OpenSceneGraph

Building Geometry Model

Topics

- How to Draw Basic Object
- How to Set Vertices and Vertex Attribute
- How to Make Use of Different Polygon
- How to Get Access to Geometry Attributes

Introduction

- *The basic operation of OpenGL's graphical pipeline is to accept vertex data (points, lines, triangles, and polygons) and pixel data (graphical image data), convert them into fragments and store them in the frame buffer*
- *The frame buffer serves as a major interface between developers and the computer display, which maps each frame of graphic contents into memory space for read-write operation.*
- *OSG encapsulates the whole OpenGL vertex transformation and primitive assembly operations in order to manage and send vertex data to the OpenGL pipeline, as well as some data transmission optimizations and additional polygonal techniques for improving rendering performance.*

How OpenGL draws objects

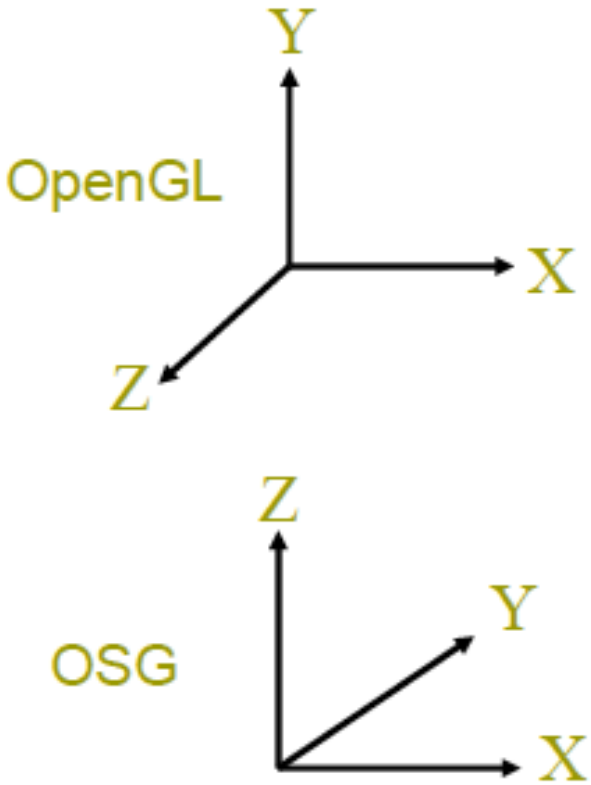
- OpenGL uses geometry primitives to draw
- A geometry primitive, which may be a set of points, lines, triangles, or polygonal faces, determines how OpenGL sorts and renders its associated vertex data.
- Specify a list of vertices between the glBegin() and glEnd() pair, which is called immediate mode

```
glBegin(GL_TRIANGLES);
        glColor3f(1, 0, 0); // set vertex color to red
        glVertex3fv(v1); // draw a triangle with v1, v2, v3
        glVertex3fv(v2);
        glVertex3fv(v3);
glEnd();
```

How OpenGL draws objects

- Vertex Array
  - Vertex data including vertex coordinates, normals, colors can be stored in various arrays
  - Primitives will be formed by indexing the array element
- Display List
  - All vertices and pixel data are compiled and copied into the graphic memory
- Vertex Buffer Object (VBO)
  - Allow vertex array data to be stored in the high performance memory
- OSG
  - Use VBO and display list to manage and render geometries

OpenSceneGraph's Coordinate System



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

Gnode and Geometry

- 사각형을 그리는 geometry
  - 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

Geode and Drawable Classes

- osg::Geode class corresponds to the lead node of scene graph
  - No child node
  - Contains geometry information for rendering
  - The geometry data to be drawn are stored in a set of osg::Drawable objects
  - Provide a few method to attach and detach drawables
    - addDrawable()
    - removeDrawable() or removeDrawables()
    - getDrawable()
    - getnumDrawable()

