OSG Building a Scene Graph

2008년 여름
박경신

Overview

- Rendering States
  - StateSet
  - Attribute & Modes
  - Texture Mapping
  - Light
  - Materials
- File I/O
- NodeKits
  - Text

Rendering State

- OSG는 대부분의 OpenGL 함수 파이프라인 렌더링 상태를(예, 알파, 블렌딩, 클리핑, 색 마스크, 컬링, 안개, 깊이, 스텐실, 래스터링, 등) 지원한다.
- OSG는 osg::StateSet에서 렌더링 상태를 지정하고 scene graph의 어떤 노드에 attach 할 수 있다.
- OSG 가 scene graph 를 traverse하면서 StateSet은 OpenGL state attribute를 관리한다. 그래서 우리 프로그램이 다른 subgraph에서 다른 상태를 지정할 수 있도록 한다.
- OSG가 각 subgraph 를 traverse하면서 렌더링 상태를 저장(store)하고 복구(restore)할 수 있다.

Rendering State

- OSG는 렌더링 상태를 attributes (fog color and blend functions, …)과 modes 로 정리하고 있다.
- Modes는 glEnable()과 glDisable()로 OpenGL state feature를 지정한 것과 거의 일대일 대응이다.
- State value을 지정하려면,
  - 지정해야 하는 Node나 Drawable의 상태에서 StateSet 를 얻어와야 한다.
  - StateSet 를 불러서 state modes와 attributes를 지정한다.
- Node나 Drawable로부터 StateSet를 얻으려면,

```
// obj is either a Node or a Drawble
osg::StateSet *state = obj->getOrCreateStateSet();
```

Rendering State – Setting Attributes

- getOrCreateStateSet() 함수는 obj's StateSet 포인터를 반환한다. 만약 obj가 StateSet이 없다면, 새로운 StateSet을 생성해서 attach 한다.
- Attribute를 지정하려면,
  - Attribute을 바꿔야하는 클래스를 instantiate 한다.
  - 이 클래스에 attribute 값을 지정하고, osg::StateSet::setAttribute 를 사용하여 StateSet에 attach 한다.

```
// Obtain the StateSet from the geom
osg::StateSet *state = geom->getOrCreateStateSet();
// Create and add the CullFace attribute
Osg::CullFace *cf = new osg::CullFace(osg::CullFace::BACK);
state->setAttribute(cf);
```


Rendering State – Setting a Mode

- osg::StateSet::setMode() 를 사용하여 mode를 enable 또 disable 한다.

```
// Obtain the StateSet
osg::StateSet *state = geom->getOrCreateStateSet();
// Enable fog in this StateSet
// The first parameter to setMode() is any GLenum that is valid
// in a call to glEnable() or glDisable()
// The second parameter can be osg::StateAttribute::ON or
// osg::StateAttribute::OFF
state->setMode(GL_FOG, osg::StateAttribute::ON);
```


Rendering State – Setting Attribute & Mode

- osg::StateSet::setAttributeAndModes() 를 사용하여 attribute과 mode를 동시에 지정할 수 있다.

```
// Create the BlendFunc attribute
osg::BlendFunc *bf = new osg::BlendFunc();
// Attach the BlendFunc attribute and enable blending
// The second parameter to setAttributeAndModes() enables or
// disabled the first parameter's attribute's corresponding mode
state->setAttributeAndModes(bf);
```


Rendering State – State Inheritance

- 노드에 상태가 지정되면, 그 상태가 그 노드와 그 노드의 children에게 적용된다.
- child node 가 같은 상태에 다른 값을 지정하고자 한다면, 그 child 상태 값을 parent state에서 override 해야 한다.
  - 즉, default behavior는 child 노드에서 바꾸지 않는 한 parent state를 inherit 한다.

