X3D C/C++/C# Language Bindings (Updates)

ISO/IEC JTC 1/SC 24/WG 9 and Web3D Meetings
Seoul, Korea

January 21-24, 2019

Myeong Won Lee

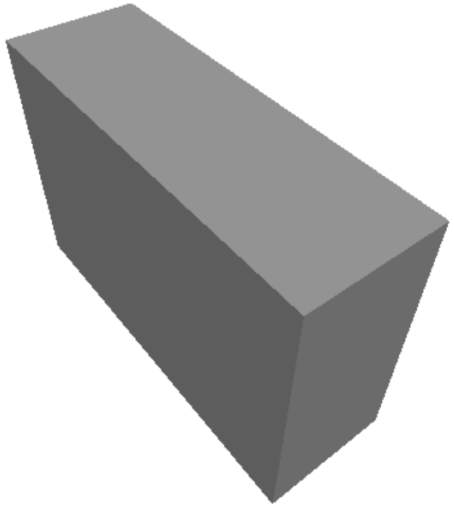
Status of Work

- ISO/IEC 19777-3 X3D C Language Binding
  - NP submitted
- ISO/IEC 19777-4 X3D C++ Language Binding
  - NP submitted
- ISO/IEC 19777-5 X3D C# Language Binding
  - NP vote ended

C/C++/C# Language Binding Concepts

- What is C/C++/C# language binding?
  - X3D scene access interface using C, C++ and C# languages
    - Specify 19775-2 X3D Scene Access Interface using C, C++, C# languages
  - Development of C, C++/C# programs using X3D data types and functions
  - X3D scene read, update, store, and exchange in C, C++/C# applications
- Scope
  - Provides a browser implementation independent way of accessing a browser's capabilities via the languages
  - Provides a set of implementation independent base classes and interfaces that represent possible interactions with an X3D scene through an SAI
  - Provides a C, C++ and C# API format for X3D scene access

A Simple Example of X3D Scene Access API



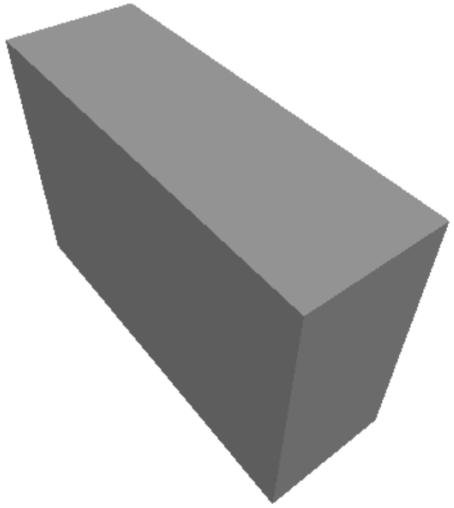
```
<X3D>
<Scene>
   <Background skyColor='1 1 1'/>
   <Viewpoint description='Book View'
orientation='-0.747 -0.624 -0.231 1.05' position='-
1.81 3.12 2.59'/>
   <Shape>
      <Box size='1 2 3'/>
      <Appearance>
         <Material/>
      </Appearance>
   </Shape>
</Scene>
</X3D>
```

X3D

getX3D	setX3D
getScene	setScene
getBackground	setBackground
getViewpoint	setViewpoint
getShape	setShape
getBox	setBox
getApperance	setApperance
getMaterial	setMaterial

X3D Scene Access Interface (SAI)

A Sample of X3D Scene Access API (C++)



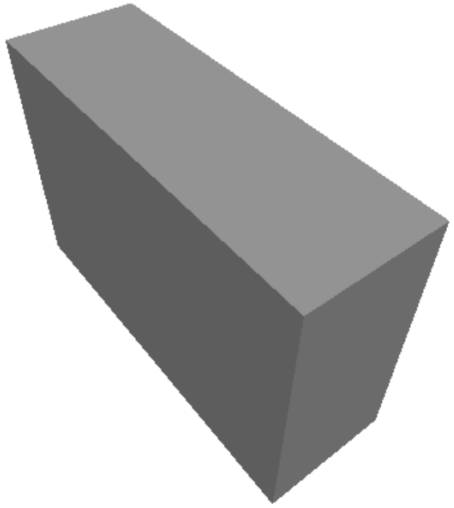
getX3D getScene getBackground getViewpoint getShape getBox getApperance getMaterial	setX3D setScene setBackground setViewpoint setShape setBox setApperance setMaterial

X3D Scene Access Interface (SAI)

getX3D (&pX3D) getScene(&pScene) getBackground(&pBackground) getViewpoint(&pViewpoint) getShape(&pShape) getBox(&pBox) getApperance(&pAppearance) getMaterial(&pMaterial)
setX3D (pX3D) setScene(pScene) setBackground(pBackground) setViewpoint(pViewpoint) setShape(pShape) setBox(pBox) setApperance(pAppearance) setMaterial(pMaterial)

X3D C++ encoding

A Sample of X3D Scene Access API (C#)



getX3D	setX3D
getScene	setScene
getBackground	setBackground
getViewpoint	setViewpoint
getShape	setShape
getBox	setBox
getApperance	setApperance
getMaterial	setMaterial

X3D Scene Access Interface (SAI)

```
getX3D (pX3D)
getScene(pScene)
getBackground(pBackground)
getViewpoint(pViewpoint)
getShape(pShape)
getBox(pBox)
getApperance(pAppearance)
getMaterial(pMaterial)
```

```
setX3D (pX3D)
setScene(pScene)
setBackground(pBackground)
setViewpoint(pViewpoint)
setShape(pShape)
setBox(pBox)
setApperance(pAppearance)
setMaterial(pMaterial)
```

X3D C# encoding

X3D C++ Binding Viewer Program Example

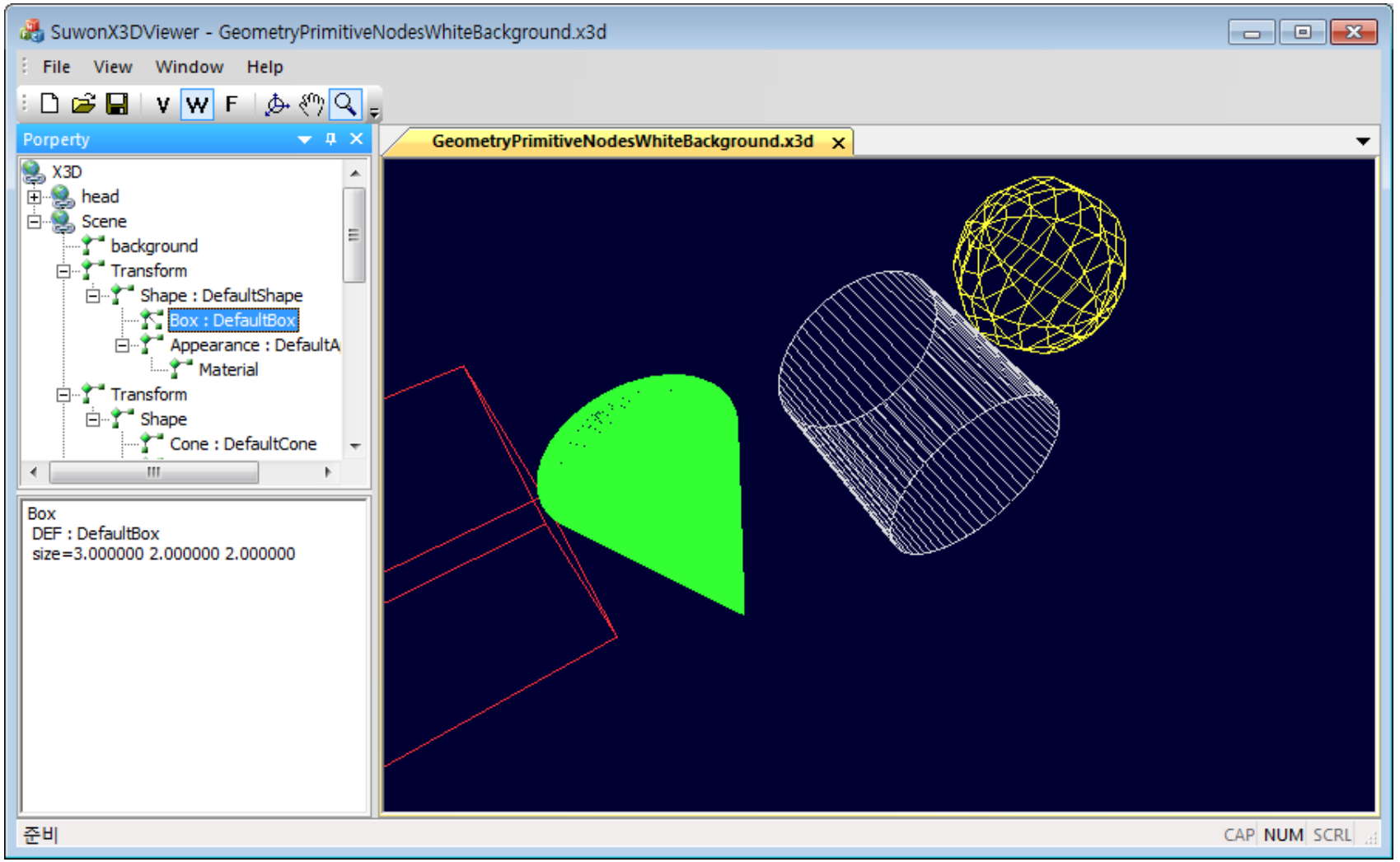
1. SuwonX3DBindingViewer
   1) Load X3DLib.dll
   2) Parse an X3D file with X3DLib
   3) Read, update, draw, and store the X3D file using X3DLib classes

2. X3DLib.dll
   1) Parse an X3D file
   2) Insert the parsed X3D into an internal class
   3) Provide an interface to read X3D

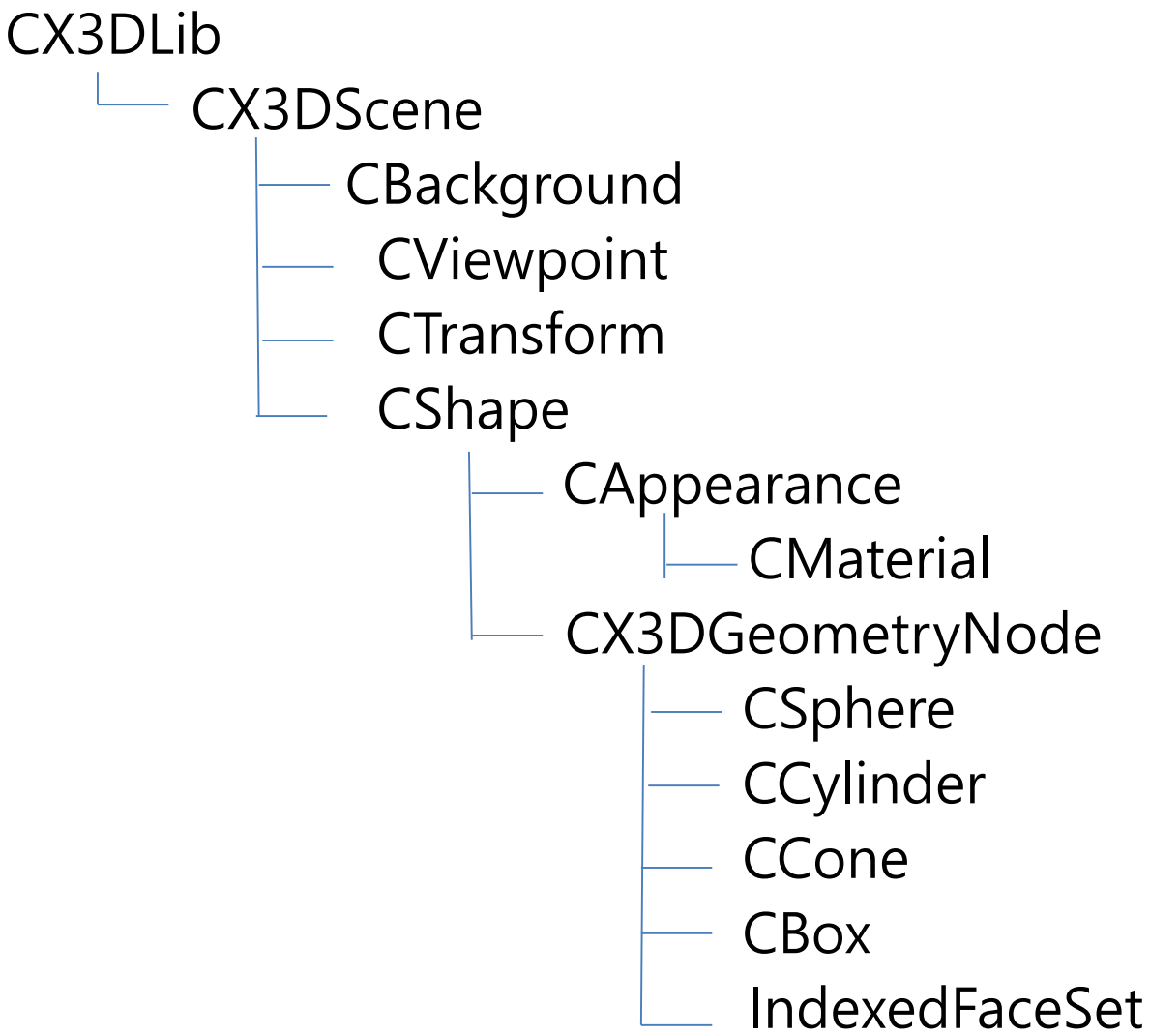
SuwonX3DBindingViewer (X3D C++ Binding Viewer)

X3D Tree View

Property View



X3D C++ Binding Class Structure (Partial)



Background

```
//C.3.6 Background
/** Background defines a concrete node interface that extends interface X3DBackgroundNode. */

class AFX_EXT_CLASS CBackground : public CX3DBackgroundNode
{
        DECLARE_DYNAMIC(CBackground);
public:
        CBackground();
        virtual ~CBackground();
//Implimentation
public:
        virtual void Draw();
        virtual CString toXMLString();
        virtual CString getPropertyString();

        /** Return array of String results array [] from MFString inputOutput field named "backUrl" */
        CString* getBackUrl ();

        /** Return number of primitive values in "backUrl" array */
        int getNumBackUrl ();

        /** Assign String array [] to MFString inputOutput field named "backUrl" */
        void setBackUrl (CString* values, int size);
```

```
X3DBackgroundNode : X3DBindableNode {
  SFBool   [in]     set_bind
  MFFloat  [in,out] groundAngle   []       [0,π/2]
  MFColor  [in,out] groundColor   []       [0,1]
  SFNode   [in,out] metadata      NULL     [X3DMetadataObject]
  MFFloat  [in,out] skyAngle      []       [0,π]
  MFColor  [in,out] skyColor      0 0 0    [0,1]
  SFFloat  [in,out] transparency  0        [0,1]
  SFTime   [out]    bindTime
  SFBool   [out]    isBound
}
```

Background

```
/** Assign single String value [] as the MFString array for inputOutput field named "backUrl" */
void setBackUrl (CString value);

/** Return array of String results array [] from MFString inputOutput field named "bottomUrl" */
CString* getBottomUrl ();

/** Return number of primitive values in "bottomUrl" array */
int getNumBottomUrl ();

/** Assign String array [] to MFString inputOutput field named "bottomUrl" */
void setBottomUrl (CString* values, int size);

/** Assign single String value [] as the MFString array for inputOutput field named "bottomUrl" */
void setBottomUrl (CString value);

/** Return array of String results array [] from MFString inputOutput field named "frontUrl" */
CString* getFrontUrl ();

/** Return number of primitive values in "frontUrl" array */
int getNumFrontUrl ();

/** Assign String array [] to MFString inputOutput field named "frontUrl" */
void setFrontUrl (CString* values, int size);
```

Background

```
/** Assign single String value [] as the MFString array for inputOutput field named "frontUrl" */
void setFrontUrl (CString value);

/** Return array of String results array [] from MFString inputOutput field named "leftUrl" */
CString* getLeftUrl ();

/** Return number of primitive values in "leftUrl" array */
int getNumLeftUrl ();

/** Assign String array [] to MFString inputOutput field named "leftUrl" */
void setLeftUrl (CString* values, int size);

/** Assign single String value [] as the MFString array for inputOutput field named "leftUrl" */
void setLeftUrl (CString value);

/** Return array of String results array [] from MFString inputOutput field named "rightUrl" */
CString* getRightUrl ();

/** Return number of primitive values in "rightUrl" array */
int getNumRightUrl ();

/** Assign String array [] to MFString inputOutput field named "rightUrl" */
void setRightUrl (CString* values, int size);
```

Background

```
    /** Assign single String value [] as the MFString array for inputOutput field named "rightUrl" */
    void setRightUrl (CString value);

    /** Return array of String results array [] from MFString inputOutput field named "topUrl" */
    CString* getTopUrl ();

    /** Return number of primitive values in "topUrl" array */
    int getNumTopUrl ();

    /** Assign String array [] to MFString inputOutput field named "topUrl" */
    void setTopUrl (CString* values, int size);

    /** Assign single String value [] as the MFString array for inputOutput field named "topUrl" */
    void setTopUrl (CString value);
}
```

Viewpoint

```
//C.3.242 Viewpoint
/** Viewpoint defines a concrete node interface that extends interface X3DViewpointNode. */
class AFX_EXT_CLASS CViewpoint : public CX3DViewpointNode
{
        DECLARE_DYNAMIC(CViewpoint);
public:
        CViewpoint();
        virtual ~CViewpoint();
//Implimentation
public:
        virtual void Draw();
        virtual CString toXMLString();
        virtual CString getPropertyString();

        /** Return array of 3-tuple float results array in radians from SFVec3f inputOutput field named "centerOfRotation" */
        float* getCenterOfRotation ();

        /** Assign 3-tuple float array in radians to SFVec3f inputOutput field named "centerOfRotation" */
        void setCenterOfRotation (float* value);
```

```
Viewpoint : X3DViewpointNode {
  SFBool     [in]     set_bind
  SFVec3f    [in,out] centerOfRotation  0 0 0    (-∞,∞)
  SFString   [in,out] description       ""
  SFFloat    [in,out] fieldOfView       π/4      (0,π)
  SFBool     [in,out] jump              TRUE
  SFNode     [in,out] metadata          NULL     [X3DMetadataObject]
  SFRotation [in,out] orientation       0 0 1 0  [-1,1],(-∞,∞)
  SFVec3f    [in,out] position          0 0 10   (-∞,∞)
  SFBool     [in,out] retainUserOffsets FALSE
  SFTime     [out]    bindTime
  SFBool     [out]    isBound
}
```

Viewpoint

```
        /** Return float result [] from SFFloat inputOutput field named "fieldOfView" */
        float getFieldOfView ();

        /** Assign float value [] to SFFloat inputOutput field named "fieldOfView" */
        void setFieldOfView (float value);

        /** Return array of 3-tuple float results array [] from SFVec3f inputOutput field named "position" */
        float* getPosition ();

        /** Assign 3-tuple float array [] to SFVec3f inputOutput field named "position" */
        void setPosition (float* value);
}
```

Transform

```
//C.3.232 Transform
/** Transform defines a concrete node interface that extends interface X3DGroupingNode. */

class AFX_EXT_CLASS CTransform : public CX3DGroupingNode
{
        DECLARE_DYNAMIC(CTransform);

public:
        CTransform();
        virtual ~CTransform();

//Implimentation
public:
        virtual void Draw();
        virtual CString toXMLString();
        virtual CString getPropertyString();

        /** Return array of 3-tuple float results array [] from SFVec3f inputOutput field named "center" */
        float* getCenter ();

        /** Assign 3-tuple float array [] to SFVec3f inputOutput field named "center" */
        void setCenter (float* value);
```

```
Transform : X3DGroupingNode {
  MFNode     [in]     addChildren               [X3DChildNode]
  MFNode     [in]     removeChildren            [X3DChildNode]
  SFVec3f    [in,out] center            0 0 0    (-∞,∞)
  MFNode     [in,out] children          []       [X3DChildNode]
  SFNode     [in,out] metadata          NULL     [X3DMetadataObject]
  SFRotation [in,out] rotation          0 0 1 0  [-1,1] or (-∞,∞)
  SFVec3f    [in,out] scale             1 1 1    (-∞, ∞)
  SFRotation [in,out] scaleOrientation  0 0 1 0  [-1,1] or (-∞,∞)
  SFVec3f    [in,out] translation       0 0 0    (-∞,∞)
  SFVec3f    []       bboxCenter        0 0 0    (-∞,∞)
  SFVec3f    []       bboxSize          -1 -1 -1 [0,∞) or -1 -1 -1
}
```

Transform

```
        /** Return array of 4-tuple float results array in radians from SFRotation inputOutput field named "rotation" */
        float* getRotation ();

        /** Assign 4-tuple float array in radians to SFRotation inputOutput field named "rotation" */
        void setRotation (float* value);

        /** Return array of 3-tuple float results array [] from SFVec3f inputOutput field named "scale" */
        float* getScale ();

        /** Assign 3-tuple float array [] to SFVec3f inputOutput field named "scale" */
        void setScale (float* value);

        /** Return array of 4-tuple float results array in radians from SFRotation inputOutput field named "scaleOrientation" */
        float* getScaleOrientation ();

        /** Assign 4-tuple float array in radians to SFRotation inputOutput field named "scaleOrientation" */
        void setScaleOrientation (float* value);

        /** Return array of 3-tuple float results array [] from SFVec3f inputOutput field named "translation" */
        float* getTranslation ();

        /** Assign 3-tuple float array [] to SFVec3f inputOutput field named "translation" */
        void setTranslation (float* value);
}
```

Shape

```
//C.3.199 Shape
/** Shape defines a concrete node interface that extends interface X3DShapeNode. */
class AFX_EXT_CLASS CShape : public CX3DShapeNode
{
		DECLARE_DYNAMIC(CShape);
public:
		CShape();
		virtual ~CShape();

//Implimentation
public:
		virtual void Draw();
		virtual CString toXMLString();
		virtual CString getPropertyString();

		/** Return X3DAppearanceNode result (using a properly typed node or X3DPrototypeInstance) from SFNode inputOutput field named "appearance" */
		void getAppearance (CX3DNode result);

		/** Assign X3DAppearanceNode value (using a properly typed node) to SFNode inputOutput field named "appearance" */
		void setAppearance (CX3DAppearanceNode node);
```

```
Shape : X3DShapeNode {
  SFNode   [in,out] appearance NULL      [X3DAppearanceNode]
  SFNode   [in,out] geometry   NULL      [X3DGeometryNode]
  SFNode   [in,out] metadata   NULL      [X3DMetadataObject]
  SFVec3f  []       bboxCenter 0 0 0     (-∞,∞)
  SFVec3f  []       bboxSize   -1 -1 -1  [0,∞) or -1 -1 -1
}
```

Shape

```
        /** Assign X3DAppearanceNode value (using a properly typed protoInstance) */
        void setAppearance (CX3DPrototypeInstance protoInstance);

        /** Return X3DGeometryNode result (using a properly typed node or X3DPrototypeInstance) from SFNode inputOutput field named "geometry" */
        void getGeometry (CX3DNode result);

        /** Assign X3DGeometryNode value (using a properly typed node) to SFNode inputOutput field named "geometry" */
        void setGeometry (CX3DGeometryNode node);

        /** Assign X3DGeometryNode value (using a properly typed protoInstance) */
        void setGeometry (CX3DPrototypeInstance protoInstance);
}
```

Appearance

```
//C.3.2 Appearance
/** Appearance defines a concrete node interface that extends interface X3DAppearanceNode. */
class AFX_EXT_CLASS CAppearance : public CX3DAppearanceNode
{
        DECLARE_DYNAMIC(CAppearance);
public:
        CAppearance();
        virtual ~CAppearance();

//Implimentation
public:
        virtual void Draw();
        virtual CString toXMLString();
        virtual CString getPropertyString();

        /** Return array of X3DShaderNode results array (using a properly typed node array or X3DPrototypeInstance array) from
MFNode inputOutput field named "shaders" */
        CX3DNode* getShaders ();

        /** Return number of nodes in "shaders" array */
        int getNumShaders ();

        /** Assign X3DShaderNode array (using a properly typed node array) to MFNode inputOutput field named "shaders" */
        void setShaders (CX3DShaderNode* nodes, int size);
```

```
Appearance : X3DAppearanceNode {
  SFNode [in,out] fillProperties   NULL [FillProperties]
  SFNode [in,out] lineProperties   NULL [LineProperties]
  SFNode [in,out] material         NULL [X3DMaterialNode]
  SFNode [in,out] metadata         NULL [X3DMetadataObject]
  MFNode [in,out] shaders          []   [X3DShaderNode]
  SFNode [in,out] texture          NULL [X3DTextureNode]
  SFNode [in,out] textureTransform NULL [X3DTextureTransformNode]
}
```

Appearance

```
/** Assign single X3DShaderNode value (using a properly typed node) as the MFNode array for inputOutput field named "shaders" */
void setShaders (CX3DShaderNode node);

/** Assign X3DShaderNode array (using a properly typed protoInstance array) to MFNode inputOutput field named "shaders" */
void setShaders (CX3DPrototypeInstance node);

/** Assign X3DShaderNode array (using a properly typed node array) to MFNode inputOutput field named "shaders" */
void setShaders (CX3DNode* nodes, int size);

/** Return FillProperties result (using a properly typed node or X3DPrototypeInstance) from SFNode inputOutput field named "fillProperties" */
void getFillProperties (CX3DNode result);

/** Assign FillProperties value (using a properly typed node) to SFNode inputOutput field named "fillProperties" */
void setFillProperties (CFillProperties node);

/** Assign FillProperties value (using a properly typed protoInstance) */
void setFillProperties (CX3DPrototypeInstance protoInstance);

/** Return LineProperties result (using a properly typed node or X3DPrototypeInstance) from SFNode inputOutput field named "lineProperties" */
void getLineProperties (CX3DNode result);
```

