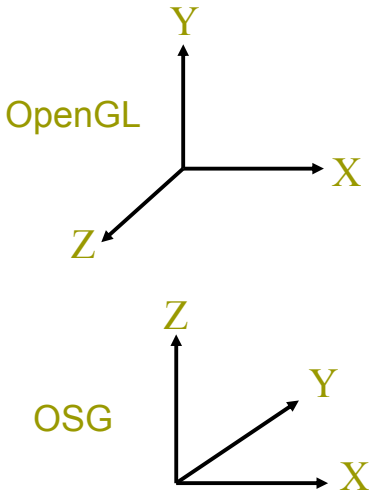
OSG Building a Scene Graph

2008년 여름
박경신

Overview

- OSG Coordinate System
- Geode & Geometry
- Group Nodes
- Rendering States
- File I/O
- NodeKits

OpenSceneGraph's Coordinate System



- OpenGL은 오른손 좌표계 (Right-Handed Coordinate System) x+ 오른쪽. y+ 위쪽. z+ 화면 밖으로 나오는 방향.
- **OSG**은 오른손 좌표계 (Right-Handed Coordinate System) x+ 오른쪽. y+ 화면 안으로 향하는 방향. z+ 위쪽.

Geodes and Geometry

- 다음 예제는 사각형을 그리는 geometry 예제이다.
  - 정점 배열, 법선 벡터, 정점에 대한 색 배열을 생성
  - osg::Geometry 개체를 생성하고 배열을 추가. 또한, 위의 데이터를 어떻게 그릴 것인지 지정하는 osg::DrawArrays 개체를 추가.
  - osg::Geode scene graph node를 생성하고 이 Geometry 개체를 추가

```
#include <osg/Geode>
#include <osg/Geometry>

osg::ref_ptr<osg::Node> CreateSceneGraph()
{
    // Create an object to store geometry in
    osg::ref_ptr<osg::Geometry> geom = new osg::Geometry;
```

Geodes and Geometry

```
// Create an array of four vertices
osg::ref_ptr<osg::Vec3Array> v = new osg::Vec3Array;
geom->setVertexArray(v.get());
v->push_back(osg::Vec3(-1.f, 0.f, -1.f));
v->push_back(osg::Vec3(1.f, 0.f, -1.f));
v->push_back(osg::Vec3(1.f, 0.f, 1.f));
v->push_back(osg::Vec3(-1.f, 0.f, 1.f));

// Create an array of four colors
osg::ref_ptr<osg::Vec4Array> c = new osg::Vec4Array;
geom->setColorArray(c.get());
geom->setColorBinding(osg::Geometry::BIND_PER_VERTEX);
c->push_back(osg::Vec4(1.f, 0.f, 0.f, 1.f));
c->push_back(osg::Vec4(0.f, 1.f, 0.f, 1.f));
c->push_back(osg::Vec4(0.f, 0.f, 1.f, 1.f));
c->push_back(osg::Vec4(1.f, 1.f, 1.f, 1.f));
```


Geodes and Geometry

```
// Create an array for the single normal
osg::ref_ptr<osg::Vec3Array> n = new osg::Vec3Array;
geom->setNormalArray(n.get());
geom->setNormalBinding(osg::Geometry::BIND_OVERALL);
n->push_back(osg::Vec3(0.f, -1.f, 0.f));

// Draw a four-vertex quad from the stored data
geom->addPrimitiveSet(
      new osg::DrawArrays(osg::PrimitiveSet::QUADS, 0, 4));

// Add the Geometry (Drawable) to a Geode and return the Geode
osg::ref_ptr<osg::Geode> geode = new osg::Geode;
geode->addDrawable(geom.get());
return geode.get();
}
```


