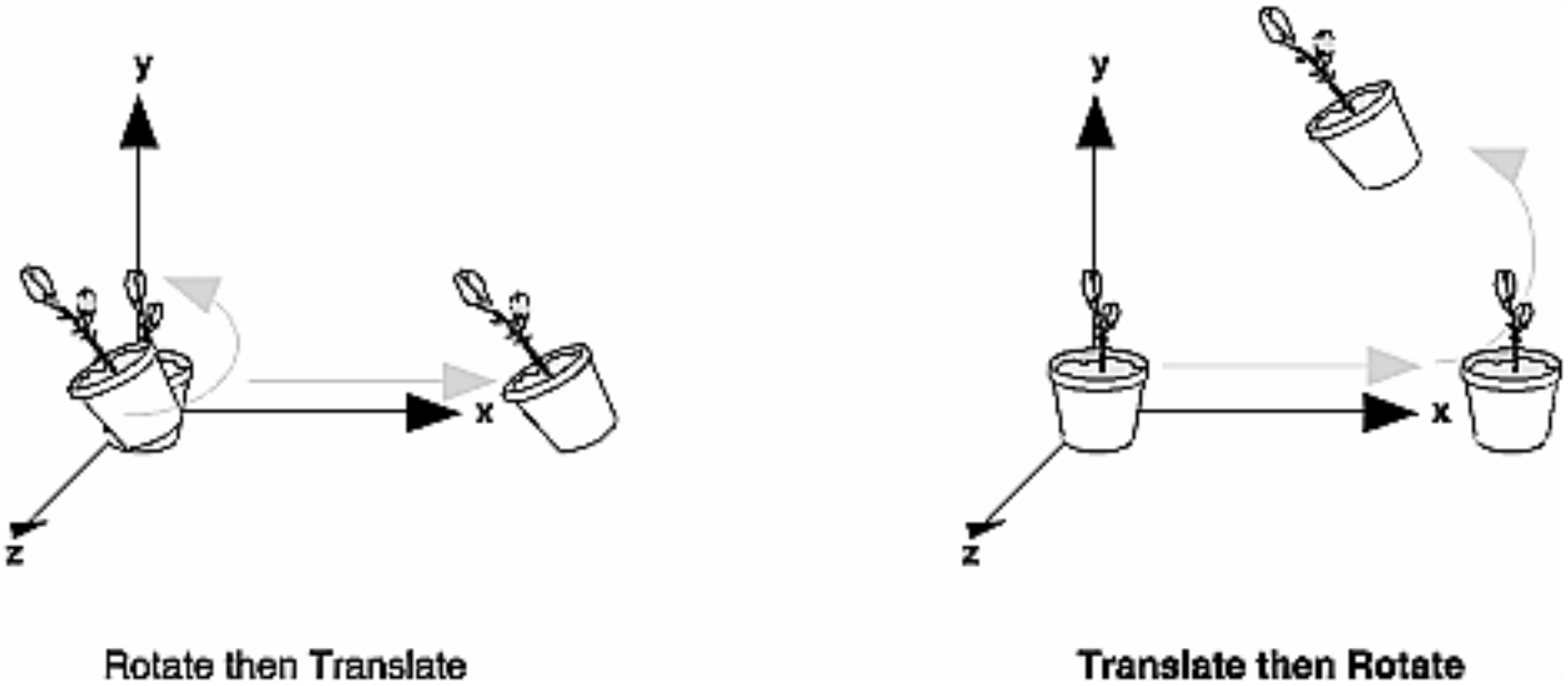
OpenSceneGraph

Transformation

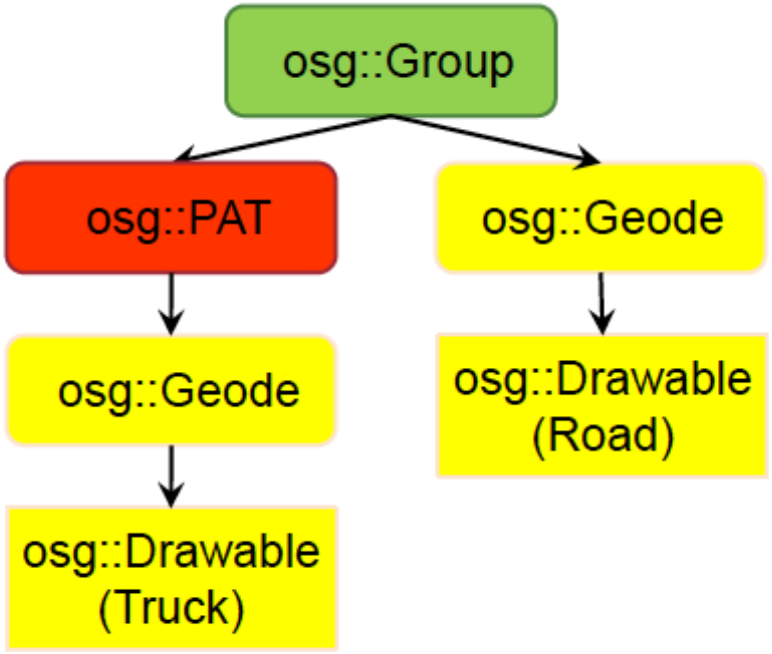
Topics

- Transformation
- OSG Transformation Node
- How to Make Use of Different Polygon
- How to Get Access to Geometry Attributes