Rendering basic shapes

- Provide an osg::ShapeDrawable class
  - Inherites from the osg::Drawable to render basic geometry sharps quickly with plain paramterts
  - Always include osg::Shape object to specify geometry's type and properties

  - sharpDrawable->setShape(new osg::Box(osg::Vec3(1.0f, 0.0f, 0.0f), 10.0f, 10.0f, 5.0f))

  - Center point (1.0, 0.0, 0.0)
  - Width : 10, height : 10, depth : 5
- osgVec3 : three element vector in OSG
- osgVec2, osgVec4
- osgVec3d
- Osg::Box, osg::Capsule, osg::cone, osg::Cyliner, osg::Sphere

Drawable

- 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 (predefinedshapes)의 사용을 위해 제공
- osg::Geometry
  - 일반적인 기하학적 개체 (geometry) 데이터의 저장 및 렌더링에 사용

Quickly Creating Simple Objects

```
osg::ref_ptr<osg::ShapeDrawable> shape1 = new osg::ShapeDrawable;
shape1->setShape( new osg::Box(osg::Vec3(-3.0f, 0.0f, 0.0f), 2.0f, 2.0f, 1.0f) );

osg::ref_ptr<osg::ShapeDrawable> shape2 = new osg::ShapeDrawable;
shape2->setShape( new osg::Sphere(osg::Vec3(3.0f, 0.0f, 0.0f),1.0f) );
shape2->setColor( osg::Vec4(0.0f, 0.0f, 1.0f, 1.0f) );

osg::ref_ptr<osg::ShapeDrawable> shape3 = new osg::ShapeDrawable;
shape3->setShape( new osg::Cone(osg::Vec3(0.0f, 0.0f, 0.0f),1.0f, 1.0f) );
shape3->setColor( osg::Vec4(0.0f, 1.0f, 0.0f, 1.0f) );
```

Quickly Creating Simple Objects

```
osg::ref_ptr<osg::ShapeDrawable> shape1 = new osg::ShapeDrawable;
shape1->setShape( new osg::Box(osg::Vec3(-3.0f, 0.0f, 0.0f), 2.0f, 2.0f, 1.0f) );

osg::ref_ptr<osg::ShapeDrawable> shape2 = new osg::ShapeDrawable;
shape2->setShape( new osg::Sphere(osg::Vec3(3.0f, 0.0f, 0.0f),1.0f) );
shape2->setColor( osg::Vec4(0.0f, 0.0f, 1.0f, 1.0f) );

osg::ref_ptr<osg::ShapeDrawable> shape3 = new osg::ShapeDrawable;
shape3->setShape( new osg::Cone(osg::Vec3(0.0f, 0.0f, 0.0f),1.0f, 1.0f) );
shape3->setColor( osg::Vec4(0.0f, 1.0f, 0.0f, 1.0f) );
```

