OpenSceneGraph 控制模型

一、简介

对模型的控制就是修改模型的位置和方向属性，使模型的位置和方向发生改变，通常通过移动、旋转、缩放来实现。在三维 ***CAD*** 软件中通常要对模型的位置进行修改，如装配模型时把其中一个零件模型移动一个位置。由计算机图形学知识得三维图形的几何变换可用一个四阶齐次矩阵来表示，即模型的几何变换都是对矩阵进行操作。

二、OSG 模型控制

在 ***OSG*** 中，加入模型的默认位置是屏幕中心，对模型的位置、方向控制是通过类 ***osg::MatrixTransform*** 来实现。由类图知，类 ***osg::MatrixTransform*** 继承自类 ***osg::Transform***，而类 ***osg::Transform*** 是由类 ***osg::Group*** 继承而来。

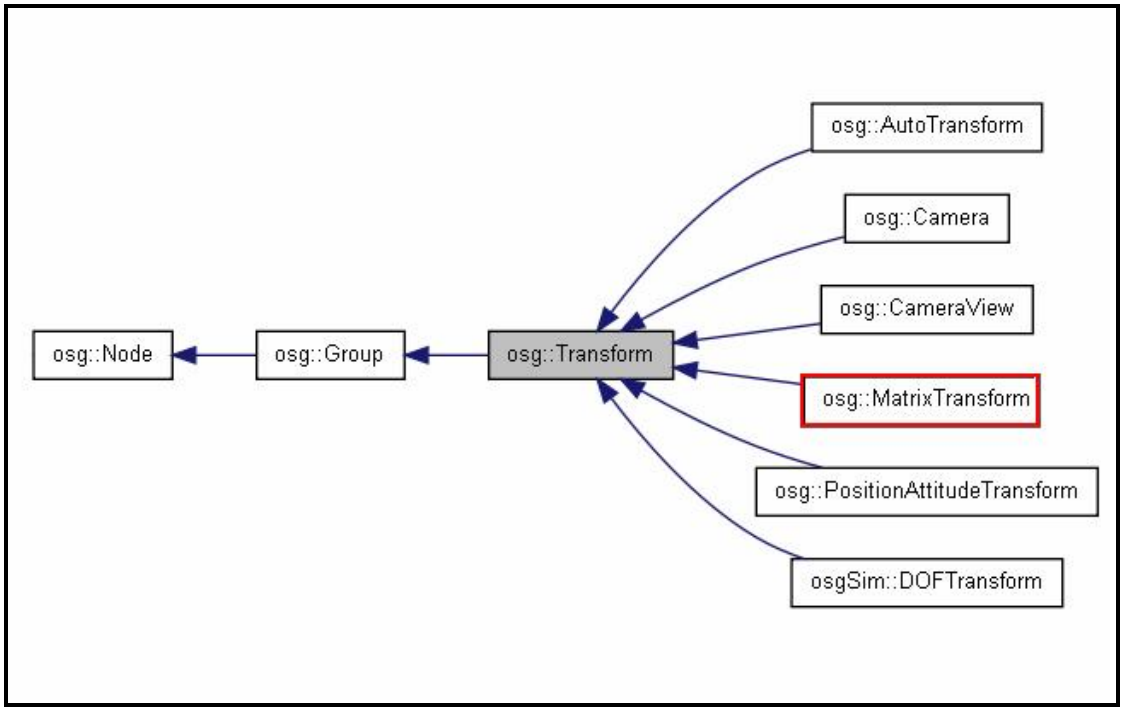

Figure 3.1 Inheritance Diagram for osg::MatrixTransform

声明类 ***osg::MatrixTransform*** 中的注释为：

```
/** MatrixTransform - is a subclass of Transform which has an osg::Matrix
  * which represents a 4x4 transformation of its children from local coordinates
  * into the Transform's parent coordinates.
*/
```