Model View Transformation

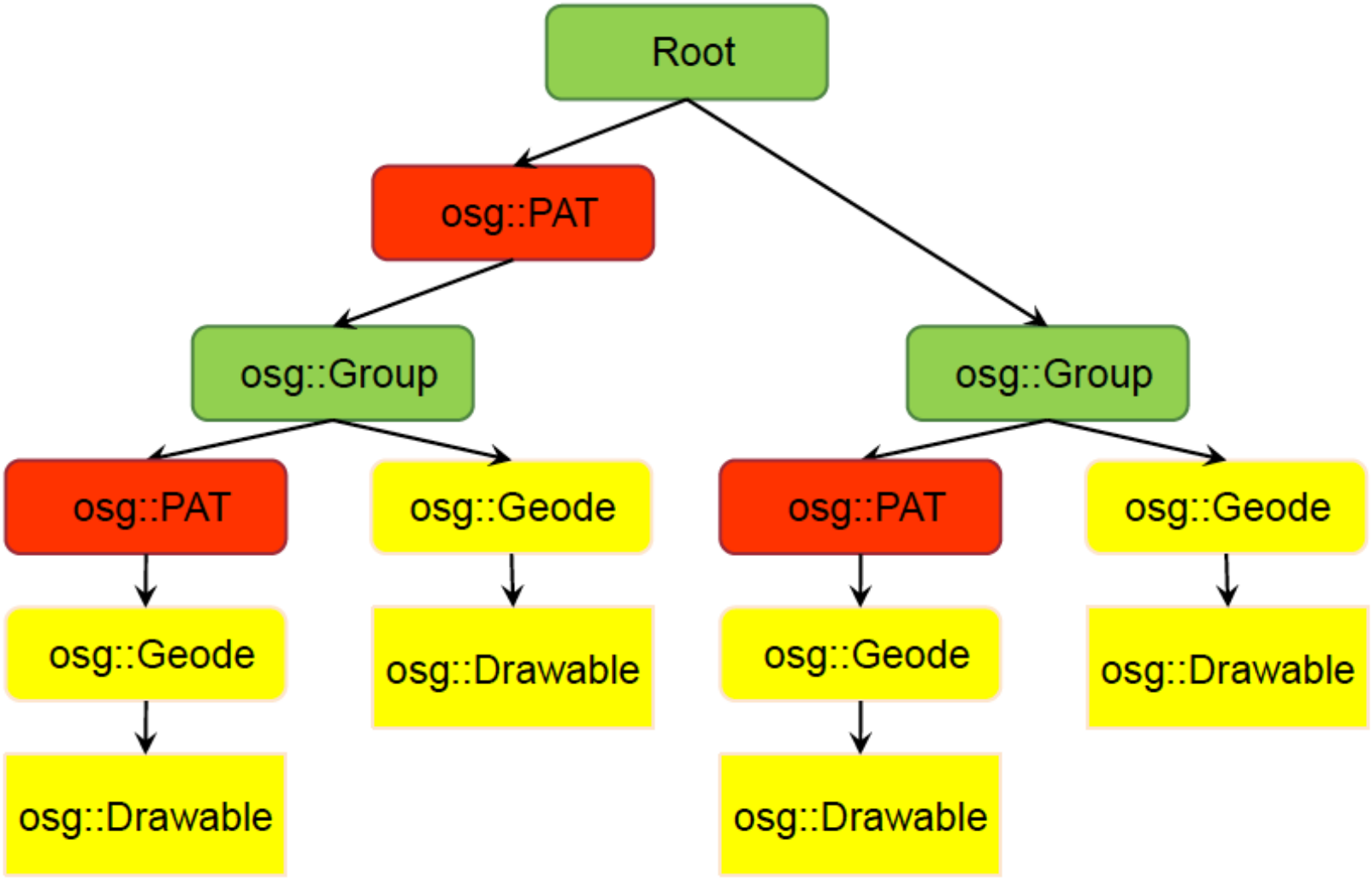


Transformation in OSG

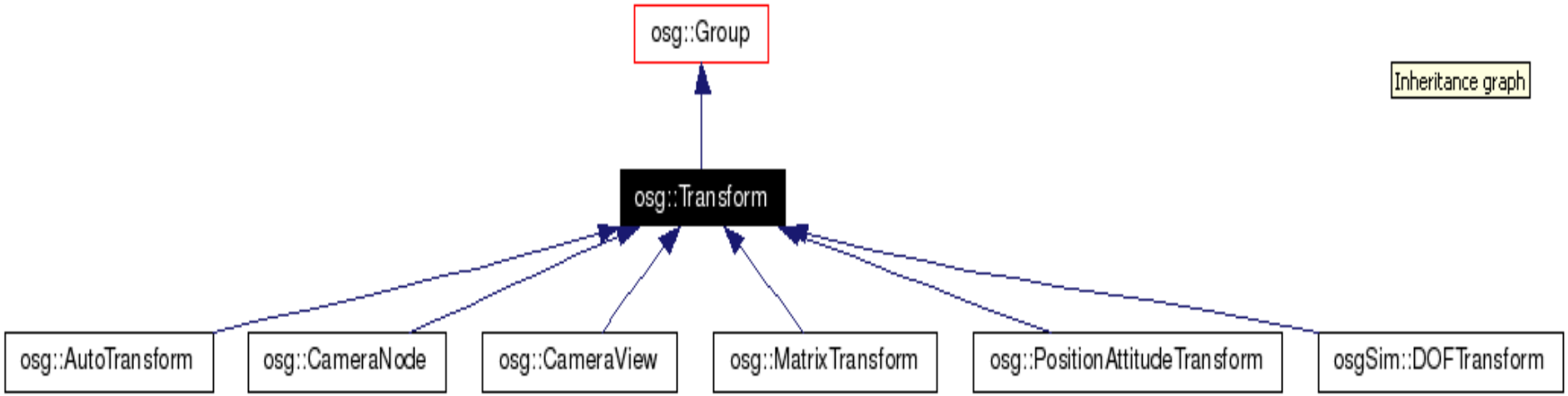


PAT : PositionAttitudeTransform

Typical Graph



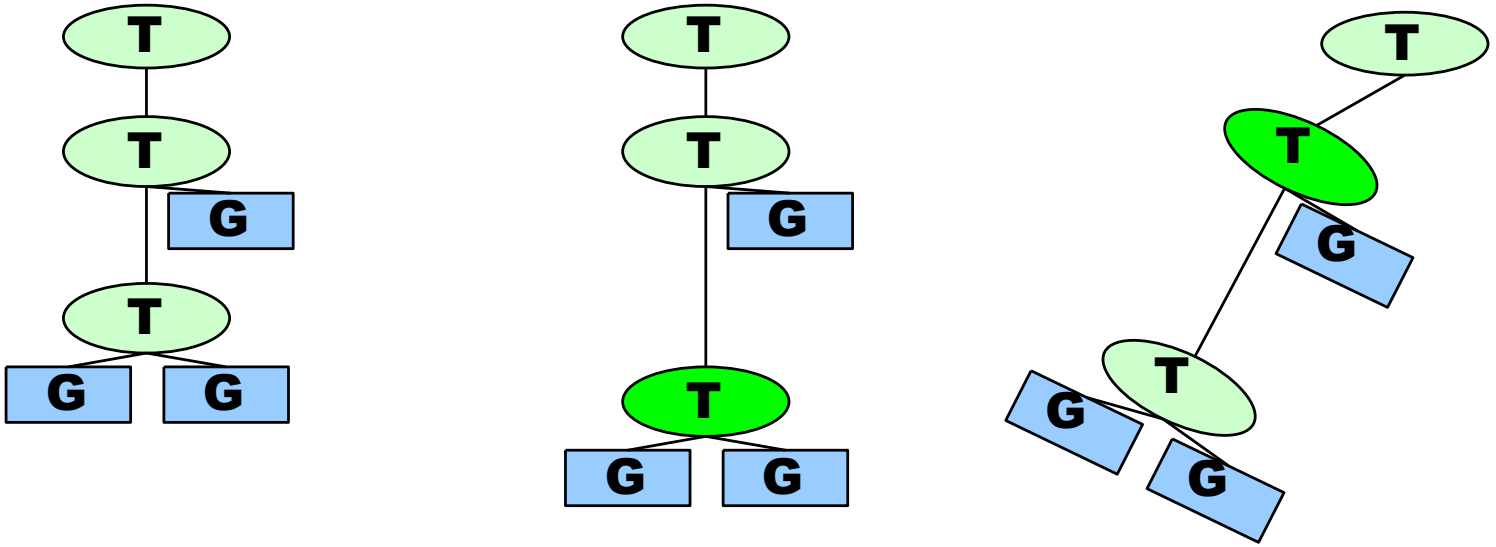
Transform Nodes