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

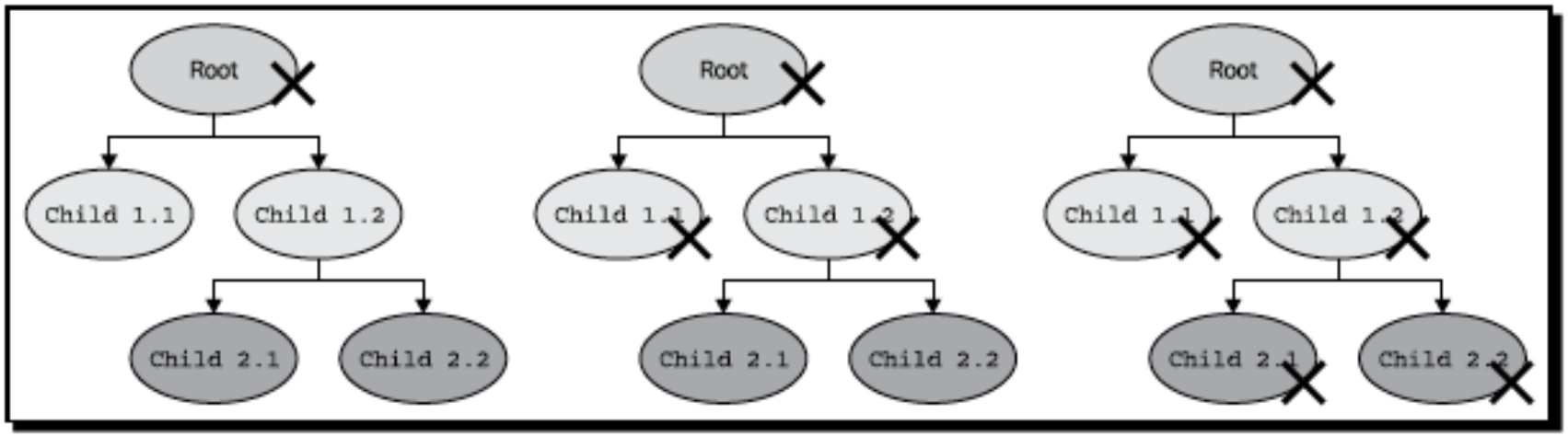
ref_ptr<>

- Typical programming scenarios, developer should create a pointer to the root node, which manages all other child nodes of the scene graph
  - Traverse the scene graph and delete each node and its internal data while they no longer need to be rendered
  - Without writing the management code, data segments occupied by all scene nodes will never be deleted, which will lead to unexpected memory leaks
  - Garbage : memory block that are unreachable from any program variables

- osg::ref_ptr<> template class
  - A native smart pointer
  - Automatic garbage collection and deallocation
  - get(), release()

Collecting Garbage: Why and How

- Reason for using smart pointers and garbage collection system
  - Fewer bugs
  - Efficient management
  - Easy to debug



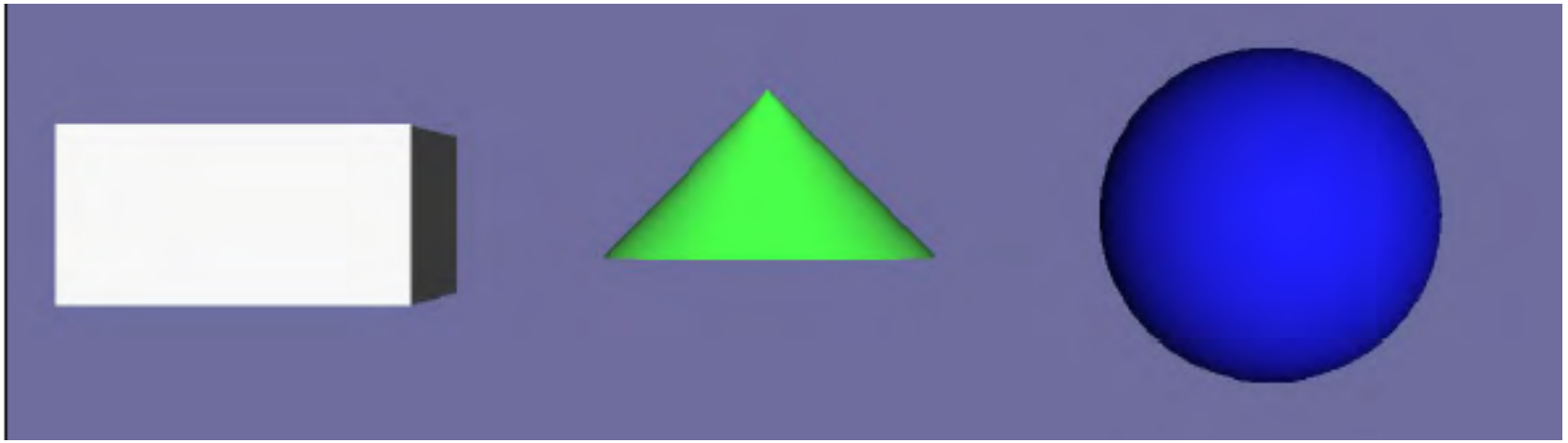
ref_ptr<>

```
osg:: ShapeDrawable *drawable = new ShapeDrawable(cone);
drawable->setColor(osg::Vec4(1.0f, 0.0f, 0.0f, 1.0f));
osg:: Geode* geode = new Geode();
geode->addDrawable(drawable);
```