Geodes and Geometry

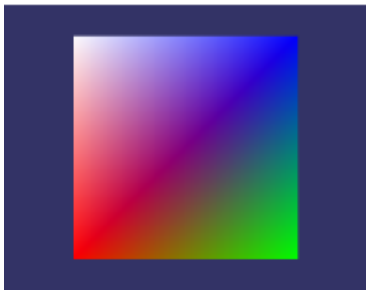
```
#include <osg/ref_ptr>
…

osg::ref_ptr<osg::Node> CreateSceneGraph();

int main( int, char** )
{
    osg::ref_ptr<osg::Node> root = CreateSceneGraph();
    if (!root.valid())
        osg::notify(osg::FATAL) <<
            "Failed in createSceneGraph()." << endl;
    bool result = osgDB::writeNodeFile(*(root.get()), "Simple.osg" );
    if ( !result )
        osg::notify(osg::FATAL) <<
            "Failed in osgDB::writeNodeFile()." << endl;
}
```


Geodes and Geometry

- ~>osgviewer Simple.osg



Geometry Classes

- Vector and Array Classes
  - osg::Vec2, osg:: Vec3, osg:: Vec4
  - osg:: Vec2Array, osg:: Vec3Array, osg:: Vec4Array
- Drawables
  - osg::DrawPixels
  - osg::ShapeDrawable
  - osg::Geometry
- Geodes
  - osg::Geode

Vector and Array Classes

- OSG는 정점 (vertices), 법선벡터 (normals), 색 (colors), 텍스쳐 좌표와 같은 벡터 자료의 저장을 위해 다양한 클래스를 제공하고 있다.
- osg::Vec3
  - 정점, 벡터, 법선 벡터
- osg::Vec4
  - 색
- osg::Vec2
  - 2차원 텍스쳐 좌표
- osg::Vec2Array, osg::Vec3Array, osg::Vec4Array
  - STL's vector class에서 파생된 클래스
  - STL's vector class 의 operator[]() 와 push_back()을 사용

Vector and Array Classes

```
osg::ref_ptr<osg::Vec3Array> v = new osg::Vec3Array;
geom->setVertexArray(v.get());
v->resize(4);
(*v)[0] = osg::Vec3(-1.f, 0.f, -1.f);
(*v)[1] = osg::Vec3(1.f, 0.f, -1.f);
(*v)[2] = osg::Vec3(1.f, 0.f, 1.f);
(*v)[3] = osg::Vec3(-1.f, 0.f, 1.f);
```


Drawables

- osg::Drawable
  - 렌더링 자료를 저장하는 데에 사용하는 virtual base class
  - 파생 클래스로 osg::DrawPixels, osg::ShapeDrawable, osg::Geometry이 있다.
- osg::DrawPixels
  - OpenGL의 glDrawPixels()를 wrapper
- osg::ShapeDrawable
  - 원통 (cylinder)나 구 (sphere)같이 미리 지정된 형태 (predefined shapes)의 사용을 위해 제공
- osg::Geometry
  - 일반적인 기하학적 개체 (geometry) 데이터의 저장 및 렌더링에 사용

osg::Geometry

- osg::Geometry
  - OpenGL의 정점 배열 함수들인, glVertexPointer(), glNormalPointer(), glDrawArrays(), glDrawElements()와 유사
- 사각형을 그리는 예제에서 사용됐던 Geometry 함수들
  - setVertexArray(), setColorArray(), setNormalArray()
    - OpenGL의 glVertexPointer(), glColorPointer(), glNormalPointer()와 유사
    - 정점 (vertex), 색 (color), 법선벡터 (normal vector) 자료를 지정
  - setColorBinding(), setNormalBinding()
    - Geometry에게 어떻게 색과 법선벡터를 그려야 할 지를 지정
    - osg::Geometry::BIND_PER_VERTEX는 각 정점마다 다른 색으로 지정
    - osg::Geometry::BIND_OVERALL는 전체 기하학적 개체 (geometry)에 하나의 법선벡터를 지정

osg::Geometry

- addPrimitiveSet()
  - Geometry에게 자료를 어떻게 렌더링할 지를 지정한다.
  - osg::PrimitiveSet의 포인터를 파라메터로 받는다.
  - osg::PrimitiveSet은 virtual class로 그 자체로 instance할 수 없다.
  - osg::PrimitiveSet의 파생 클래스인 osg::DrawArrays 개체를 넣는다.
  - osg::DrawArrays 클래스는 OpenGL의 glDrawArrays() 의 wrapper로 생각하면 된다.

```
osg::DrawArrays::DrawArrays(GLenum mode, GLint first, GLsizei
  count );
```

```
geom->addPrimitiveSet(
  new osg::DrawArrays(osg::PrimitiveSet::TRIANGLE_STRIP, 0, 6 );
```