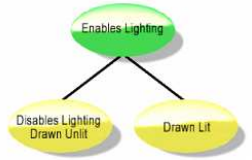

Figure 2-7
State inheritance
In this scene graph, the root node enables lighting. Its first child disables lighting, which overrides the lighting state from its parent. OSG renders the first child with lighting disabled. The second child doesn't change state. As a result, OSG renders the second child using its parent's state, with lighting enabled.

Rendering State – State Inheritance

- OSG는 scene graph의 어느 지점에서 attribute 와 mode에 대한 state inheritance behavior를 개별적으로 고칠 수 있도록 해준다.
- setAttribute(), setMode(), and setAttributeAndModes() 함수의 second parameter에 bitwise OR를 해서 다음 중 하나와 같이 지정할 수 있다.
  - osg::StateAttribute::OVERRIDE
    - attribute 이나 mode를 OVERRIDE로 지정하면, 모든 children이 자신의 state를 바꾸던 아니던 상관없이 attribute와 mode를 상속한다.
  - osg::StateAttribute::PROTECTED
    - Attribute이나 mode를 overriding 으로 부터 보호하려면 PROTECTED를 사용한다.
    - 이것은 parent state가 child state를 절대 override 하지 않는다.
  - osg::StateAttribute::INHERIT
    - 이것은 child state를 무조건 parent로부터 상속받도록 한다. 그 결과, child state을 지우로 parent state 를 사용하게 한다.

Rendering State – State Inheritance

```
// Obtain the root node's StateSet
osg::StateSet *state = root->getOrCreateStateSet();

// Create a PolygonMode attribute
osg::PolygonMode *pm =  new osg::PolygonMode(
                            osg::PolygonMode::FRONT_AND_BACK,
                            osg::PolygonMode::LINE);
// Force wireframe rendering
state->setAttributeAndModes(pm,
            osg::StateAttribute::ON | osg::StateAttribute::OVERRIDE);
```


Texture Mapping

- OSG는 OpenGL texture mapping 을 지원한다.
- Texture mapping을 사용하고자 한다면, 반드시 다음과 같이 지정해야 한다.
  - Geometry에 **textuere coordinates**이 있어야 한다.
  - **Texture attribute** 개체를 생성하고, **texture image** 데이터를 저장한다.
  - **SetState**에 **texture attrubute**과 **mode**를 지정한다.

Texture Coordinates

- OSG의 geometry 개체에 texture coordinates을 지정한다.
- 여러 개의 텍스쳐를 하나의 geometry에 지정하려면 여러 개의 텍스쳐 좌표 (texture coordinates) 배열을 geometry에 attach 한다.

```
// create a geometry object
osg::ref_ptr<osg::Geometry> geom = new osg::Geometry;
// create a Vec2Array of texture coordinates for texture unit 0
// and attach it to the geom
osg::ref_ptr<osg::Vec2Array> tc = new osg::Vec2Array;
geom->setTexCoordArray(0, tc.get());
tc->push_back(osg::Vec2(0.f, 0.f));
tc->push_back(osg::Vec2(1.f, 0.f));
tc->push_back(osg::Vec2(1.f, 1.f));
tc->push_back(osg::Vec2(0.f, 1.f));
```

Loading Images

- 기본적인 2D Texture Mapping을 만들려면 osg::Texture2D와 osg::Image 클래스를 사용한다.

```
osg::StateSet* state = node->getOrCreateStateSet();
// load the texture image
osg::ref_ptr<osg::Image> image = osgDB::readImageFile("sun.tif");
// attach the image in a Texture2D object
osg::ref_ptr<osg::Texture2D> tex = new osg::Texture2D;
tex->setImage(image.get());
// after creating the OpenGL texture object,
// release the internal ref_ptr<Image> (delete the Image)
tex->setUnRefImageDataAfterApply(true);
```


Texture State

- osg::StateSet::setTextureAttribute()은 texture attribute을 지정한다.

```
// create a texture3D attribute
osg::ref_ptr<osg::Texture2D> tex = new osg::Texture2D;
….
// attach the texture attribute for texture unit 0
state->setTextureAttribute(0, tex.get());
```