类 ***osg::MatrixTransform*** 是类 ***osg::Transform*** 的子类，它有一个类 ***osg::Matrix*** 的成员变量 ***_matrix*** 来表示其子节点到其父节点的四阶齐次坐标变换。

声明类 ***osg::Transform*** 中的注释为：

```
/** A Transform is a group node for which all children are transformed by
  * a 4x4 matrix. It is often used for positioning objects within a scene,
  * producing trackball functionality or for animation.
  *
  * Transform itself does not provide set/get functions, only the interface
```

```
 * for defining what the 4x4 transformation is.  Subclasses, such as
 * MatrixTransform and PositionAttitudeTransform support the use of an
 * osg::Matrix or a osg::Vec3/osg::Quat respectively.
 *
 * Note: If the transformation matrix scales the subgraph then the normals
 * of the underlying geometry will need to be renormalized to be unit
 * vectors once more.  This can be done transparently through OpenGL's
 * use of either GL_NORMALIZE and GL_RESCALE_NORMAL modes. For further
 * background reading see the glNormalize documentation in the OpenGL
 * Reference Guide (the blue book). To enable it in the OSG, you simply
 * need to attach a local osg::StateSet to the osg::Transform, and set
 * the appropriate mode to ON via
 *   stateset->setMode(GL_NORMALIZE, osg::StateAttribute::ON);
*/
```

OSG 通过 ***osg::Transform*** 节点类家族来实现几何数据的变换。***osg::Transform*** 类继承自 ***osg::Group*** 类，它可以有多个子节点。但是 ***osg::Transform*** 类是一个无法由程序实例化的虚基类。用户应当使用 ***osg::MatrixTransform*** 或 ***osg::PositionAttitudeTransform*** 来替代它，这两个类均继承自 ***osg::Transform*** 类。根据用户程序的需要，可以使用其中任意一个或者同时使用他们。关于类 ***osg::Transform*** 和 ***osg::MatrixTransform*** 类的更多内容，请参考《***OpenSceneGraph*** 快速入门》书中的组节点一章。

三、OSG 中模型控制实现方法

在 ***OSG*** 中，因为矩阵变换类 ***osg::MatrixTransform*** 继承自 ***osg::Group***，所以矩阵变换类可以当作一个特殊节点加入到场景中，矩阵变换类中也可以加入节点，加入的节点就会被这个矩阵变换类处理，可以对加入的节点模型进行移动、旋转、缩放操作。

编程实现模型控制程序，为了简便起见，模型节点仍然从文件中得到。得到模型节点后，分别对其进行移动、旋转和缩放操作。程序代码如下：

```
//-----------------------------------------------------------------------------
//  Copyright (c) 2012 eryar All Rights Reserved.
//
//      File    : Main.cpp
//      Author  : eryar@163.com
//      Date    : 2012-1-5 21:42
//      Version : 1.0v
//
//  Description : Model transformations: Translate, Rotate, Scale.
//
//=============================================================================

#include <osgDB/ReadFile>
#include <osgViewer/Viewer>
#include <osg/MatrixTransform>
#include <osgViewer/ViewerEventHandlers>
```

```
int main(int argc, char* argv[])
{
    osgViewer::Viewer   viewer;
    viewer.addEventHandler( new osgViewer::WindowSizeHandler);
    viewer.addEventHandler( new osgViewer::StatsHandler);

    osg::ref_ptr<osg::Group> root   = new osg::Group;
    osg::ref_ptr<osg::Node> axes   = osgDB::readNodeFile( "axes.osgt");

    // Translate: Offset along X axis 2 unit;
    osg::ref_ptr<osg::MatrixTransform> mtMove =  new osg::MatrixTransform;
    mtMove->setMatrix(osg::Matrix::translate(-2, 0, 0));
    mtMove->addChild(axes.get());

    // Rotate: Rotate along Z axis about 45 degree then translate along x axis
2 unit.
    osg::ref_ptr<osg::MatrixTransform> mtRotate =  new osg::MatrixTransform;
    mtRotate->setMatrix(osg::Matrix::rotate(
       osg::DegreesToRadians(45.0),osg::Z_AXIS) *
osg::Matrix::translate(2,0,0));
    mtRotate->addChild(axes.get());

    // Scale
    osg::ref_ptr<osg::MatrixTransform> mtScale  =  new osg::MatrixTransform;
    mtScale->setMatrix(osg::Matrix::scale(0.5,0.5,0.5));
    mtScale->addChild(axes.get());

    root->addChild(mtMove);
    root->addChild(mtRotate);
    root->addChild(mtScale);

    viewer.setSceneData(root.get());
    viewer.realize();
    return viewer.run();
}
```