Geodes

- osg::Geode
  - osg::Node에서 파생된 클래스
  - 렌더링을 위한 geometry를 저장하는 OSG leaf node
  - 스크린에 렌더링 되는 모든 geometry는 Geode의 addDrawable()을 이용하여 Geode 에 attach 되어야 한다.

Group Nodes

- osg::Group
  - Child nodes를 추가하는데 사용한다.
  - Geode를 제외한 대부분의 OSG nodes는 Group에서 파생되었다.
- osg::Group의 파생 클래스
  - osg::Transform
  - osg::LOD
  - osg::Switch

Group Node's Child Interface

- Group's Child interface
  - Group은 std::vector<ref_ptr<Node>>에 그것의 child를 가리키는 포인터를 저장한다.
  - Child를 index로 access할 수 있다.
  - Child를 add, remove, replace하는 등 다양한 함수를 제공한다.

Group Nodes

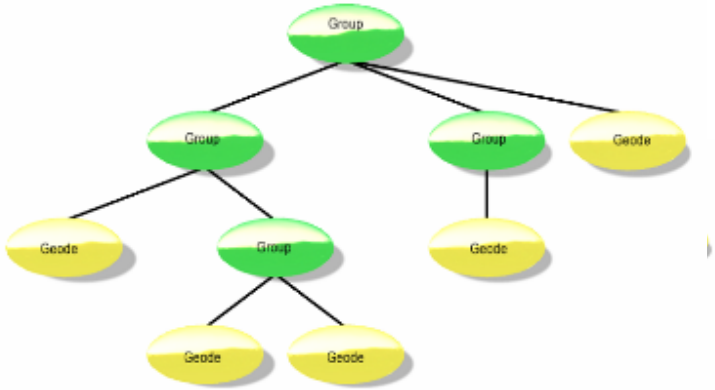


Figure 2-5
The Group node
Shown in green, the **Group** node can have multiple children, which in turn can also be **Group** nodes with their own children.

Group Node's Child Interface

```
class Group : public Node {
public:
    // add a child node
    bool addChild(Node *child);
    // remove a child node. If the node isn't a child, do nothing and return
    // false
    bool removeChild(Node *child);
    // replace a child node with a new child node
    bool replaceChild(Node *origChild, Node *newChild);
    // return the number of children
    unsigned int getNumChildren() const;
    // return true if the specified node is a child node
    bool containsNode(const Node *node) const;
    …
};
```

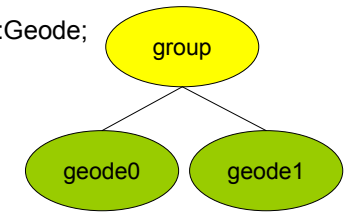

Group Node's Child Interface

- 예제

```
osg::ref_ptr<osg::Group> group = new osg::Group;

osg::ref_ptr<osg::Geode> geode0 = new osg::Geode;
group->addChild(geode0.get());

osg::ref_ptr<osg::Geode> geode1 = new osg::Geode;
group->addChild(geode1.get());
```



Group Node's Parent Interface

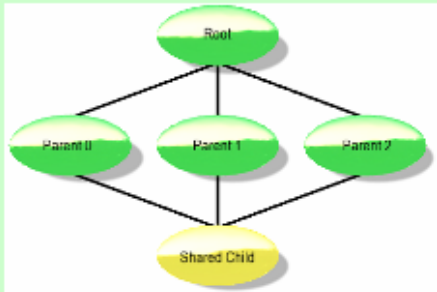
```
class Node : public Object {
public:
    …
    typedef std::vector<Group *> ParentList;
    // return a list of all parents
    const ParentList & getParents() const;
    // return a pointer to the parent with the specified index
    Group * getParent (unsigned int i);
    // return the number of parents
    unsigned int getNumParents() const;
};
```

- 일반적인 경우에, 노드는 하나의 parent (getNumParents() returns 1)를 가지고 있고, 그 parent를 가리키는 포인터를 얻고자 하려면 getParent(0)를 사용한다.