- osg::StateSet::setTextureMode() 사용하여 texture mode를 지정한다.

```
// disable 2D texture mapping for texture unit 1
state->setTextureMode(1, GL_TEXTURE_2D, osg::StateAttribute::OFF);
// attach 2D texture attribute and enable GL_TEXTURE_2D both on
// texture unit 0
state->setTextureAttributeAndModes(0, tex.get());
```


Lighting

- OSG는 OpenGL lighting (material properties, light properties, lighting models) 을 지원한다.
- OpenGL과 같이, OSG는 light source나 shadows를 렌더링하지 않는다.
- Lighting을 사용하고자 한다면, 반드시 다음과 같이 지정해야 한다.
  - Geometry에는 **normal**이 있어야 한다.
  - **Lighting**을 **enable** (활성화)시키고, **lighting state**을 지정한다.
  - **Light source properties**를 지정하고, 그것을 scene graph에 attach하여 넣는다.
  - Surface **material properties**를 지정한다.

Lighting – Normals

- OpenGL같이 normals은 반드시 unit vector이어야 한다.
- 그러나, scale 변환은 normals의 길이를 변형시킨다.
- Normal을 unit vector로 지정하는 가장 효과적인 해법은 **StateSet 에서 normal rescaling을 활성화하는 것이다.**

```
osg::StateSet *state = geode->getOrCreateStateSet();
state->setMode(GL_RESCALE_NORMAL, osg::StateAttribute::ON);
```

- 이것은 uniform scaling 에 의해 영향을 받으면 normals 을 unit vector로 복원시켜주는 기능이다.

Lighting – Normals

- 전체 scene graph에 전역적으로 normalization 을 활성화하려면 다음과 같이 한다.

```
osg::StateSet *state = geode->getOrCreateStateSet();
state->setModel(GL_NORMALIZE, osg::StateAttribute::ON);
```

- NOTE: normal rescaling 보다 렌더링 효과를 떨어뜨린다.

Lighting State

- Light과 light source를 활성화(enable) 한다.
- osgViewer는 root node의 state set에 light을 지정한다.
- 다음은 lighting enable하고 root 노드의 StateSet에 2개 light sources (GL_LIGHT0 & GL_LIGHT1) 지정하는 예제이다.

```
osg::StateSet *state = root->getOrCreateStateSet();
state->setMode(GL_LIGHTING, osg::StateAttribute::ON);
state->setMode(GL_LIGHT0, osg::StateAttribute::ON);
state->setMode(GL_LIGHT1, osg::StateAttribute::ON);
```


Light Sources

- OSG는 동시에 8개의 light source (GL_LIGHT0 ~ GL_LIGHT7)를 지정할 수 있다. OpenGL implementation에 따라 좀더 추가적으로 light source를 지정할 수도 있다.
- Scene graph 에 light source를 추가하려면
  - osg::Light 개체를 생성하고 light source parameters를 정의한다.
  - osg::LightSource 를 Light에 추가한다.
  - Scene graph 에 LightSource를 추가한다.
- LightSource는 하나의 light definition을 가진 leaf node로 전체 신그라프에 영향을 미친다.
- LightSource는 group 노드의 파생 노드이므로, light에 geometry를 붙히는 것도 가능하다 – 그렇게 해서 light을 표시할 수 있다.