With ref_ptr<> VS. without ref_ptr

```
osg::ref_ptr<osg::ShapeDrawable> shape1 = new osg::ShapeDrawable;
shape1->setShape( new osg::Cone(osg::Vec3(0.0f, 0.0f, 0.0f), 1.0f, 1.0f) );
shape1->setColor( osg::Vec4(1.0f, 0.0f, 0.0f, 1.0f) );
osg::ref_ptr<osg::Geode> root = new osg::Geode;
root->addDrawable( shape1.get() );
```

Result of Shape Rendering



Osg::ShapeDrawable class

- Useful for quick display, but not an efficient way of drawing geometry primitives
- Only be used for quick prototyping and dubbging

- Osg::Geometry
  - Create geometries with high performance computation and visualization requirements
  - osg::GLBeginEndAdapter Class
    - Used to perform basic shape drawing operation
    - The use of vetex array in the style of a glBegin and glEnd pair
  - To get and use an intialized osg::GLBeginEndAdapter, define class derived from the osg::Drawable class and re-implement its drawImplenation() method

Osg::ShapeDrawable class

```
void drawImpletation(osg::RenderInfo & renderInfo ) const
{
    osg::GLBeginEndAdapter& gl =
    renderInfo.getState()-> getGLBeignEndAdapter()l
    gl.Begin( ..);
    gl.Vertext3fv(…);
    gl.End();
}
```

Storing Array Data

- Support vertex arrays and VBO to speed up the rendering process
- To manage the vertext data -> osg::Array

- osg::Array
  - Can't be instantiated
  - Declare interfaces to exchange with OpenGL calls and buffer data modifier
  - Osg::Vec2Array, osg::Vec3Array inherit the characterstics of the STL vector class
  - Make use of std::vector members (push_back(), pop_back())

Vertics and Vertex Attributes

- Vertex
  - Atomic element of geometry primitives
  - Several number attributes – verext position, color, fog coordinate
  - Position is always required
  - osg::Geometry class with set *Array() method

Specifying Drawing Types

- How to render Vertex
  - osg::primitiveSet is used to manage a geometry primitive set
  - addPrimitiveSet()
  - RemovePrimitiveSet()
  - getPrimitiveSet()
  - getNumPRivmiteSets()
- osg::DrawArrays class
  - Use a number of sequential elements from vertex arrays to constructu a sequence of geometry primitives
  - Attached to osg::Geometry object
    - geom->addPrimitiveSet (new osg::DrawArrays (mode, first, count))
    - Mode : GL_POINTS, GL_LINE_STRIP, GL_LINE_LOOP, GL_LINES, GL_TRIANGLE_STRIP, GL_TRIANGLE_FAN, GL_TRIANGLES, GL_QUAD_STRIP, GL_QUADS, and GL_POLYGON

Storing Array Data

- Support vertex arrays and VBO to speed up the rendering process
- To manage the vertext data -> osg::Array

- osg::Array
  - Can't be instantiated
  - Declare interfaces to exchange with OpenGL calls and buffer data modifier
  - Osg::Vec2Array, osg::Vec3Array inherit the characterstics of the STL vector class
  - Make use of std::vector members (push_back(), pop_back())