Group Node's Parent Interface

Multiple Parents

When you add the same node as a child to multiple nodes, the child node has multiple parents, as the diagram illustrates. You might want to do this to render the same subgraph many times without creating multiple copies of the subgraph. Section 2.4.3 Example Code for Setting State shows an example of how to render the same **Geode** with different transformations and different rendering state.



When a node has multiple parents, OSG traverses the node multiple times. Each parent keeps its own **ref_ptr<>** to the child, so the child isn't deleted until all parents have stopped referencing it.

Transform Nodes

- osg::Transform
  - Group에서 파생된 클래스 (multiple children을 가질 수 있다)
  - virtual base class로 그냥 instantiate 할 수 없고 osg::MatrixTransform, osg::PositionAttitudeTransform를 사용한다.
- Transform은 reference frame을 지정해서 사용해야 한다.
  - Default reference frame은 osg::Transform::RELATIVE_RF 로 OpenGL의 glRotatef(), glScalef(), glTranslatef()를 겹쳐서 사용할 수 있는 것처럼 사용한다.
  - 만약, OpenGL의 glLoadMatrixf()같이 absolute reference frame을 사용하고자 한다면 osg::Transform::ABSOLUTE_RF를 사용한다.

```
osg::ref_ptr<osg::MatrixTransform> mt =
                                        new osg::MatrixTransform;
mt->setReferenceFrame(osg::Transform::ABSOLUTE_RF);
```


MatrixTransform Node

- MatrixTransform는 osg::Matrix 를 내부적으로 사용한다.
- 이동을 원한다면, translation 행렬을 생성하고 그 행렬을 MatrixTransform에 지정한다.

```
osg::ref_ptr<osg::MatrixTransform> mt = new osg::MatrixTransform;
osg::Matrix m;
m.makeTranslate(x, y, z);
mt->setMatrix(m);
```

- NOTE: osg::Matrix는 osg::Referenced에서 파생되지 않음

PositionAttitudeTransform Node

- PositionAttitudeTransform는 Vec3 position과 quaternion을 사용하여 transformation을 지정할 때 사용한다.

```
// create a quaternion rotated theta radians around axis
float theta(M_PI * .5f);
osg::Vec3 axis(.707f, .707f, 0.f);
osg::Quat q0(theta, axis);

// create a quaternion using yaw/pitch/roll angles
osg::Vec3 yawAxis(0.f, 0.f, 1.f);
osg::Vec3 pitchAxis(1.f, 0.f, 0.f);
osg::Vec3 rollAxis(0.f, 1.f, 0.f);
```


Matrix Transform Node

```
// This example assumes yawRad, pitchRad, rollRad are defined and
// declared as floats externally
osg::Quat q1(yawRad, yawAxis, pitchRad, pitchAxis, rollRad, rollAxis);

// concatenate the two quaternions
q0 *= q1;

// configure a PositionAttitudeTransform by using its setPosition() and
// setAttitude() methods
// (x, y, z, theta, and axis are externally defined and declared)
osg::Vec3 pos(x, y, z);
osg::Quat att(theta, axis);
osg::ref_ptr<osg::PositionAttitudeTransform> pat =
                                        new osg::PositionAttitudeTransform;
pat->setPosition(pos);
pat->setAttitude(att);
```


Matrix Transform and the Order of the Transform

- Vec3 v를 new origin T를 중심으로 rotation R하는 변환의 예

```
osg::Matrix T;
T.makeTranslate(x, y, z);
osg::Matrix R;
R.makeRotate(angle, axis);
Vec3 vPrime = v * R * T;
```

- OpenGL과는 반대로 행렬이 premultiplication 방식으로 계산된다.