Light Sources

- OpenGL light source 에 light 을 연결짓기 위해 light 의 숫자를 지정한다.
- Default상태에 light number는 0이다.
- 다음 예제는 GL_LIGHT2에 light 개체를 연결한 것을 보여준다.

```
// Create a Light object to control GL_LIGHT2's parameters
osg::ref_ptr<osg::Light> light = new osg::Light;
light->setLightNum(2);
```

Light Sources

- Light 클래스는 OpenGL의 glLight()에서 볼 수 있는 기능들을 제공한다.
- Light 클래스 함수로 light의 ambient, diffuse, specular 색을 지정할 수 있다; point light, directional light, spot lights을 지정할 수 있다; light의 거리에 따라 밝기 감쇠 (lihgt attenuation)을 지정할 수 있다.

```
// create a white spot light source
osg::ref_ptr<osg::Light> light = new osg::Light;
light->setAmbient(osg::Vec4(.1f, .1f, .1f, 1.f));
light->setDiffuse(osg::Vec4(.8f, .8f, .8f, 1.f));
light->setSpecular(osg::Vec4(.8f, .8f, .8f, 1.f));
light->setPosition(osg::Vec3(0.f, 0.f, 0.f));
light->setDirection(osg::Vec3(1.f, 0.f, 0.f));
light->setSpotCutoff(25.f);
```


Light Sources

- OSG는 LightSource 노드에 변환행렬을 적용해서 light의 위치를 변환할 수 있다.
  - 일반적으로 LightSource 를 MatrixTransform의 child로 attach하고 MatrixTransform으로 light 의 위치를 제어한다.

```
// Create the Light and set its properties
osg::ref_ptr<osg::Light> light = new osg::Light;
…
// Create a MatrixTransform to position the Light
osg::ref_ptr<osg::MatrixTransform> mt = new osg::MatrixTransform;
osg::Matrix m;
m.makeTranslate(osg::Vec3(-3.f, 2.f, 5.f));
mt->setMatrix(m);
```


Light Sources

```
// Add the Light to a LightSource. Add the LightSource and
// MatrixTransform to the scene graph
osg::ref_ptr<osg::LightSource> ls = new osg::LightSource;
parent->addChild(mt.get());
mt->addChild(ls.get());
ls->setLight(light.get());
```

- By default, OSG는 LightSource 에 현재 변환행렬에 의하여 light 위치를 변환한다.
- 다음 예제는 LightSource 변환을 무시하고 light 위치를 절대값에 의거하여 적용되도록 하는 예이다.

```
osg::ref_ptr<osg::LightSource> ls = new osg::LightSource;
ls->setReferenceFrame(osg::LightSource::ABSOLUTE_RF);
```


Light Material Properties

- osg::MaterialStateAttribute은 OpenGL의 glMaterial() 또는 glColorMaterial() 기능을 제공한다.
  - Ambient, diffuse, specular, emissive material colors
  - Specular exponent (or shininess)
- Material properties를 지정하려면
  - Material 개체를 생성한다.
  - Colors와 그 외의 파라메타를 지정한다.
  - Scene graph 에 StateSet 을 attach 한다.

Light Material Properties

- Material properties를 지정하는 예제
  - Shininess (specular exponent)는 반드시 1.0 ~ 128.0 사이 값을 사용하도록 한다.

```
osg::StateSet *state = node->getOrCreateStateSet();
osg::ref_prt<osg::Material> mat = new osg::Material;
mat->setDiffuse(osg::Material::FRONT, osg::Vec4(.2f, .9f, .9f, 1.f));
mat->setSpecular(osg::Material::FRONT, osg::Vec4(1.f, 1.f, 1.f, 1.f));
mat->setShininess(osg::Material::FRONT, 96.f);
state->setAttribute(mat.get());
```


FileIO

- Reading Files
  - osgDB::readNodeFile는 3차원 모델 파일을 읽어들여서 Node 개체로 반환한다.
  - osgDB::readImageFile은 2차원 이미지 파일을 읽어들여서 Image 개체로 반환한다.
  - osgDB::Registry::getDataFilePathList() 함수를 사용하여 데이터 파일 디렉토리를 추가한다.
- Write a Scene into a File
  - osgDB::writeNodeFile는 Node 데이터를 3차원 모델 파일로 저장한다.
  - osgDB::writeImageFile는 Image 데이터를 2차원 이미지 파일로 저장한다.

