui sera de type entier

```
#début de l'edition du shapefile line qui aura un seule champ c'est id
lines_layer.startEditing()
pr.addAttributes ([ QgsField('id', QVariant.Int)] )
```

Nous allons créer un dictionnaire qui va avoir comme valeur cordonné de chaque point, et comme clef un identifiant auto-incrément. Puis nous allons ajouter le fichier de forme point dans notre carte

```
#creating point coordinate dict
points = processing.features(point_layer)
points_dict = {}
i = 0
for p in points:
    geom = p.geometry()  # récuper la geométrie du point
    attrs = p.attributes() # récuper ces attributs
    p = geom.asPoint()  # la considérant autant que piont
    time.sleep(0.01)
    points_dict[str(i)] = p
    i+=1
QgsMapLayerRegistry().instance().addMapLayer(point_layer)
```

Pour donner plus d'interactivité de notre plugin nous avons utilisé une barre de progression qui permet de donner un retour visuel a l'utilisateur pour savoir l'état d'avancement du traitement, ce

retour visuel est très important surtout lorsqu'il s'agit d'une grande quantité de points a lié

```
#Progress bar widget
progressMessageBar = iface.messageBar().createMessage("edition des lignes...")
progress = QProgressBar()
progress.setMaximum(len(points_dict))
progress.setAlignment(Qt.AlignLeft|Qt.AlignVCenter)
progressMessageBar.layout().addWidget(progress)
iface.messageBar().pushWidget(progressMessageBar, iface.messageBar().INFO)
```

C'est la partie la plus important celle qui permet de lier entre les points, chaque point avec le point qui le suit, en utilisant la méthode frompolyline qui prend deux paramétré de type QgsPoint, en ajoutant la valeur de barre de progression pour synchronisé l'état d'avancement avec le retour visuel de barre de progression.

```
i=1
j=0
for p in points_dict:

    if(i<len(points_dict)):
        frPoint = points_dict[str(j)]
        toPoint = points_dict[str(i)]
        attrs = [j]
        new_line = QgsGeometry.fromPolyline([QgsPoint(frPoint), QgsPoint(toPoint)])
        feat = QgsFeature()
        feat.setGeometry(new_line)
        feat.setAttributes(attrs)
        (res, outFeats) = pr.addFeatures([feat])
        lines_layer.commitChanges()
        progress.setValue(i)
    i+=1
    j+=1
    progress.setValue(i)

iface.messageBar().clearWidgets()
```