  v' = TRv

Matrix Transform and the Order of the Transform

- 전 슬라이드의 예는 다음 diagram과 OpenGL commands와 같다.


```
// 이것은, 다음 OpenGL commands와 같다
glMatrixMode(GL_MODEL_VIEW);
glTranslatef(…) ; // Translation T
glRotatef(…); // Rotation R
  …
glVertex3f(…);
glVertex3f(…);
```

Switch Node

- osg::Switch node 는 선택적인 렌더링 또는 특별한 child node를 빼고자 할 때 사용된다.
- 스위치 노드는 현재 렌더링 상태를 기반으로 특정 children을 그려주거나 또는 게임에서 스크린이나 레벨을 스위치할 때 쓰인다.
- 스위치는Group 노드에서 상속받았으며 child interface를 가지고 있다.
- 각 Switch child node는 관련된 Boolean 값을 가지고 있으며, 이 값이 true가 되면 그린다.
- 다음 예제는 Switch node 생성해서 하나는 보이고 (visible) 다른 하나는 보이지 않는 (not visible) 2개의 Group children을 추가한 예를 보여주고 있다.

Switch Node

```
osg::ref_ptr<osg::Group> group0, group1;
…
// Create a Switch parent node and add two Group children
osg::;ref_ptr<osg::Switch> sw = new osg::Switch;
// Render the first child
sw->addChild(group0.get(), true); // sw->addChild(group0.get());
// Don't render the second child
sw->addChild(group1.get(), false);

// Possible to change the default value for new children
Switch::setNewChildDefaultValue(false);
// the new children will be turned off by default
sw->addChild(group0.get());
sw->addChild(group1.get());
```


Switch Node

```
// After you've added a child to a Switch, you can change its value

// Add a child, initially turned on
sw->addChild(group0.get(), true);

// Disable group0
sw->setChildValue(group0.get(), false);
```


LOD Node

Figure 2-8
The LOD node

- LOD node는 여러 개의 Levels of Detail을 가진 개체를 렌더링할 때 사용한다.
- 각 child마다 최소 최대값을 가진 range를 지정한다.
- LOD node는 viewer와의 거리가 child에 지정되어 있는 range안에 들어오는 child를 그린다.
- LOD children은 어떤 순서로 넣어져 있어도 상관없으며 detail에 따른 거리로 정렬(sort)되어 있지 않아도 된다.

LOD Node

- 다음 LOD 예제는 0부터 1000까지의 range를 가진 Geode child를 추가한 것으로, 이 Parent node는 viewer 까지의 거리가 1000 이내일 때 렌더링한다.

```
osg::ref_ptr<osg::Geode> geode;
…
osg::ref_ptr<osg::LOD> lod = new osg::LOD;
// Display geode when 0.f <= distance < 1000.f
lod->addChild(geode.get(), 0.f, 1000.f);
```


LOD Node

- 만약 이것들의 range에 다 들어오게 되면 여러 개의 children을 동시에 보여주게 된다.

```
osg::ref_ptr<osg::Geode> geode0, geode1;
// Initialize the Geodes
…
osg::ref_ptr<osg::LOD> lod = new osg::LOD;
// Display geode0 when 0.f <= distance < 1050.f
lod->addChild(geode0.get(), 0.f, 1050.f);
// Display geode1 when 950.f <= distance < 2000.f
lod->addChild(geode1.get(), 950.f, 2000.f);
// Result: display geode0 and geode1 when 950.f <= distance < 1050.f
```


LOD Node

- By default, LOD 는 viewer와 개체의 bounding volume의 중앙점까지의 거리를 계산한다.
- 다음은 LOD node에 user-defined center를 사용한 예이다.

```
osg::ref_ptr<osg::LOD> lod = new osg::LOD;
// Use a user-defined center for distance computation
lod->setCenterMode(osg::LOD::USER_DEFINED_CENTER);
// specify the user-defined center x = 10 y = 100
lod->setCenter(osg::Vec3(10.f, 100.f, 0.f));
```

- 다시 bounding sphere center를 사용하는 default 계산으로 돌리고자 한다면, 다음과 같이 사용한다.

```
osg::LOD::setCenterMode(
      osg::LOD::USE_BOUNDING_SPHERE_CENTER);
```