osg::Transform - 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.

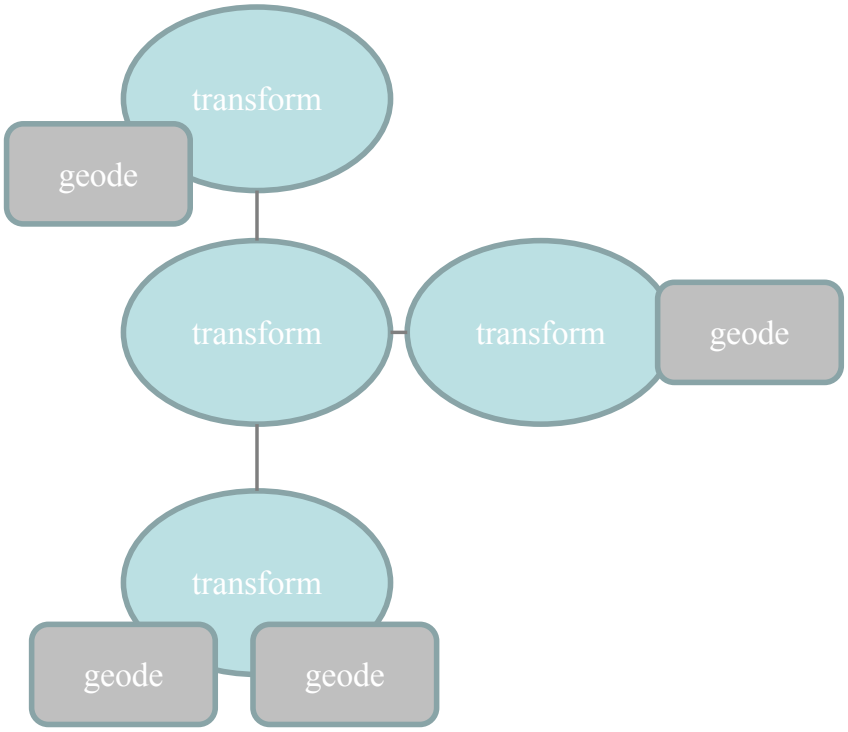
Transformations

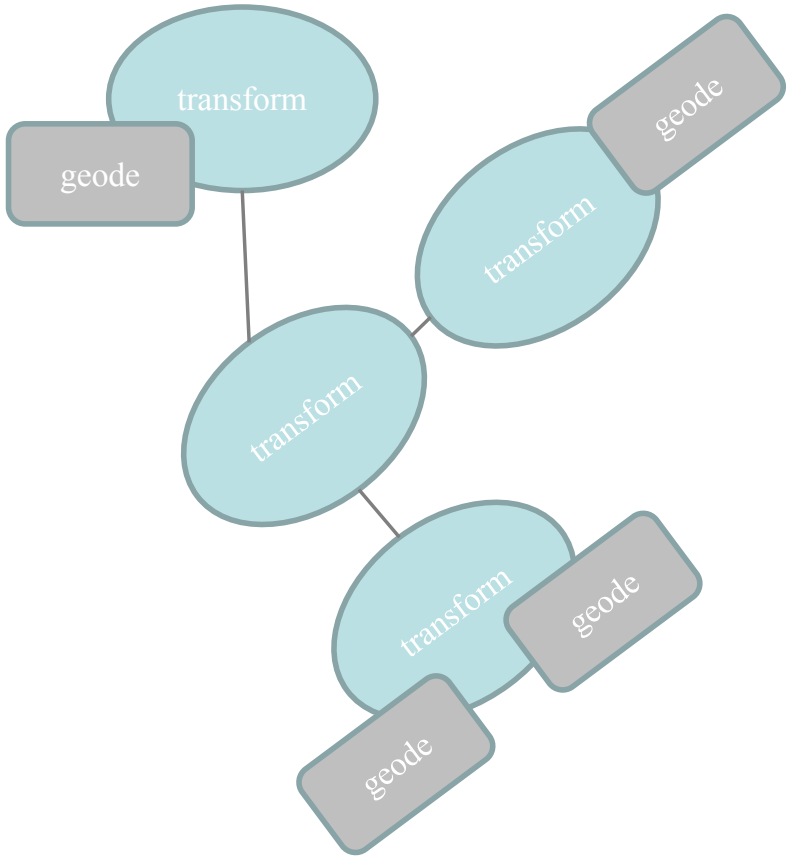
- Transformations apply to all child nodes in the tree
- Allows hierarchies, e.g. limbs on a human body