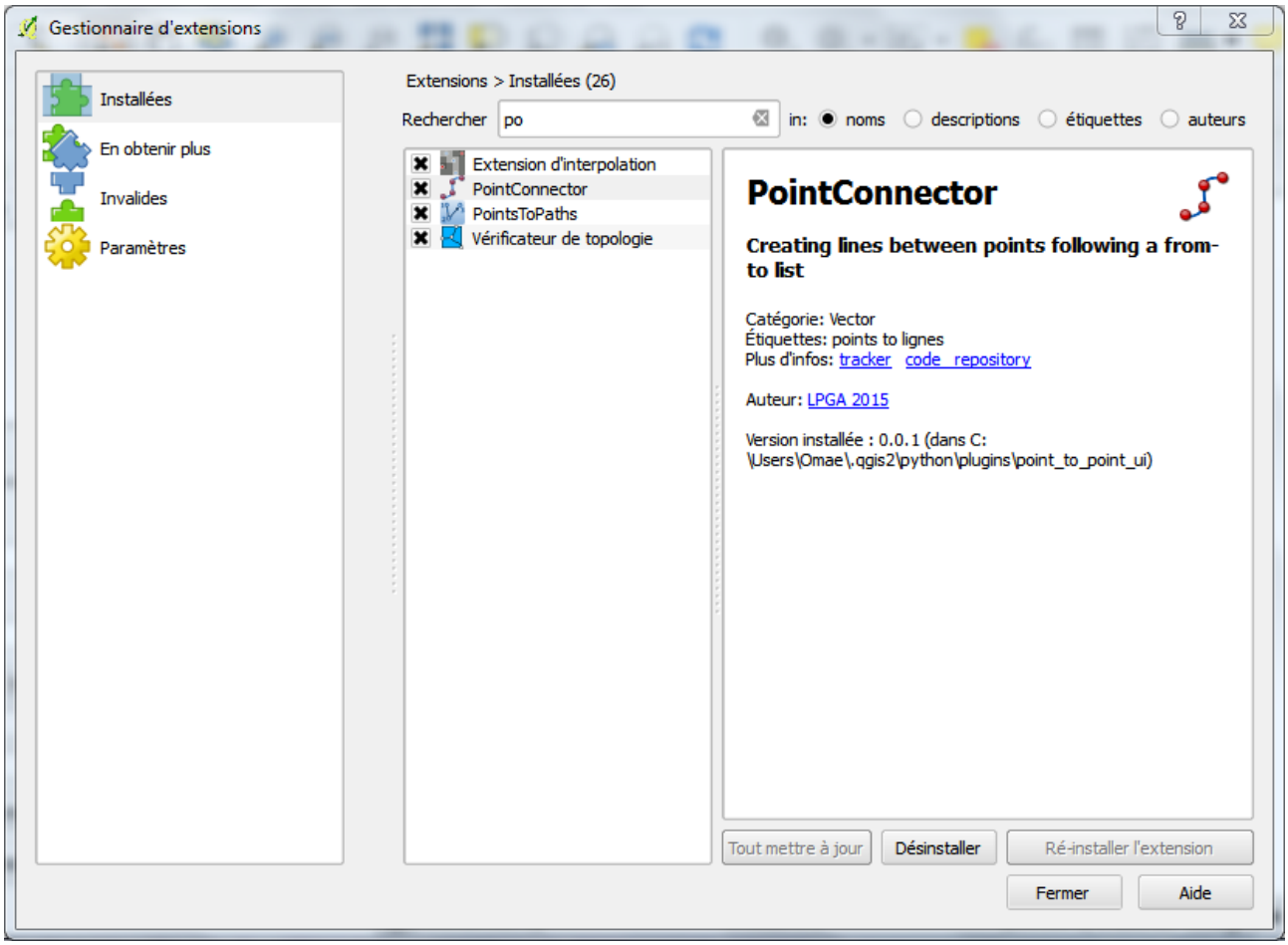
Une fois le traitement est finit on ajoute la couche vectoriel a notre interface Qgis exactement dans la carte et on fait un retour visuel sous forme d'un message comme quoi le traitement est réussi.

```
QgsMapLayerRegistry().instance().addMapLayer(lines_layer)
QMessageBox.information(None, 'Success', 'All lines drawn without error')
```

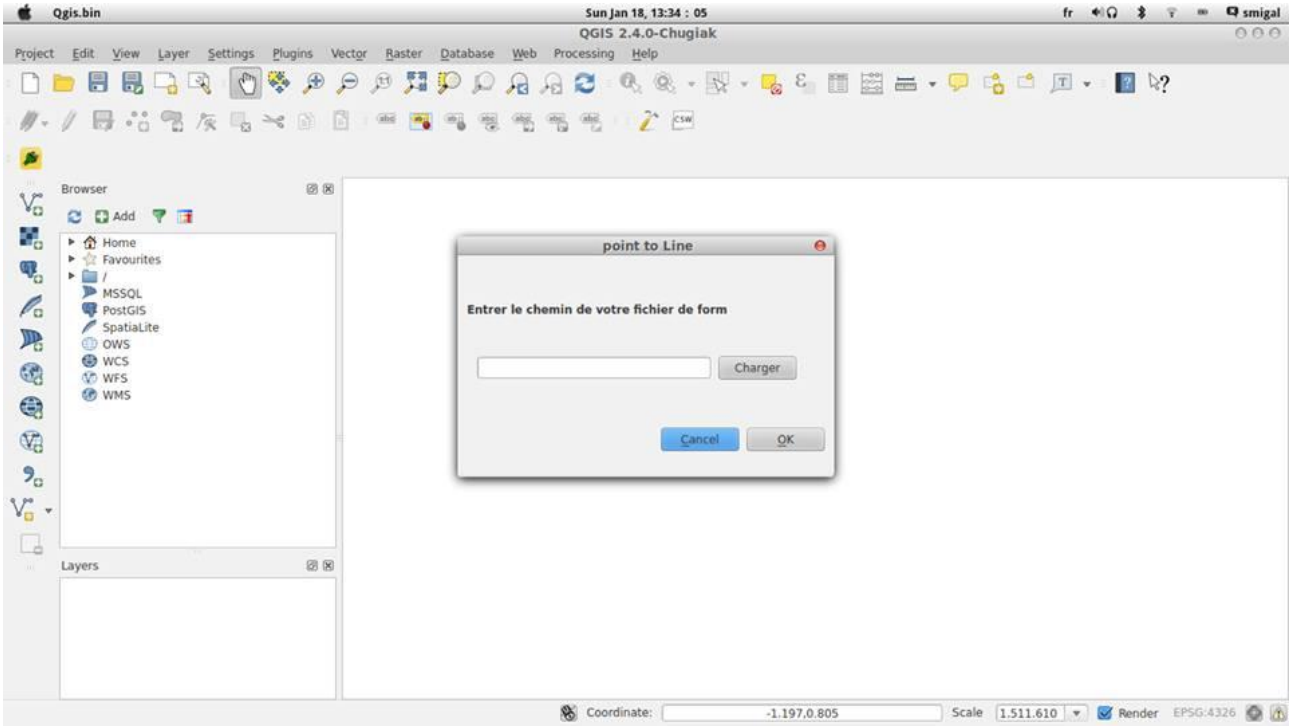
Installation du Plugin :