Appearance

```
/** Assign LineProperties value (using a properly typed node) to SFNode inputOutput field named "lineProperties" */
void setLineProperties (CLineProperties node);

/** Assign LineProperties value (using a properly typed protoInstance) */
void setLineProperties (CX3DPrototypeInstance protoInstance);

/** Return X3DMaterialNode result (using a properly typed node or X3DPrototypeInstance) from SFNode inputOutput field named "material" */
void getMaterial (CX3DNode result);

/** Assign X3DMaterialNode value (using a properly typed node) to SFNode inputOutput field named "material" */
void setMaterial (CX3DMaterialNode node);

/** Assign X3DMaterialNode value (using a properly typed protoInstance) */
void setMaterial (CX3DPrototypeInstance protoInstance);

/** Return X3DTextureNode result (using a properly typed node or X3DPrototypeInstance) from SFNode inputOutput field named "texture" */
void getTexture (CX3DNode result);

/** Assign X3DTextureNode value (using a properly typed node) to SFNode inputOutput field named "texture" */
void setTexture (CX3DTextureNode node);
```

Appearance

```
        /** Assign X3DTextureNode value (using a properly typed protoInstance) */
        void setTexture (CX3DPrototypeInstance protoInstance);

        /** Return X3DTextureTransformNode result (using a properly typed node or X3DPrototypeInstance) from SFNode inputOutput field named "textureTransform" */
        void getTextureTransform (CX3DNode result);

        /** Assign X3DTextureTransformNode value (using a properly typed node) to SFNode inputOutput field named "textureTransform" */
        void setTextureTransform (CX3DTextureTransformNode node);

        /** Assign X3DTextureTransformNode value (using a properly typed protoInstance) */
        void setTextureTransform (CX3DPrototypeInstance protoInstance);
}
```

Material

```
//C.3.121 Material
/** Material defines a concrete node interface that extends interface X3DMaterialNode. */
class AFX_EXT_CLASS CMaterial : public CX3DMaterialNode
{
		DECLARE_DYNAMIC(CMaterial);
public:
		CMaterial();
		virtual ~CMaterial();
//Implimentation
public:
		virtual void Draw();
		virtual CString toXMLString();
		virtual CString getPropertyString();

		/** Return float result [] from intensityType type inputOutput field named "ambientIntensity" */
		float getAmbientIntensity ();

		/** Assign float value [] to intensityType type inputOutput field named "ambientIntensity" */
		void setAmbientIntensity (float value);

		/** Return array of 3-tuple float results array using RGB values [0..1] from SFColor inputOutput field named "diffuseColor" */
		float* getDiffuseColor ();
		/** Assign 3-tuple float array using RGB values [0..1] to SFColor inputOutput field named "diffuseColor" */
		void setDiffuseColor (float* color);
```

```
Material : X3DMaterialNode {
  SFFloat [in,out] ambientIntensity 0.2          [0,1]
  SFColor [in,out] diffuseColor     0.8 0.8 0.8  [0,1]
  SFColor [in,out] emissiveColor    0 0 0        [0,1]
  SFNode  [in,out] metadata         NULL         [X3DMetadataObject]
  SFFloat [in,out] shininess        0.2          [0,1]
  SFColor [in,out] specularColor    0 0 0        [0,1]
  SFFloat [in,out] transparency     0            [0,1]
}
```

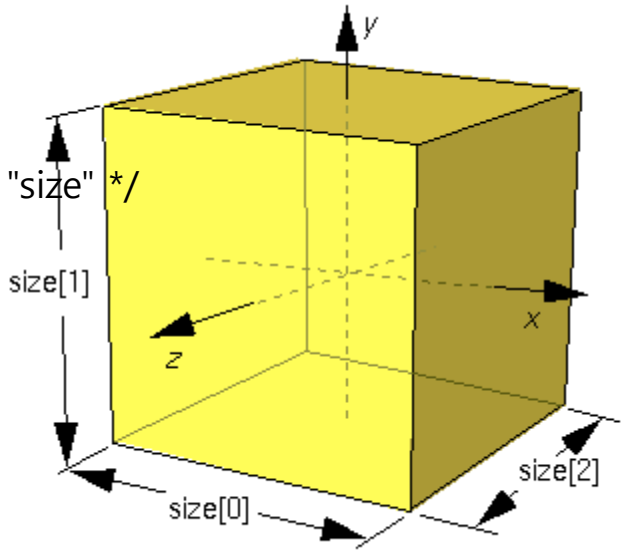
Material

```
        /** Return array of 3-tuple float results array using RGB values [0..1] from SFColor inputOutput field named "emissiveColor" */
        float* getEmissiveColor ();

        /** Assign 3-tuple float array using RGB values [0..1] to SFColor inputOutput field named "emissiveColor" */
        void setEmissiveColor (float* color);

        /** Return float result [] from intensityType type inputOutput field named "shininess" */
        float getShininess ();

        /** Assign float value [] to intensityType type inputOutput field named "shininess" */
        void setShininess (float value);

        /** Return array of 3-tuple float results array using RGB values [0..1] from SFColor inputOutput field named "specularColor" */
        float* getSpecularColor ();

        /** Assign 3-tuple float array using RGB values [0..1] to SFColor inputOutput field named "specularColor" */
        void setSpecularColor (float* color);

        /** Return float result [] from intensityType type inputOutput field named "transparency" */
        float getTransparency ();

        /** Assign float value [] to intensityType type inputOutput field named "transparency" */
        void setTransparency (float value);
}
```

Box

```
//C.3.16 Box
/** Box defines a concrete node interface that extends interface X3DGeometryNode. */
class AFX_EXT_CLASS CBox : public CX3DGeometryNode
{
		DECLARE_DYNAMIC(CBox);
public:
		CBox();
		virtual ~CBox();
//Implimentation
public:
		virtual void Draw();
		virtual CString toXMLString();
		virtual CString getPropertyString();
		/** Return array of 3-tuple float results array [] from SFVec3f initializeOnly field named "size" */
		float* getSize ();
		/** Assign 3-tuple float array [] to SFVec3f initializeOnly field named "size" */
		void setSize (float* value);
		/** Return boolean result from SFBool initializeOnly field named "solid" */
		bool getSolid ();
		/** Assign boolean value to SFBool initializeOnly field named "solid" */
		void setSolid (bool value);
}
```

```
Box : X3DGeometryNode {
  SFNode  [in,out] metadata NULL  [X3DMetadataObject]
  SFVec3f []       size     2 2 2 (0,∞)
  SFBool  []       solid    TRUE
}
```



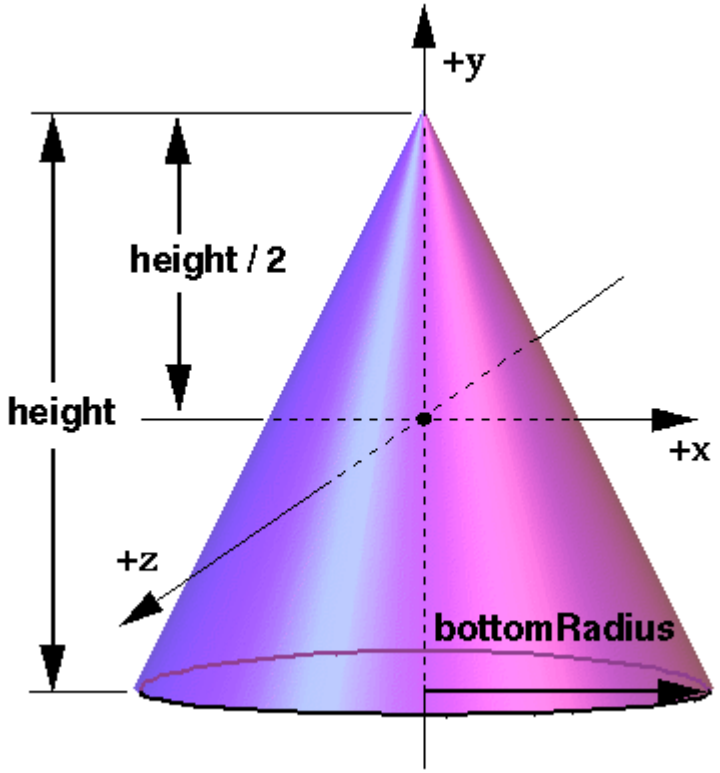
Cone

```
//C.3.40 Cone
/** Cone defines a concrete node interface that extends interface X3DGeometryNode. */

class AFX_EXT_CLASS CCone : public CX3DGeometryNode
{
		DECLARE_DYNAMIC(CCone);
public:
		CCone();
		virtual ~CCone();
//Implementation
public:
		virtual void Draw();
		virtual CString toXMLString();
		virtual CString getPropertyString();

		/** Return float result [] from  type initializeOnly field named "bottomRadius" */
		float getBottomRadius ();

		/** Assign float value [] to  type initializeOnly field named "bottomRadius" */
		void setBottomRadius (float value);
```

```
Cone : X3DGeometryNode {
  SFNode  [in,out] metadata     NULL [X3DMetadataObject]
  SFBool  []       bottom       TRUE
  SFFloat []       bottomRadius 1    (0,∞)
  SFFloat []       height       2    (0,∞)
  SFBool  []       side         TRUE
  SFBool  []       solid        TRUE
}
```

Cone

```
/** Return float result [] from  type initializeOnly field named "height" */
float getHeight ();
/** Assign float value [] to  type initializeOnly field named "height" */
void setHeight (float value);
/** Return boolean result from SFBool initializeOnly field named "side" */
bool getSide ();

/** Assign boolean value to SFBool initializeOnly field named "side" */
void setSide (bool value);

/** Return boolean result from SFBool initializeOnly field named "bottom" */
bool getBottom ();

/** Assign boolean value to SFBool initializeOnly field named "bottom" */
void setBottom (bool value);

/** Return boolean result from SFBool initializeOnly field named "solid" */
bool getSolid ();

/** Assign boolean value to SFBool initializeOnly field named "solid" */
void setSolid (bool value);
}
```



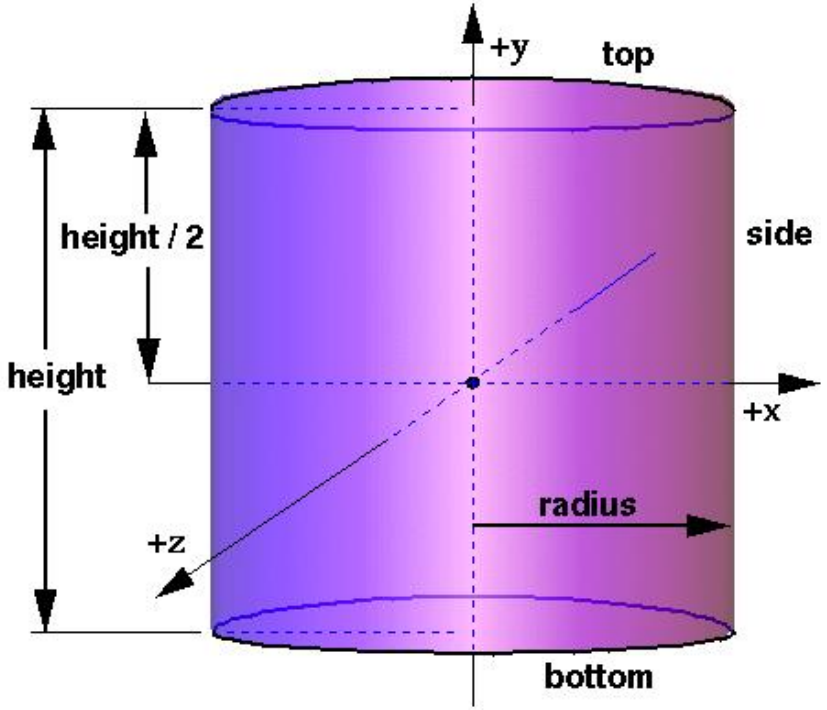
Cylinder

```
//C.3.52 Cylinder
/** Cylinder defines a concrete node interface that extends interface X3DGeometryNode. */

class AFX_EXT_CLASS CCylinder : public CX3DGeometryNode
{
        DECLARE_DYNAMIC(CCylinder);
public:
        CCylinder();
        virtual ~CCylinder();
//Implimentation
public:
        virtual void Draw();
        virtual CString toXMLString();
        virtual CString getPropertyString();

        /** Return boolean result from SFBool initializeOnly field named "bottom" */
        bool getBottom ();

        /** Assign boolean value to SFBool initializeOnly field named "bottom" */
        void setBottom (bool value);

        /** Return float result [] from  type initializeOnly field named "height" */
        float getHeight ();
```

```
Cylinder : X3DGeometryNode {
  SFNode  [in,out] metadata NULL [X3DMetadataObject]
  SFBool  []       bottom   TRUE
  SFFloat []       height   2     (0,∞)
  SFFloat []       radius   1     (0,∞)
  SFBool  []       side     TRUE
  SFBool  []       solid    TRUE
  SFBool  []       top      TRUE
}
```

Cylinder

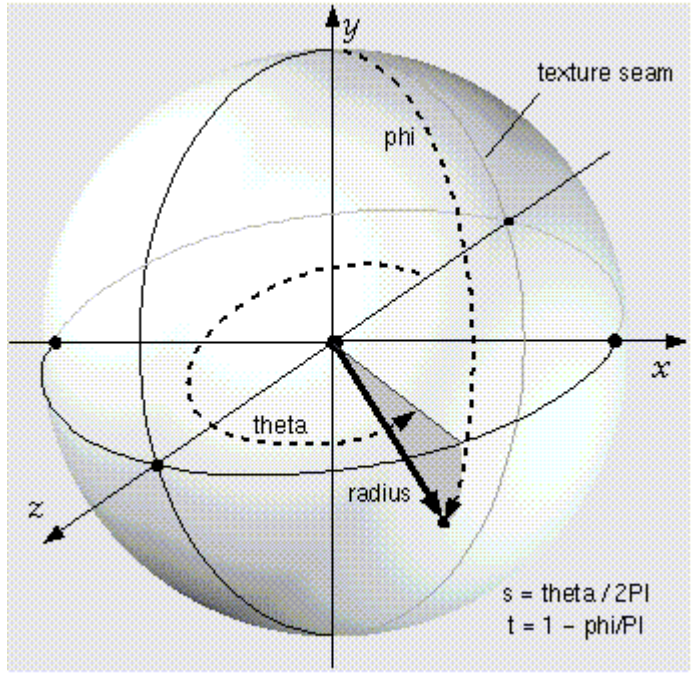
```
/** Assign float value [] to  type initializeOnly field named "height" */
void setHeight (float value);
/** Return float result [] from  type initializeOnly field named "radius" */
float getRadius ();

/** Assign float value [] to  type initializeOnly field named "radius" */
void setRadius (float value);
/** Return boolean result from SFBool initializeOnly field named "side" */
bool getSide ();

/** Assign boolean value to SFBool initializeOnly field named "side" */
void setSide (bool value);

/** Return boolean result from SFBool initializeOnly field named "top" */
bool getTop ();
/** Assign boolean value to SFBool initializeOnly field named "top" */
void setTop (bool value);

/** Return boolean result from SFBool initializeOnly field named "solid" */
bool getSolid ();
/** Assign boolean value to SFBool initializeOnly field named "solid" */
void setSolid (bool value);
}
```



Sphere

```
//C.3.205 Sphere
/** Sphere defines a concrete node interface that extends interface X3DGeometryNode. */
class AFX_EXT_CLASS CSphere : public CX3DGeometryNode
{
		DECLARE_DYNAMIC(CSphere);
public:
		CSphere();
		virtual ~CSphere();
//Implimentation
public:
		virtual void Draw();
		virtual CString toXMLString();
		virtual CString getPropertyString();
		/** Return float result [] from  type initializeOnly field named "radius" */
		float getRadius ();
		/** Assign float value [] to  type initializeOnly field named "radius" */
		void setRadius (float value);
		/** Return boolean result from SFBool initializeOnly field named "solid" */
		bool getSolid ();
		/** Assign boolean value to SFBool initializeOnly field named "solid" */
		void setSolid (bool value);
}
```

```
Sphere : X3DGeometryNode {
  SFNode  [in,out] metadata NULL [X3DMetadataObject]
  SFFloat []       radius   1    (0,∞)
  SFBool  []       solid    TRUE
}
```



IndexedFaceSet

```
//C.3.98 IndexedFaceSet
/** IndexedFaceSet defines a concrete node interface that extends interface X3DComposedGeometryNode. */

class AFX_EXT_CLASS CIndexedFaceSet : public CX3DComposedGeometryNode
{
        DECLARE_DYNAMIC(CIndexedFaceSet);

public:
        CIndexedFaceSet();
        virtual ~CIndexedFaceSet();

//Implimentation
public:
        virtual void Draw();
        virtual CString toXMLString();
        virtual CString getPropertyString();

        /** Assign MFInt32 value using RGB values [0..1] to
            MFInt32 inputOnly field named "set_colorIndex" */
        void setColorIndex (int32_t* colors, int size);
```

```
IndexedFaceSet : X3DComposedGeometryNode {
  MFInt32 [in]      set_colorIndex
  MFInt32 [in]      set_coordIndex
  MFInt32 [in]      set_normalIndex
  MFInt32 [in]      set_texCoordIndex
  MFNode  [in,out] attrib             []    [X3DVertexAttributeNode]
  SFNode  [in,out] color              NULL  [X3DColorNode]
  SFNode  [in,out] coord              NULL  [X3DCoordinateNode]
  SFNode  [in,out] fogCoord           NULL  [FogCoordinate]
  SFNode  [in,out] metadata           NULL  [X3DMetadataObject]
  SFNode  [in,out] normal             NULL  [X3DNormalNode]
  SFNode  [in,out] texCoord           NULL  [X3DTextureCoordinateNode]
  SFBool  []       ccw                TRUE
  MFInt32 []       colorIndex         []    [0,∞) or -1
  SFBool  []       colorPerVertex     TRUE
  SFBool  []       convex             TRUE
  MFInt32 []       coordIndex         []    [0,∞) or -1
  SFFloat []       creaseAngle        0     [0,∞)
  MFInt32 []       normalIndex        []    [0,∞) or -1
  SFBool  []       normalPerVertex    TRUE
  SFBool  []       solid              TRUE
  MFInt32 []       texCoordIndex      []    [-1,∞)
}
```

IndexedFaceSet

```
/** Assign single SFInt32 value using RGB values [0..1] as the MFInt32 array for inputOnly field named "set_colorIndex" */
void setColorIndex (int32_t color);

/** Assign MFInt32 value [] to MFInt32 inputOnly field named "set_coordIndex" */
void setCoordIndex (int32_t* values, int size);

/** Assign single SFInt32 value [] as the MFInt32 array for inputOnly field named "set_coordIndex" */
void setCoordIndex (int32_t value);

/** Assign MFInt32 value [] to MFInt32 inputOnly field named "set_normalIndex" */
void setNormalIndex (int32_t* values, int size);

/** Assign single SFInt32 value [] as the MFInt32 array for inputOnly field named "set_normalIndex" */
void setNormalIndex (int32_t value);

/** Assign MFInt32 value [] to MFInt32 inputOnly field named "set_texCoordIndex" */
void setTexCoordIndex (int32_t* values, int size);

/** Assign single SFInt32 value [] as the MFInt32 array for inputOnly field named "set_texCoordIndex" */
void setTexCoordIndex (int32_t value);
```

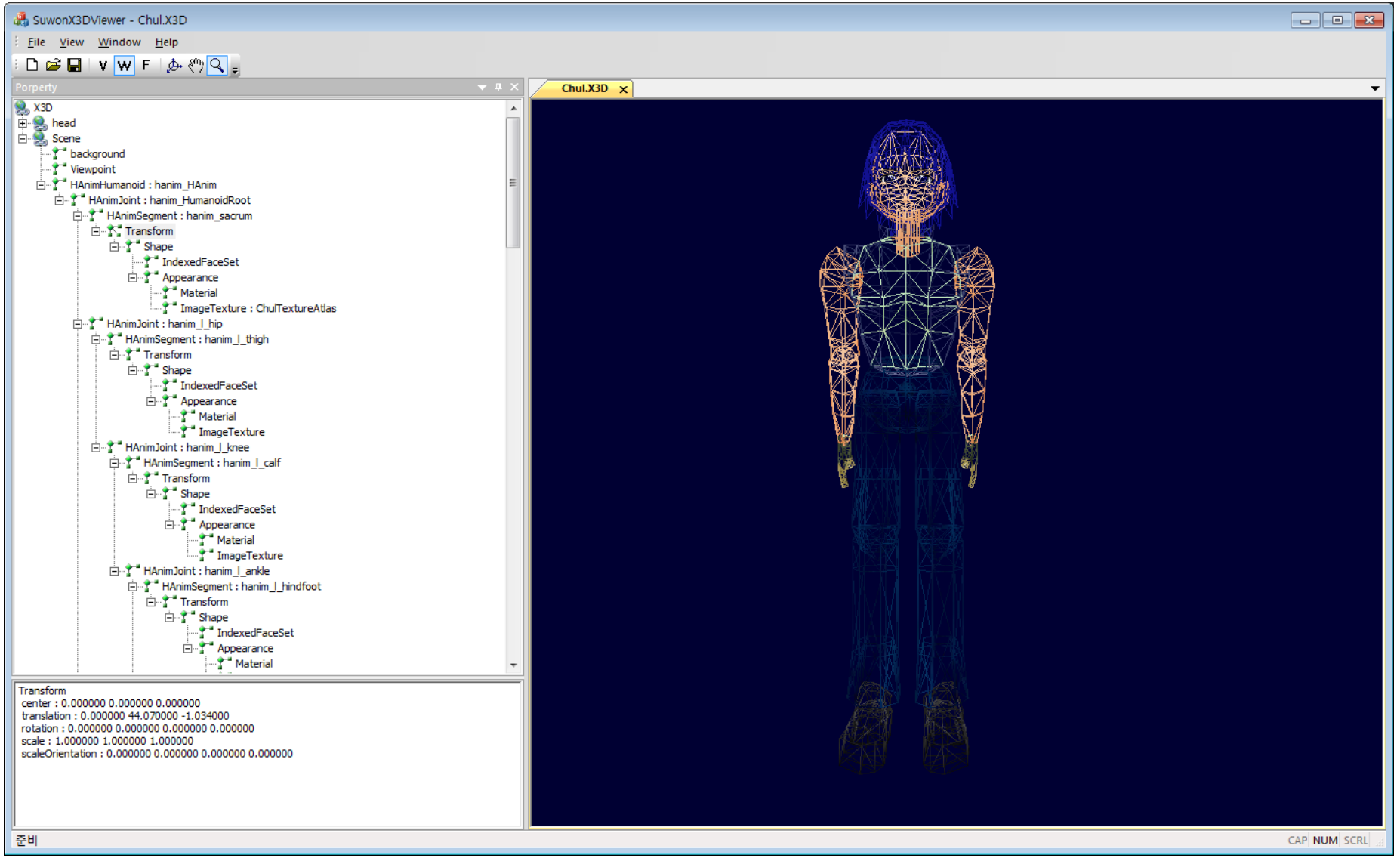
IndexedFaceSet

```
/** Return boolean result from SFBool initializeOnly field named "convex" */
bool getConvex ();

/** Assign boolean value to SFBool initializeOnly field named "convex" */
void setConvex (bool value);

/** Return float result in radians from  type initializeOnly field named "creaseAngle" */
float getCreaseAngle ();

/** Assign float value in radians to  type initializeOnly field named "creaseAngle" */
void setCreaseAngle (float angle);

/** Return MFInt32 result using RGB values [0..1] from MFInt32 initializeOnly field named "colorIndex" */
int32_t* getColorIndex ();

/** Return number of primitive values in "colorIndex" array */
int getNumColorIndex ();

/** Return MFInt32 result [] from MFInt32 initializeOnly field named "coordIndex" */
int32_t* getCoordIndex ();
```

IndexedFaceSet

```
    /** Return number of primitive values in "coordIndex" array */
    int getNumCoordIndex ();

    /** Return MFInt32 result [] from MFInt32 initializeOnly field named "normalIndex" */
    int32_t* getNormalIndex ();

    /** Return number of primitive values in "normalIndex" array */
    int getNumNormalIndex ();

    /** Return MFInt32 result [] from MFInt32 initializeOnly field named "texCoordIndex" */
    int32_t* getTexCoordIndex ();

    /** Return number of primitive values in "texCoordIndex" array */
    int getNumTexCoordIndex ();
}
```

IndexedFaceSet Sample



X3DComposedGeometryNode (IndexedFaceSet Public Node)

```
//B.2.9 X3DComposedGeometryNode
/** X3DComposedGeometryNode defines an abstract node interface that extends interfaces X3DNode.
* Composed geometry nodes produce renderable geometry, can contain Color Coordinate Normal TextureCoordinate, and are contained by a Shape node. */

class AFX_EXT_CLASS CX3DComposedGeometryNode : public CX3DGeometryNode
{
		DECLARE_DYNAMIC(CX3DComposedGeometryNode);

public:
		CX3DComposedGeometryNode();
		virtual ~CX3DComposedGeometryNode();

//Implimentation
public:
		virtual void Draw();
		virtual CString toXMLString();
		virtual CString getPropertyString();

		/** Return bool result from SFBool initializeOnly field named "ccw" */
		bool getCcw ();
```

