Travailler avec un maillage (Mesh)

Dans cette section nous allons voir comment :

- sont stockés les faces et arêtes dans le système
- créer un Mesh
- parcourir les points/faces/arêtes d'un mesh

Veuillez tout d'abord lire la section concernant l'utilisation des points Fr_points

Les nuages global des faces et arêtes

Tout comme les points lors du lancement de l'application un vecteur de faces ainsi qu'un vecteur d'arêtes sont créés afin de contenir les faces/arêtes que vous allez fabriquer dans vos steps/readers. Le mesh ne contiendra que les indices des points, faces et arêtes créés (tout comme une scène) et vous pourrez utiliser les itérateurs **CT_PointIterator**, **CT_FaceIterator** et **CT_EdgeIterator** pour parcourir les éléments du mesh.

Création d'un Mesh

L'ItemDrawable à utiliser pour créer un mesh est un **CT_MeshModel**. Celui-ci utilise un **CT_Mesh** qui est la classe contenant les indices des points, faces et arêtes (edge) créé.

Imaginons que vous vouliez créé un carré centré en position [80000, 0, 50000] et de dimension [5, 5]. Celui-ci est constitué de 4 points, 2 faces et 6 arêtes, il nous faudra alors créer un nuage de points, un nuage de faces et un nuage d'arêtes :

```
// création du mesh
CT_Mesh *mesh = new CT_Mesh();

/*****************************/
/********** POINTS ***********/
/*****************************/

// création de 4 points
CT_MutablePointIterator itP = CT_MeshAllocator::AddVertices(mesh, 4);

// passe au premier point
itP.next();

// on récupère son index global
size_t globalIndexFirstPoint = itP.currentGlobalIndex();

// modification du point 1
itP.replaceCurrentPoint(createCtPoint(79997.5, -2.5, 50000));

// modification du point 2 en une seule ligne
itP.next().replaceCurrentPoint(createCtPoint(80002,5, -2.5, 50000));

// modification du point 3 en une seule ligne
itP.next().replaceCurrentPoint(createCtPoint[1];

// modification du point 4 en une seule ligne
itP.next().replaceCurrentPoint(createCtPoint(80002,5, 2.5, 50000));

/*****************************/
/*********** FACES ***********/
/*****************************/

// création de 2 faces
CT_MutableFaceIterator itF = CT_MeshAllocator::AddFaces(mesh, 2);
```

```
// récupère la première face
CT_Face &face1 = itF.next().cT();

// récupère l'index de la face 1
size_t faceIndex = itF.cIndex();

// création de 3 arêtes
CT_MutableEdgeIterator itE = CT_MeshAllocator::AddHEdges(mesh, 3);

// création de variable contenant les indices des arêtes
size_t e1Index = itE.next().cIndex();
size_t e2Index = e1Index + 1;
size_t e3Index = e1Index + 2;

// on passe l'index de la première arête à la face. Une face doit connaitre au moins une arête.
face1.setEdge(e1Index);

/*****************************/
/*********** ARETES ***********/
/******* première partie ******/
/*****************************/

// création de variables contenant les indices des points de la face
size_t p0 = globalIndexFirstPoint;
size_t p1 = p0+1;
size_t p2 = p0+2;

// on récupère l'arête 1
CT_Edge &e1 = itE.next().cT();

// et on lui affecte les indices des points qui la compose
e1.setPoint0(p0);
e1.setPoint1(p1);

// ainsi que l'indice de la face
e1.setFace(faceIndex);

// on récupère l'arête 2
CT_Edge &e2 = itE.next().cT();
e2.setPoint0(p1);
e2.setPoint1(p2);
e2.setFace(faceIndex);

// on récupère l'arête 3
CT_Edge &e3 = itE.next().cT();
e3.setPoint0(p2);
e3.setPoint1(p0);
e3.setFace(faceIndex);

// on définit qui est la précédente, qui est la suivante parmis les arêtes
e1.setNext(e2Index);
e1.setPrevious(e3Index);
e2.setNext(e3Index);
e2.setPrevious(e1Index);
e3.setNext(e1Index);
e3.setPrevious(e2Index);

/*****************************/
/*********** ARETES ***********/
/******* deuxième partie ******/
/*****************************/

// création de 3 nouvelles arêtes
itE = CT_MeshAllocator::AddHEdges(mesh, 3);

// création de variable contenant les indices des arêtes
e1Index = itE.next().cIndex();
```

```
e2Index = e1Index + 1;
e3Index = e1Index + 2;

// récupère la deuxième face
CT_Face &face2 = itF.next().cT();

// récupère l'index de la face 2
faceIndex = itF.cIndex();

// on passe l'index de la première arête à la face. Une face doit connaitre au moins une arête.
face2.setEdge(e1Index);

// création de variables contenant les indices des points de la face
p0 = globalIndexFirstPoint+3;
p1 = p0+1;
p2 = p0+2;

// on récupère l'arête 1
CT_Edge &e12 = itE.next().cT();

// et on lui affecte les indices des points qui la compose
e12.setPoint0(p0);
e12.setPoint1(p1);

// ainsi que l'indice de la face
e12.setFace(faceIndex);

// on récupère l'arête 2
CT_Edge &e22 = itE.next().cT();
e22.setPoint0(p1);
e22.setPoint1(p2);
e22.setFace(faceIndex);

// on récupère l'arête 3
CT_Edge &e32 = itE.next().cT();
e32.setPoint0(p2);
e32.setPoint1(p0);
e32.setFace(faceIndex);

// on définit qui est la précédente, qui est la suivante parmis les arêtes
e12.setNext(e2Index);
e12.setPrevious(e3Index);
e22.setNext(e3Index);
e22.setPrevious(e1Index);
e32.setNext(e1Index);
e32.setPrevious(e2Index);

/*****************************/
/******** Finalisation ********/
/*****************************/

// création du CT_MeshModel
CT_MeshModel *meshModel  = new CT_MeshModel(DEF_SearchMesh, out_res, mesh);
```

Parcours des points/faces/arêtes d’un mesh

Pour le parcours il suffit d’utiliser les itérateurs :

```
CT_PointIterator itP(meshModel->getPointCloudIndex());

while(itP.hasNext()) {
      itP.next();
```

```
	CT_Point p = itP.currentPoint();

	// traitement
	....
}

// création d'un objet permettant de récupérer un point à partir de son indice global
CT_PointAccessor pAccess;

CT_FaceIterator itF(meshModel->getFaceCloudIndex());

while(itF.hasNext()) {
	itF.next();
	const CT_Face &f = itF.cT();

// on demande à la face l'indice global du point 0 puis on demande au nuage global le point à cet indice
	CT_Point p0 = pAccess.pointAt( f.iPointAt(0) );

	// on peu demander à la face les points de 0 à 2 inclus
	CT_Point p1 = pAccess.pointAt( f.iPointAt(1) );
	CT_Point p2 = pAccess.pointAt( f.iPointAt(2) );

	// traitement
	....
}

CT_EdgeIterator itE(meshModel->getFaceCloudIndex());

while(itE.hasNext()) {
	itE.next();
	const CT_Edge &e = itE.cT();

// on demande à l'arête l'indice global du point 0 puis on demande au nuage global le point à cet indice
	CT_Point p0 = pAccess.pointAt( e.iPointAt(0) );

	// on peu demander à l'arête les points de 0 à 1 inclus
	CT_Point p1 = pAccess.pointAt( e.iPointAt(1) );

	// traitement
	....
}
```

[1] 79997.5, 2.5, 50000