Le dossier de plugin doit être placé dans le dossier python/plugins de Qgis.

Il suffit de lancer le gestionnaire d'extensions et rechercher votre plugin par son nom dans notre cas c'est pointConnector une fois sélectionner cocher la case à coché pour l'activer.



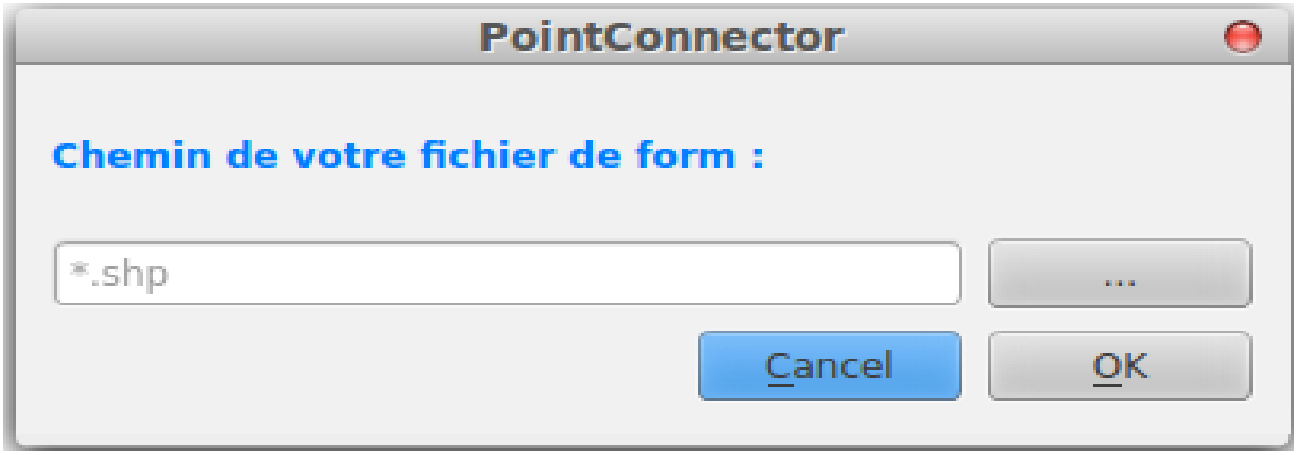
Lancement du plugin :

