Travailler avec les points

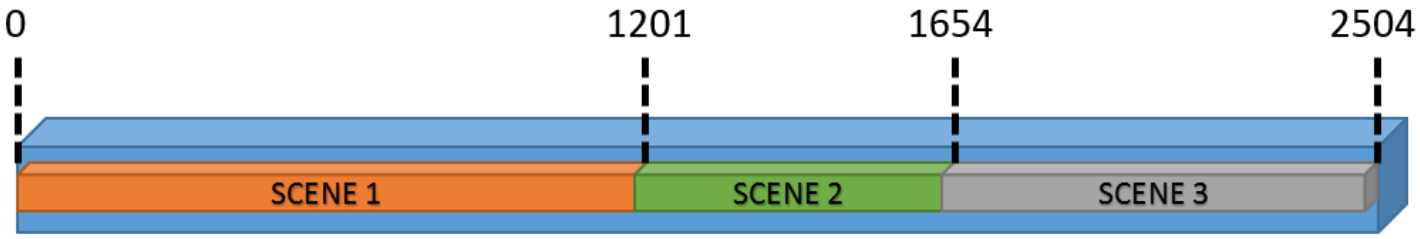
Cette section vous explique en détails comment :

- sont stocké les points
- créer un nouveau nuage de points sans perdre de précision à l'affichage
- parcourir un nuage de points à partir d'un nuage d'index
- obtenir un point à partir de son index global

Le système de nuage de points global et les indices de points

Lors du lancement de l'application le système crée un vecteur de points qui contiendra l'ensemble des points créés dans vos étapes / readers.

Voici par exemple le nuage de point global qui contient les points de 3 scènes différentes (créées à partir de 3 étapes différentes) :



La classe **CT_Scene** ne retient qu'une liste de tous les indices des points créés (classe **CT_NMPCIR**). Par exemple :

- La scène 1 a créé 1201 points et contient un **CT_NMPCIR** qui est un nuage d'index non modifiable contenant les indices des points [0;1200]
- La scène 2 a créé 453 points et contient un **CT_NMPCIR** qui est un nuage d'index non modifiable contenant les indices des points [1201;1653]
- La scène 3 a créé 850 points et contient un **CT_NMPCIR** qui est un nuage d'index non modifiable contenant les indices des points [1654;2503]

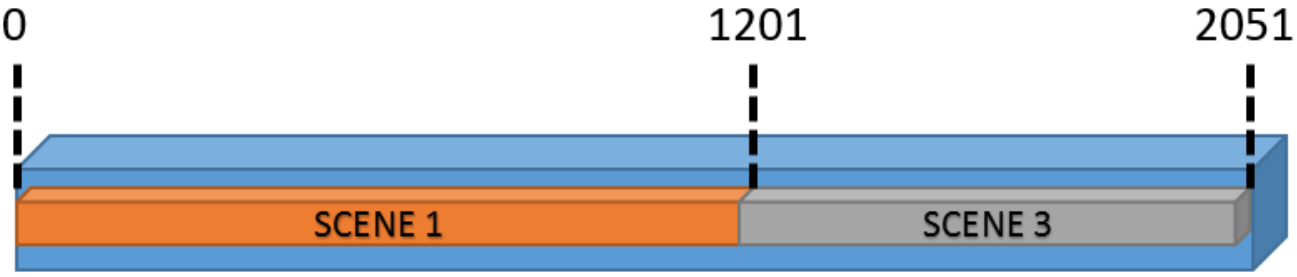
Le nuage de points global contient 2504 points au total.

CT_NMPCIR (Not Modifiable Point Cloud Index Registered) est une classe qui contient les indices des points créés et informent le système lorsqu'elle est supprimé de la mémoire. Ainsi le système peut supprimer les points créés et synchroniser tous les autres indices si cet élément n'est plus utilisé.

La synchronisation des indices de points

En utilisant ce système il a fallu mettre en place un élément qui va synchroniser les nuages d'indices si certains points sont supprimés du nuage de points global.

Par exemple si la scène 2 est supprimée de la mémoire il va falloir synchroniser les indices de la scène 3 puisqu'il ne correspondront plus avec les bon points. Pour cela le système va décaler tous les indices de la scène 3 automatiquement lorsque la scène 2 est supprimée. Vous n'avez donc pas à vous préoccuper de cet aspect puisqu'il est gérer automatiquement et est totalement transparent.