Drawing a Colored Quad

```
osg::ref_ptr<osg::Vec3Array> vertices = new osg::Vec3Array;
vertices->push_back( osg::Vec3(0.0f, 0.0f, 0.0f) );
vertices->push_back( osg::Vec3(1.0f, 0.0f, 0.0f) );
vertices->push_back( osg::Vec3(1.0f, 0.0f, 1.0f) );
vertices->push_back( osg::Vec3(0.0f, 0.0f, 1.0f) );

osg::ref_ptr<osg::Vec4Array> colors = new osg::Vec4Array;
colors->push_back( osg::Vec4(1.0f, 0.0f, 0.0f, 1.0f) );
colors->push_back( osg::Vec4(0.0f, 1.0f, 0.0f, 1.0f) );
colors->push_back( osg::Vec4(0.0f, 0.0f, 1.0f, 1.0f) );
colors->push_back( osg::Vec4(1.0f, 1.0f, 1.0f, 1.0f) );
```

Drawing a Colored Quad

```
osg::ref_ptr<osg::Geometry> quad = new osg::Geometry;
quad->setVertexArray( vertices.get() );
quad->setNormalBinding( osg::Geometry::BIND_OVERALL );
quad->setColorArray( colors.get() );
quad->setColorBinding( osg::Geometry::BIND_PER_VERTEX );
quad->addPrimitiveSet( new osg::DrawArrays(GL_QUADS, 0, 4) );

osg::ref_ptr<osg::Geode> root = new osg::Geode;
root->addDrawable( quad.get() );
osgViewer::Viewer viewer;
viewer.setSceneData( root.get() );
return viewer.run();
```

osg::Geometry

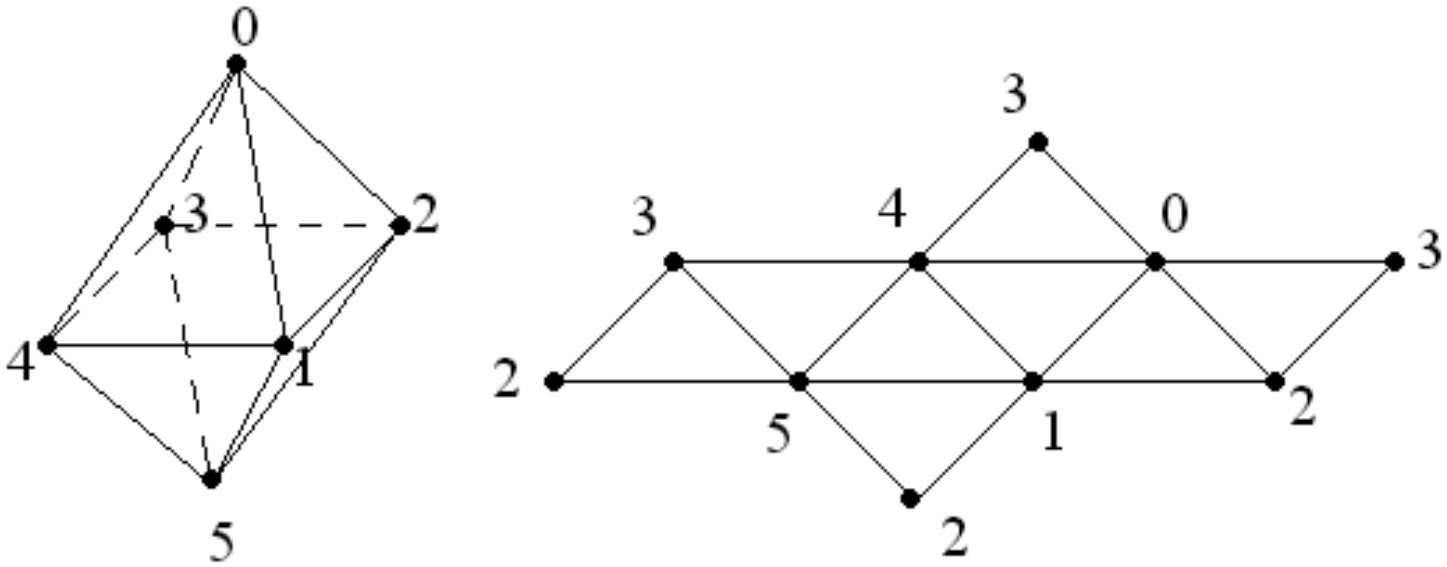
- 사각형을 그리는 예제에서 사용됐던 Geometry 함수들
  - setVertexArray(), setColorArray(), setNormalArray()
    - OpenGL의 glVertexPointer(), glColorPointer(), glNormalPointer()와 유사
    - 정점 (vertex), 색 (color), 법선벡터 (normal vector) 자료를 지정
  - setColorBinding(), setNormalBinding()
    - Geometry에게 어떻게 색과 법선벡터를 그려야 할 지를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