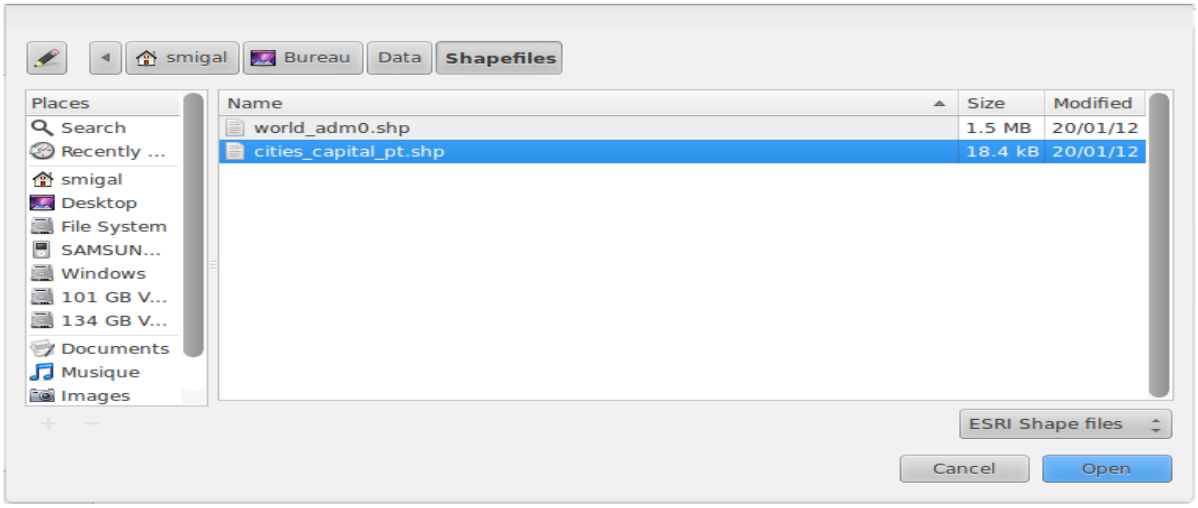


TEST :

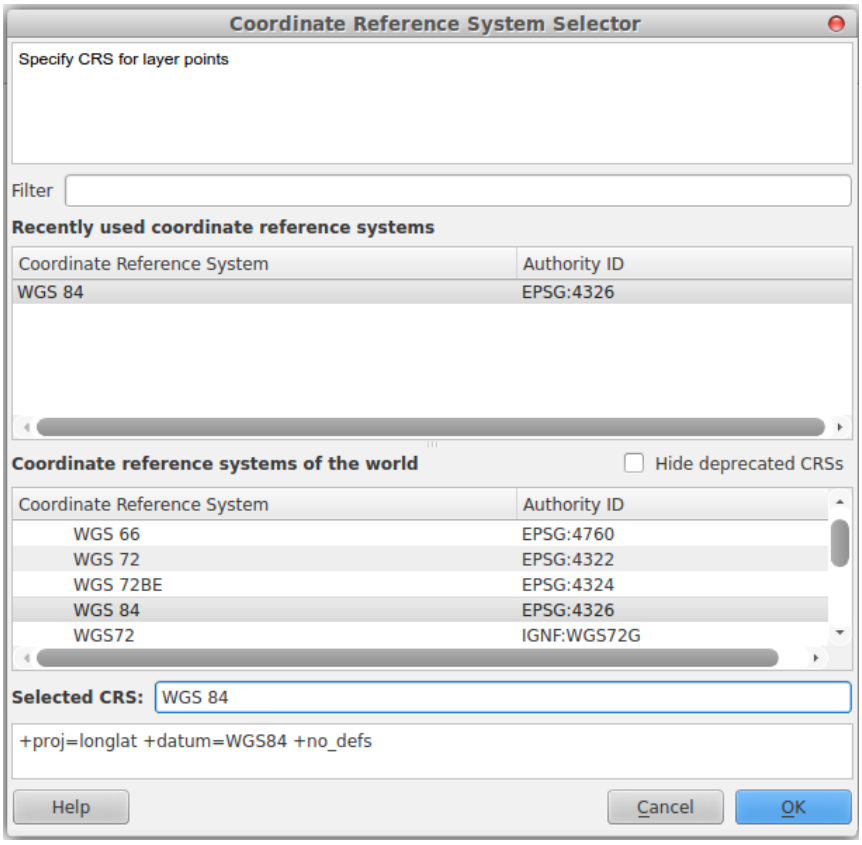
Une fois installée vous aurez l'icône suivant qui s'ajoute à votre interface Qgis

Une fois cliquer dessus, vous aurez le formulaire qui s'afficher cliquer sur le bouton … pour parcourir ou taper le chemin de votre fichier de points.