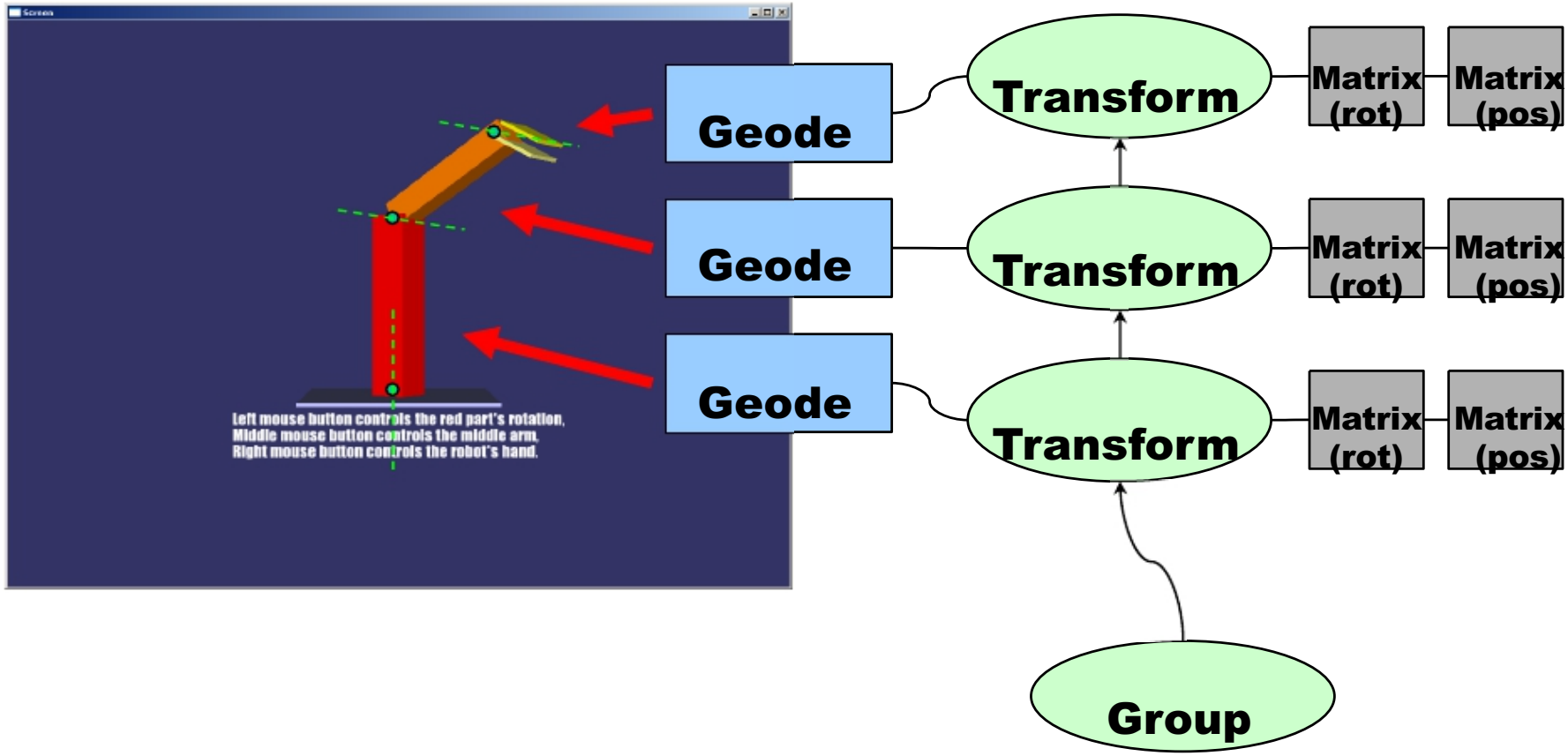
OpenSceneGraph - Transformation

- OSG allows for hierarchies of transformation node. Such structure makes it much easier to control the motions of each limb, part or the whole body.

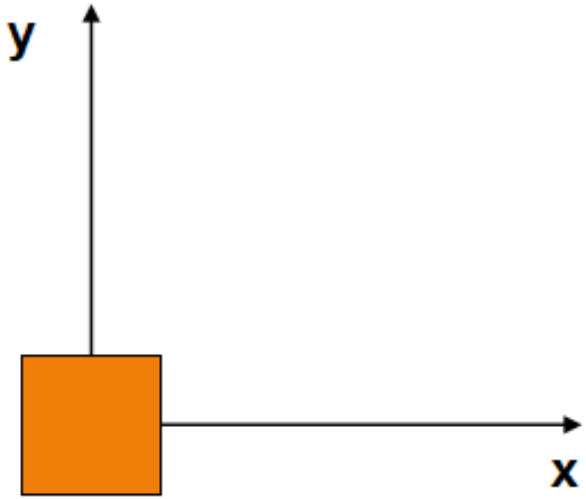


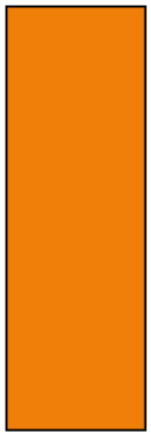


Example


Left mouse button controls the red part's rotation.
Middle mouse button controls the middle arm.
Right mouse button controls the robot's hand.
Geode
Geode
Geode
Transform
Transform
Transform
Matrix (rot)
Matrix (pos)
Matrix (rot)
Matrix (pos)
Matrix (rot)
Matrix (pos)
Group

Transformation as a Tool



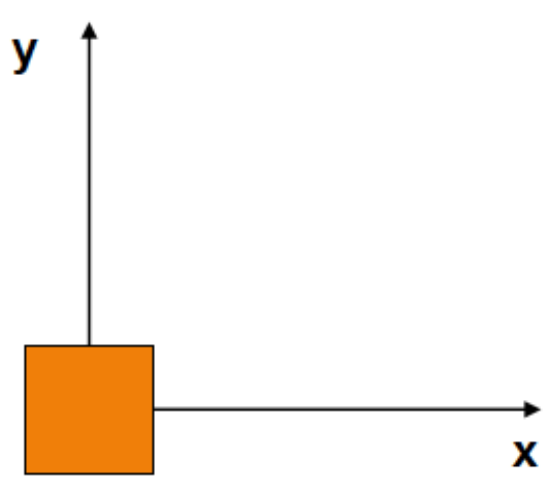


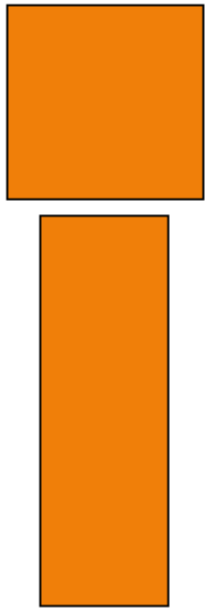
Translation = [5,0]