X3DComposedGeometryNode (IndexedFaceSet Public Node)

```
/** Assign bool value to SFBool initializeOnly field named "ccw" */
void setCcw (bool value);

/** Return bool result from SFBool initializeOnly field named "colorPerVertex" */
bool getColorPerVertex ();

/** Assign bool value to SFBool initializeOnly field named "colorPerVertex" */
void setColorPerVertex (bool color);

/** Return bool result from SFBool initializeOnly field named "normalPerVertex" */
bool getNormalPerVertex ();

/** Assign bool value to SFBool initializeOnly field named "normalPerVertex" */
void setNormalPerVertex (bool value);

/** Return bool result from SFBool initializeOnly field named "solid" */
bool getSolid ();

/** Assign bool value to SFBool initializeOnly field named "solid" */
void setSolid (bool value);
```

X3DComposedGeometryNode (IndexedFaceSet Public Node)

```
        /** Return array of X3DVertexAttributeNode results array (using a properly typed node array or X3DPrototypeInstance array) from MFNode inputOutput field named "attrib" */
        CX3DNode* getAttrib ();

        /** Return number of nodes in "attrib" array */
        int getNumAttrib ();

        /** Assign X3DVertexAttributeNode array (using a properly typed node array) to MFNode inputOutput field named "attrib" */
        void setAttrib (CX3DVertexAttributeNode* nodes);

        /** Assign single X3DVertexAttributeNode value (using a properly typed node) as the MFNode array for inputOutput field named "attrib" */
        void setAttrib (CX3DVertexAttributeNode node);

        /** Assign X3DVertexAttributeNode array (using a properly typed protoInstance array) to MFNode inputOutput field named "attrib" */
        void setAttrib (CX3DPrototypeInstance node);

        /** Assign X3DVertexAttributeNode array (using a properly typed node array) to MFNode inputOutput field named "attrib" */
        void setAttrib (CX3DNode* nodes);

        /** Return X3DColorNode result (using a properly typed node or X3DPrototypeInstance) from SFNode inputOutput field named "color" */
        void getColor (CX3DNode result);
```

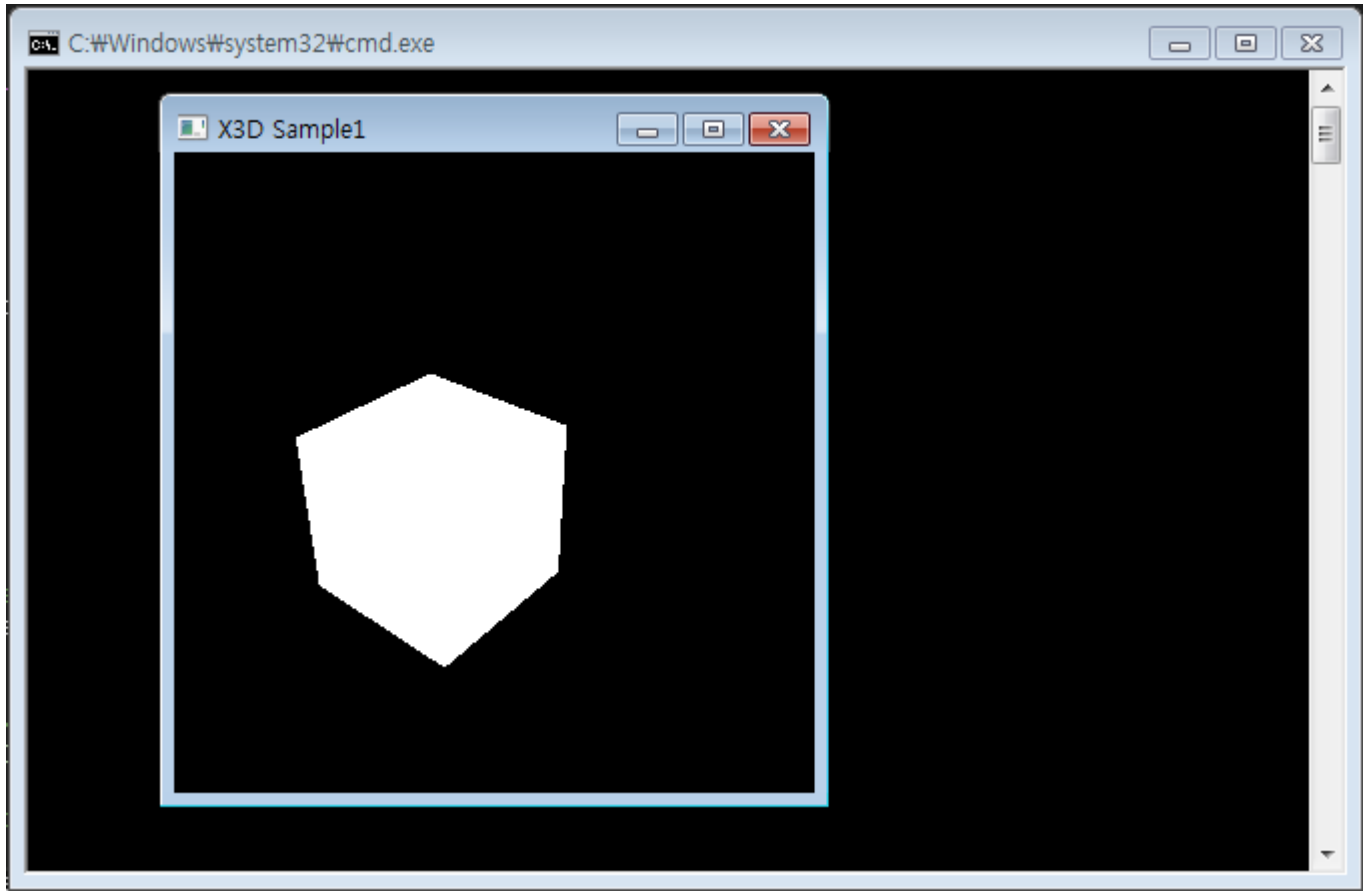
X3DComposedGeometryNode (IndexedFaceSet Public Node)

```
        /** Assign X3DColorNode value (using a properly typed node) to SFNode inputOutput field named "color" */
        void setColor (CX3DColorNode color);

        /** Assign X3DColorNode value (using a properly typed protoInstance) */
        void setColor (CX3DPrototypeInstance protoInstance);

        /** Return X3DCoordinateNode result (using a properly typed node or X3DPrototypeInstance) from SFNode inputOutput field named "coord" */
        //CX3DNode* getCoord ();
        void getCoord (CX3DNode result);

        /** Assign X3DCoordinateNode value (using a properly typed node) to SFNode inputOutput field named "coord" */
        void setCoord (CX3DCoordinateNode node);

        /** Assign X3DCoordinateNode value (using a properly typed protoInstance) */
        void setCoord (CX3DPrototypeInstance protoInstance);

        /** Return FogCoordinate result (using a properly typed node or X3DPrototypeInstance) from SFNode inputOutput field named "fogCoord" */
        void getFogCoord (CX3DNode result);

        /** Assign FogCoordinate value (using a properly typed node) to SFNode inputOutput field named "fogCoord" */
        void setFogCoord (CFogCoordinate node);
```

X3DComposedGeometryNode (IndexedFaceSet Public Node)

```
        /** Assign FogCoordinate value (using a properly typed protoInstance) */
        void setFogCoord (CX3DPrototypeInstance protoInstance);

        /** Return X3DNormalNode result (using a properly typed node or X3DPrototypeInstance) from SFNode inputOutput field named "normal" */
        //CX3DNode* getNormal ();
        void getNormal (CX3DNode result);

        /** Assign X3DNormalNode value (using a properly typed node) to SFNode inputOutput field named "normal" */
        void setNormal (CX3DNormalNode node);

        /** Assign X3DNormalNode value (using a properly typed protoInstance) */
        void setNormal (CX3DPrototypeInstance protoInstance);

        /** Return X3DTextureCoordinateNode result (using a properly typed node or X3DPrototypeInstance) from SFNode inputOutput field named "texCoord" */
        void getTexCoord (CX3DNode result);

        /** Assign X3DTextureCoordinateNode value (using a properly typed node) to SFNode inputOutput field named "texCoord" */
        void setTexCoord (CX3DTextureCoordinateNode node);

        /** Assign X3DTextureCoordinateNode value (using a properly typed protoInstance) */
        void setTexCoord (CX3DPrototypeInstance protoInstance);
}
```

X3D C++ Binding Sample



X3D C++ Binding Sample Source (Win32)

```
#include <glut.h>
#include "..₩X3DLib₩X3DLib.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#endif


CWinApp theApp;
using namespace std;
CX3DScene* m_pScene = NULL;

void changeSize(int w, int h) {

    if(h == 0)
        h = 1;

    float ratio = 1.0* w / h;

    // Reset the coordinate system before modifying
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();

    // Set the viewport to be the entire window
    glViewport(0, 0, w, h);
```

GLUT Library include

X3D C++ Library include

X3D C++ Binding Sample Source (Win32)

```
// Set the correct perspective.
    gluPerspective(45,ratio,1,1000);
    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
    gluLookAt(5.0,5.0,5.0,
              0.0,0.0,-1.0,
              0.0f,1.0f,0.0f);
}

void DrawNode(CX3DNode *pNode)
{
        if(pNode==NULL)
                return;

        ::glPushMatrix();
        if (pNode->isType(NODE_SHAPE))
                ((CX3DShapeNode*)pNode)->geometry->Draw();

        ::glPopMatrix();
}

void renderScene(void)
{
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    glPushMatrix();
```

X3D C++ Binding Sample Source (Win32)

```
int nCount = m_pScene->m_Objects.GetCount();
                for(int i=0; i<nCount; i++)
                {
                                CX3DNode* pChild =(CX3DNode*)(m_pScene->m_Objects).GetAt(i);
                                if (pChild)
                                                DrawNode(pChild);
                }

    glPopMatrix();
    glutSwapBuffers();
}


void initialize(void)
{
                m_pScene = new CX3DScene();

                CX3DShapeNode* shape = new CX3DShapeNode;
                CBox* box = new CBox;
                shape->setGeometry((CX3DGeometryNode*)box);

                m_pScene->AddNode(shape, NULL);
}
```

X3D C++ Binding Sample Source (Win32)

```
int _tmain(int argc, TCHAR* argv[], TCHAR* envp[])
{
		int nRetCode = 0;

		HMODULE hModule = ::GetModuleHandle(NULL);

		if (hModule != NULL)
		{
			if (!AfxWinInit(hModule, NULL, ::GetCommandLine(), 0))
			{
				_tprintf(_T("심각한 오류: MFC를 초기화하지 못했습니다.₩n"));
				nRetCode = 1;
			}
			else
			{
				glutInit(&argc, argv);
				glutInitDisplayMode(GLUT_DEPTH | GLUT_DOUBLE | GLUT_RGBA);
				glutInitWindowPosition(100,100);
				glutInitWindowSize(320,320);
				glutCreateWindow("X3D Sample1");

				initialize();

				glutDisplayFunc(renderScene);
				glutIdleFunc(renderScene);
				glutReshapeFunc(changeSize);
```

X3D C++ Binding Sample Source (Win32)

```
                                        glutMainLoop();
                                }
        }
    else
        {
                                _tprintf(_T("심각한 오류: GetModuleHandle 실패₩n"));
                                nRetCode = 1;
        }


        return nRetCode;
}
```

X3D C++ Binding Sample Source (Win32)

```
#include <glut.h>
#include "..₩X3DLib₩X3DLib.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#endif


CWinApp theApp;
using namespace std;

CX3DScene* m_pScene = NULL;

void changeSize(int w, int h) {

    if(h == 0)
        h = 1;

    float ratio = 1.0* w / h;

    // Reset the coordinate system before modifying
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();

    // Set the viewport to be the entire window
    glViewport(0, 0, w, h);

    // Set the correct perspective.
    gluPerspective(45,ratio,1,1000);
    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
    gluLookAt(5.0,5.0,5.0,
              0.0,0.0,-1.0,
              0.0f,1.0f,0.0f);
}
```

Include GLUT Library

Include X3D C++ Library

Window size changed

X3D C++ Binding Sample Source (Win32)

```
void DrawNode(CX3DNode *pNode)
{
    if(pNode==NULL)
        return;

    ::glPushMatrix();
    if (pNode->isType(NODE_SHAPE))
        ((CX3DShapeNode*)pNode)->geometry->Draw();

    ::glPopMatrix();
}

void renderScene(void)
{
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    glPushMatrix();

    int nCount = m_pScene->m_Objects.GetCount();
    for(int i=0; i<nCount; i++)
    {
        CX3DNode* pChild =(CX3DNode*)(m_pScene->m_Objects).GetAt(i);
        if (pChild)
            DrawNode(pChild);
    }

    glPopMatrix();

    glutSwapBuffers();
}


void initialize(void)
{
    m_pScene = new CX3DScene();

    CX3DShapeNode* shape = new CX3DShapeNode;
    CBox* box = new CBox;
    shape->setGeometry((CX3DGeometryNode*)box);

    m_pScene->AddNode(shape, NULL);
}
```

Draw a Box

Draw Child Nodes

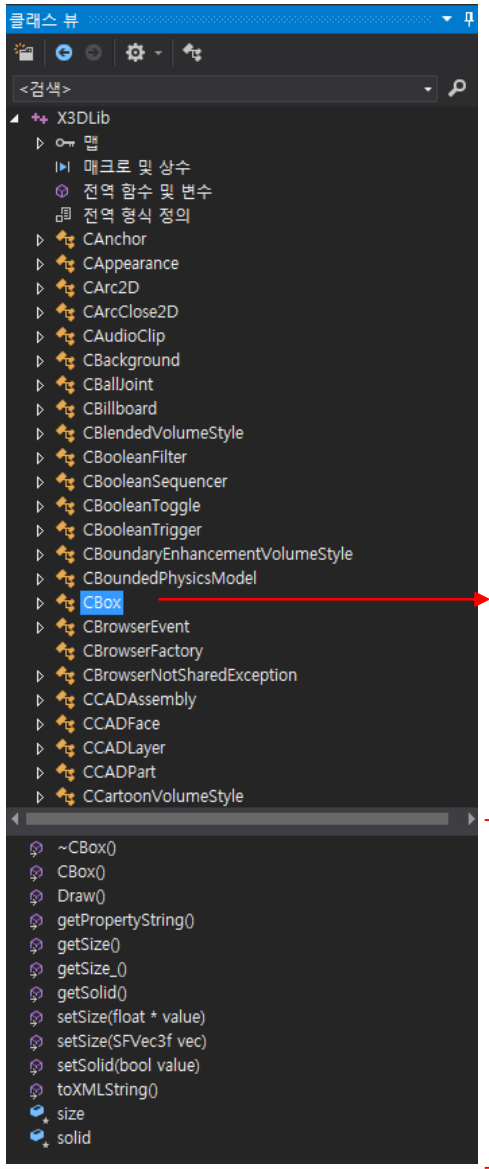
Create a Box

X3D C++ Binding Sample Source (Win32)

```
int _tmain(int argc, TCHAR* argv[], TCHAR* envp[])
{
    int nRetCode = 0;

    HMODULE hModule = ::GetModuleHandle(NULL);

    if (hModule != NULL)
    {
        if (!AfxWinInit(hModule, NULL, ::GetCommandLine(), 0))
        {
            _tprintf(_T("심각한 오류: MFC를 초기화하지 못했습니다.₩n"));
            nRetCode = 1;
        }
        else
        {
            glutInit(&argc, argv);
            glutInitDisplayMode(GLUT_DEPTH | GLUT_DOUBLE | GLUT_RGBA);
            glutInitWindowPosition(100,100);
            glutInitWindowSize(320,320);
            glutCreateWindow("X3D Sample1");

            initialize();

            glutDisplayFunc(renderScene);
            glutIdleFunc(renderScene);
            glutReshapeFunc(changeSize);

            glutMainLoop();
        }
    }
    else
    {
        _tprintf(_T("심각한 오류: GetModuleHandle 실패₩n"));
        nRetCode = 1;
    }


    return nRetCode;
}
```

Create a Window and initialization

X3D C++ Library

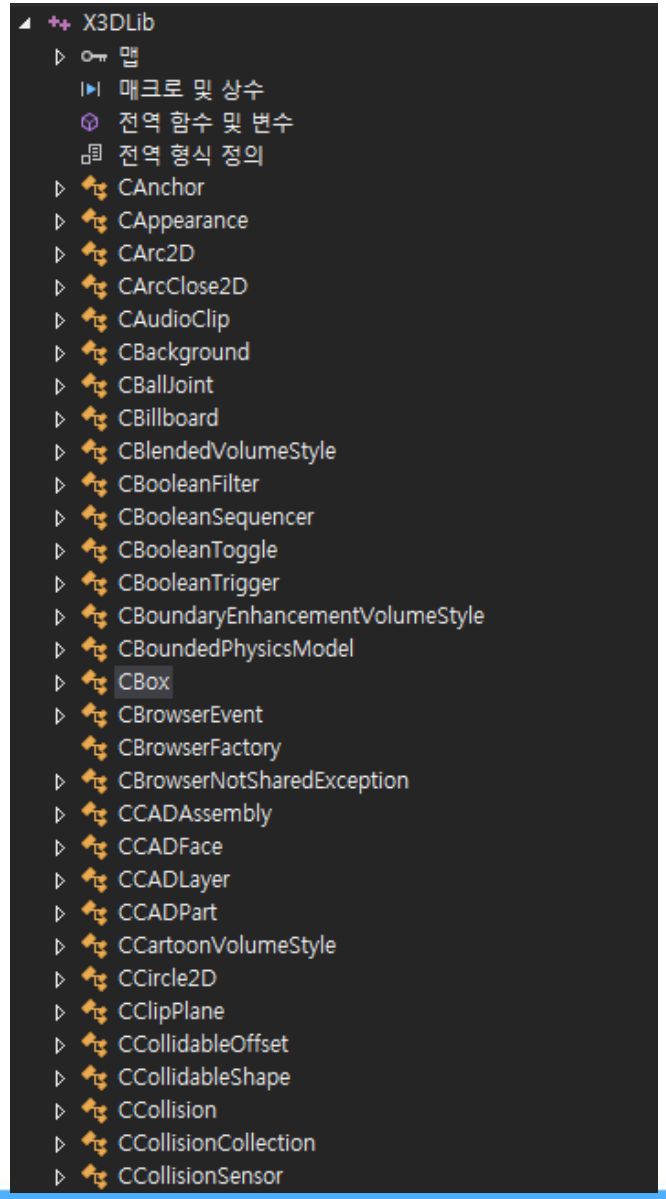


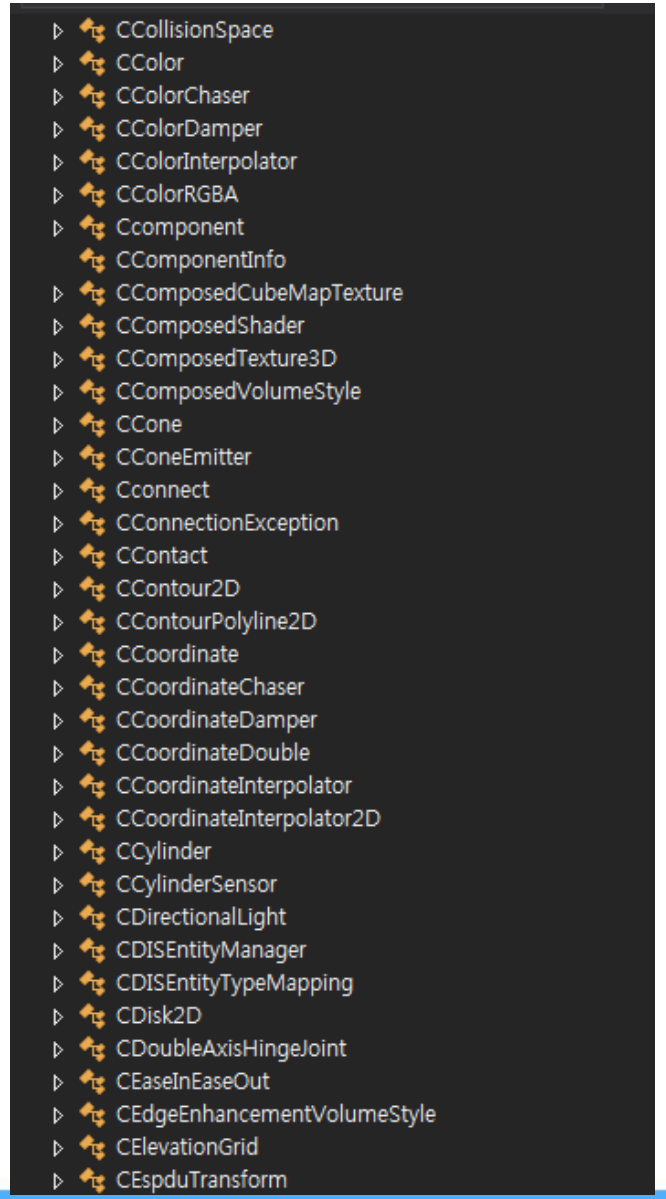
Box Node CBox Class

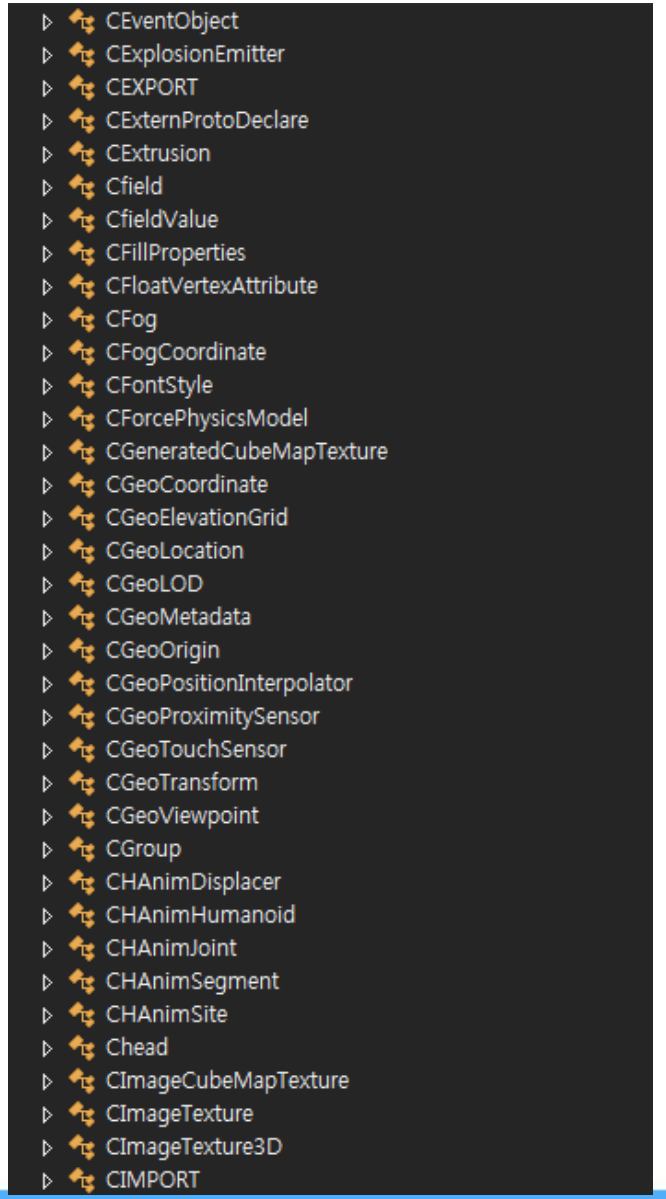
CBox Class function

```
//C.3.16 Box
/** Box defines a concrete node interface that extends interface X3DGeometryNode. */

class AFX_EXT_CLASS CBox : public CX3DGeometryNode
{
    DECLARE_DYNAMIC(CBox);

public:
    CBox( );
    virtual ~CBox( );

//Implimentation
public:
    virtual void Draw( );
    virtual CString toXMLString( );
    virtual CString getPropertyString( );

    /** Return array of 3-tuple float results array [] from SFVec3f initializeOnly field named "size" */
    float* getSize ( );

    /** Assign 3-tuple float array [] to SFVec3f initializeOnly field named "size" */
    void setSize (float* value);

    /** Return boolean result from SFBool initializeOnly field named "solid" */
    bool getSolid ( );

    /** Assign boolean value to SFBool initializeOnly field named "solid" */
    void setSolid (bool value);

//Attributes
protected:
    SFVec3f     size;
    bool        solid;
};
```

X3D C++ Library







X3D C++ Library

- CImportedNodeException
- CIndexedFaceSet
- CIndexedLineSet
- CIndexedQuadSet
- CIndexedTriangleFanSet
- CIndexedTriangleSet
- CIndexedTriangleStripSet
- CInline
- CInsufficientCapabilitiesException
- CIntegerSequencer
- CIntegerTrigger
- CInvalidBrowserException
- CInvalidDocumentException
- CInvalidExecutionContextException
- CInvalidFieldException
- CInvalidFieldValueException
- CInvalidNodeException
- CInvalidOperationTimingException
- CInvalidProtoException
- CInvalidRouteException
- CInvalidURLException
- CInvalidX3DException
- CIS
- CIsoSurfaceVolumeData
- CKeySensor
- CLayer
- CLayerSet
- CLayout
- CLayoutGroup
- CLayoutLayer
- CLinePickSensor
- CLineProperties
- CLineSet
- CLoadSensor
- CLocalFog
- CLOD
- CMaterial
- CMatrix3VertexAttribute
- CMatrix4VertexAttribute
- Cmeta
- CMetadataBoolean
- CMetadataDouble
- CMetadataFloat
- CMetadataInteger
- CMetadataSet
- CMetadataString
- CMFBool
- CMFColor
- CMFColorRGBA
- CMFDouble
- CMFFloat
- CMField
- CMFImage
- CMFInt32
- CMFMatrix3d
- CMFMatrix3f
- CMFMatrix4d
- CMFMatrix4f
- CMFRotation
- CMFString
- CMFTime
- CMFVec2d
- CMFVec2f
- CMFVec3d
- CMFVec3f
- CMFVec4d
- CMFVec4f
- CMotorJoint
- CMovieTexture
- CMultiTexture
- CMultiTextureCoordinate
- CMultiTextureTransform
- CNavigationInfo
- CNodeInUseException
- CNodeUnavailableException
- CNormal
- CNormalInterpolator
- CNoSuchBrowserException
- CNurbsCurve
- CNurbsCurve2D
- CNurbsOrientationInterpolator
- CNurbsPatchSurface
- CNurbsPositionInterpolator
- CNurbsSet
- CNurbsSurfaceInterpolator
- CNurbsSweptSurface
- CNurbsSwungSurface
- CNurbsTextureCoordinate
- CNurbsTrimmedSurface
- COpacityMapVolumeStyle
- COrientationChaser
- COrientationDamper
- COrientationInterpolator
- COrthoViewpoint
- CPackagedShader
- CParticleSystem
- CPickableGroup
- CPixelTexture
- CPixelTexture3D
- CPlaneSensor
- CPointEmitter
- CPointLight
- CPointPickSensor
- CPointSet
- CPolyline2D
- CPolylineEmitter
- CPolypoint2D
- CPositionChaser

