RaFiltre

Le but de ce projet est de creer un outil de filtrage automatique de nuages de points. Il prends en entree des fichiers e57, pts, ptx, las, laz et permet de ressortir des fichiers des meme formats.
Pour faire un filtrage efficace, il y aura 2 etapes. Dans un premier temps, on genere un Octree avec les points issus du nuages, puis on va filtrer blocs par blocs. Ca nous permet d'avoir un filtrage homogene, sans avoir besoin de faire de memory mapping (j'ai pas eu le temps de le faire :().

Format General:

Octhread : Creation de l'arbre
ProceedOcthread : Lire un fichier et creer un Octhread avec
FinishOcthread : Ecrire un fichier a partir de points
Filtering : Passe de filtrage
RaFiltre-Ninja : Utilisation des 4 projets precedent, en plus du GUI afin d'appliquer un filtre sur un fichier et le sauvegarder
OutilsExternes-RafiltreNinja : Outils externe. Pour les petites fonctions sur les fichiers.

Octhread-GUI : Gui pour lancement du filtrage.
OutilsExternes-GUI : Gui pour le lancement des fonctions externes au filtrage

Format du dossier :

- RaFiltre-Ninja.exe *(fichier)*
- datas : *(dossier)*
    - laszip.dll *(fichier)*
    - nepascliquer.exe *(fichier)*
    - PotreeConverter.exe *(fichier)*
    - xerces-c_3_2.dll *(fichier)*
    - OutilsExternes.exe *(fichier)*
    - OutilsExternes-GUI.exe *(fichier)*

Description en details:

Octhread : Le projet (.lib) qui va nous permettre de generer un Octree a partir du nuage de points. Il va creer un nouveau dossier dans lequel on va sauvegarder les feuilles de l'arbre en binaire.
Octree : Fonctions pour la gestion de l'arbre global et toutes ses informations.
Node : Fonctions pour la gestion des noeuds de l'arbre.
MyFile : Fonctions qui vont gerer l'input et l'output de fichier de nuage de points binaire. On est oblige de passer par la, car les nuages de points sont trop gros pour la RAM, et pour eviter les stacks overflow, on doit sauvegarder les points sur le disque. Or, ecrire et lire dans un fichier ASCII c'est tres long. Ici, on va les lire bloc par bloc c'est plus rapide. *base.hpp* : Contient le format de points qu'on va utiliser avec PCL

ProceedOcthread : Le projet (.lib) qui va nous permettre de lire les fichiers et de generer des octree avec.
*E57** : C'est tous les fichiers qui vont etre utile pour lire et ecrire des fichiers E57 (c'est la misere)
OpenableFile : La classe mere pour l'ouverture des fichiers. Les fichier de lecture vont devoir derives de cette classe.
e57File : Pour ouvrir des fichiers e57
LASfile : Pour ouvrir des fichiers LAS et LAZ

PTSfile : Pour ouvrir des fichiers PTS
OpenFactor.hpp : Gerer l'ouverture des differents fichier.

Filtering : Le projet (.lib) qui va nous permettre de filtrer un Octree.
Filtering : La classe qui fait la liaison entre les filtre et le nuage de points en octree et la sauvegarde. C'est ici que beaucoup de choses vont se passer.
Filtre : Contient tous les filtres disponibles
FiltersParam.hpp : Les parametres qui vont etre utilises lors du lancement du programme. Puisuqe le GUI a ete fait en C# et le programme en C++, j'ai fait comme ca. Ca permet de reunir tous les parametres qui ont ete utilises. (c'est vraiment pas terrible...).

FinishOcthread : Le projet (.lib) qui va nous permettre de sauvegarder les fichiers.
*E57** : Ils sont de nouveau la, parceque y'a un probleme avec les libs si jamais ils sont pas present... J'ai jamais trouve pourquoi...
SavableFile : La classe mere pour l'ecriture des fichier. Toutes les classes d'ecriture de fichier vont devoir derive de cette classe.
SaveFactor.hpp : Voir OpenFactor.hpp mais pour la sauvegarde. *SaveE57* : Pour ecrire des fichiers E57
SaveLAS : Pour ecrire des fichiers LAS/LAZ
SavePTX : Pour ecrire des fichiers PTX
SavePTS : Pour ecrire des fichiers PTS

RaFiltre-Ninja : Le projet (.exe) principal. C'est lui qui va lancer toutes les commandes en utilisant les autres projets. *MultiScan* : Enorme erreur de ma part que j'ai fait trop tard et que j'ai pas eu le temps de corriger... . Je pensais qu'on ne pourrait filtrer que vers des fichiers unifie (qui ne possedent qu'un seul scan). Mais il s'est trouve qu'on peut faire des fichiers contenant plusieurs scans vers des fichiers qui contiennent plusieurs scans. Cette classe est la pour gerer ce cas : Ouvrir des fichiers e57 non unifies et sortir des fichiers e57 non unifies ou des PTX non unifies.
main : Le lancement du programme principale avec tous les parametres.

OutilsExternes-RafiltreNinja : Le projet (.exe) qui contient les fonctions qui ne sont "pas utiles" dans le programme de base:
CutIntoParts : C'est la classe qui va nous permettre de creer un decoupage du nuage de points en fonctions de valeurs definies. Par exemple 10 10 10 coupera le nuage en "petit nuages" de tailles 10m*10m*10m. Ou en fonction d'un fichier donner en entree.
Optimizer : Permet d'optimiser un fichier e57. Mais on en a plus besoin, puisque la routine est deja applique a tout les fichiers e57.
FichierSansPoint : Permet de creer un fichier e57 sans points (enfin, avec 100 points, sinon Cyclone n'accepte pas). Juste pour recuperer la position des scans.
Unique/multipleFiles : Des fonctions diverse sur un ou plusieurs fichiers. Pas trop utilise, sauf la fonction qui va chercher le nombre des points d'un fichier et le nombre de scan qui le compose.

Octhread-GUI : : Le projet (.exe) qui va lancer le GUI pour lancer le rafiltre. Lorsqu'on va cliquer sur OK, il va call ./RafiltreNinja [args 1] [args 2], ..., [args n] (rafiltreninja.exe se nomme "nepascliquer.exe")

OutilsExternes-GUI : Le projet (.exe) qui va lancer le GUI pour lancer les fonctions externes au rafiltre.