Rotation = (0^{O}, [1,1])

Scale = [1,3]

Transformation as a Tool



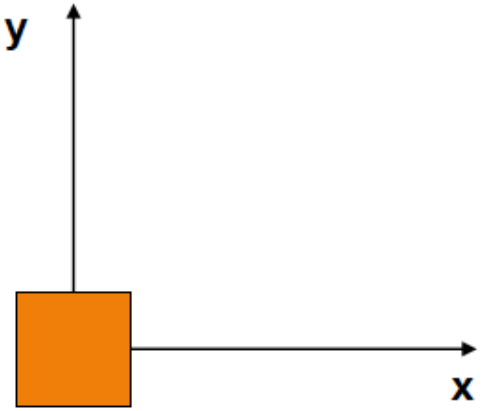


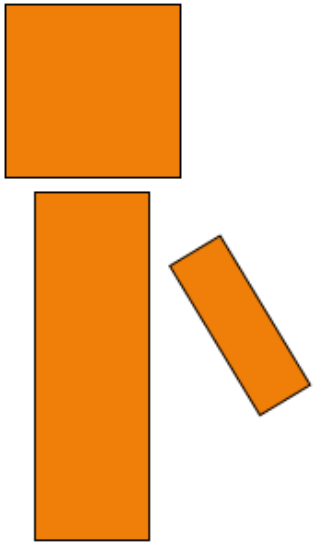
Translation = [5,2]

Rotation = (0^{O}, [1,1])

Scale = [1.5,1.5]

Transformation as a Tool



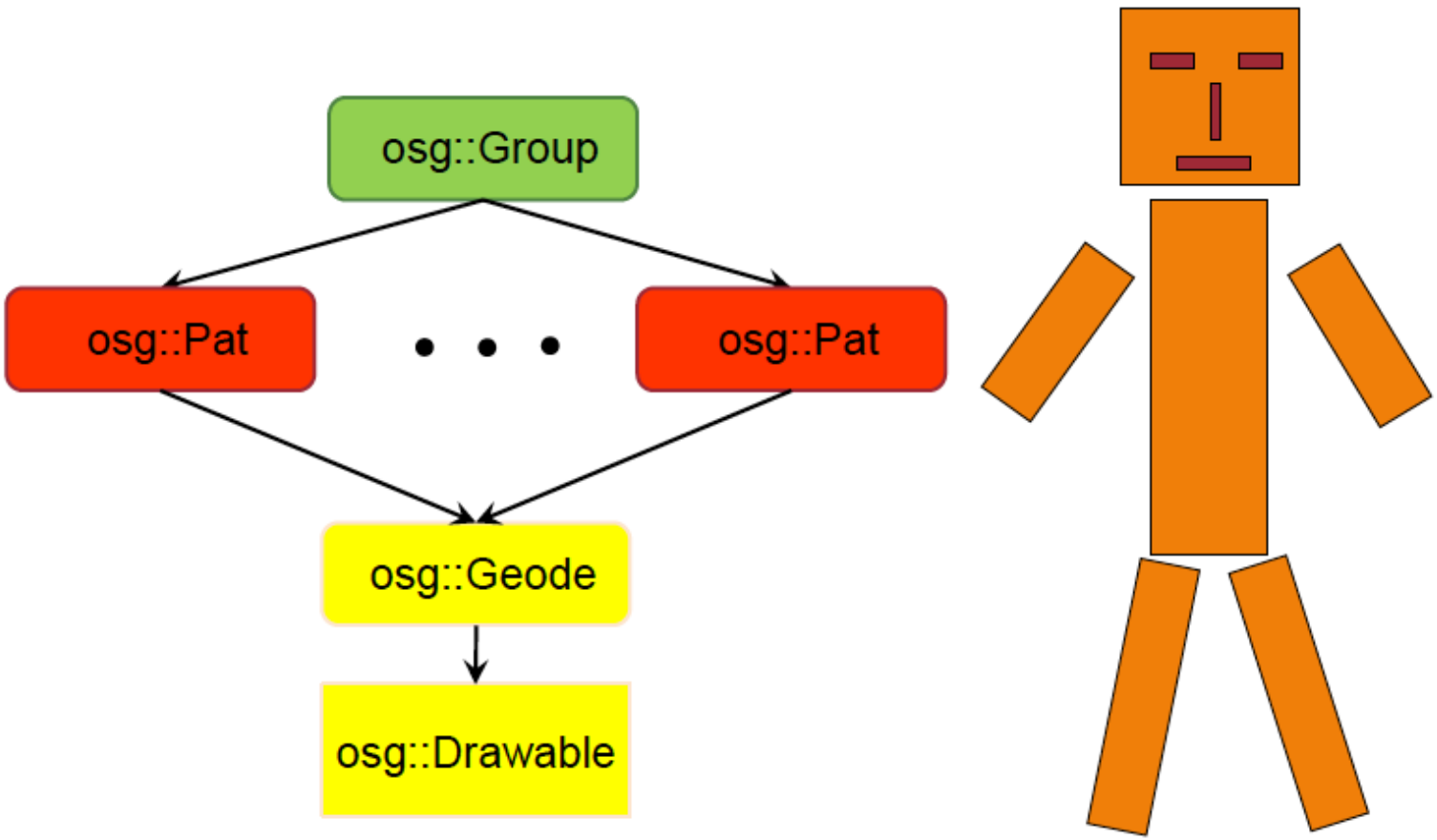


Translation = [6,0]

Rotation = (45^{O}, [1,0])

Scale = [0.5,1.5]