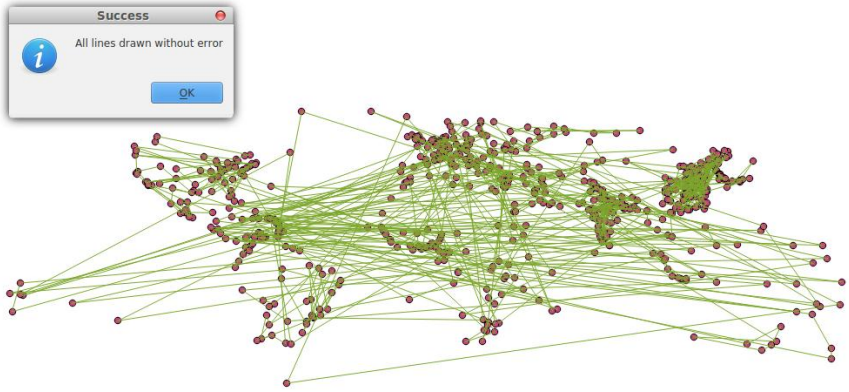




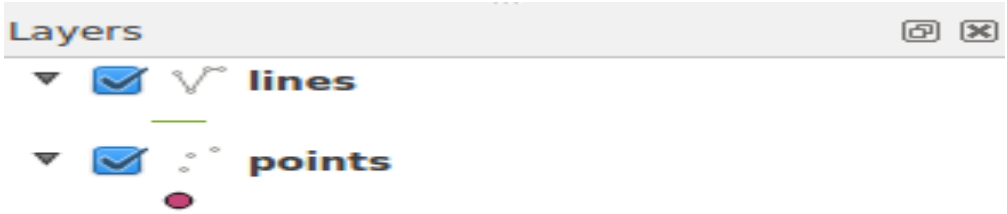
Une fois vous sélectionner votre fichier de points et vous appuie sur ok, vous devez sélectionner le type de projection que vous voulez utiliser.



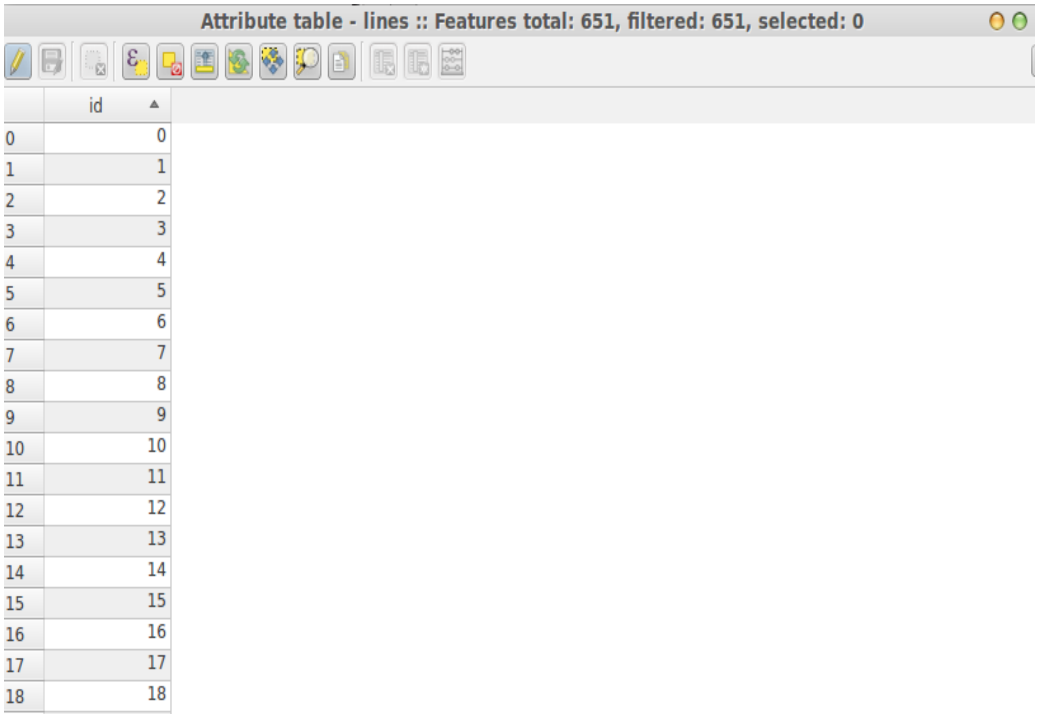
Voici a quoi va ressembler le résultat. Chaque point va être lié avec celle qui la suit en donnant un identifiant pour chaque ligne