运行效果如下图所示：

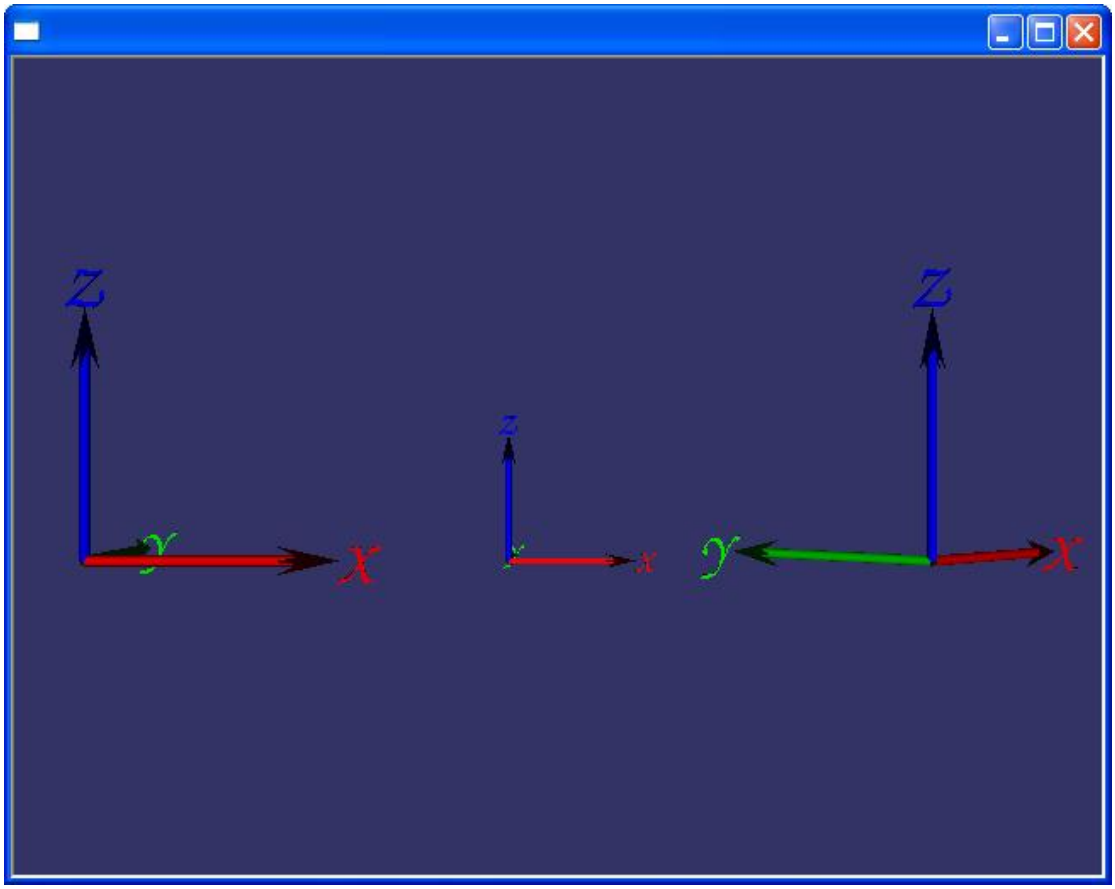

Figure 3.2 Translate, Scale, Rotate Model