Le décalage des indices des points est transparent pour le développeur car il est géré automatiquement par le système

Les points et les systèmes de coordonnées

Le point utilisé en interne

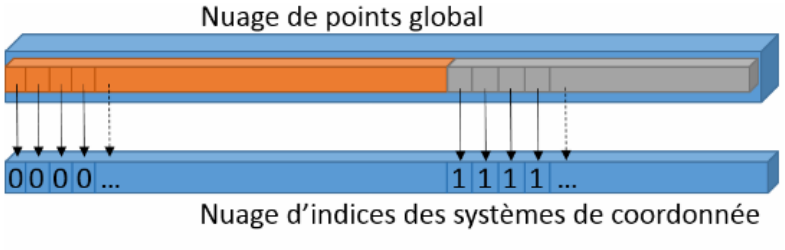
En interne le système stocke les points dans le nuage de points global en utilisant le type **CT_PointData** qui est une classe dérivant de **Eigen::Vector3f**.

En interne les points sont stockés en utilisant des **float** pour gagner de l'espace mémoire et pour la visualisation avec OpenGL.

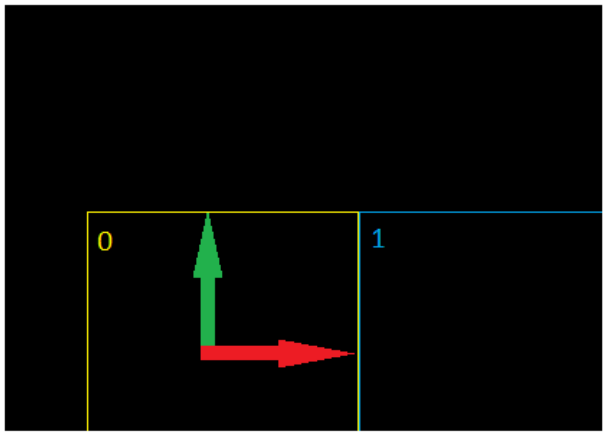
En effet **OpenGL n'utilise pas le type double** pour dessiner des points. De ce fait, en considérant une précision au mm, les coordonnées peuvent aller de **-16 777, 216 m à 16 777, 216 m**. Ce qui signifie qu'on ne peut avoir des scènes excédant **32 km de diamètre** à cette précision, et que des données en projection géographique dont les coordonnées sont en million de mètres perdent (fortement) en précision. Lors de l'affichage en 3D d'une scène dépassant la précision d'un float les points "sautent" d'un endroit à l'autre puisqu'OpenGL arrondi les valeurs.

Système de coordonnée

Les systèmes de coordonnées sont des éléments qui transforment les points à la lecture et à l'export pour ne pas perdre en précision lors de l'affichage. Ce chapitre n'est qu'à titre d'information puisque l'utilisation des systèmes de coordonnées est totalement transparent pour le développeur.



Liste des systèmes de coordonnée
0. Système de coordonnée [0; 0; 0]
1. Système de coordonnée [33554.432; 0; 0]
....



L'image ci-dessus présente comment les systèmes de coordonnées sont stockés/utilisés.

- L'espace 3D est découpé en cube suivant la précision que vous désiré (mm, 1/10mm, etc...). Pour une précision au mm les cubes font 33554.432 mètres de large (la distance maximum entre deux points d'un nuage pour avoir une précision au mm lors de l'affichage).
- Lors du chargement d'un point on calcul dans quel cube il se trouve et si un système de coordonnée y est associé, si non on le crée. Une liste de système de coordonnée est alors constituée. Cette liste contiendra toujours le système de coordonnée [0;0;0] à l'indice 0.
- Un nuage d'indice est synchronisé avec le nuage de points global. Ce nuage contient pour chaque point du nuage global l'indice du système de coordonnée utilisé.

A l'affichage, lors des traitements ou lors de l'exportation les points sont convertis (float → double) automatiquement en utilisant les systèmes de coordonnées pour que la précision ne soit pas perdu.

Les systèmes de coordonnées sont créés automatiquement au chargement des points. Ils sont transparents pour le développeur.