Et la couche de ligne va être créée



Avec un identifiant unique pour chaque ligne.

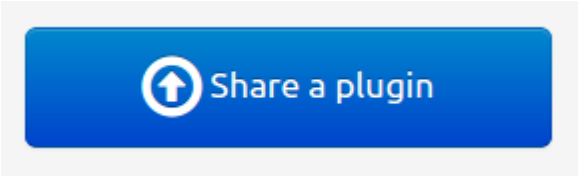


	id
0	0
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9
10	10
11	11
12	12
13	13
14	14
15	15
16	16
17	17
18	18

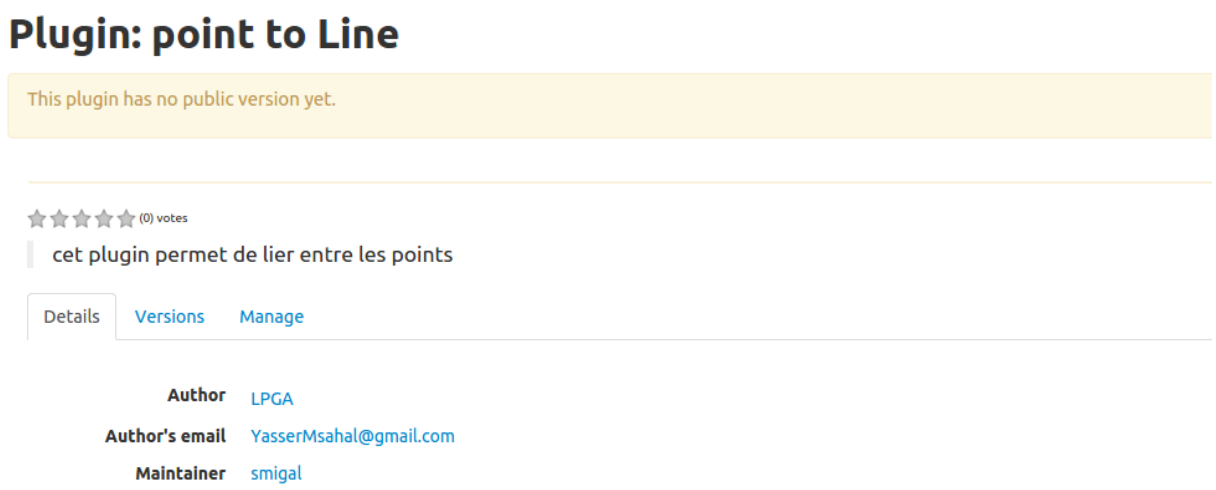
Publication du plugin dans les dépôts de Qgis

Vous allez sur le lien suivant : https://plugins.qgis.org/plugins/

Cliquer sur share à plugin pour partager votre plugin vous vous identifié et charger votre fichier compressé de votre plugin .zip une fois c'est fait



Il suffit d'attendre la validation du modérateur des dépôts pour valider votre plugin



https://github.com/lpga/point_to_point

Conclusion

Ce projet s'est révélé très enrichissant dans la mesure où il a consisté en une approche concrète du métier de géomaticien En effet, la prise d'initiative, le respect de tous les détails Et le travail en équipe sera des aspects essentiels pour notre futur métier.

Géomatique Appliquée

Licence Professionnelle

Université Chouaïb Doukkali

LPGA 2013-2015

Merci

tous les contributeurs