Transformation as a Tool



Transformation Matrix

- $u = M \cdot v = [R|T] \cdot v$
- Rotation about an arbitrary axis:

$$R(k_x, k_y, k_z, \theta) =$$

$$\begin{pmatrix} k_x k_x(1-c)+c & k_x k_y(1-c)-k_z s & k_x k_z(1-c)+k_y s & 0 \\ k_y k_x(1-c)+k_z s & k_y k_y(1-c)+c & k_y k_z(1-c)+k_x s & 0 \\ k_z k_x(1-c)-k_y s & k_y k_z(1-c)+k_x s & k_z k_z(1-c)+c & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

- $c = cos(\theta), s = sin(\theta)$

Transformation in OSG

- osg::Transform
  - osg::MatrixTransform
  - osg::PositionAttitudeTransform (PAT)

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

Transform nodes

- Transform
  - **”MatrixTransform”**
    - Has a 4x4 matrix (RefMatrixd) representing a transformation
  - **”PositionAttitudeTransform”**
    - Sets transform via Vec3 position and Quat attitude
  - **”AutoTransform”**
    - Automatically aligns children with screen coordinates

Transform Nodes

- osg::Transform
  - Group에서 파생된 클래스 (multiple children을 가질 수 있다)
  - virtual base class로 그냥 instantiate 할 수 없음
- osg::MatrixTransform, osg::PositionAttitudeTransform를 사용한다.

MatrixTransform Node

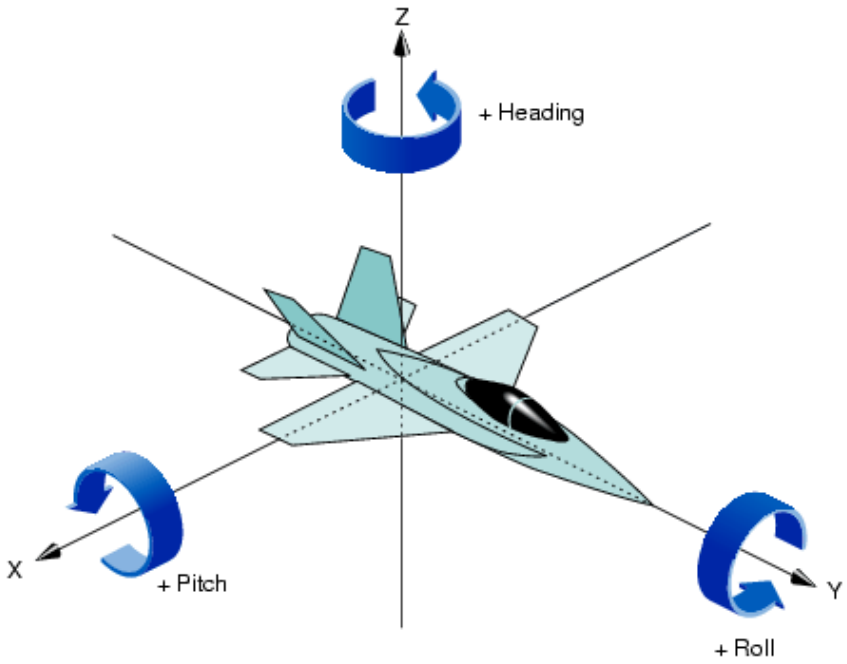
- MatrixTransform는 osg::Matrix 를 내부적으로 사용한다.
- 이동을 원한다면, translation 행렬을 생성하고 그 행렬 MatrixTransform 에 지정한다.

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

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 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;
```

OSG Matrix

void makeRotate (const Vec3d &from, const Vec3d &to)

void makeRotate (value_type angle, const Vec3d &axis)

void makeRotate (value_type angle, value_type x, value_type y, value_type z)

void makeTranslate (const Vec3d &)

void makeTranslate (value_type, value_type, value_type)

void makeScale (const Vec3d &)

void makeScale (value_type, value_type, value_type)

The MatrixTransform class

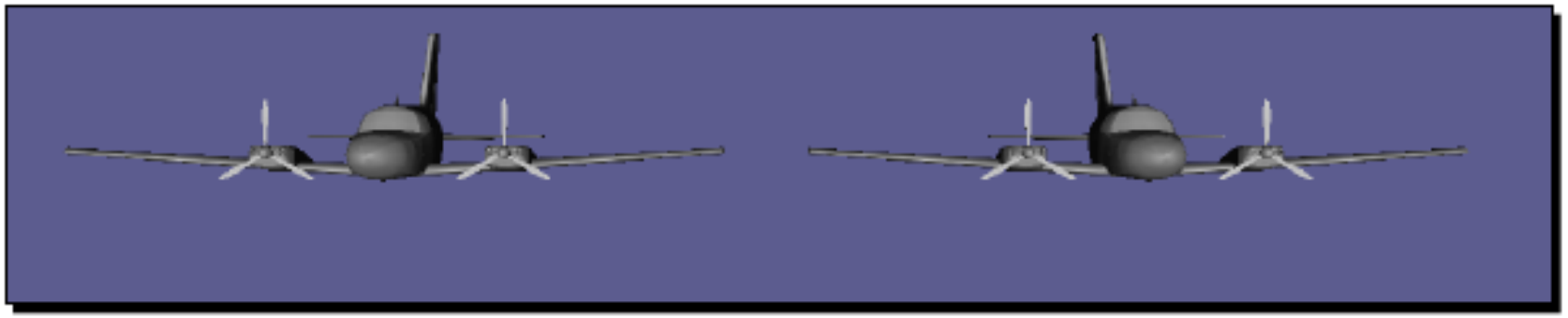
- The osg::MatrixTransform class
  - Derived from osg::Transform.
  - It uses an osg::Matrix variable internally to apply 4x4 double type matrix transformations.
  - The public methods setMatrix() and getMatrix() will assign an osg::Matrix parameter onto the member variable of osg::MatrixTransform.

Example Code