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

geom->addPrimitiveSet(new

osg::DrawArrays(osg::PrimitiveSet::TRIANGLE_STRIP, 0, 6 );
```

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

Drawing An Octahedron



Set the Vertex

```
osg::ref_ptr<osg::Vec3Array> vertices = new osg::Vec3Array(6);


(*vertices)[0].set( 0.0f, 0.0f, 1.0f);
(*vertices)[1].set(-0.5f,-0.5f, 0.0f);
(*vertices)[2].set( 0.5f,-0.5f, 0.0f);
(*vertices)[3].set( 0.5f, 0.5f, 0.0f);
(*vertices)[4].set(-0.5f, 0.5f, 0.0f);
(*vertices)[5].set( 0.0f, 0.0f,-1.0f);
```

Make Triangle

```
osg::ref_ptr<osg::DrawElementsUInt> indices =
new osg::DrawElementsUInt(GL_TRIANGLES, 24);

(*indices)[0] = 0; (*indices)[1] = 1; (*indices)[2] = 2;
(*indices)[3] = 0; (*indices)[4] = 2; (*indices)[5] = 3;
(*indices)[6] = 0; (*indices)[7] = 3; (*indices)[8] = 4;
(*indices)[9] = 0; (*indices)[10]= 4; (*indices)[11]= 1;
(*indices)[12]= 5; (*indices)[13]= 2; (*indices)[14]= 1;
(*indices)[15]= 5; (*indices)[16]= 3; (*indices)[17]= 2;
(*indices)[18]= 5; (*indices)[19]= 4; (*indices)[20]= 3;
(*indices)[21]= 5; (*indices)[22]= 1; (*indices)[23]= 4;
```

Create Geometry & Add Geometry

```
osg::ref_ptr<osg::Geometry> geom = new osg::Geometry;


geom->setVertexArray( vertices.get() );
geom->addPrimitiveSet( indices.get() );
osgUtil::SmoothingVisitor::smooth( *geom );