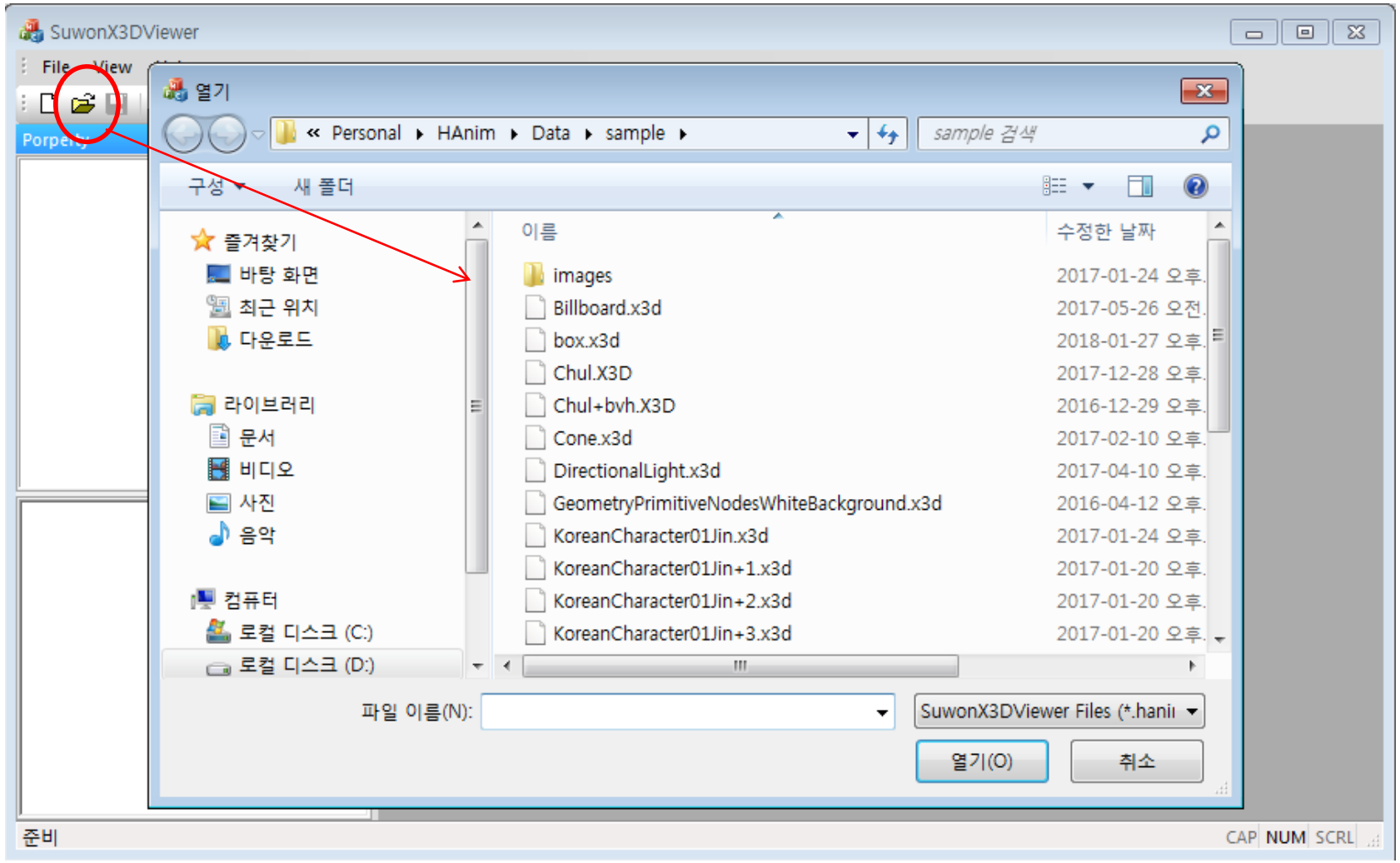
X3D C++ Library

- CPositionChaser2D
- CPositionDamper
- CPositionDamper2D
- CPositionInterpolator
- CPositionInterpolator2D
- CPrimitivePickSensor
- CProfileInfo
- CProgramShader
- CProjectionVolumeStyle
- CProtoBody
- CProtoDeclare
- CProtoInstance
- CProtoInterface
- CProximitySensor
- CQuadSet
- CReceiverPdu
- CRectangle2D
- CRigidBody
- CRigidBodyCollection
- CROUTE
- CScalarChaser
- CScalarDamper
- CScalarInterpolator
- CScene
- CSceneGraphStructureStatement
- CScreenFontStyle
- CScreenGroup
- CScript
- CSegmentedVolumeData
- CSFBool
- CSFColor
- CSFColorRGBA
- CSFDouble
- CSFFloat
- CSFImage
- CSFInt32
- CSFMatrix3d
- CSFMatrix3f
- CSFMatrix4d
- CSFMatrix4f
- CSFRotation
- CSFString
- CSFTime
- CSFVec2d
- CSFVec2f
- CSFVec3d
- CSFVec3f
- CSFVec4d
- CSFVec4f
- CShadedVolumeStyle
- CShaderPart
- CShaderProgram
- CShape
- CSignalPdu
- CSilhouetteEnhancementVolumeStyle
- CSingleAxisHingeJoint
- CSliderJoint
- CSound
- CSphere
- CSphereSensor
- CSplinePositionInterpolator
- CSplinePositionInterpolator2D
- CSplineScalarInterpolator
- CSpotLight
- CSquadOrientationInterpolator
- CStaticGroup
- CStdioFileEx
- CStringSensor
- CSurfaceEmitter
- CSwitch
- CTexCoordChaser2D
- CTexCoordDamper2D
- CText
- CTextureBackground
- CTextureCoordinate
- CTextureCoordinate3D
- CTextureCoordinate4D
- CTextureCoordinateGenerator
- CTextureProperties
- CTextureTransform
- CTextureTransform3D
- CTextureTransformMatrix3D
- CTimeSensor
- CTimeTrigger
- CToneMappedVolumeStyle
- CTouchSensor
- CTransform
- CTransformSensor
- CTransmitterPdu
- CTriangleFanSet
- CTriangleSet
- CTriangleSet2D
- CTriangleStripSet
- CTwoSidedMaterial
- Cunit
- CUniversalJoint
- CURLUnavailableException
- CViewpoint
- CViewpointGroup
- CViewport
- CVisibilitySensor
- CVolumeData
- CVolumeEmitter
- CVolumePickSensor
- CWindPhysicsModel
- CWorldInfo
- CX3D
- CX3DAppearanceChildNode

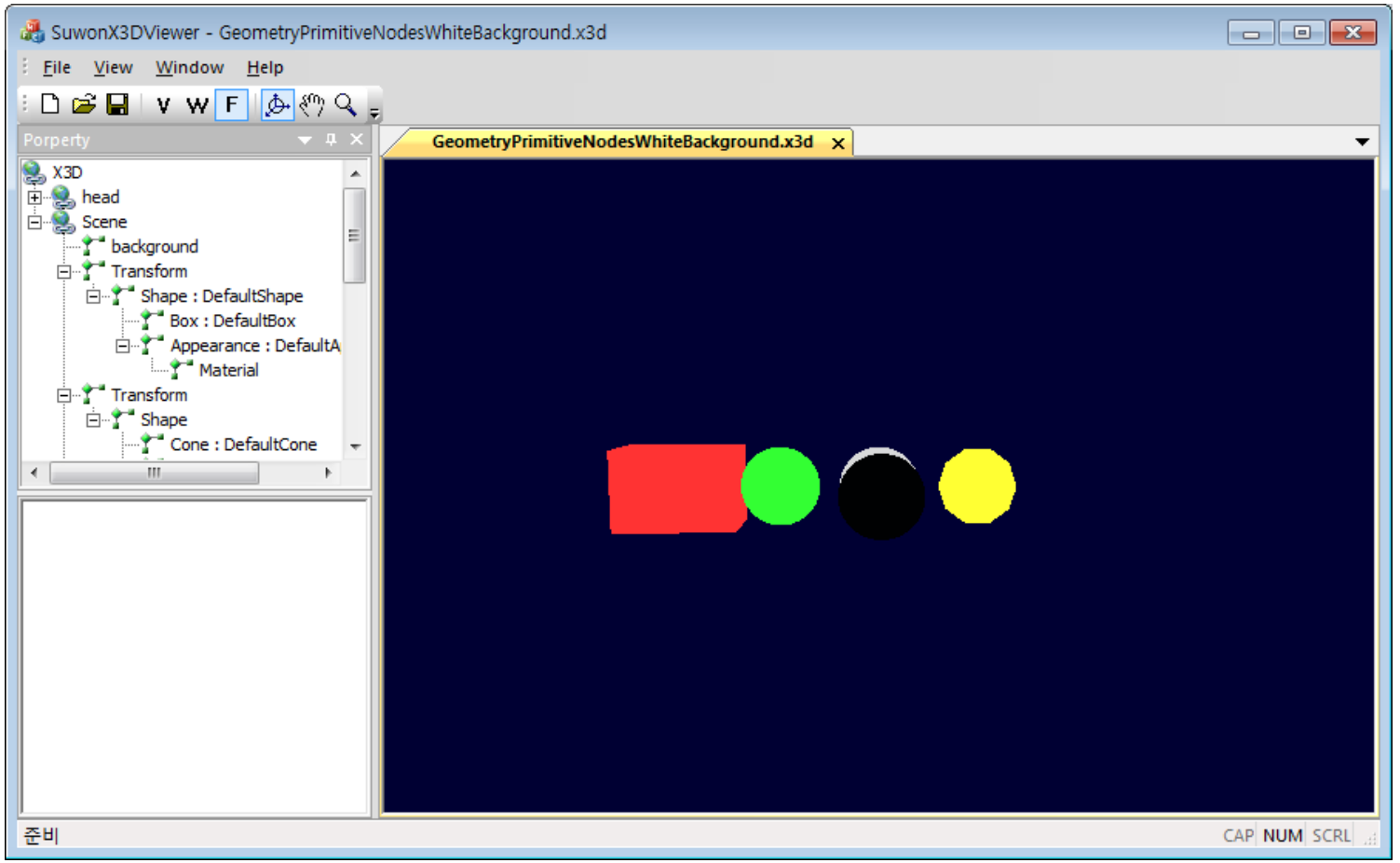
X3D C++ Library

- CX3DAppearanceNode
- CX3DArrayField
- CX3DBackgroundNode
- CX3DBindableNode
- CX3DBoundedObject
- CX3DChaserNode
- CX3DChildNode
- CX3DColorNode
- CX3DComposableVolumeRenderStyleNode
- CX3DComposedGeometryNode
- CX3DCoordinateNode
- CX3DDamperNode
- CX3DDragSensorNode
- CX3DEnvironmentalSensorNode
- CX3DEnvironmentTextureNode
- CX3DException
- CX3DField
- CX3DFieldDefinition
- CX3DFieldEvent
- CX3DFieldEventListener
- CX3DFogObject
- CX3DFollowerNode
- CX3DFontStyleNode
- CX3DGeometricPropertyNode
- CX3DGeometryNode
- CX3DGroupingNode
- CX3DInfoNode
- CX3DInterpolatorNode
- CX3DKeyDeviceSensorNode
- CX3DLayerNode
- CX3DLayoutNode
- CX3DLib
- CX3DLightNode
- CX3DMaterialNode
- CX3DMeta
- CX3DMetadataObject
- CX3DNBodyCollidableNode
- CX3DNBodyCollisionSpaceNode
- CX3DNetworkSensorNode
- CX3DNode
- CX3DNodeArray
- CX3DNodeMixedContent
- CX3DNormalNode
- CX3DNurbsControlCurveNode
- CX3DNurbsSurfaceGeometryNode
- CX3DParametricGeometryNode
- CX3DParticleEmitterNode
- CX3DParticlePhysicsModelNode
- CX3DPickableObject
- CX3DPickSensorNode
- CX3DPointingDeviceSensorNode
- CX3DProductStructureChildNode
- CX3DProgrammableShaderObject
- CX3DPrototypeInstance
- CX3DRigidJointNode
- CX3DScene
- CX3DScriptNode
- CX3DSensorNode
- CX3DSequencerNode
- CX3DShaderNode
- CX3DShapeNode
- CX3DSoundNode
- CX3DSoundSourceNode
- CX3DTexture2DNode
- CX3DTexture3DNode
- CX3DTextureCoordinateNode
- CX3DTextureNode
- CX3DTextureTransformNode
- CX3DTimeDependentNode
- CX3DTouchSensorNode
- CX3DTriggerNode
- CX3DUrlObject
- CX3DVertexAttributeNode
- CX3DViewpointNode
- CX3DViewportNode
- CX3DVolumeDataNode
- CX3DVolumeRenderStyleNode
- CX3Node
- Matrix3
- Matrix4

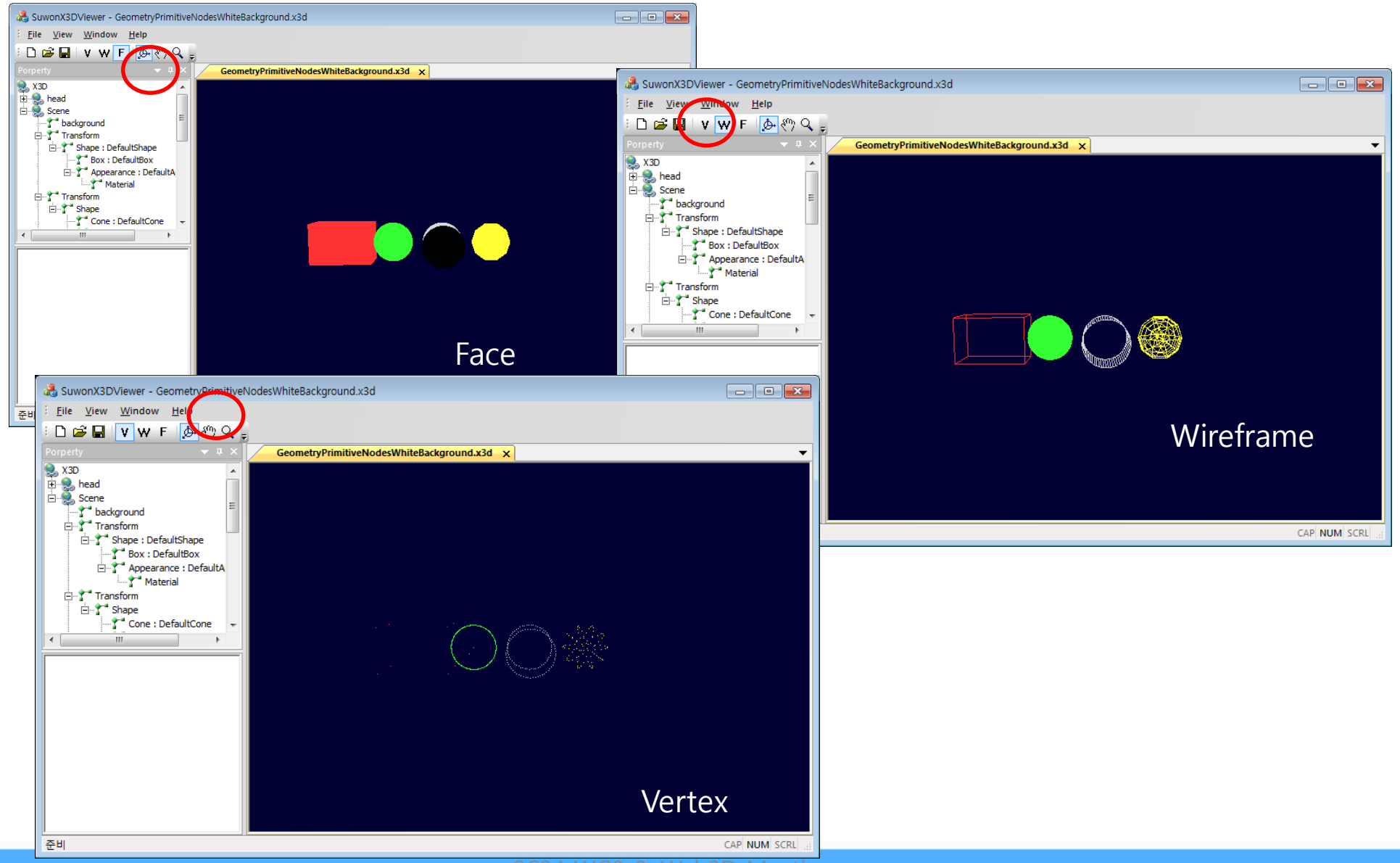
X3D C++ Binding Viewer (X3D File Open)



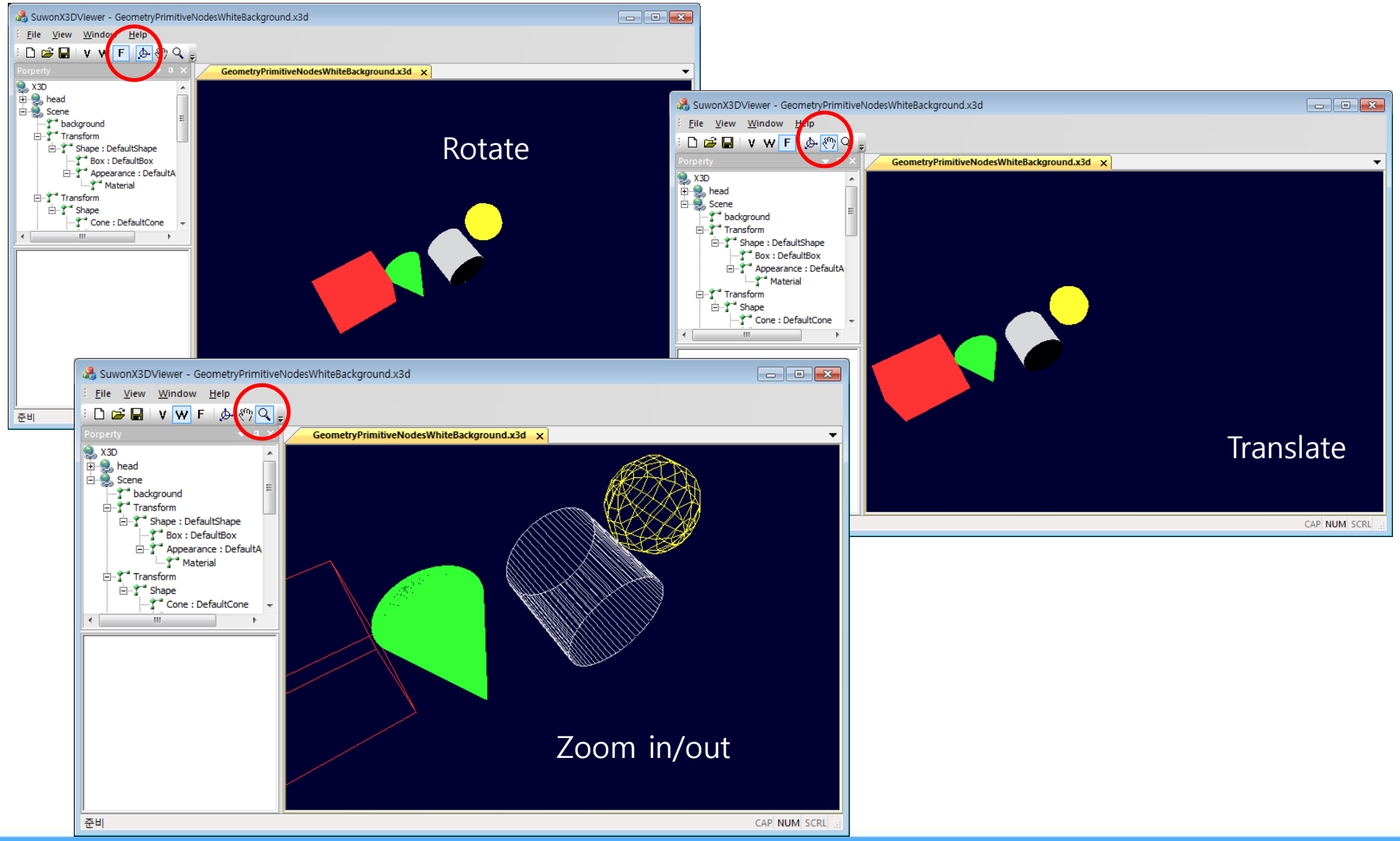
X3D C++ Binding Viewer (X3D Load)



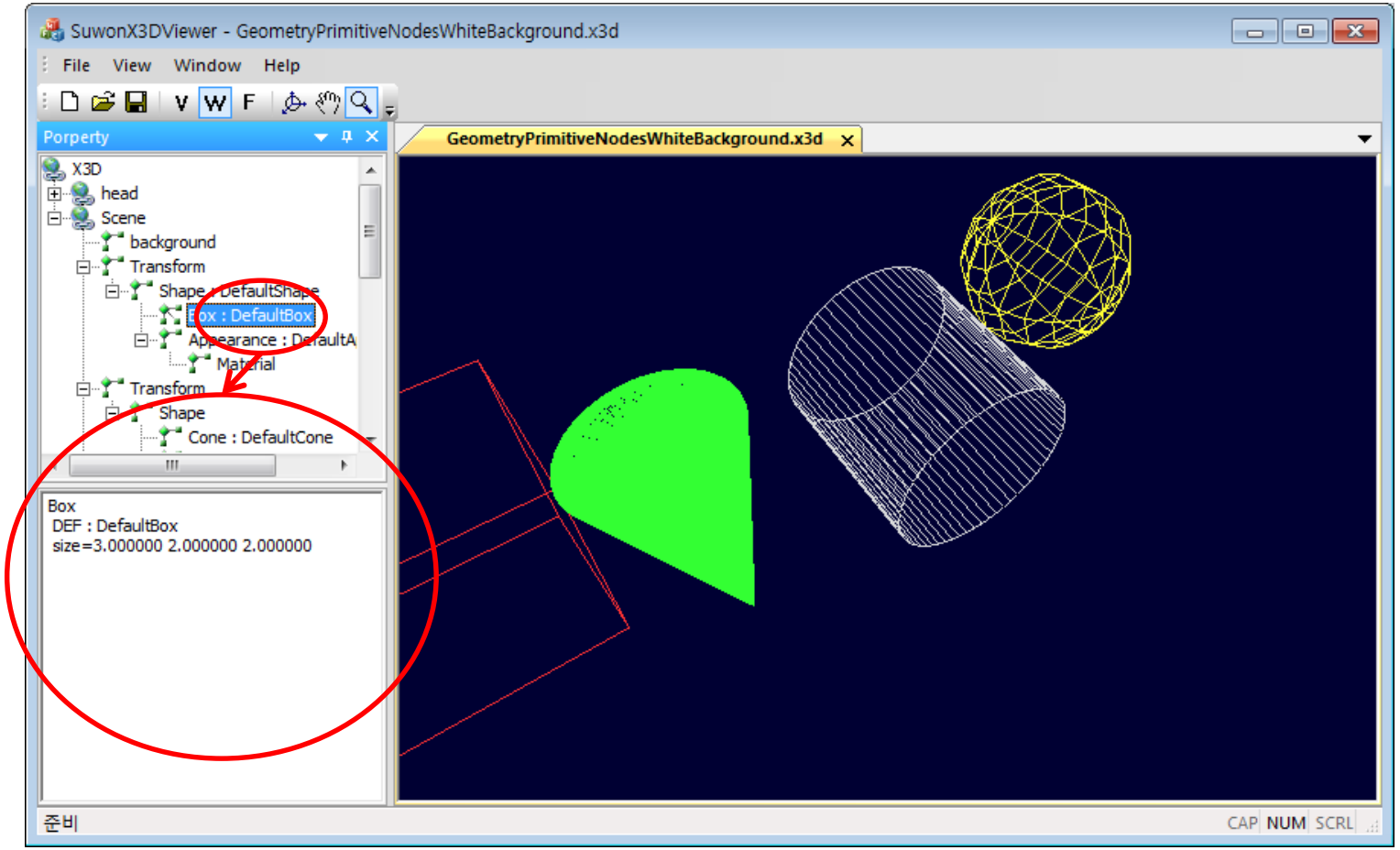
X3D C++ Binding Viewer (Viewer Mode)