```
osg::ref_ptr<osg::Node> model = osgDB::readNodeFile("cessna.osg" );

osg::ref_ptr<osg::MatrixTransform> transformation1 = new osg::MatrixTransform;
transform1->setMatrix( osg::Matrix::translate(-25.0f, 0.0f, 0.0f) );
transform1->addChild( model.get() );

osg::ref_ptr<osg::MatrixTransform> transform2 = new osg::MatrixTransform;
transform2->setMatrix( osg::Matrix::translate(25.0f, 0.0f, 0.0f) );
transform2->addChild( model.get());
```

Example Code

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


root->addChild( transformation1.get() );
root->addChild( transformation2.get() );


osgViewer::Viewer viewer;
viewer.setSceneData( root.get() );
return viewer.run();
```



States

- *Rendering States*
  - Rendering – simulation of physical interaction of light and matter
  - Physically correct simulation is too complex
  - Good approximations are needed
  - **State** controls the approximations
- Recall: OSG traverses the scene graph and renders geometry according to the current state

Rendering State

- Application controls state by *osg::StateSet*
- *StateSet* node can be attached to any node including *Drawable*.
  - State changes are expensive in hardware
  - Minimize the number of *StateSet* objects

Obtain State

- *osg::StateSet* state = obj->getOrCreateStateSet();*
- *obj* is a *Node* or *Drawable*.
  - Return a pointer to *obj*'s *StateSet*.
  - If *StateSet* doesn't exist, create it and attach to *obj*

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 attribute 를 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 StateSet 에 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 또는 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.

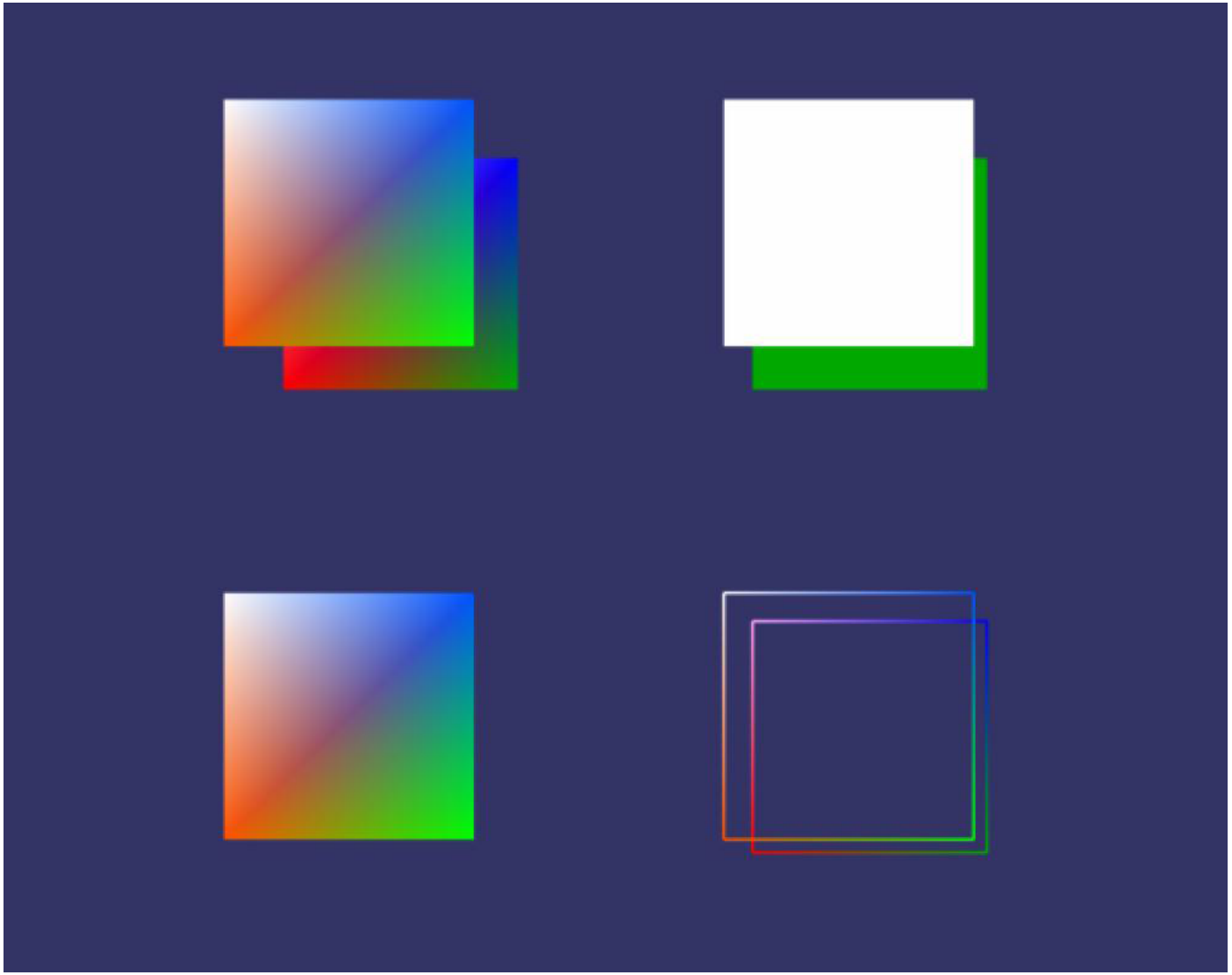
Rendering State – State Inheritance

- 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 지우고 parent state 를 사용하게 한다.

