

Travailler avec Open Scene Graph

3. Transformations et États avec Osg::Shape

Traduction par Thomas Baquet



Source: <http://www.openscenegraph.org/projects/osg/wiki/Support/Tutorials/> -by **Joseph Sullivan**

Traduction: Février 2008

Sources: 2003-2006

NdT: Avant de commencer

Il est possible qu'il y ait encore des erreurs de traduction, si c'est le cas, prévenez-moi. Aucune responsabilité ne peut cependant être engagée contre le traducteur ou l'auteur suite à ces erreurs, ou à quelque erreur à la lecture de la traduction.

Il y a eu également eu des mots impossibles à retraduire en les mettant en contexte dans la traduction (je pense à des mots comme « geometry » par exemple qui est employé comme nom simple). Dans ces cas, j'ai soit employé le mot en anglais, soit tourné la phrase de façon à garder le même sens.

En lisant ce document, le lecteur prend pleinement conscience de ce qui vient d'être écrit.

But du tutoriel

Dans ce tutoriel, nous verrons comment construire une scène avec l'aide des instances d'`osg::Shape`. Nous utiliserons également `osg::StateSet` pour contrôler le rendu des formes, ainsi que les noeuds de transformation.

Utiliser la classe « Shape »

La classe `Shape` est la classe de base pour tous les types de forme. Les formes sont utilisées pour le culling et la détection de collisions, ou pour définir les formes géométriques affichables. Les classes suivantes sont héritières de la classe `Shape`:

- `TriangleMesh`
- `Sphere`
- `InfinitePlane`
- `HeightField`
- `Cylinder`
- `Cone`
- `CompositeShape`
- `Box`

Pour utiliser une de ces formes dans notre rendu, nous avons besoin de l'associer à une instance de la classe «`Drawable`». C'est la classe «`ShapeDrawable`» qui nous permettra de le faire. Cette dernière est dérivée de la classe `Drawable` et nous permet d'assigner à une instance `Shape` quelque chose que nous voulons rendre. Comme `ShapeDrawable` dérive de `Drawable`, elle peut être ajoutée à une instance de la classe `Geode`. Le code suivant montre comment faire ceci pour ajouter un cube à une scène vide:

```
// Déclare un groupe servant de noeud principal:
osg::Group* root = new osg::Group();

// Déclare une instance de la classe « box » (dérivée de la classe Shape)
// Ce constructeur prend un argument osg::Vec3 pour définir le centre
// et des nombres flottants pour définir la hauteur, la largeur et la
// profondeur (il est possible de ne donner que les dimensions via un
// autre constructeur).
osg::Box* unitCube = new osg::Box( osg::Vec3(0,0,0), 1.0f);

// Déclare une instance de la ShapeDrawable et l'initialise avec le cube
// que nous venons de créer. Cette classe est dérivée de Drawable, on
// peut donc l'ajouter à une instance Geode.
osg::ShapeDrawable* unitCubeDrawable = new osg::ShapeDrawable(unitCube);

// Déclare une instance de la classe Geode
osg::Geode* basicShapesGeode = new osg::Geode();

// Ajoute unitCubeDrawable à ce dernier...
```

```
basicShapesGeode->addDrawable(unitCubeDrawable);  
  
// Ajoute le Geode à la scène  
root->addChild(basicShapesGeode);
```

Créer une sphère est similaire au code précédent. Sans trop de commentaires, le code ressemble à quelque chose comme ceci:

```
// Crée une sphère centrée sur l'origine des axes, avec une unité de  
// rayon.  
osg::Sphere* unitSphere = new osg::Sphere( osg::Vec3(0,0,0), 1.0);  
osg::ShapeDrawable* unitSphereDrawable = new  
osg::ShapeDrawable(unitSphere);
```