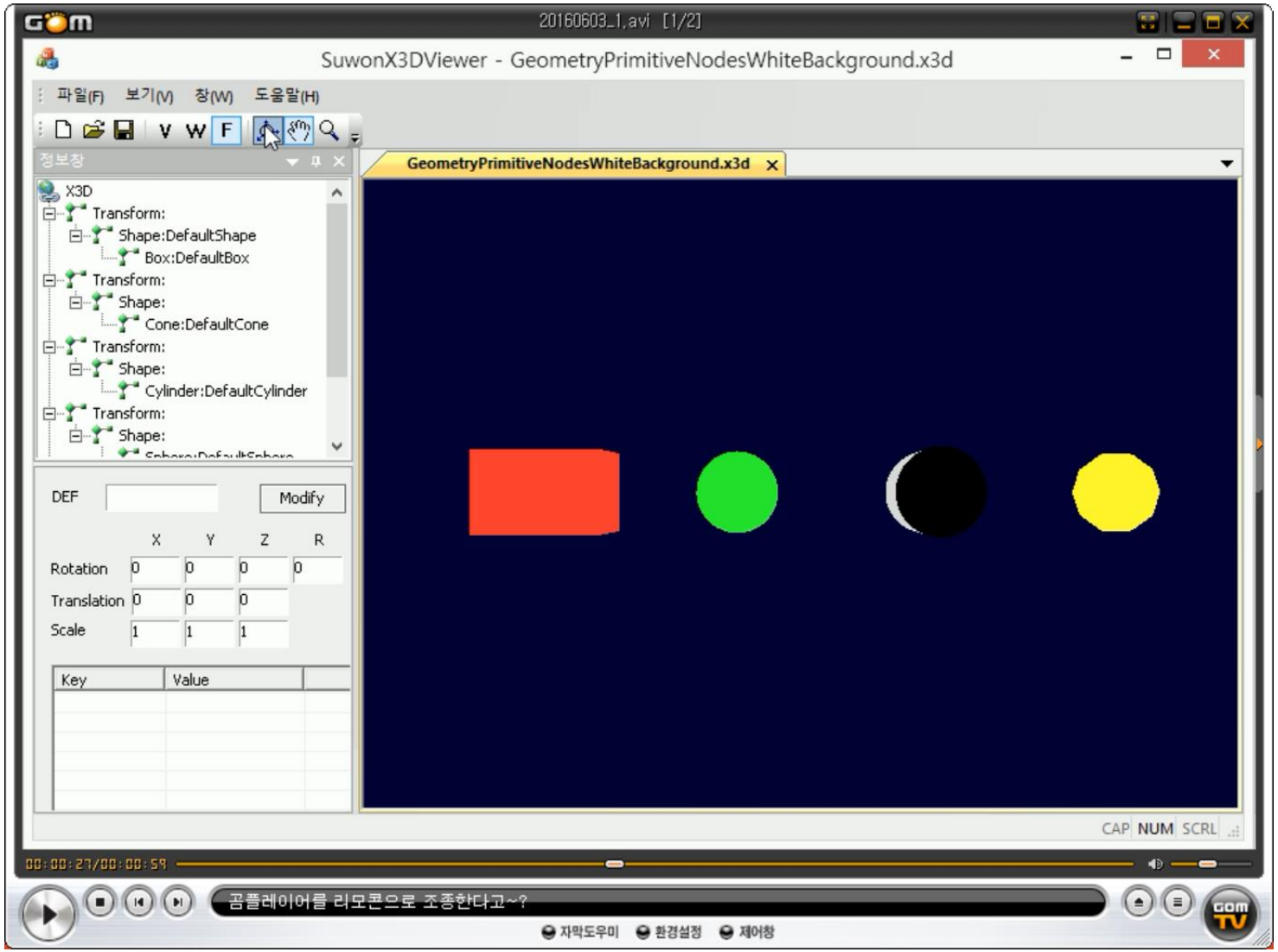
X3D C++ Binding Viewer (Viewer Mode)



X3D C++ Binding Viewer (Property Window)



X3D C++ Binding Viewer Demo



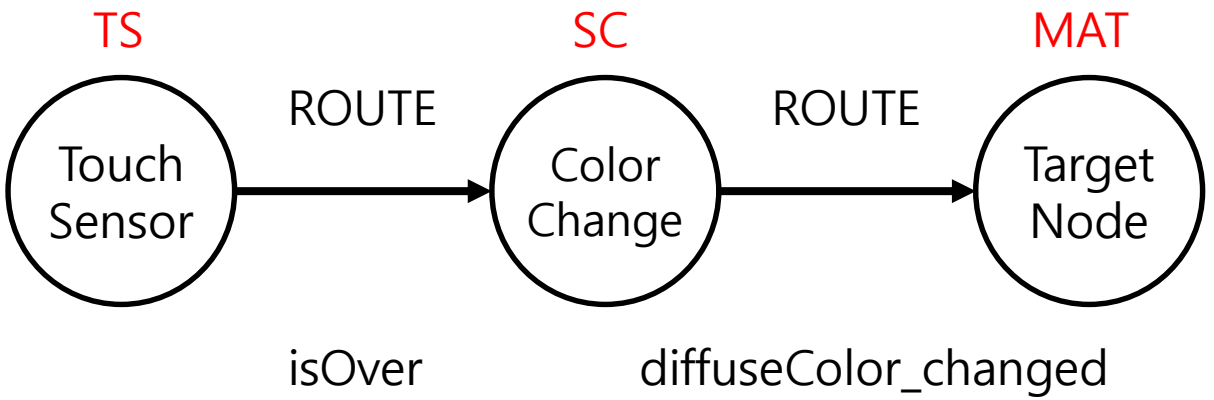
ISO/IEC NP 19777-4
X3D C++ Language Binding
Annex C Examples
(Implementation)

Example 1. TouchSensor isOver event.x3d

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE X3D PUBLIC "ISO//Web3D//DTD X3D 3.0//EN" "http://www.web3d.org/specifications/x3d-3.0.dtd">
<X3D profile="Immersive">
  <head>
    <meta content="TouchSensorIsOverEvent.x3d" name="filename"/>
    <meta content="Xeena VRML importer" name="translator"/>
    <meta content="23 February 2005" name="imported"/>
    <meta content="23 February 2005" name="revised"/>
    <meta
content="X3D-Edit, http://www.web3D.org/TaskGroups/x3d/translation/README.X3D-Edit.html" name="generator"/>
    <meta content="Vrml97ToX3dNist, http://ovrt.nist.gov/v2_x3d.html" name="generator"/>
  </head>
  <Scene>
    <Group>
      <Shape>
        <Appearance>
          <Material DEF="MAT" diffuseColor="0 0 1"/>
        </Appearance>
        <Box/>
      </Shape>
      <TouchSensor DEF="TS"/>
    </Group>
    <Script DEF="SC" url="&amp;quot;SAIExample1.class&amp;quot; ">
      <field accessType="inputOnly" name="isOver" type="SFBool"/>
      <field accessType="outputOnly" name="diffuseColor_changed" type="SFColor"/>
    </Script>
    <ROUTE fromField="isOver" fromNode="TS" toField="isOver" toNode="SC"/>
    <ROUTE fromField="diffuseColor_changed" fromNode="SC"
     toField="set_diffuseColor" toNode="MAT"/>
  </Scene>
</X3D>
```

Example 1. TouchSensor isOver event.x3d (Implementation Overview)

```
<Material DEF="MAT" diffuseColor="0 0 1"/>
<TouchSensor DEF="TS"/>
<Script DEF="SC" url="&quot;SAIExample1.class&quot; ">
    <field accessType="inputOnly" name="isOver" type="SFBool"/>
    <field accessType="outputOnly" name="diffuseColor_changed" type="SFColor"/>
</Script>
<ROUTE fromField="isOver" fromNode="TS"  toField="isOver" toNode="SC"/>
<ROUTE fromField="diffuseColor_changed" fromNode="SC" toField="set_diffuseColor" tooNode="MAT"/>
```



Example 1. TouchSensor isOver event.x3d (Implementation) (1)

```
#include "stdafx.h"
#include "X3D_Sample2.h"

#include <glut.h>
#include "..₩X3DLib₩X3DLib.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#endif

CWinApp theApp;
using namespace std;

CX3DScene m_pScene;
float ratio = 0;

/** The isOver field */
CSFBool* isOver=NULL;

/** The diffuseColor_changed field */
CSFColor* diffuseColor=NULL;

/** Color Constant, RED */
float RED[] = {1.0f, 0, 0};

/** Color Constant, BLUE */
float BLUE[] = {0, 0, 1.0f};

CX3DFieldEventListener lis;
```

Example 1. TouchSensor isOver event.x3d (Implementation) (2)

```
void initialize()
{
		/*  specify clear values for the color buffers  */
		glClearColor(0.0, 0.0, 0.0, 0.0);

		/*  enable various capabilities */
		glEnable(GL_DEPTH_TEST);

		/* select flat or smooth shading */
		glShadeModel(GL_FLAT);

		/* specify mapping of depth values from normalized device coordinates to window coordinates */
		glDepthRange(0.0, 1.0);

		/* x3d File loading */
		m_pScene.createX3DFromFile(_T("2.x3d"));

    /**
     * Notification that the script has completed the setup and should go
     * about its own internal initialization.
     */

		isOver = (CSFBool*)(m_pScene.m_fields.get("isOver"));
		diffuseColor = (CSFColor*)(m_pScene.m_fields.get("diffuseColor_changed"));

		isOver->addX3DEventListener(lis);
}
```

Example 1. TouchSensor isOver event.x3d (Implementation) (3)

```
void display(void)
{
		/* sets the bitplane area of the window to values previously selected by glClearColor */
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);

		/* X3d Library Draw function Call */
		m_pScene.Draw();

		glFlush();
}

/**
* Handle field changes.
*
* @param evt The field event
*/
void readableFieldChanged(CX3DFieldEvent* evt)
{
		if (evt->getSource() == isOver)
		{
			if (isOver->getValue() == true)
				diffuseColor->setValue(RED);
			else
				diffuseColor->setValue(BLUE);
		}
		else
		{
			printf("Unhandled event: %d", evt);
		}
}
```

Example 1. TouchSensor isOver event.x3d (Implementation) (4)

```
/* Mouse Hit Object Porcess */
void processHits(GLint hits, GLuint buffer[])
{
        unsigned int i, j;
        GLuint names, *ptr;

        ptr = (GLuint *) buffer;
        for (i = 0; i < hits; i++) {  /* for each hit  */
                names = *ptr;
                for (j = 0; j < names; j++) {  /* for each name */
                        diffuseColor->setValue(RED);
                }
        }
}
#define BUFSIZE 512
void pickRects(int button, int state, int x, int y)
{
        GLuint selectBuf[BUFSIZE];
        GLint hits;
        GLint viewport[4];

        if (button != GLUT_LEFT_BUTTON || state != GLUT_DOWN)
                return;

        /* return the value or values of a selected parameter */
        glGetIntegerv(GL_VIEWPORT, viewport);
```

Example 1. TouchSensor isOver event.x3d (Implementation) (5)

```
        /* establish a buffer for selection mode values */
        glSelectBuffer(BUFSIZE, selectBuf);

        /* sets the rasterization mode */
        glRenderMode(GL_SELECT);
        /* initializes the name stack */
        glInitNames();

        /* push and pop the name stack  */
        glPushName(0);

        glMatrixMode(GL_PROJECTION);
        glPushMatrix();
        glLoadIdentity();

        /* forces execution of OpenGL functions in finite time */
        glFlush();

        hits = glRenderMode(GL_RENDER);
        processHits(hits, selectBuf);
}
```

Example 1. TouchSensor isOver event.x3d (Implementation) (6)

```
void reshape(int w, int h)
{
		if(h == 0)
			h = 1;
		ratio = 1.0* w / h;

		/* set the viewport */
		glViewport(0, 0, w, h);

		/* defines a picking region */
		gluPickMatrix((GLdouble) x, (GLdouble) (viewport[3] - y),
			5.0, 5.0, viewport);


		/* set up a perspective projection matrix */
		gluPerspective(45,ratio,1,1000);

		m_pScene.Draw(GL_SELECT);

		glPopMatrix();
```

Example 1. TouchSensor isOver event.x3d (Implementation) (7)

```
                /* specify which matrix is the current matrix */
                glMatrixMode(GL_PROJECTION);

                /* replace the current matrix with the identity matrix */
                glLoadIdentity();

                /* set up a perspective projection matrix */
                gluPerspective(45,ratio,1,1000);

                glMatrixMode(GL_MODELVIEW);

                glLoadIdentity();

                /* defines a viewing transformation */
                gluLookAt(5.0,5.0,5.0,
        0.0,0.0,-1.0,
        0.0f,1.0f,0.0f);

}

int _tmain(int argc, TCHAR* argv[], TCHAR* envp[])
{
                int nRetCode = 0;

                /* Retrieves a module handle for the specified module. The module must have been loaded by the calling
process */
                HMODULE hModule = ::GetModuleHandle(NULL);
```

Example 1. TouchSensor isOver event.x3d (Implementation) (8)

```
if (hModule != NULL)
{
        // to initialize MFC
        if (!AfxWinInit(hModule, NULL, ::GetCommandLine(), 0))
        {
                _tprintf(_T("Error: Initialze failed.₩n"));
                nRetCode = 1;
        }
        else
        {
          /* Initializes the COM library on the current thread and identifies the concurrency model
as single-thread apartment  */
          HRESULT hr = CoInitialize(0);

          /* initialize the GLUT library */
          glutInit(&argc, argv);

          /* sets the  initial display mode */
          glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB | GLUT_DEPTH);

          /* set the  initial window position and  size respectively */
          glutInitWindowSize (200, 200);
          glutInitWindowPosition (100, 100);

          /* creates a top-level window. */
          glutCreateWindow(argv[0]);

          initialize();
```

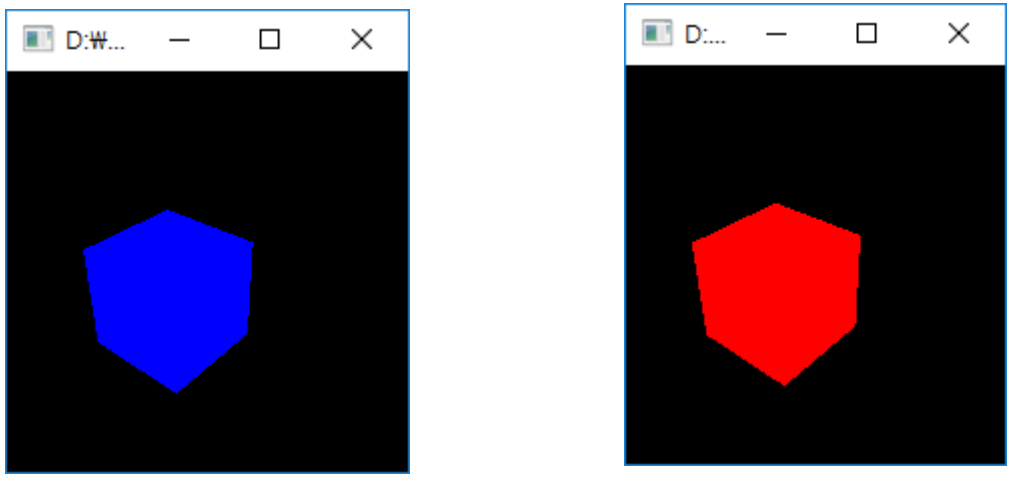
Example 1. TouchSensor isOver event.x3d (Implementation) (9)

```
                    /* sets the mouse callback for the  current window. */
                    glutMouseFunc(pickRects);

                    /* sets the reshape callback for the  current window. */
                    glutReshapeFunc(reshape);

                    /* sets the display callback for the  current window. */
                    glutDisplayFunc(display);

                    /* enters the GLUT event processing loop. */
                    glutMainLoop();
                }
        }
        else
        {
                _tprintf(_T("Error: GetModuleHandle failed₩n"));
                nRetCode = 1;
        }

        return nRetCode;
}
```

Example 1. TouchSensor isOver event.x3d (Implementation) (10)

On click, an event is generated

- On click, the color of the box is changed from blue to red.
- On click in the outside of the box, the color of the box is changed to blue.

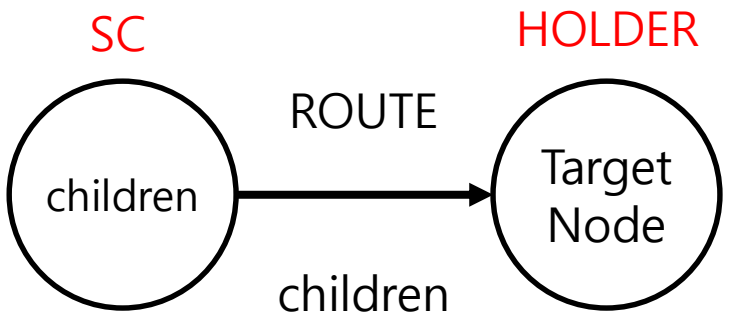


Example 2. Create nodes.x3d

```
<?xml version="1.0" encoding="UTF-8"?>
<X3D profile="Immersive">
  <head>
    <meta content="CreateNodes.x3d" name="filename"/>
    <meta content="Xeena VRML importer" name="translator"/>
    <meta content="*enter date of initial version here*" name="created"/>
    <meta content="23 February 2005" name="imported"/>
    <meta content="23 February 2005" name="revised"/>
    <meta content="X3D-Edit, http://www.web3D.org/TaskGroups/x3d/translation/README.X3D-
     Edit.html" name="generator"/>
    <meta content="Vrml97ToX3dNist, http://ovrt.nist.gov/v2_x3d.html" name="generator"/>
  </head>
  <Scene>
    <Transform DEF="HOLDER" translation="-2.0 0.0 0.0"/>
    <Script DEF="SC" url="SAIExample.class">
      <field accessType="outputOnly" name="children" type="MFNode"/>
    </Script>
    <ROUTE fromField="children" fromNode="SC" toField="children" toNode="HOLDER"/>
  </Scene>
</X3D>
```

Example 2. Create nodes.x3d (Implementation Overview)

```
<Transform DEF="HOLDER" translation="-2.0 0.0 0.0"/>
    <Script DEF="SC" url="SAIExample.class">
      <field accessType="outputOnly" name="children" type="MFNode"/>
    </Script>
    <ROUTE fromField="children" fromNode="SC"  toField="children"
    toNode="HOLDER"/>
```



Example 2. Create nodes.x3d (Implementation) (1)

```
// X3D_Sample3.cpp : 콘솔 응용 프로그램에 대한 진입점을 정의합니다.
//

#include "stdafx.h"
#include "X3D_Sample3.h"

/* include glut library */
#include <glut.h>

/* include X3D library */
#include "..₩X3DLib₩X3DLib.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#endif

/* Global varialble */
CWinApp theApp;
using namespace std;

/* X3D Library pointer */
CX3DScene m_pScene;
float ratio = 0;

/**
* Notification that the script has completed the setup and should go
* about its own internal initialization.
*/
```

Example 2. Create nodes.x3d (Implementation) (2)

```
void initialize()
{
        /*  specify clear values for the color buffers  */
        glClearColor(0.0, 0.0, 0.0, 0.0);

        /*  enable various capabilities */
        glEnable(GL_DEPTH_TEST);

        /* select flat or smooth shading */
        glShadeModel(GL_FLAT);

        /* specify mapping of depth values from normalized device coordinates to window coordinates */
        glDepthRange(0.0, 1.0);  /* The default z mapping */

        /* x3d File loading */
        m_pScene.createX3DFromFile(_T("3.x3d"));

    /**
     * Notification that the script has completed the setup and should go
     * about its own internal initialization.
     */

        CShape* shape = (CShape*) m_pScene.createNode("Shape");
        CBox* box = (CBox*) m_pScene.createNode("Box");
        shape->setGeometry(box);
        m_pScene.addRootNode(shape);
}
```

Example 2. Create nodes.x3d (Implementation) (3)

```
void display(void)
{
                /* sets the bitplane area of the window to values previously selected by glClearColor */
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);

                /* X3d Library Draw function Call */
                m_pScene.Draw();

                /* forces execution of OpenGL functions in finite time */
                glFlush();
}

void reshape(int w, int h)
{
                if(h == 0)
                                h = 1;

                ratio = 1.0* w / h;

                /* set the viewport */
                glViewport(0, 0, w, h);

                /* GL_PROJECTION specify which matrix is the current matrix */
                glMatrixMode(GL_PROJECTION);

                /* replace the current matrix with the identity matrix */
                glLoadIdentity();
```

Example 2. Create nodes.x3d (Implementation) (4)

```
                /* set up a perspective projection matrix */
                gluPerspective(45,ratio,1,1000);

                /* GL_MODELVIEW specify which matrix is the current matrix */
                glMatrixMode(GL_MODELVIEW);

                /* replace the current matrix with the identity matrix */
                glLoadIdentity();

                /* defines a viewing transformation */
                gluLookAt(5.0,5.0,5.0,
            0.0,0.0,-1.0,
            0.0f,1.0f,0.0f);

}

int _tmain(int argc, TCHAR* argv[], TCHAR* envp[])
{
                int nRetCode = 0;

                /* Retrieves a module handle for the specified module. The module must have been loaded by the calling
process */
                HMODULE hModule = ::GetModuleHandle(NULL);

                if (hModule != NULL)
                {
```

Example 2. Create nodes.x3d (Implementation) (5)

```
// to initialize MFC
if (!AfxWinInit(hModule, NULL, ::GetCommandLine(), 0))
{
        _tprintf(_T("Error: Initialze failed.₩n"));
        nRetCode = 1;
}
else
{
   /* Initializes the COM library on the current thread and identifies the concurrency
model as single-thread apartment  */
   HRESULT hr = CoInitialize(0);

   /* initialize the GLUT library */
   glutInit(&argc, argv);

   /* sets the  initial display mode */
   glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB | GLUT_DEPTH);

   /* set the  initial window position and  size respectively */
   glutInitWindowSize (200, 200);

   glutInitWindowPosition (100, 100);

   /* creates a top-level window. */
   glutCreateWindow(argv[0]);
```

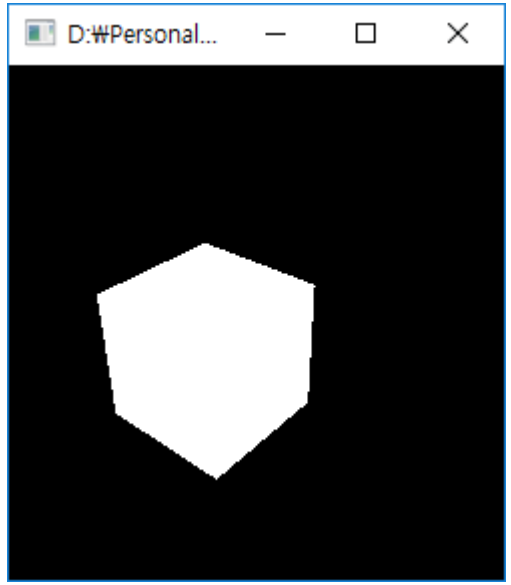
Example 2. Create nodes.x3d (Implementation) (6)

```
                    /* Data Initialize */
                    initialize();

                    /* sets the reshape callback for the  current window. */
                    glutReshapeFunc(reshape);

                    /* sets the display callback for the  current window. */
                    glutDisplayFunc(display);

                    /* enters the GLUT event processing loop. */
                    glutMainLoop();
                }
        }
        else
        {
                _tprintf(_T("Error: GetModuleHandle failed₩n"));
                nRetCode = 1;
        }

        return nRetCode;

}
```

Example 2. Create nodes.x3d (Implementation Results)

In the initialization stage,

- A Box under Shape is created in the Scene

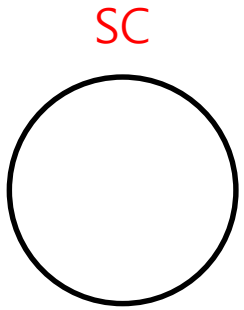


Example 3. Per frame notification.x3d

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE X3D PUBLIC "http://www.web3d.org/specifications/x3d-3.0.dtd"
                     "file:///www.web3d.org/TaskGroups/x3d/translation/x3d-3.0.dtd">
<X3D profile="Immersive">
  <head>
    <meta content="PerFrameNotification.x3d" name="filename"/>
    <meta content="Xeena VRML importer" name="translator"/>
    <meta content="23 February 2005" name="imported"/>
    <meta content="23 February 2005" name="revised"/>
    <meta
content="X3D-Edit, http://www.web3D.org/TaskGroups/x3d/translation/README.X3D-Edit.html"
     name="generator"/>
    <meta content="Vrml97ToX3dNist, http://ovrt.nist.gov/v2_x3d.html" name="generator"/>
  </head>
  <Scene>
    <Script DEF="SC" url="&quot;SAIExample3.class&quot; "/>
  </Scene>
</X3D>
```

Example 3. Per frame notification.x3d (Implementation Overview)

<Script DEF="SC" url=""SAIExample3.class" "/>



Initialize(): Initialize Time value

prepareEvents(): Calculate and output FPS

Example 3. Per frame notification.x3d (Implementation) (1)

```
#include "stdafx.h"
#include "X3D_Sample4.h"

/* include glut library */
#include <glut.h>

/* include X3D library */
#include "..₩X3DLib₩X3DLib.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#endif

/* Global varialble */
CWinApp theApp;
using namespace std;

/* X3D Library pointer */
CX3DScene m_pScene;
float wratio = 0;

int frame,currentTime,lastStartTime=0;
char s[30];
```

Example 3. Per frame notification.x3d (Implementation) (2)

```
/**
* Notification that the script has completed the setup and should go
* about its own internal initialization.
*/

void initialize()
{
		/*  specify clear values for the color buffers  */
		glClearColor(0.0, 0.0, 0.0, 0.0);

		/*  enable various capabilities */
		glEnable(GL_DEPTH_TEST);

		/* select flat or smooth shading */
		glShadeModel(GL_FLAT);

		/* specify mapping of depth values from normalized device coordinates to window coordinates */
		glDepthRange(0.0, 1.0);  /* The default z mapping */

		/* x3d File loading */
		m_pScene.createX3DFromFile(_T("4.x3d"));

		// current time initialize
		currentTime=glutGet(GLUT_ELAPSED_TIME);
}
```