osg::ref_ptr<osg::Geode> root = new osg::Geode;
root->addDrawable( geom.get() );
osgViewer::Viewer viewer;
viewer.setSceneData( root.get() );
return viewer.run();
```

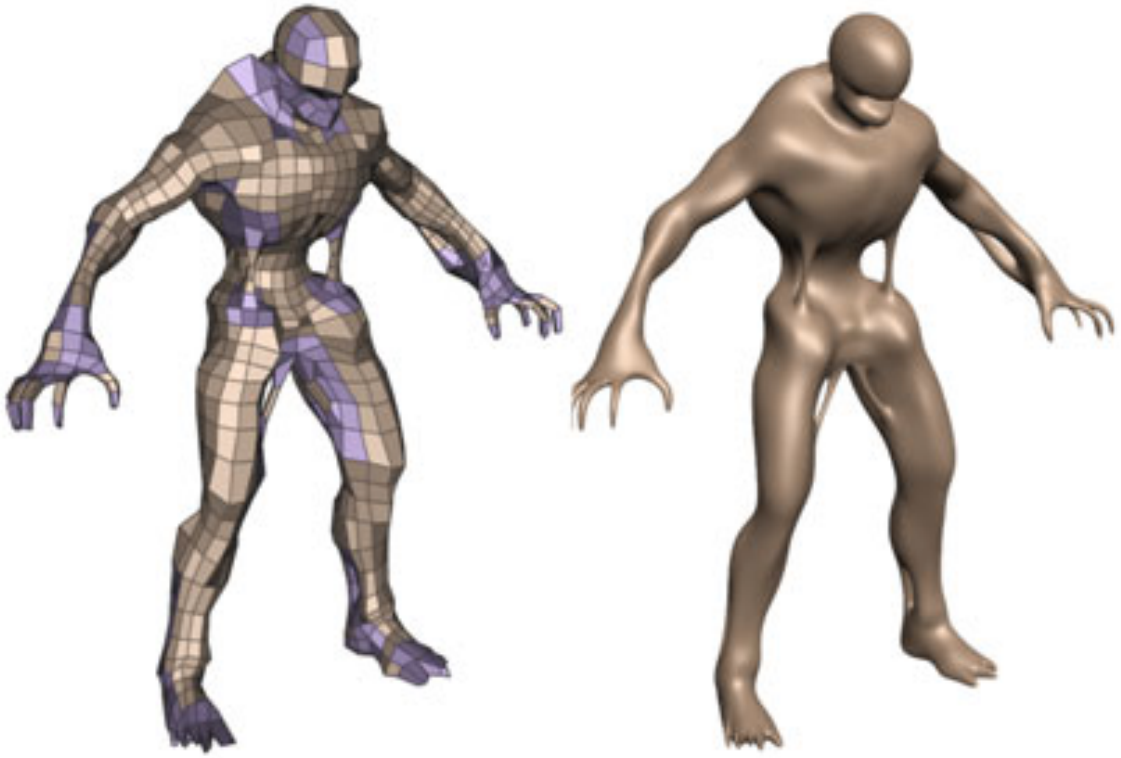
Using Polygonal Techniques

- Supports various polygon technique for manipulating the geometry objects
  - Polygon reduction and tessllation
  - osgUtil::Simplifier
  - osgUtil::Tesselator
  - osgUtil::TriStripVisitor

Tessellating A Polygon

❖ **Tessellation**

- 폴리곤을 미세한 조각으로 세분화하는 방법
- 정사각형을 대각선으로 자른다면 이는 사각형을 두 개의 삼각형으로 “조각 내는” 것



Tessellating A Polygon

- Creating a concave polygon by using the osg::Geometry

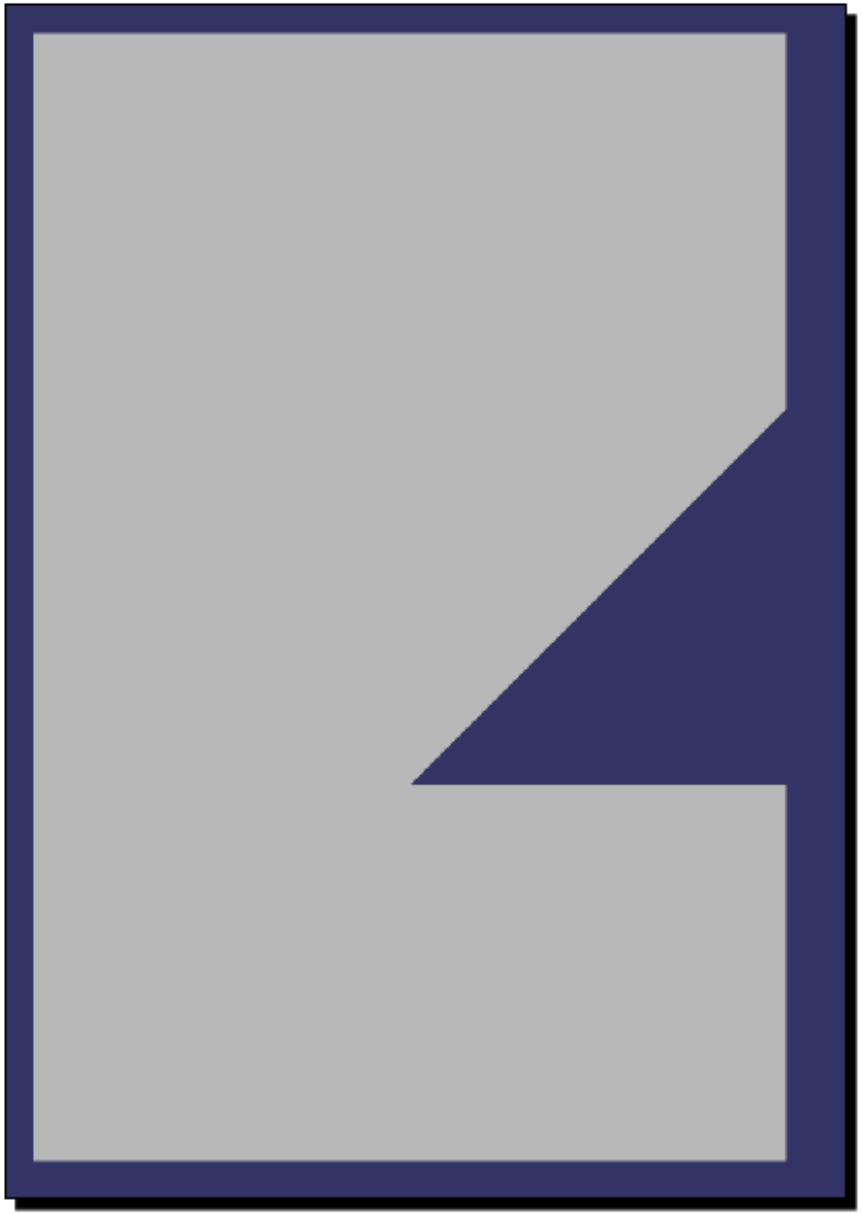
```
osg::ref_ptr<osg::Vec3Array> vertices = new osg::Vec3Array;