Text Output with OSG—osgText NodeKit

- OSG는 core OSG를 확장인 nodekits 를 제공하고 있다.
- osgText nodekit를 사용한 texture mapped text를 그릴 수 있다.
- osgText Components
  - osgText namespace를 정의한다.
  - 이 namespace에서, 폰트 로딩이나 텍스트 스트링 렌더링 등 관련된 클래스를 제공한다.
  - osgText::Text 클래스가 이중에서 주요 콤포넌트이다.
  - osgText::Font 클래스는 font filename으로 폰트를 생성해준다.
  - Font는 FreeType plugin을 사용하여 font file을 로딩한다.

Using osgText

- osgText/Font와 osgText/Text header files이 필요하다.
- osgText를 사용하기 위해서, 다음 3 스텝을 진행해야 한다.
  - 같은 폰트를 사용해서 여러 개의 텍스트 스트링을 그리기 위해서는, 모든 텍스트 개체들과 공유하는 하나의 Font 개체를 생성한다.
  - 그리는 텍스트 스트링 하나마다, Text 개체를 생성한다. 텍스트의 정렬 (alignment), 방향 (orientation), 위치 (position), 크기 (size) 등의 옵션을 지정한다. 스텝 1에서 생성한 폰트 개체에 새로운 폰트 개체를 지정한다.
  - addDrawable()을 사용해서 Geode에 텍스트 개체를 추가한다. 여러 개의 텍스트 개체를 하나의 Geode에 추가하거나 또는 여러 개의 Geode 개체를 생성할 수 있다. Scene graph에 Geode 개체를 child node로 추가한다.

Using osgText

- 다음은 Font object을 Courier New TrueType font file인 cour.ttf 를 생성하는 예를 보여주고 있다.

```
osg::ref_ptr<osgText::Font> font =
                        osgText::readFontFile("fonts/cour.ttf");
```

- osgText::osgReadFontFile는 폰트 파일을 로딩하기 위해 FreeType OSG plugin를 사용한다.
- osgReadFontFile() 은 OSG_FILE_PATH 에 지정되어 있는 디렉토리에 플랫폼에 맞는 여러 폰트를 찾는다.
- readFromFile()이 지정한 파일을 못 찾거나 파일이 폰트가 아니면, NULL을 반환한다.

Using osgText

- 다음과 같이 Text object를 생성하고, 폰트를 지정하고, 텍스트 출력을 지정한다.

```
osg::ref_ptr<osgText::Text> text = new osgText::Text;
text->setFont(font.get());
text->setText("Display this message.");
```

- setText 함수는 osgText::String를 받는다.
- osgText::Text는 텍스트의 크기(size), 모양 (appearance), 방향 (orientation), 위치 (position) 를 제어하는 여러 가지 함수를 제공하고 있다.

Text Position

- Text는 cull 과 draw traversal중에 텍스트 개체의 좌표 위치를 변환시킨다.
- By default, 텍스트 개체의 위치는 텍스트 개체 (object) 좌표의 중심에 있다.
- Text::setPosition를 사용하여 텍스트 개체의 위치를 변환한다.

```
// Draw the text at (10., 0., 1.)
text->setPosition(osg::Vec3(10.f, 0.f, 1.f));
```


Text Orientation

- 텍스트 방향은 3차원 공간에서 렌더링괴는 텍스트의 전면의 방향을 결정한다.
- Text::setAxisAlignment() 를 사용하여 텍스트의 방향을 지정한다.
- Billboard-style 텍스트를 생성하려면 Text::Screen을 사용한다.

```
text->setAxisAlignment(osgText::Text::SCREEN);
```

- 이와 다른 방법으로는, 텍스트를 an axis-aligned plane 에 놓는다 (텍스트의 기본 방향은 Text::XY_PLANE에 있으며 텍스트는 xy plane에서 +z로 바라보는 곳에 위치하고 있다).

```
text->setAxisAlignment(osgText::Text::XY_PLANE);
```