Example 3. Per frame notification.x3d (Implementation) (3)

```
void display(void)
{
		/* sets the bitplane area of the window to values previously selected by glClearColor */
		glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);

		/* X3d Library Draw function Call */
		m_pScene.Draw();

		/* forces execution of OpenGL functions in finite time */
		glFlush();
}

/**
* Start of frame notification.
*/
void prepareEvents()
{
		/* sets the bitplane area of the window to values previously selected by glClearColor */
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);

		/* FPS (frames per second) calculate */
    frame++;

		/*  retrieves simple GLUT state represented by integers */
    currentTime=glutGet(GLUT_ELAPSED_TIME);
```

Example 3. Per frame notification.x3d (Implementation) (4)

```
if (currentTime - lastStartTime > 1000) {
        sprintf_s(s,"FPS:%4.2f", (frame * 1000.0 / (currentTime-lastStartTime)));
        lastStartTime = currentTime;
        frame = 0;
    }

    /* RGB color Sets the current color */
    glColor3f(1.0f,1.0f,1.0f);

    glPushMatrix();

                    char *c;
                    /* specify the raster position for pixel operations */
                    glRasterPos2f(-2, -2);

                    for (c=s; *c != '₩0'; c++) {
                                /* Draw a bitmapped character.  */
                                glutBitmapCharacter(GLUT_BITMAP_8_BY_13, *c);
                    }
    glPopMatrix();

                    /* forces execution of OpenGL functions in finite time */
                    glFlush();
}
```

Example 3. Per frame notification.x3d (Implementation) (5)

```
void reshape(int w, int h)
{
		if(h == 0)
			h = 1;

		wratio = 1.0* w / h;

		/* set the viewport */
		glViewport(0, 0, w, h);

		/* GL_PROJECTION specify which matrix is the current matrix */
		glMatrixMode(GL_PROJECTION);

		/* replace the current matrix with the identity matrix */
		glLoadIdentity();

		/* set up a perspective projection matrix */
		gluPerspective(45,wratio,1,1000);

		/* GL_MODELVIEW specify which matrix is the current matrix */
		glMatrixMode(GL_MODELVIEW);

		/* replace the current matrix with the identity matrix */
		glLoadIdentity();
```

Example 3. Per frame notification.x3d (Implementation) (6)

```
		/* defines a viewing transformation */
		gluLookAt(5.0,5.0,5.0,
	0.0,0.0,-1.0,
	0.0f,1.0f,0.0f);

}

int _tmain(int argc, TCHAR* argv[], TCHAR* envp[])
{
		int nRetCode = 0;

		/* Retrieves a module handle for the specified module. The module must have been loaded by the calling
process */

		HMODULE hModule = ::GetModuleHandle(NULL);

		if (hModule != NULL)
		{
			// to initialize MFC
			if (!AfxWinInit(hModule, NULL, ::GetCommandLine(), 0))
			{
				_tprintf(_T("Error: Initialze failed.₩n"));
				nRetCode = 1;
			}
			else
			{
```

Example 3. Per frame notification.x3d (Implementation) (7)

```
/* Initializes the COM library on the current thread and identifies the concurrency
model as single-thread apartment  */
HRESULT hr = CoInitialize(0);   // 컴 객체 초기화..

/* initialize the GLUT library */
glutInit(&argc, argv);

/* sets the  initial display mode */
glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB | GLUT_DEPTH);

/* set the  initial window position and  size respectively */
glutInitWindowSize (200, 200);
glutInitWindowPosition (100, 100);

/* creates a top-level window. */
glutCreateWindow(argv[0]);

/* Data Initialize */
initialize();

/* sets the reshape callback for the  current window. */
glutReshapeFunc(reshape);

/* sets the display callback for the  current window. */
glutDisplayFunc(display);

/* sets the global idle callback. */
glutIdleFunc(prepareEvents);
```

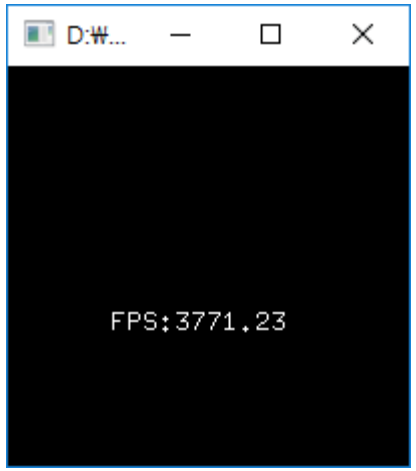
Example 3. Per frame notification.x3d (Implementation) (8)

```
                    /* enters the GLUT event processing loop. */
                    glutMainLoop();
                }
            }
            else
            {
                _tprintf(_T("Error: GetModuleHandle failed₩n"));
                nRetCode = 1;
            }

            return nRetCode;
}
```

Example 3. Per frame notification.x3d (Implementation Results)

In the prepareEvents()

- FPS is calculated and displayed

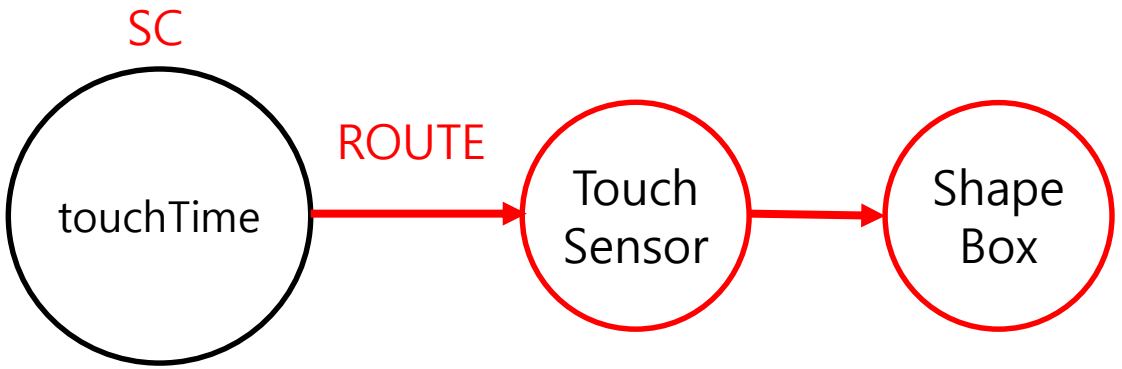


Example 4. Add dynamic routes.x3d

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE X3D PUBLIC "ISO//Web3D//DTD X3D 3.0//EN"
    "http://www.web3d.org/specifications/x3d-3.0.dtd">
<X3D profile="Immersive">
  <Scene>
    <Script DEF="SC" url="SAIExample4.class">
      <field accessType="inputOnly" name="touchTime" type="SFTime"/>
    </Script>
  </Scene>
</X3D>
```

Example 4. Add dynamic routes.x3d (Implementation Overview)

```
<Script DEF="SC" url="SAIExample4.class">
      <field accessType="inputOnly" name="touchTime" type="SFTime"/>
</Script>
```



Initialize() :

- Read "touchTime" field value
- Create Shape(Box) and TouchSesor under Group in the Scene
- Create "touchTime" ROUTE in the scene, connect the TouchSensor to the scene

readableFieldChanged()

- On the event of TouchSensor, "Poke!" is displayed

Example 4. Add dynamic routes.x3d (Implementation) (1)

```
#include "stdafx.h"
#include "X3D_Sample5.h"

/* include glut library */
#include <glut.h>

/* include X3D library */
#include "..₩X3DLib₩X3DLib.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#endif

/* Global varialble */
CWinApp theApp;
using namespace std;

/* X3D Library pointer */
CX3DScene m_pScene;
float ratio = 0;

CSFTime* touchTime = NULL;
```

Example 4. Add dynamic routes.x3d (Implementation) (2)

```
/**
* Notification that the script has completed the setup and should go
* about its own internal initialization.
*/
void initialize()
{
        /*  specify clear values for the color buffers  */
        glClearColor(0.0, 0.0, 0.0, 0.0);

        /*  enable various capabilities */
        glEnable(GL_DEPTH_TEST);

        /* select flat or smooth shading */
        glShadeModel(GL_FLAT);

        /* specify mapping of depth values from normalized device coordinates to window coordinates */
        glDepthRange(0.0, 1.0);  /* The default z mapping */

        /* x3d File loading */
        m_pScene.createX3DFromFile(_T("5.x3d"));
```

Example 4. Add dynamic routes.x3d (Implementation) (3)

```
    /**
     * Notification that the script has completed the setup and should go
     * about its own internal initialization.
     */

                    touchTime = (CSFTime*)(m_pScene.m_fields.get("touchTime"));

                    CShape* shape = (CShape*) m_pScene.createNode("Shape");
                    CBox* box = (CBox*) m_pScene.createNode("Box");
    CX3DNode* touchSensor = m_pScene.createNode("TouchSensor");

                    shape->setGeometry(box);

                    CGroup* group = (CGroup*)m_pScene.createNode("Group");

                    // Add the shape and sensor to the group
                    group->addChildren(shape);
                    group->addChildren(touchSensor);

                    m_pScene.addRootNode(group);

                    CX3DScriptNode* selfRef =(CX3DScriptNode*)m_pScene.getNode("SC");
    // Get a handle to the toplevel execution context
    m_pScene.addRoute(touchSensor, "touchTime", selfRef, "touchTime");

}
```

Example 4. Add dynamic routes.x3d (Implementation) (4)

```
void display(void)
{
		/* sets the bitplane area of the window to values previously selected by glClearColor */
   glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);

		/* X3d Library Draw function Call */
		m_pScene.Draw();

		/* forces execution of OpenGL functions in finite time */
		glFlush();
}

/* Mouse Hit Object Porcess */
void processHits(GLint hits, GLuint buffer[])
{
		unsigned int i, j;
		GLuint names, *ptr;

		//printf("hits = %d₩n", hits);
		ptr = (GLuint *) buffer;
		for (i = 0; i < hits; i++) {  /* for each hit  */
				names = *ptr;
		//		printf(" number of names for hit = %d₩n", names); ptr++;
		//		printf("  z1 is %g;", (float) *ptr/0x7fffffff); ptr++;
		//		printf(" z2 is %g₩n", (float) *ptr/0x7fffffff); ptr++;
		//		printf("    the name is ");
```

Example 4. Add dynamic routes.x3d (Implementation) (5)

```
                                        for (j = 0; j < names; j++) {  /* for each name */
                                                printf("Poke!"); ptr++;
                                        }
                                        printf("₩n");
                        }
}

#define BUFSIZE 512

void pickRects(int button, int state, int x, int y)
{
                        GLuint selectBuf[BUFSIZE];
                        GLint hits;
                        GLint viewport[4];

                        if (button != GLUT_LEFT_BUTTON || state != GLUT_DOWN)
                                        return;

                        /* return the value or values of a selected parameter */
                        glGetIntegerv(GL_VIEWPORT, viewport);

                        /* establish a buffer for selection mode values */
                        glSelectBuffer(BUFSIZE, selectBuf);

                        /* sets the rasterization mode */
                        glRenderMode(GL_SELECT);
```

Example 4. Add dynamic routes.x3d (Implementation) (6)

```
                /* defines a picking region */
                gluPickMatrix((GLdouble) x, (GLdouble) (viewport[3] - y),
                                                                5.0, 5.0, viewport);

                /* set up a perspective projection matrix */
                gluPerspective(45,ratio,1,1000);

                /* X3d Library Draw function Call */
                m_pScene.Draw(GL_SELECT);

                glPopMatrix();

                /* forces execution of OpenGL functions in finite time */
                glFlush();

                hits = glRenderMode(GL_RENDER);
                processHits(hits, selectBuf);
}

void reshape(int w, int h)
{
                if(h == 0)
                                h = 1;

                ratio = 1.0* w / h;

                /* set the viewport */
                glViewport(0, 0, w, h);
```

Example 4. Add dynamic routes.x3d (Implementation) (7)

```
		/* specify which matrix is the current matrix */
		glMatrixMode(GL_PROJECTION);

		/* replace the current matrix with the identity matrix */
		glLoadIdentity();

		/* set up a perspective projection matrix */
		gluPerspective(45,ratio,1,1000);

		glMatrixMode(GL_MODELVIEW);

		glLoadIdentity();

		/* defines a viewing transformation */
		gluLookAt(5.0,5.0,5.0,
	0.0,0.0,-1.0,
	0.0f,1.0f,0.0f);

}

int _tmain(int argc, TCHAR* argv[], TCHAR* envp[])
{
		int nRetCode = 0;

		/* Retrieves a module handle for the specified module. The module must have been loaded by the calling
process */
		HMODULE hModule = ::GetModuleHandle(NULL);
```

Example 4. Add dynamic routes.x3d (Implementation) (8)

```
if (hModule != NULL)
{
        // to initialize MFC
        if (!AfxWinInit(hModule, NULL, ::GetCommandLine(), 0))
        {
                _tprintf(_T("Error: Initialze failed.₩n"));
                nRetCode = 1;
        }
        else
        {
          /* Initializes the COM library on the current thread and identifies the concurrency
model as single-thread apartment  */
          HRESULT hr = CoInitialize(0);   // 컴 객체 초기화..

          /* initialize the GLUT library */
          glutInit(&argc, argv);

          /* sets the  initial display mode */
          glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB | GLUT_DEPTH);

          /* set the  initial window position and  size respectively */
          glutInitWindowSize (200, 200);
          glutInitWindowPosition (100, 100);

          /* creates a top-level window. */
          glutCreateWindow(argv[0]);
```

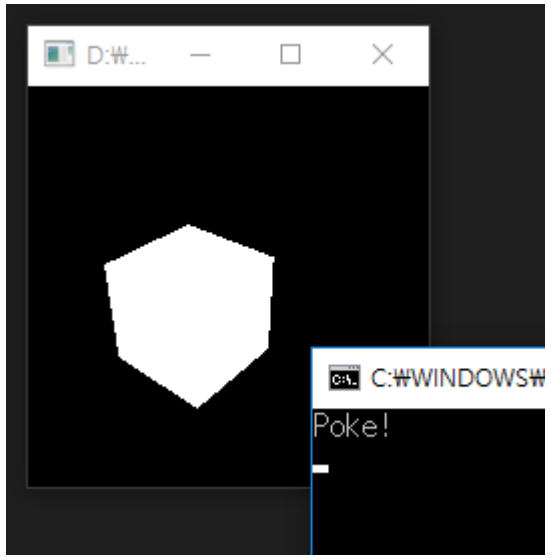
Example 4. Add dynamic routes.x3d (Implementation) (9)

```
                    /* Data Initialize */
                    initialize();

                    /* sets the mouse callback for the  current window. */
                    glutMouseFunc(pickRects);

                    /* sets the reshape callback for the  current window. */
                    glutReshapeFunc(reshape);

                    /* sets the display callback for the  current window. */
                    glutDisplayFunc(display);

                    /* enters the GLUT event processing loop. */
                    glutMainLoop();
                }
        }
        else
        {
                _tprintf(_T("Error: GetModuleHandle failed₩n"));
                nRetCode = 1;
        }

