Types of items available in Computree

fr_FR.png ...version française de cette page

Computree includes items of different types, to manage different types of data necessary for the algorithms developed. A number of standard types of items are available in the **core**, what allow compatibility between plugins.

However, if necessary, a plugin can define specific types of items, used solely by this plugin.

Points managment and corresponding types of items

Computree is designed to handle 3D point clouds. As such points have a specific place in the platform. And points #□□ □□□□Point#□□ are not items to optimize the memory management.

Classes in Computree are prefixed by a code to determine their membership plugin. *CT* Prefix means that the class is implemented in the core of Computree.

A point is simply an array of three decimal numbers (float, 7 significant digits) containing its coordinates (x , y, z). All points created in a session are grouped in a **points repository** and indexed continuously.

To date, the use of float decimal type (for memory optimization purposes) poses problems when point clouds are projected in geographic coordinates such as Lambert 93, due to the large number of significant digits needed. In these cases there will be loss of precision when importing points in **Computree**. At the moment, it is best to work in sensor geometry (scan center (0,0,0)).

It is expected in the short term to integrate a system of coordinates offsets to avoid this problem . This will be particularly necessary for the use of aerial LiDAR data.

To manage / display the points, they are managed within “containing points items”, which inheriting the class *CT_AbstractItemDrawableWithPointCloud*.

There are two types of items like this so far:

- The **scenes** (class *CT_Scene*) designed to manage point clouds of important size, optimizing memory usage of indexes. In the case of a scene, points are added all at once during the creation.
- The **clusters** (class *CT_PointCluster*) are rather adapted to the management of “small” sets of points (even if there is no defined limit to the number of points in a cluster). In this case the items may be added successively during the processing. Moreover, clusters update their centroid (which are displayable) in real-time.x

It is possible to have generic algorithms that can use either any type of derivative *CT_AbstractItemDrawableWithPointCloud* item.

Meshes

On the same model as points, **Computree** can handle meshes. A mesh is constituted of **points**, **faces** and **half edges**. Each of these elements is operated similarly to the point (they are not items). There is therefore a repository of faces and a repository of **half-edges**.

As for points, there is a type of item managing the arrangement of points, faces and half-edges : the **mesh model** #□□ □□□□□ *CT_MeshModel*). It will manage the construction, modification and display of individual elements of the mesh.

Attributes

We discussed about the items for storing 3D information for points, faces and half-edges. However, data associated with these elements, such as intensity, color or normal are frequently used.

So there are types of items responsible for storing the information related to the mentioned elements: the **attributes**.

There is an item type attribute for each crossing **type of data** / **element type**.

To date, **Computree handle** following types of attributes:

- **Scalar Attributes**: A value is attached to each element. The value type (integer, float, boolean) is specified as a *template*. Items scalar attributes belong to classes of items *CT_PointsAttributesScalarTemplated*, *CT_FaceAttributesScalarTemplated* and *CT_EdgeAttributesScalarTemplated* respectively for points, faces and half-edges.

- **Colors**: For each element, a color is defined by 4 integer values, between 0 and 255 (red, green, blue, alpha = transparency). Items color attributes belong to classes of items *CT_PointsAttributesColor*, *CT_FaceAttributesColor* and *CT_EdgeAttributesColor* respectively for points, faces and half-edges.

- **Normals** / **Curvature**: for each element 4 decimal numbers are stored (float): the vector of normal direction (dx, dy, dz) and the curvature. Items **Normals** / **Curvature** attributes belong to classes of items *CT_PointsAttributesNormal*, *CT_FaceAttributesNormal* and *CT_EdgeAttributesNormal* respectively for points, faces and half-edges.

Geometric shapes

Computree offers a variety of types of items to manage 3D geometric shapes on a common logic.

Thus there is a type of item for each geometric shape, inheriting all of the class *CT_AbstractShape*:

- **Circles** (class *CT_Circle*): a center (x, y, x), a normalized direction (x, y , z)
- **Ellipses** (class *CT_Ellipse*): a center (x, y, x), a normalized direction (x , y, z), a small radius and a large radius
- **Cylinders** (class *CT_Cylinder*): a center (x, y, x), a normalized direction (x , y, z), a radius and an height
- **Lines** (class *CT_Line*), which consist of two segments ends (x , y, z)
- **Polygon 2D** (class *CT_Polygon2D*): a list of ordered vertices in horizontal plane

It is expected in the medium term to add other geometric shapes : 3D polygon, rectangle, cube, cone, bezier curves...

Data grids

Computree offers two types of item to manage data in a "grid":

- The **2D grids** (class *CT_Grid2D*) to manage grids in two dimensions or **raster**. These items contain a two-dimensional table, each cell containing a value.
- The **3D grids** (class *CT_Grid3D*) to manage grids in three dimensions or **voxels grid**. These items contain a three-dimensional table, each cell containing a value.

These classes of items are **templates**, so they can be declined for various types of data in the cells.

For example, voxels containing interger would be of class *CT_Grid3D<int>*.

These structures are working in a very similar way. It handle missing values (except in the case of booleans).

Other types of items

There exist more types of items for specific uses:

- The **scanner position** (class *CT_Scanner*) store the position and parameters of a T- Lidar scan.
- The **beam** (class *CT_Beam*) manage a laser beam emitted from a scanner.

Previous page(General Presentation)	Back to summary	Next (Standard Graphic Interface)