Le point utilisé par les développeurs

Afin de rendre transparent la conversion d'un point interne vers sa représentation en double nous avons créé la classe **CT_Point** qui dérive de Eigen::Vector3d (double) et qui est renvoyé par les objets permettant d'accéder aux point d'un nuage. Un **CT_PointData** est automatiquement convertit en un **CT_Point** à l'aide de son système de coordonnée assigné. Nous verrons dans un prochain chapitre comment récupérer facilement un **CT_Point**.

Tous les ItemDrawable tel que les cercles/ellipses/rectangles/etc.... utilisent des **double** pour faire leurs calculs et stocker les résultats mais n'utilisent pas de système de coordonnée.

Création d'un nuage de points de taille fixe

Dans cet exemple nous allons voir comment créer un nouveau nuage de points dont la taille est connue par avance.

```
// variable qui va contenir le nombre de points (lu dans le fichier)
size_t size = 0;

// initialisation pour la lecture du fichier
.......

// création du nuage de point. On récupère le nuage d'indice des points créés qu'il faudra donner
à la scène.
CT_NMPCIR pcir = PS_REPOSITORY->createNewPointCloud(size);

// création d'un itérateur qui va permettre de parcourir les points créés et les modifier
CT_MutablePointIterator it(pcir);

// tant qu'il y a des points à parcourir et tant que l'utilisateur n'a pas stoppé le traitement
while(it.hasNext() && !isStopped()) {

      // récupération du point du fichier
      CT_Point p = nextPointInFile();

      // passe au point suivant du nuage
      it.next();


// modification du point courant afin qu'il prenne la nouvelle valeur et utilise le système de coo
rdonnée passés en paramètre
      it.replaceCurrentPoint(p);
}

// création de la scène
CT_Scene *scene = new CT_Scene(....);

// définit les indices des points que la scène doit dessiner
scene->setPointCloudIndexRegistered(pcir);
```

Création d'un nuage de points de taille inconnu

Dans cet exemple nous allons voir comment créer un nouveau nuage de points dont la taille n'est pas connue par avance.

```
// initialisation pour la lecture du fichier
.......

// Demande de création d'un nuage de points dont la taille est inconnu
CT_AbstractUndefinedSizePointCloud *cloud = PS_REPOSITORY->createNewUndefinedSizePointCloud();

// tant qu'il y a des points dans le fichier à parcourir et tant que l'utilisateur n'a pas stoppé
le traitement
while(hasNextPointInFile() && !isStopped()) {

      // récupération du point du fichier
      CT_Point p = nextPointInFile();

      // ajout du point avec l'indice du système de coordonnée qu'il doit utiliser
```

```
	cloud->addPoint(p);
}

// récupération du nuage d'indice des points qu'on vient de créer
CT_NMPCIR pcir = PS_REPOSITORY->registerUndefinedSizePointCloud(cloud);

// création de la scène
CT_Scene *scene = new CT_Scene(....);

// définit les indices des points que la scène doit dessiner
scene->setPointCloudIndexRegistered(pcir);
```

Parcours de points à partir d'un nuage d'indice CT_PointIterator

Lorsque vous récupérer un nuage d'indices d'une scène (par exemple) et que vous souhaitez parcourir les points il vous faut utiliser un itérateur de lecture.

```
// on récupère le nuage d'indices de la scène
CT_PCIR pcir = scene->getPointCloudIndexRegistered();

// création de l'itérateur
CT_PointIterator it(pcir);

// parcours tant qu'il y a des points et tant que l'utilisateur n'a pas stoppé le traitement
while(it.hasNext() && !isStopped()) {

	it.next();

	// récupération du point
	const CT_Point &p = it.currentPoint();

	// on peut si l'on veut récupérer son index dans le nuage global
	size_t globalIndex = it.currentGlobalIndex();

	// traitement
	.......
}
```

Parcours du nuage de points global

Si vous voulez accéder directement à un point à partir de son indice global voici l'exemple qui vous montre comment faire :

```
// on crée un objet qui permet d'accéder au nuage de point global
CT_PointAccessor pAccess;

while(...) {

	// traitement
	.....

	// récupération d'un point à partir de son indice global
	CT_Point point = pAccess.pointAt(globalIndex);
}
```

Files

globalpointcloud.png	10.2 KB	02/13/2015	Krebs Michaël
globalpointcloudmodified.png	7.35 KB	02/13/2015	Krebs Michaël
coordinatesystem.png	38.6 KB	02/23/2015	Krebs Michaël