Maintenant nous pouvons ajouter la scène en utilisant un noeud de transformation pour l'éloigner du cube que nous avons ajouté à l'origine. `unitSphereDrawable` ne peut pas être directement ajouté à la scène (par qu'elle n'est pas dérivée de la classe 'node'), nous aurons donc besoin d'un nouveau Geode pour l'ajouter:

```
osg::PositionAttitudeTransform* sphereXForm =  
    new osg::PositionAttitudeTransform();  
sphereXForm->setPosition(osg::Vec3(2.5,0,0));  
  
osg::Geode* unitSphereGeode = new osg::Geode();  
root->addChild(sphereXForm);  
  
sphereXForm->addChild(unitSphereGeode);  
unitSphereGeode->addDrawable(unitSphereDrawable);
```

Régler les états

Le tutoriel précédent nous expliquait comment créer une texture, l'assigner à une image chargée depuis un fichier, et créer un état `StateSet` dans laquelle la texture était activée. Le code suivant utilise deux états – un pour le mode de texture `BLEND` et un pour le mode de texture `DECAL`. D'abord `BLEND`:

```
// Déclare un état pour le mode de texture BLEND
osg::StateSet* blendStateSet = new osg::StateSet();

// Déclare une instance TexEnv auquel on met le mode 'BLEND'
osg::TexEnv* blendTexEnv = new osg::TexEnv();
blendTexEnv->setMode(osg::TexEnv::BLEND);

// Tourne les attributs de la texture 0 – la texture que nous avons
// chargée précédemment – à 'ON'
blendStateSet->setTextureAttributeAndModes
    (0, KLN89FaceTexture, osg::StateAttribute::ON);

// Met la texture d'environnement comme texture 0 dans notre état
blendStateSet->setTextureAttribute(0, blendTexEnv);
```

On répète cette étape pour créer le deuxième état utilisé pour le mode `DECAL`.

```
osg::StateSet* decalStateSet = new osg::StateSet();

osg::TexEnv* decalTexEnv = new osg::TexEnv();
decalTexEnv->setMode(osg::TexEnv::DECAL);

decalStateSet->setTextureAttributeAndModes
    (0, KLN89FaceTexture, osg::StateAttribute::ON);
decalStateSet->setTextureAttribute(0, decalTexEnv);
```

Maintenant que nous avons créé les états, nous pouvons les appliquer aux noeuds de notre scène. Les états sont accumulés durant le dessin de la scène (`root->leaf`). À moins qu'un noeud ait un état qui lui soit assigné, il va hériter de l'état du noeud parent. (Ce qui signifie que si un noeud a plus d'un parent, il sera rendu en utilisant plus d'un état.)¹

```
root->setStateSet(blendStateSet);
unitSphereGeode->setStateSet(decalStateSet);
```

La dernière étape est d'entrer dans la boucle de simulation:

¹ Dans ce code-ci, l'état `BLEND` est associé au noeud principal; ce qui signifie qu'il sera appliqué à tous les enfants. (Et donc dans le code source, à la pyramide...)

```
viewer.setUpViewer(osgProducer::Viewer::STANDARD_SETTINGS);
viewer.setSceneData( root );
viewer.realize();

while( !viewer.done() )
{
    viewer.sync();
    viewer.update();
    viewer.frame();
}
return 0;
```

Quelques notes par rapport au code

Ces notes sont un ajout du traducteur afin d'éclairer certains points importants

- Dans ce code-ci, nous créons un état pour l'appliquer à un noeud à l'aide de l'opérateur "new". Il est cependant préférable d'utiliser la fonction "getOrCreateStateSet();" qui à l'avantage de créer un état s'il n'existe pas pour le noeud, ou bien de retourner celui existant. Il s'utilise de cette façon:

```
osg::StateSet* stateOne = node->getOrCreateStateSet();
```

- Lors de l'utilisation de textures avec des pixels transparents, on peut utiliser une autre fonction pour définir l'état BLEND de façon plus directe que "TexEnv->setMode(osg::TexEnv::DECAL);" (qui -pour rappel- est ensuite lui même associé à l'état):

```
state->setMode(GL_BLEND,osg::StateAttribute::ON);
```