Text AxisAlignment

Text::AxisAlignment

Enumerant	*Orientation Effect*
Text::XY_PLANE	(Default.) Places text in the *xy* plane facing positive *z*.
Text::XZ_PLANE	Places text in the *xz* plane facing positive *y*.
Text::YZ_PLANE	Places text in the *yz* plane facing positive *x*.
Text::REVERSED_XY_PLANE	Places text in the *xy* plane facing negative *z*.
Text::REVERSED_XZ_PLANE	Places text in the *xz* plane facing negative *y*.
Text::REVERSED_YZ_PLANE	Places text in the *yz* plane facing negative *x*.
Text::SCREEN	Renders text that always faces the screen.

Table 2-1
Text Orientation AxisAlignment Enumerants

Text Alignment

- 워드프로세서의 text alignment 나 스프레드쉬트의 cell alignment와 비슷하다.
- 렌더링 텍스트의 수직 수평 정렬을 결정한다.
- Default 는 Text::LEFT_BASE_LINE이다.
- 텍스트 alignment을 바꾸려면, Text::setAlignment()를 사용한다.

```
text->setAlignment(osgText::Text::CENTER_TOP);
```


Text Alignment

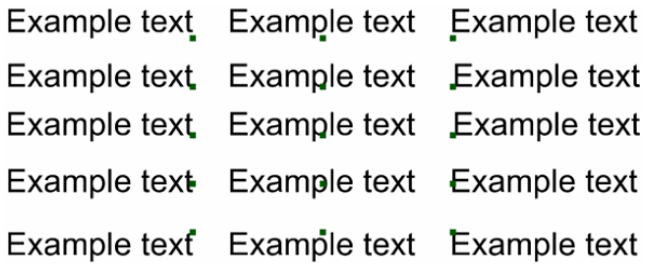

Figure 2-11
Text alignment types
This figure shows the effect of the 15 different AlignmentType enumerants. For each example, the text's position (set with **setPosition()**) is illustrated by a dark green point. Left to right, top to bottom: RIGHT_BOTTOM, CENTER_BOTTOM, LEFT_BOTTOM, RIGHT_BOTTOM_BASE_LINE, CENTER_BOTTOM_BASE_LINE, LEFT_BOTTOM_BASE_LINE, RIGHT_BASE_LINE, CENTER_BASE_LINE, LEFT_BASE_LINE, RIGHT_CENTER, CENTER_CENTER, LEFT_CENTER, RIGHT_TOP, CENTER_TOP, and LEFT_TOP.

Text Size

- Default character height는 32 object coordinate units이다.
- Character width는 폰트에 따라 다양하다.
- Default character height를 바꾸려면, Text::setCharacterSize()를 부른다.

```
// change the height to one object coordinate unit
text->setCharacterSize(1.0f);
```

- Text::setCharacterSizeMode()를 사용하여 text를 object coordinate 대신 screen coordinate에서 character height을 지정할 수 있다.

```
text->setCharacterSizeMode(osgText::Text::SCREEN_COORDS);
```

Text Resolution

- 응용 프로그램에서는 폰트 텍스쳐 맵이 사용되서 희미하게 보이는 캐릭터가 생기지 않도록 glyph resolution 를 다양하게 해야 한다.
- By default, osgText는 glyph마다 32x32 texels 를 할당한다.
- Text::setFontResolution()를 사용하여 text resolution을 바꿀 수 있다.

```
text->setFontResolution(128, 128);
```

- 폰트 resolution 증가는 hardware resource 요구를 증가시킨다.

Text Color

- Text::setColor()를 사용해서 텍스트 색을 바꾼다.
- osg::Vec4 에 rgba 색 값으로 setColor() 에 파라메터로 지정한다.

```
// Set the text color to blue
text->setColor(osg::Vec4(0.f, 0.f, 1.f, 1.f));
```