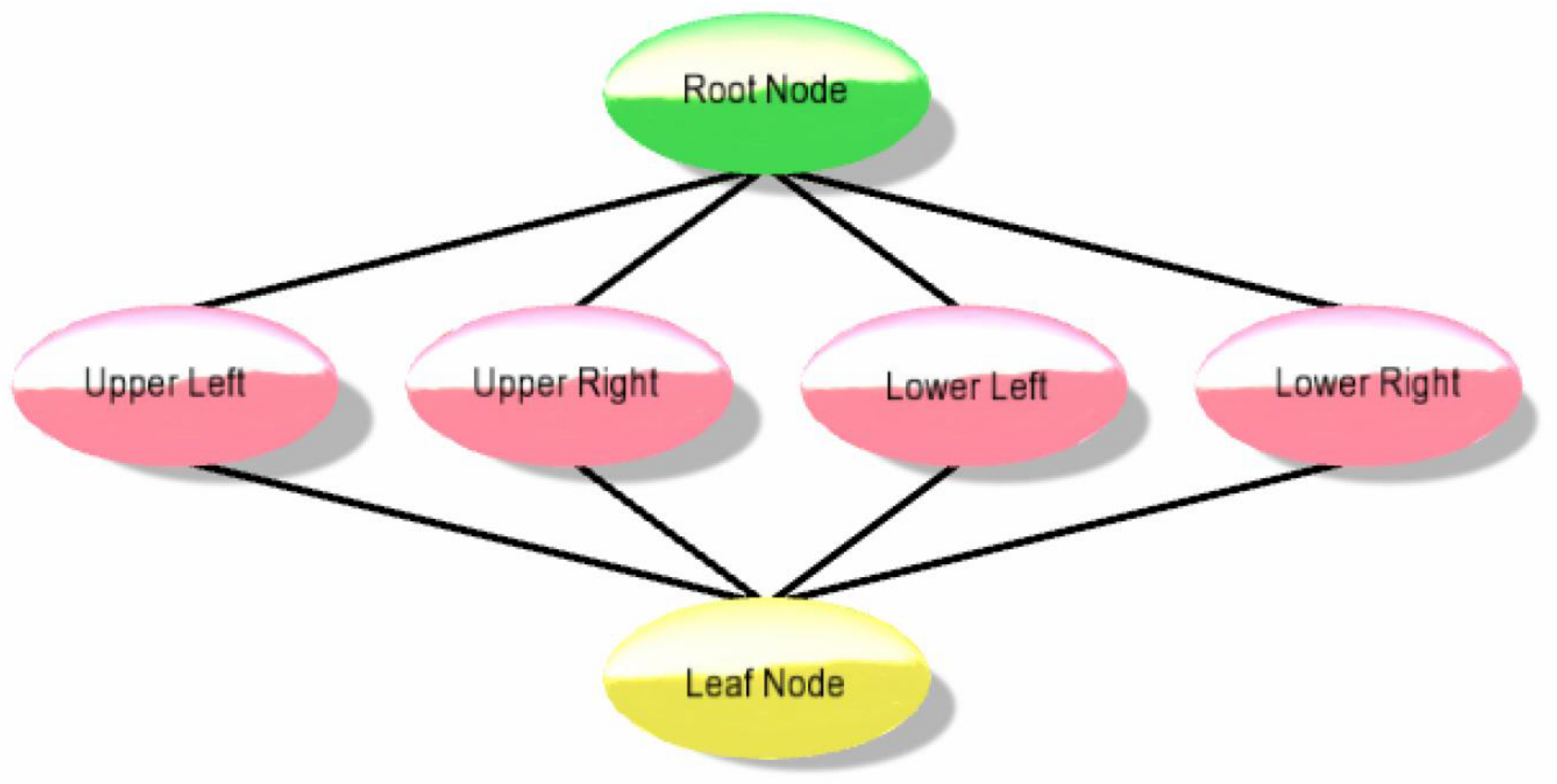
Rendering State – State Inheritance

```
// 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);
```

Example



Scene Graph of Example



Example

```
createSceneGraph()
{
        // Create the root node Group.
        osg::ref_ptr<osg::Group> root = new osg::Group;
        {

        // Disable lighting in the root node's StateSet.
        // Make it PROTECTED to prevent osgviewer from enabling it.

        osg::StateSet* state = root->getOrCreateStateSet();
        state->setMode( GL_LIGHTING, osg::StateAttribute::OFF |
        osg::StateAttribute::PROTECTED );
}
```

Example

```
// Create the leaf node Geode and attach the Drawable.
osg::ref_ptr<osg::Geode> geode = new osg::Geode;
geode->addDrawable( createDrawable().get() );


osg::Matrix m;
{
        // Upper-left: Render the drawable with default state.
        osg::ref_ptr<osg::MatrixTransform> mt =
        new osg::MatrixTransform;
        m.makeTranslate( -2.f, 0.f, 2.f );
        mt->setMatrix( m );
        root->addChild( mt.get() );
        mt->addChild( geode.get() );
}
```

Example

```
// Upper-right Set shade model to FLAT.
osg::ref_ptr<osg::MatrixTransform> mt =
new osg::MatrixTransform;

m.makeTranslate( 2.f, 0.f, 2.f );
mt->setMatrix( m );
root->addChild( mt.get() );
mt->addChild( geode.get() );

osg::StateSet* state = mt->getOrCreateStateSet();
osg::ShadeModel* sm = new osg::ShadeModel();
sm->setMode( osg::ShadeModel::FLAT );
state->setAttribute( sm );
```

Example

```
// Lower-left: Enable back face culling.

osg::ref_ptr<osg::MatrixTransform> mt =
new osg::MatrixTransform;

m.makeTranslate( -2.f, 0.f, -2.f );
mt->setMatrix( m );
root->addChild( mt.get() );
mt->addChild( geode.get() );

osg::StateSet* state = mt->getOrCreateStateSet();
osg::CullFace* cf = new osg::CullFace(); // Default: BACK
state->setAttributeAndModes( cf );
```

Example

```
// Lower-right: Set polygon mode to LINE in draw3's StateSet.

osg::ref_ptr<osg::MatrixTransform> mt = new osg::MatrixTransform;
m.makeTranslate( 2.f, 0.f, -2.f );
mt->setMatrix( m );
root->addChild( mt.get() );
mt->addChild( geode.get() );

osg::StateSet* state = mt->getOrCreateStateSet();
osg::PolygonMode* pm = new osg::PolygonMode(
osg::PolygonMode::FRONT_AND_BACK,
osg::PolygonMode::LINE );
state->setAttributeAndModes( pm );
osg::LineWidth* lw = new osg::LineWidth( 3.f );
state->setAttribute( lw );
```

FileIO

- Reading Files
  - osgDB::readNodeFile는 3차원 모델 파일을 읽어들여서 Node개체로 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.

Texture Mapping

- OSG는 OpenGL texture mapping 을 지원한다.
- Texture mapping을 사용하고자 한다면, 반드시 다음과 같이 지정해야 한다.
  - Geometry에 texture coordinates이 있어야 한다.
  - Texture attribute 개체를 생성하고, texture image 데이터를 저장한다.
  - SetState에 texture attribute과 mode를 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());
```


Q
&
A