        return nRetCode;
}
```

Example 4. Add dynamic routes.x3d (Implementation Results)

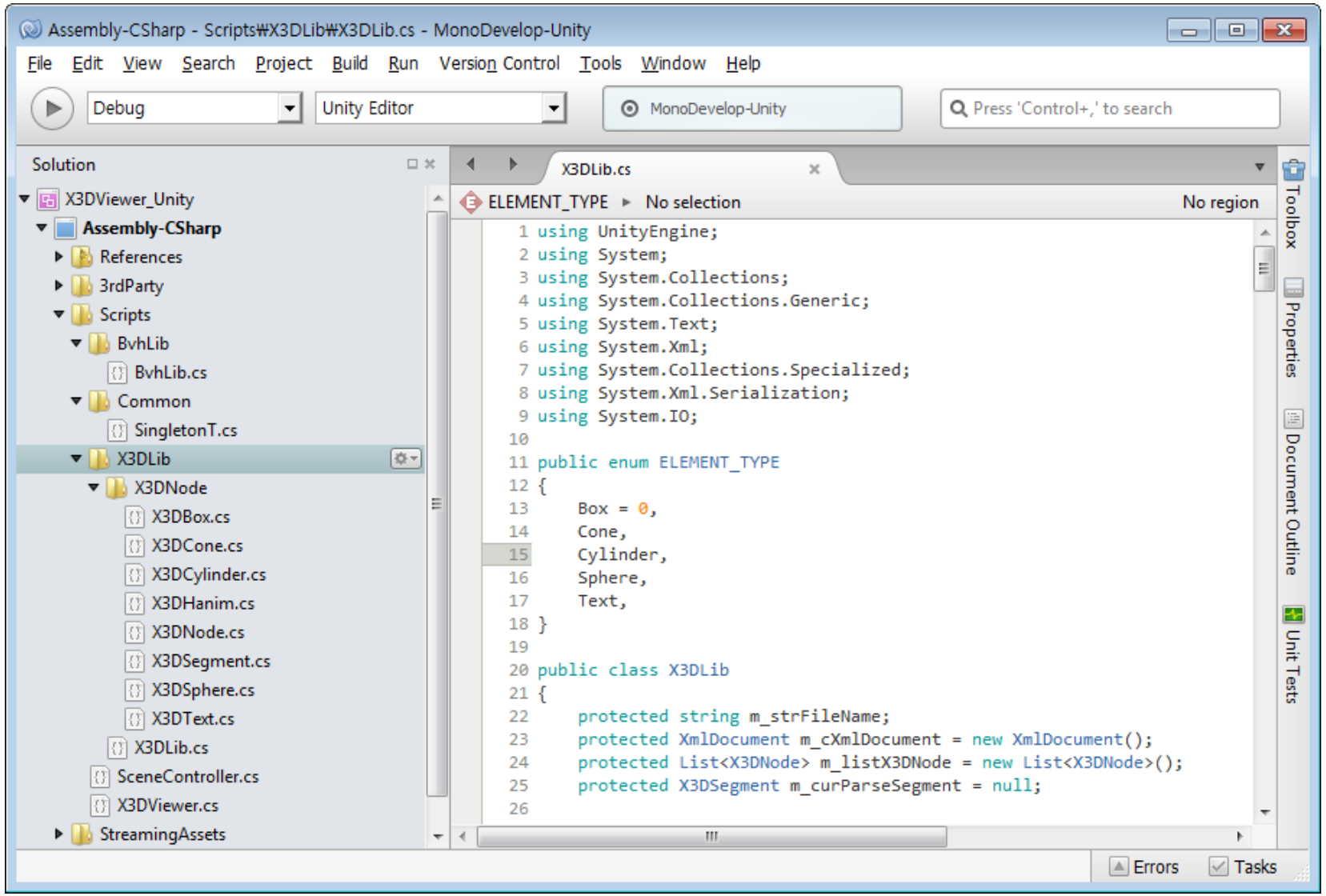
- In the initialization

  Create Scene > Group > Shape > Box > TouchSensor
- On clicking Box, “Poke!” is displayed

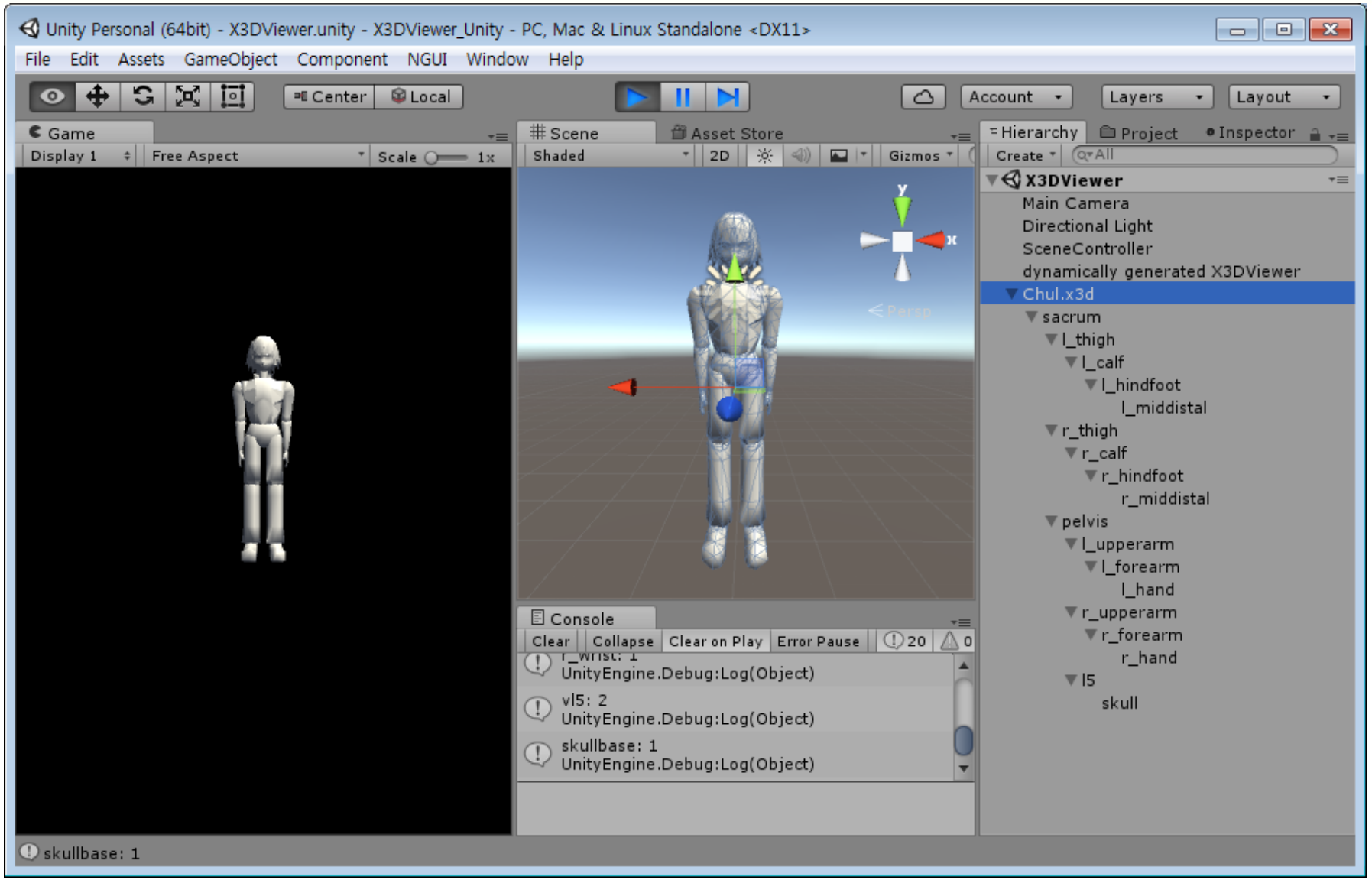


ISO/IEC NP 19777-5
X3D C# Language Binding
Annex C Examples
(Implementation)

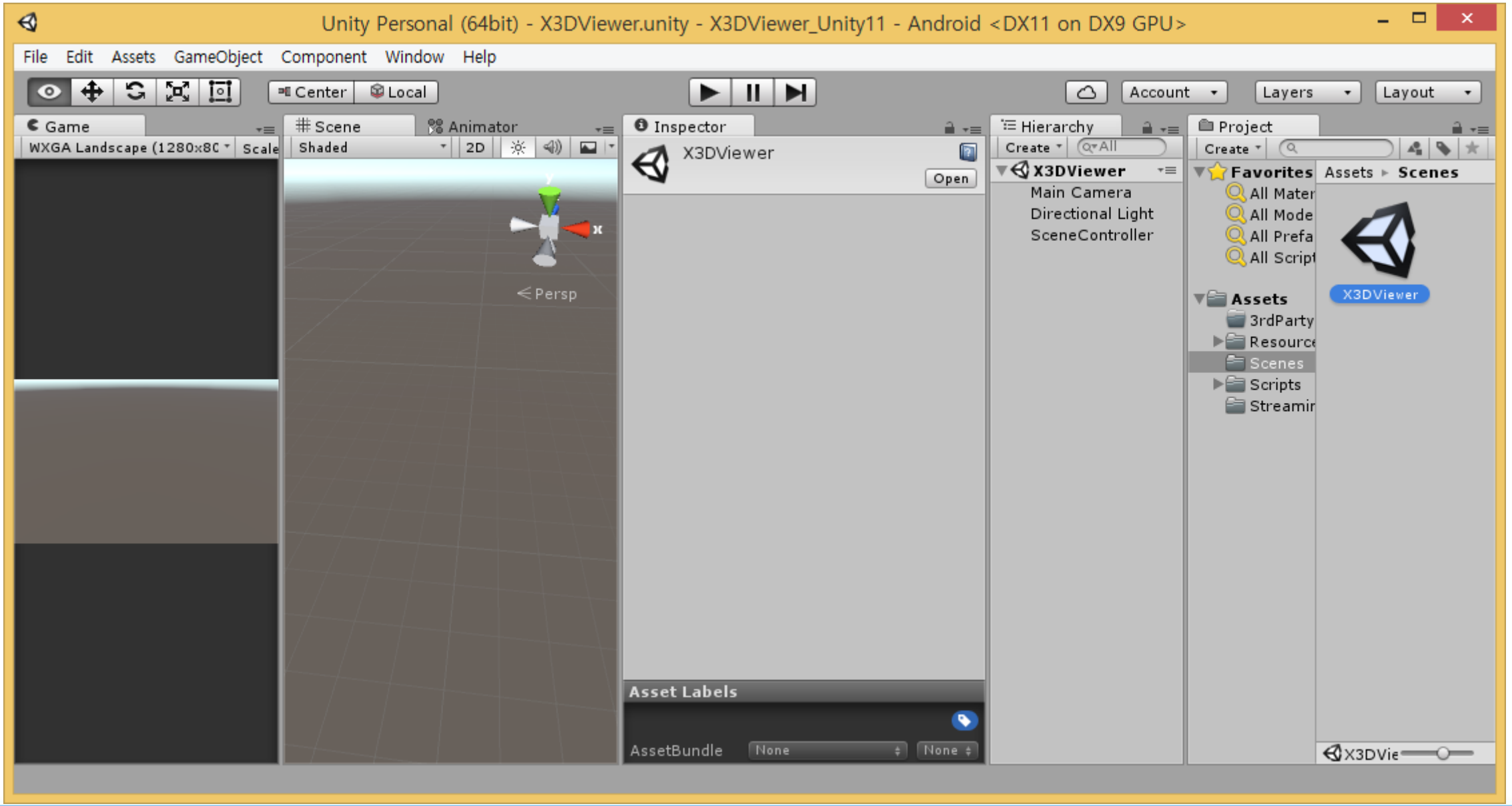
X3D C# Binding Viewer (Unity)



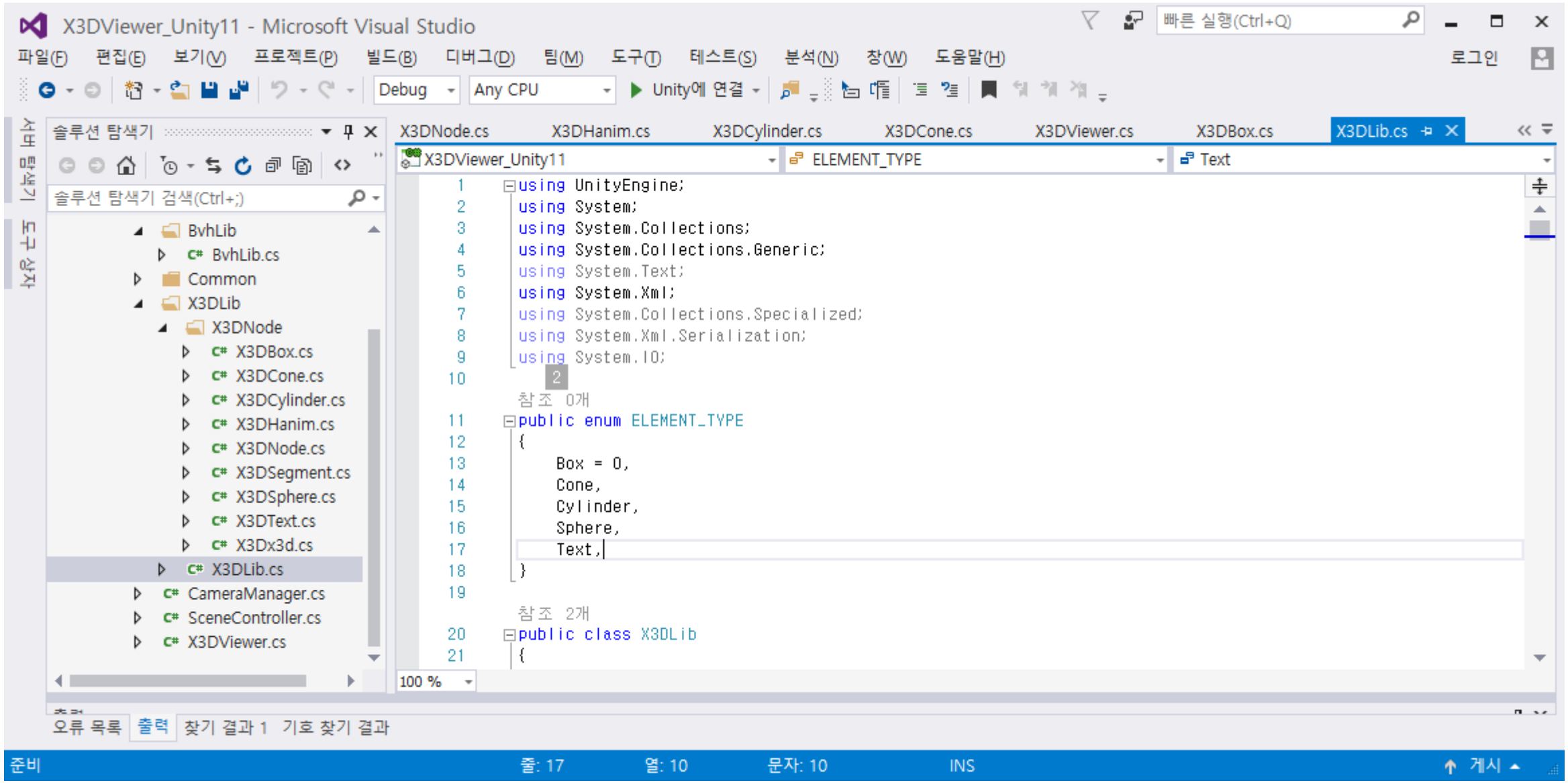
X3D C# Binding Viewer (Unity)



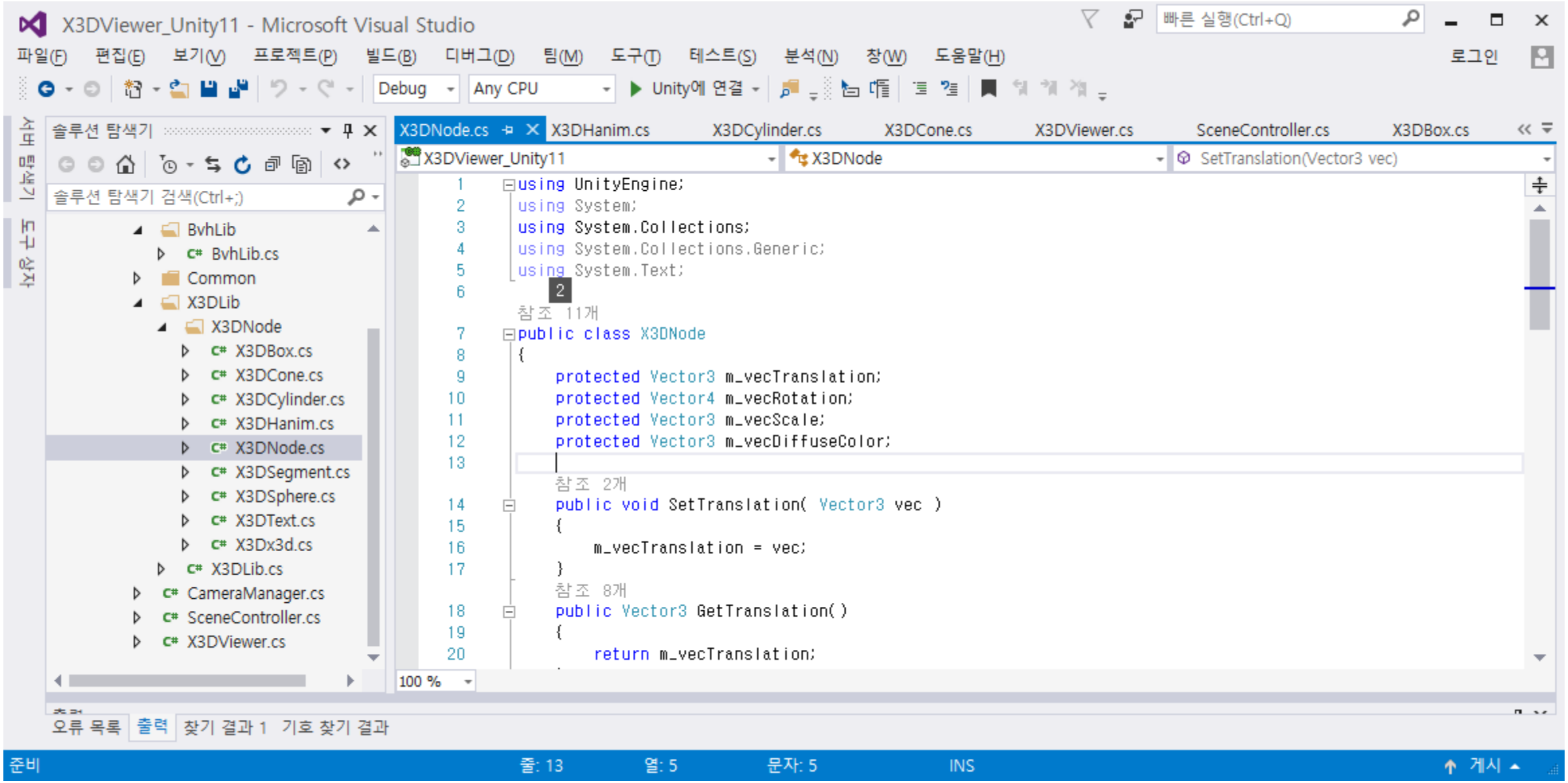
X3D C# Binding Viewer (Unity)



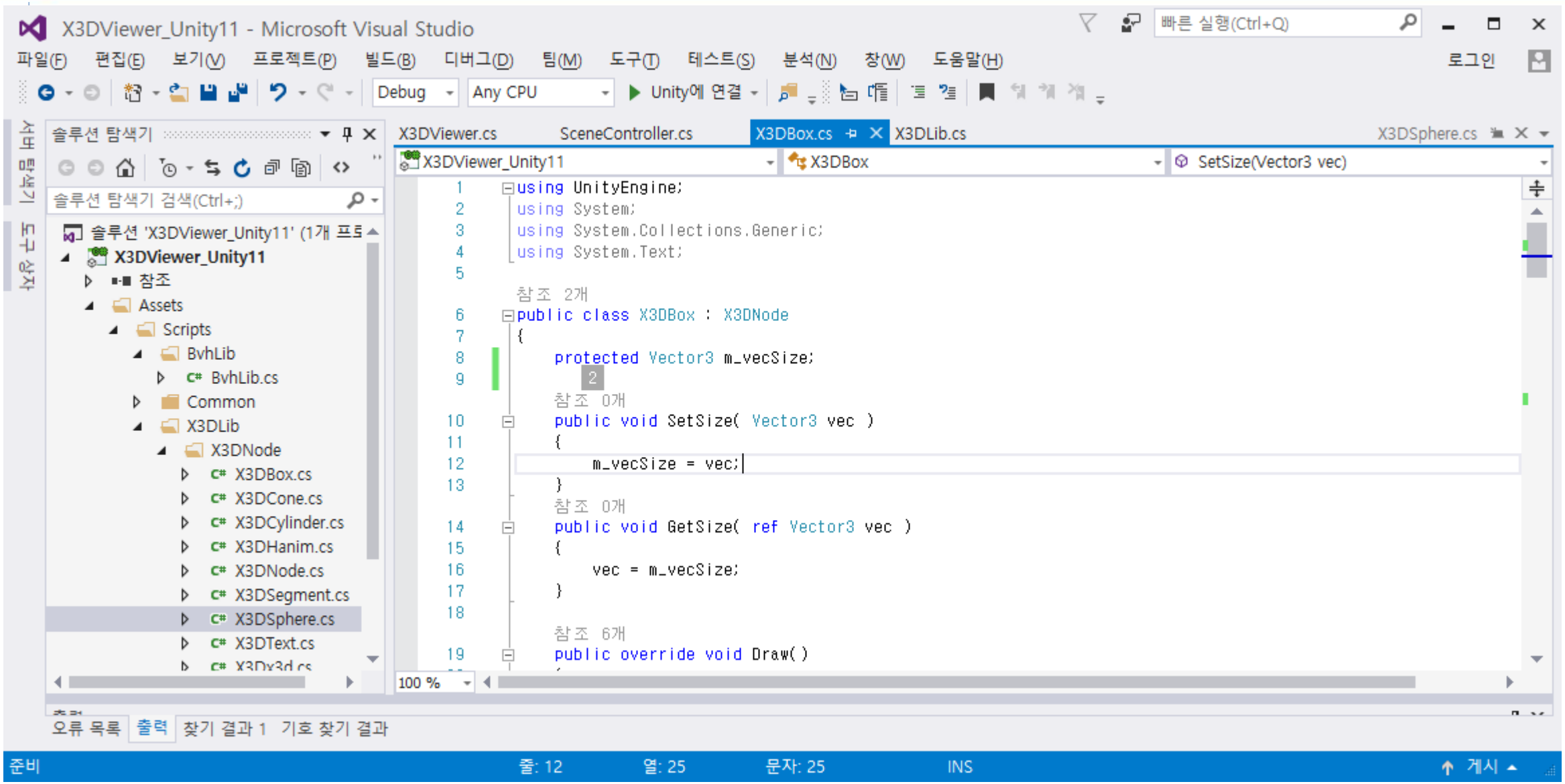
Lib Class


X3DViewer_Unity11 - Microsoft Visual Studio
빠른 실행(Ctrl+Q)
파일(F) 편집(E) 보기(V) 프로젝트(P) 빌드(B) 디버그(D) 팀(M) 도구(T) 테스트(S) 분석(N) 창(W) 도움말(H)
로그인
Debug
Any CPU
Unity에 연결
솔루션 탐색기
솔루션 탐색기 검색(Ctrl+;)
BvhLib
BvhLib.cs
Common
X3DLib
X3DNode
X3DBox.cs
X3DCone.cs
X3DCylinder.cs
X3DHanim.cs
X3DNode.cs
X3DSegment.cs
X3DSphere.cs
X3DText.cs
X3Dx3d.cs
X3DLib.cs
CameraManager.cs
SceneController.cs
X3DViewer.cs
X3DNode.cs X3DHanim.cs X3DCylinder.cs X3DCone.cs X3DViewer.cs X3DBox.cs X3DLib.cs
X3DViewer_Unity11
ELEMENT_TYPE
Text
1 using UnityEngine;
2 using System;
3 using System.Collections;
4 using System.Collections.Generic;
5 using System.Text;
6 using System.Xml;
7 using System.Collections.Specialized;
8 using System.Xml.Serialization;
9 using System.IO;
10
참조 0개
11 public enum ELEMENT_TYPE
12 {
13 Box = 0,
14 Cone,
15 Cylinder,
16 Sphere,
17 Text,
18 }
19
참조 2개
20 public class X3DLib
21 {
100 %
오류 목록 출력 찾기 결과 1 기호 찾기 결과
준비
줄: 17
열: 10
문자: 10
INS
게시

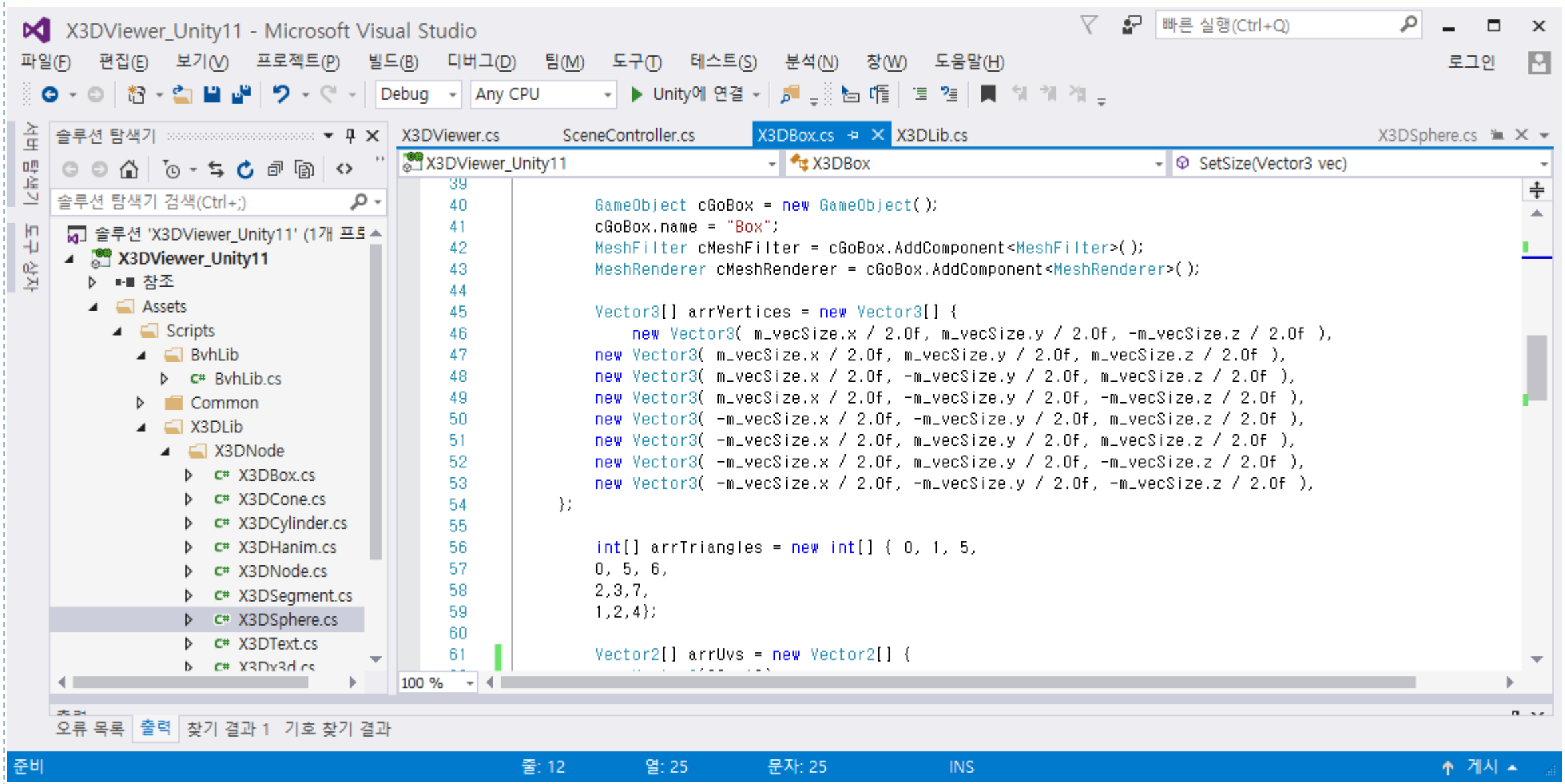
Base Node


X3DViewer_Unity11 - Microsoft Visual Studio
빠른 실행(Ctrl+Q)
파일(F) 편집(E) 보기(V) 프로젝트(P) 빌드(B) 디버그(D) 팀(M) 도구(T) 테스트(S) 분석(N) 창(W) 도움말(H)
로그인
Debug
Any CPU
Unity에 연결
솔루션 탐색기
솔루션 탐색기 검색(Ctrl+;)
BvhLib
BvhLib.cs
Common
X3DLib
X3DNode
X3DBox.cs
X3DCone.cs
X3DCylinder.cs
X3DHanim.cs
X3DNode.cs
X3DSegment.cs
X3DSphere.cs
X3DText.cs
X3Dx3d.cs
X3DLib.cs
CameraManager.cs
SceneController.cs
X3DViewer.cs
X3DNode.cs
X3DHanim.cs
X3DCylinder.cs
X3DCone.cs
X3DViewer.cs
SceneController.cs
X3DBox.cs
X3DViewer_Unity11
X3DNode
SetTranslation(Vector3 vec)
using UnityEngine;
using System;
using System.Collections;
using System.Collections.Generic;
using System.Text;
참조 11개
public class X3DNode
{
protected Vector3 m_vecTranslation;
protected Vector4 m_vecRotation;
protected Vector3 m_vecScale;
protected Vector3 m_vecDiffuseColor;
참조 2개
public void SetTranslation(Vector3 vec)
{
m_vecTranslation = vec;
}
참조 8개
public Vector3 GetTranslation( )
{
return m_vecTranslation;
100 %
오류 목록
출력
찾기 결과 1
기호 찾기 결과
준비
줄: 13
열: 5
문자: 5
INS
게시

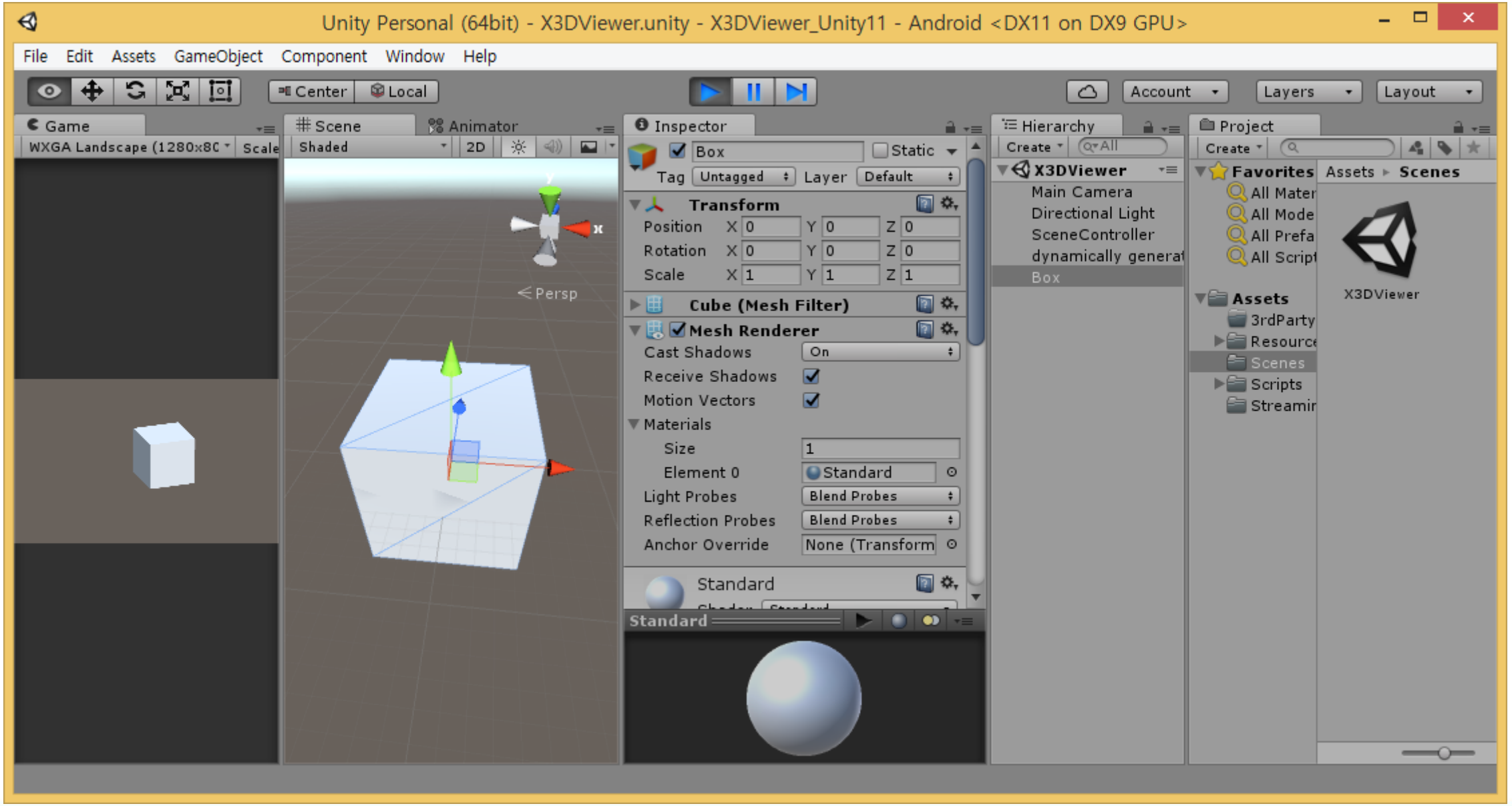
Box Class


X3DViewer_Unity11 - Microsoft Visual Studio
빠른 실행(Ctrl+Q)
파일(F) 편집(E) 보기(V) 프로젝트(P) 빌드(B) 디버그(D) 팀(M) 도구(T) 테스트(S) 분석(N) 창(W) 도움말(H)
로그인
Debug
Any CPU
Unity에 연결
솔루션 탐색기
솔루션 탐색기 검색(Ctrl+;)
솔루션 'X3DViewer_Unity11' (1개 프로
X3DViewer_Unity11
참조
Assets
Scripts
BvhLib
BvhLib.cs
Common
X3DLib
X3DNode
X3DBox.cs
X3DCone.cs
X3DCylinder.cs
X3DHanim.cs
X3DNode.cs
X3DSegment.cs
X3DSphere.cs
X3DText.cs
X3Dx3d.cs
X3DViewer.cs
SceneController.cs
X3DBox.cs
X3DLib.cs
X3DSphere.cs
X3DViewer_Unity11
X3DBox
SetSize(Vector3 vec)
1 using UnityEngine;
2 using System;
3 using System.Collections.Generic;
4 using System.Text;
5
참조 2개
6 public class X3DBox : X3DNode
7 {
8 protected Vector3 m_vecSize;
9
참조 0개
10 public void SetSize(Vector3 vec)
11 {
12 m_vecSize = vec;
13 }
참조 0개
14 public void GetSize(ref Vector3 vec)
15 {
16 vec = m_vecSize;
17 }
18
참조 6개
19 public override void Draw()
100 %
오류 목록
출력
찾기 결과 1
기호 찾기 결과
준비
줄: 12
열: 25
문자: 25
INS
게시

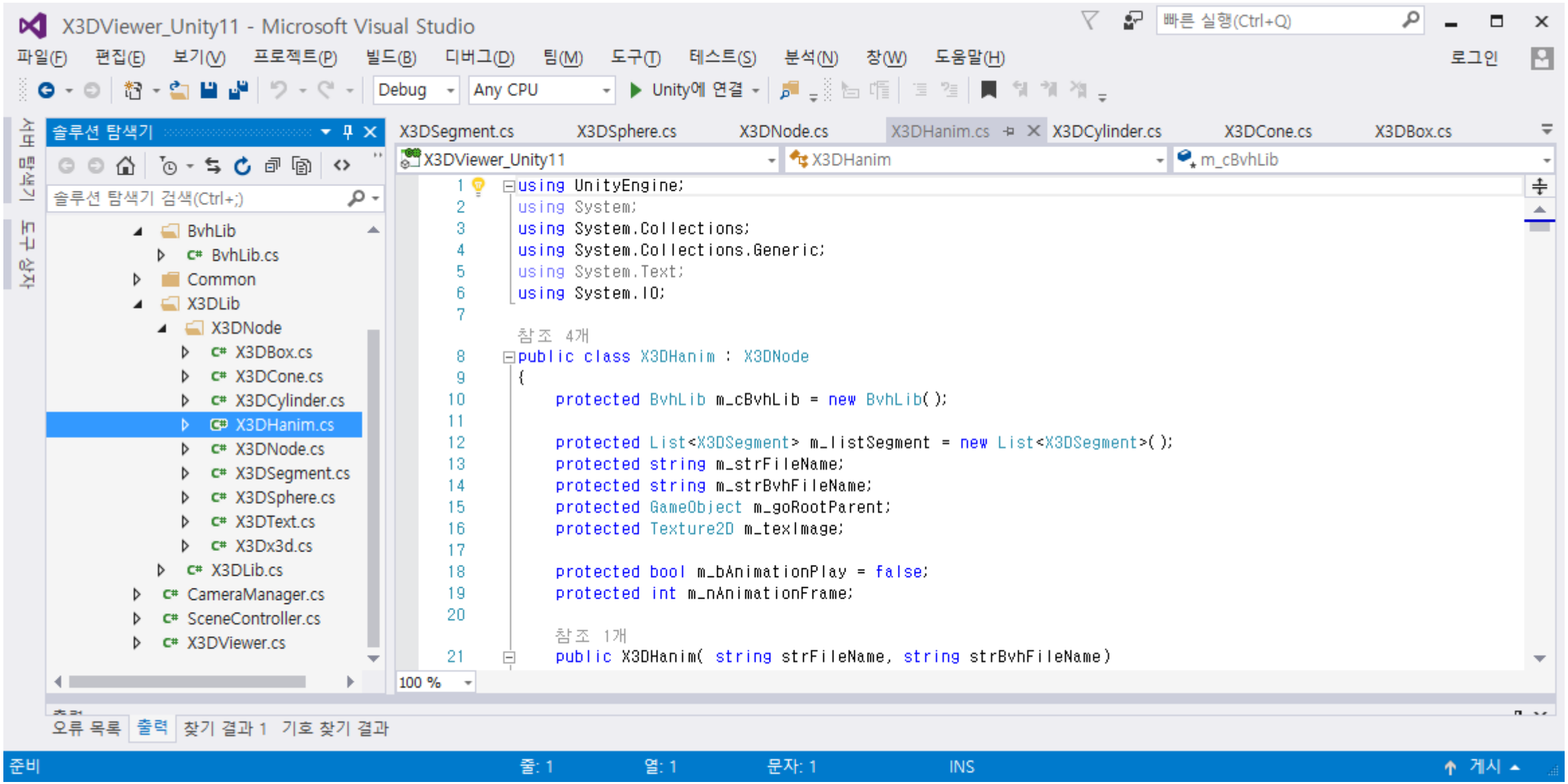
Box Class


X3DViewer_Unity11 - Microsoft Visual Studio
빠른 실행(Ctrl+Q)
파일(F) 편집(E) 보기(V) 프로젝트(P) 빌드(B) 디버그(D) 팀(M) 도구(T) 테스트(S) 분석(N) 창(W) 도움말(H)
로그인
Debug
Any CPU
Unity에 연결
솔루션 탐색기
솔루션 탐색기 검색(Ctrl+;)
솔루션 'X3DViewer_Unity11' (1개 프로
X3DViewer_Unity11
참조
Assets
Scripts
BvhLib
BvhLib.cs
Common
X3DLib
X3DNode
X3DBox.cs
X3DCone.cs
X3DCylinder.cs
X3DHanim.cs
X3DNode.cs
X3DSegment.cs
X3DSphere.cs
X3DText.cs
X3Dx3d.cs
X3DViewer.cs
SceneController.cs
X3DBox.cs
X3DLib.cs
X3DSphere.cs
X3DViewer_Unity11
X3DBox
SetSize(Vector3 vec)