vertices->push_back( osg::Vec3(0.0f, 0.0f, 0.0f) );
vertices->push_back( osg::Vec3(2.0f, 0.0f, 0.0f) );
vertices->push_back( osg::Vec3(2.0f, 0.0f, 1.0f) );
vertices->push_back( osg::Vec3(1.0f, 0.0f, 1.0f) );
vertices->push_back( osg::Vec3(1.0f, 0.0f, 2.0f) );
vertices->push_back( osg::Vec3(2.0f, 0.0f, 2.0f) );
```

Tessellating A Polygon

- Creating a concave polygon by using the osg::Geometry

```
osg::ref_ptr<osg::Vec3Array> vertices = new osg::Vec3Array;

vertices->push_back( osg::Vec3(0.0f, 0.0f, 0.0f) );
vertices->push_back( osg::Vec3(2.0f, 0.0f, 0.0f) );
vertices->push_back( osg::Vec3(2.0f, 0.0f, 1.0f) );
vertices->push_back( osg::Vec3(1.0f, 0.0f, 1.0f) );
vertices->push_back( osg::Vec3(1.0f, 0.0f, 2.0f) );
vertices->push_back( osg::Vec3(2.0f, 0.0f, 2.0f) );
```

Tessellating A Polygon



Tessellating A Polygon

```
osg::ref_ptr<osg::Vec3Array> normals = new osg::Vec3Array;
normals->push_back( osg::Vec3(0.0f,-1.0f, 0.0f) );

osg::ref_ptr<osg::Geometry> geom = new osg::Geometry;
geom->setVertexArray( vertices.get() );
geom->setNormalArray( normals.get() );
geom->setNormalBinding( osg::Geometry::BIND_OVERALL );
geom->addPrimitiveSet( new osg::DrawArrays(GL_POLYGON, 0, 8) );
```

❖

Tessellating A Polygon

- To render the concave polygon

```
osgUtil::Tessellator tessellator;
tessellator.retessellatePolygons( *geom );
```


Q
&
A