GameObject cGoBox = new GameObject();
cGoBox.name = "Box";
MeshFilter cMeshFilter = cGoBox.AddComponent<MeshFilter>();
MeshRenderer cMeshRenderer = cGoBox.AddComponent<MeshRenderer>();
Vector3[] arrVertices = new Vector3[] {
new Vector3(m_vecSize.x / 2.0f, m_vecSize.y / 2.0f, -m_vecSize.z / 2.0f),
new Vector3(m_vecSize.x / 2.0f, m_vecSize.y / 2.0f, m_vecSize.z / 2.0f),
new Vector3(m_vecSize.x / 2.0f, -m_vecSize.y / 2.0f, m_vecSize.z / 2.0f),
new Vector3(m_vecSize.x / 2.0f, -m_vecSize.y / 2.0f, -m_vecSize.z / 2.0f),
new Vector3(-m_vecSize.x / 2.0f, -m_vecSize.y / 2.0f, m_vecSize.z / 2.0f),
new Vector3(-m_vecSize.x / 2.0f, m_vecSize.y / 2.0f, m_vecSize.z / 2.0f),
new Vector3(-m_vecSize.x / 2.0f, m_vecSize.y / 2.0f, -m_vecSize.z / 2.0f),
new Vector3(-m_vecSize.x / 2.0f, -m_vecSize.y / 2.0f, -m_vecSize.z / 2.0f),
};
int[] arrTriangles = new int[] { 0, 1, 5,
0, 5, 6,
2,3,7,
1,2,4};
Vector2[] arrUvs = new Vector2[] {
100 %
오류 목록 출력 찾기 결과 1 기호 찾기 결과
준비
줄: 12
열: 25
문자: 25
INS
게시

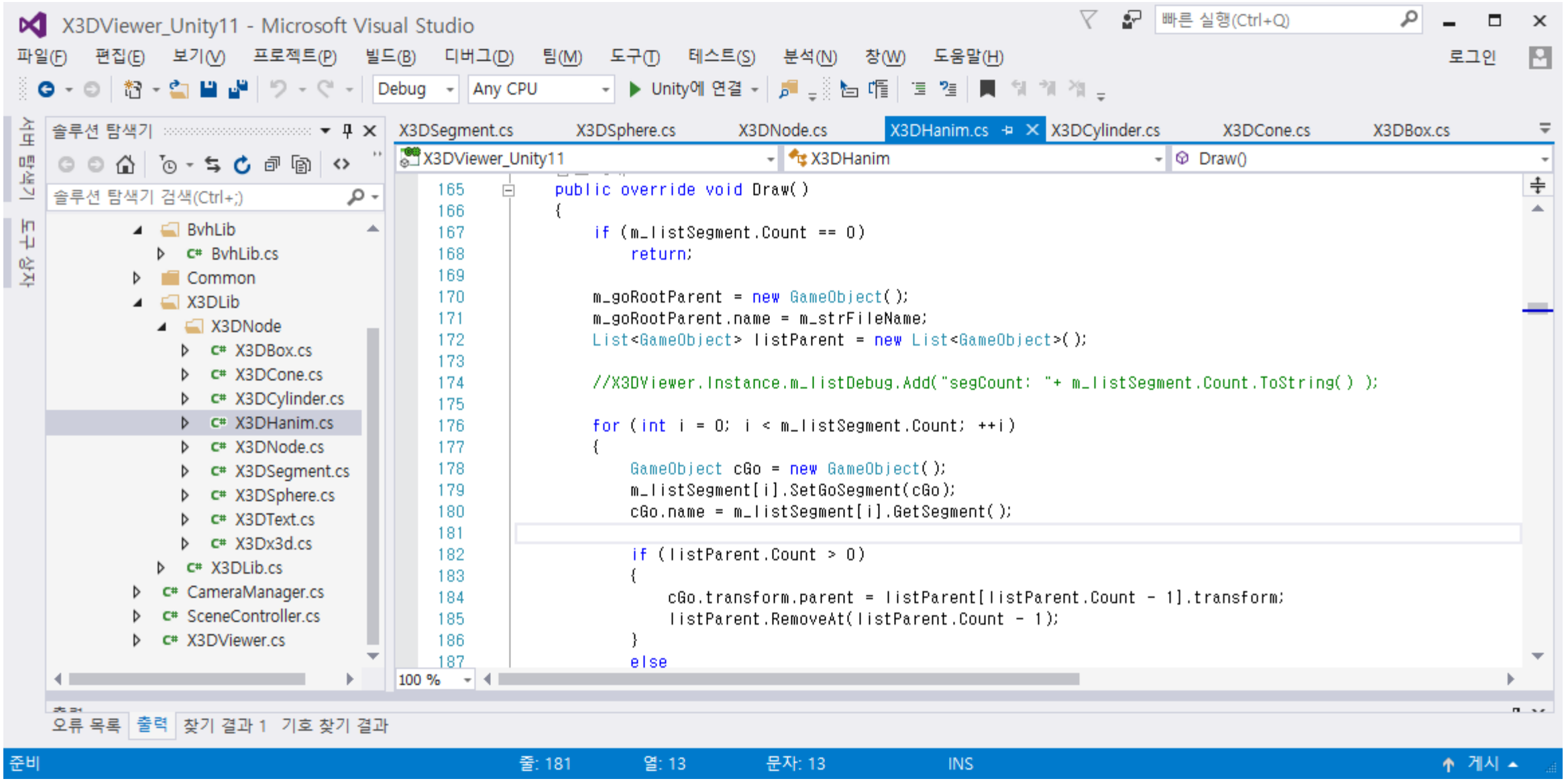
Box Class


Unity Personal (64bit) - X3DViewer.unity - X3DViewer_Unity11 - Android <DX11 on DX9 GPU>
File Edit Assets GameObject Component Window Help
Center Local
Account Layers Layout
Game
Scene
Animator
Inspector
Hierarchy
Project
Shaded
2D
Persp
Box
Static
Tag Untagged
Layer Default
Transform
Position X 0 Y 0 Z 0
Rotation X 0 Y 0 Z 0
Scale X 1 Y 1 Z 1
Cube (Mesh Filter)
Mesh Renderer
Cast Shadows On
Receive Shadows
Motion Vectors
Materials
Size 1
Element 0 Standard
Light Probes Blend Probes
Reflection Probes Blend Probes
Anchor Override None (Transform
Standard
X3DViewer
Main Camera
Directional Light
SceneController
Box
Favorites
Assets
Scenes
Scripts
X3DViewer

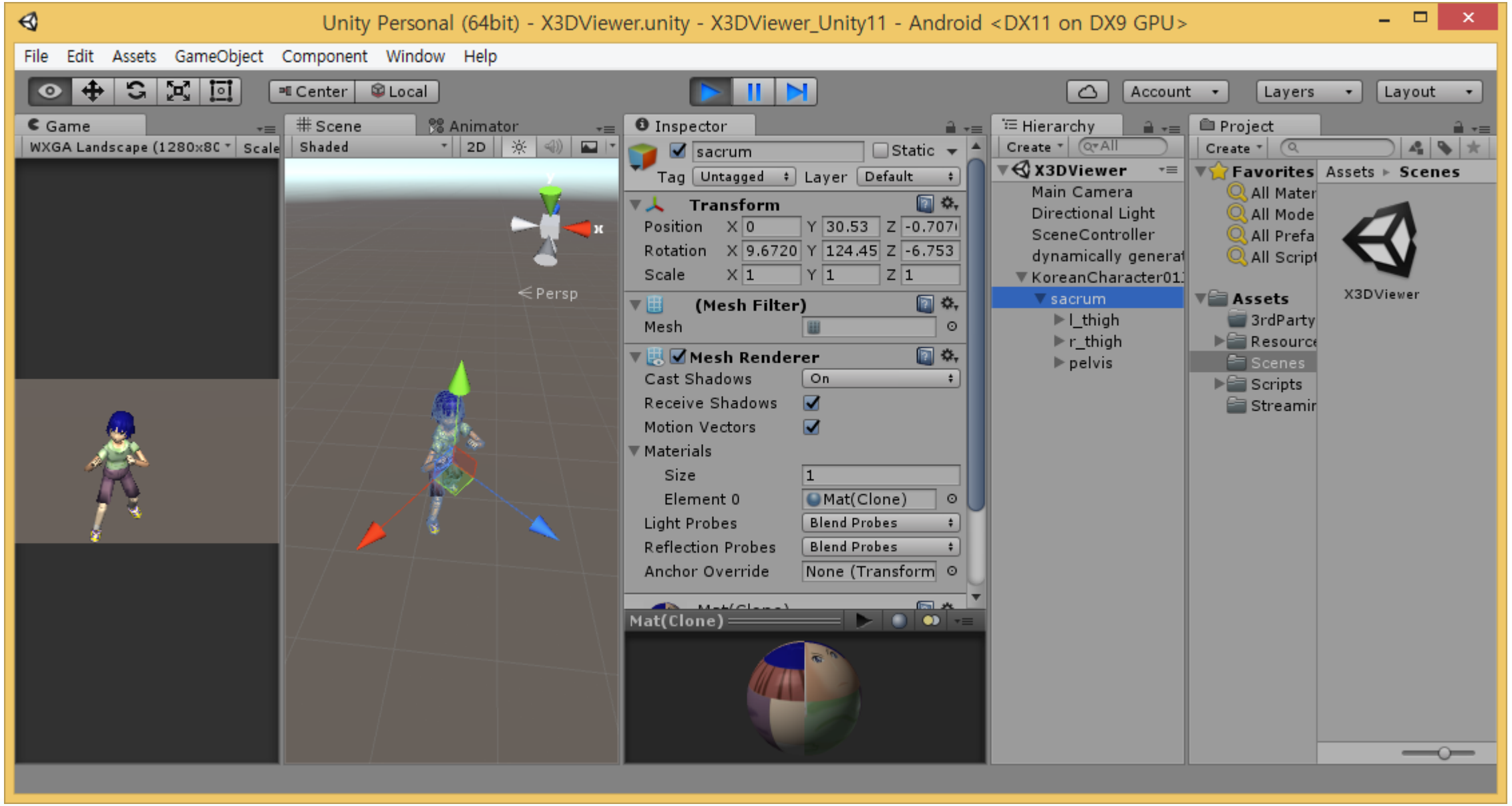
H-Anim Class


X3DViewer_Unity11 - Microsoft Visual Studio
빠른 실행(Ctrl+Q)
파일(F) 편집(E) 보기(V) 프로젝트(P) 빌드(B) 디버그(D) 팀(M) 도구(T) 테스트(S) 분석(N) 창(W) 도움말(H)
로그인
Debug
Any CPU
Unity에 연결
솔루션 탐색기
솔루션 탐색기 검색(Ctrl+;)
BvhLib
BvhLib.cs
Common
X3DLib
X3DNode
X3DBox.cs
X3DCone.cs
X3DCylinder.cs
X3DHanim.cs
X3DNode.cs
X3DSegment.cs
X3DSphere.cs
X3DText.cs
X3Dx3d.cs
X3DLib.cs
CameraManager.cs
SceneController.cs
X3DViewer.cs
X3DSegment.cs
X3DSphere.cs
X3DNode.cs
X3DHanim.cs
X3DCylinder.cs
X3DCone.cs
X3DBox.cs
X3DViewer_Unity11
X3DHanim
m_cBvhLib
using UnityEngine;
using System;
using System.Collections;
using System.Collections.Generic;
using System.Text;
using System.IO;
참조 4개
public class X3DHanim : X3DNode
{
protected BvhLib m_cBvhLib = new BvhLib();
protected List<X3DSegment> m_listSegment = new List<X3DSegment>();
protected string m_strFileName;
protected string m_strBvhFileName;
protected GameObject m_goRootParent;
protected Texture2D m_texImage;
protected bool m_bAnimationPlay = false;
protected int m_nAnimationFrame;
참조 1개
public X3DHanim(string strFileName, string strBvhFileName)
100 %
오류 목록 출력 찾기 결과 1 기호 찾기 결과
준비
줄: 1
열: 1
문자: 1
INS
게시

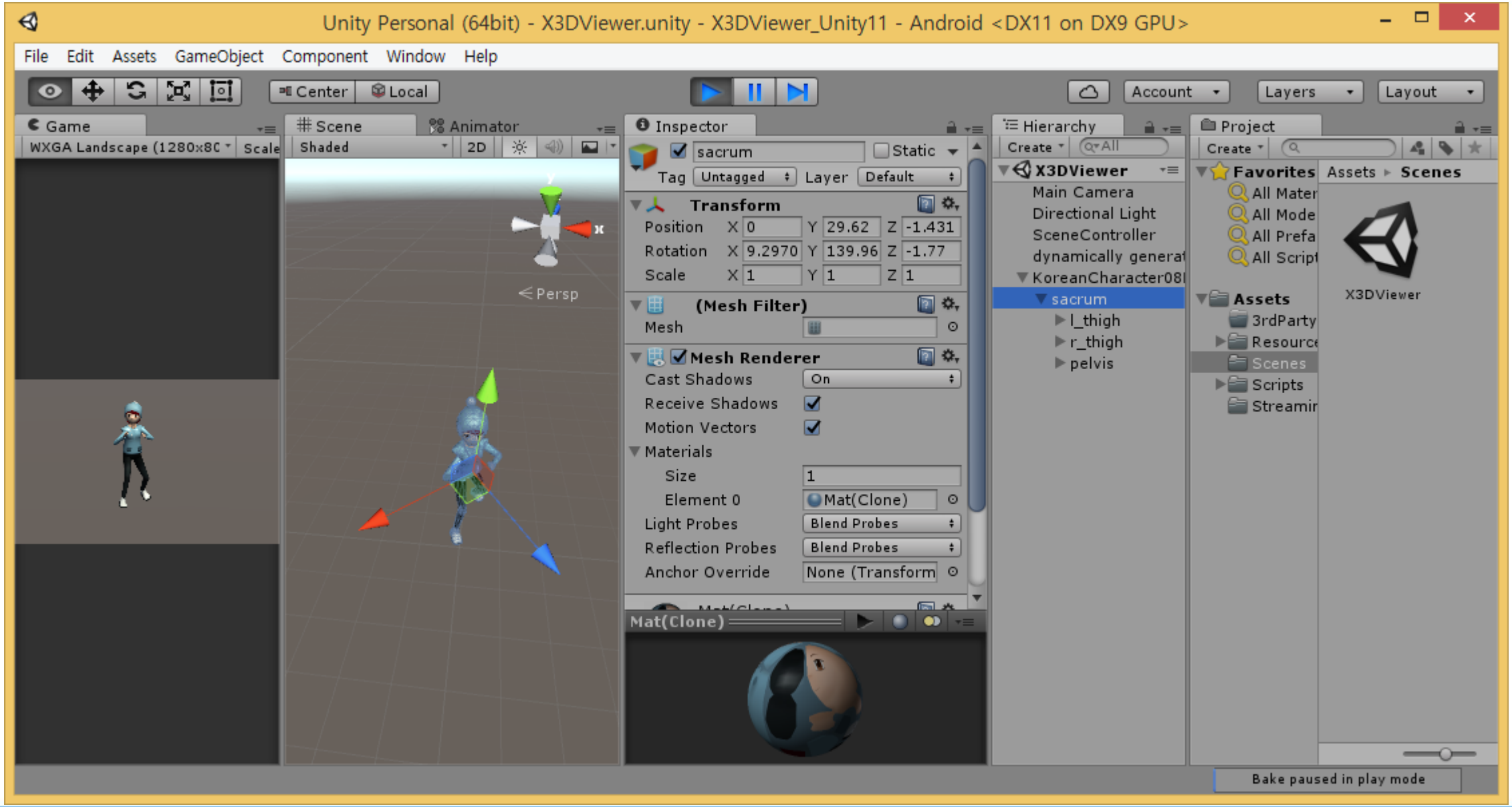
H-Anim Class


X3DViewer_Unity11 - Microsoft Visual Studio
빠른 실행(Ctrl+Q)
파일(F) 편집(E) 보기(V) 프로젝트(P) 빌드(B) 디버그(D) 팀(M) 도구(T) 테스트(S) 분석(N) 창(W) 도움말(H)
로그인
Debug
Any CPU
Unity에 연결
솔루션 탐색기
솔루션 탐색기 검색(Ctrl+;)
BvhLib
BvhLib.cs
Common
X3DLib
X3DNode
X3DBox.cs
X3DCone.cs
X3DCylinder.cs
X3DHanim.cs
X3DNode.cs
X3DSegment.cs
X3DSphere.cs
X3DText.cs
X3Dx3d.cs
X3DLib.cs
CameraManager.cs
SceneController.cs
X3DViewer.cs
X3DSegment.cs
X3DSphere.cs
X3DNode.cs
X3DHanim.cs
X3DCylinder.cs
X3DCone.cs
X3DBox.cs
X3DViewer_Unity11
X3DHanim
Draw()
public override void Draw( )
{
if (m_listSegment.Count == 0)
return;
m_goRootParent = new GameObject( );
m_goRootParent.name = m_strFileName;
List<GameObject> listParent = new List<GameObject>( );
//X3DViewer.Instance.m_listDebug.Add("segCount: "+ m_listSegment.Count.ToString( ));
for (int i = 0; i < m_listSegment.Count; ++i)
{
GameObject cGo = new GameObject( );
m_listSegment[i].SetGoSegment(cGo);
cGo.name = m_listSegment[i].GetSegment( );
if (listParent.Count > 0)
{
cGo.transform.parent = listParent[listParent.Count - 1].transform;
listParent.RemoveAt(listParent.Count - 1);
}
else
100 %
오류 목록
출력
찾기 결과 1
기호 찾기 결과
준비
줄: 181
열: 13
문자: 13
INS
게시

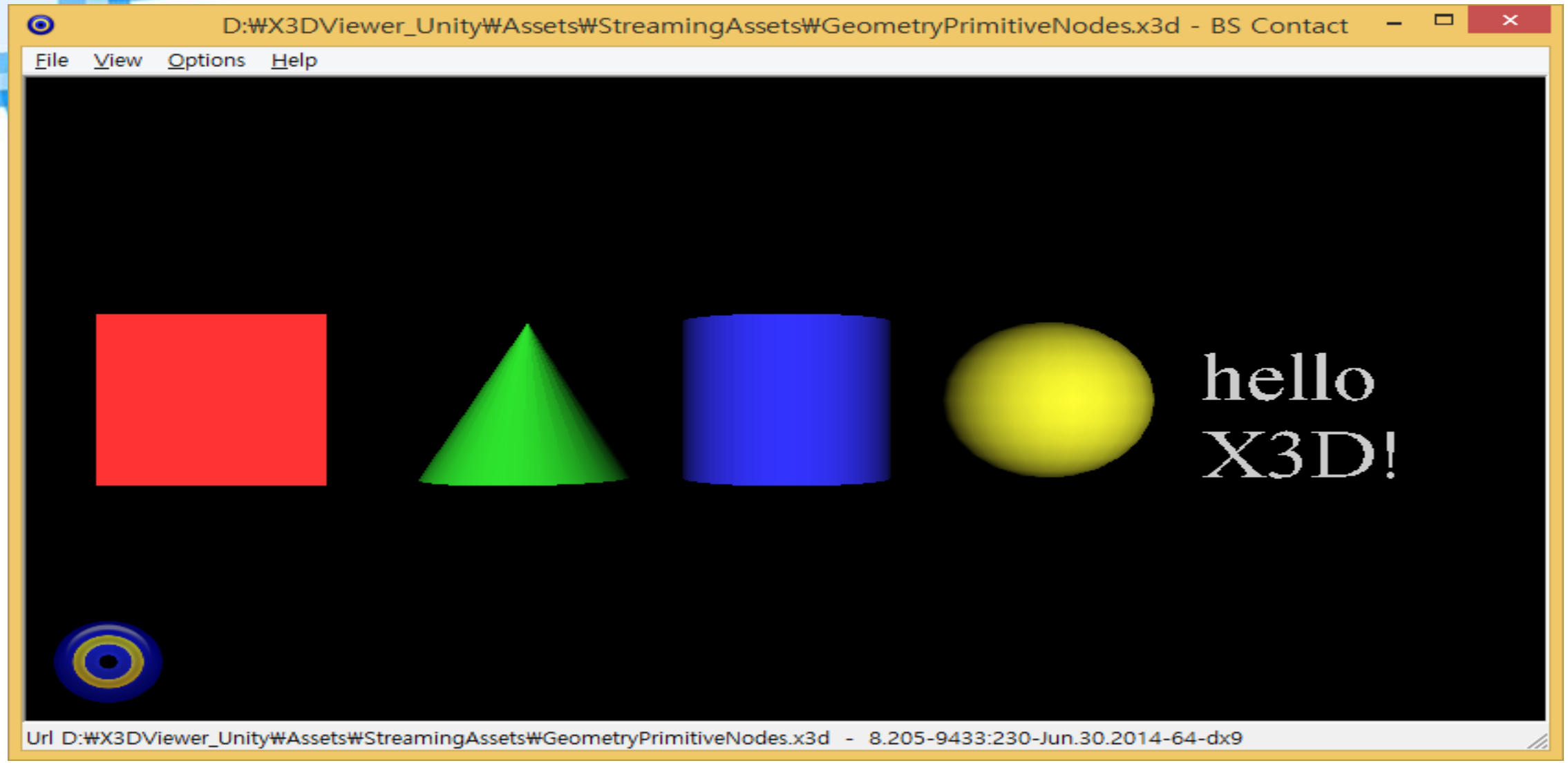
H-Anim Character Animation



H-Anim Character Animation (Video)



GeometryPrimitiveNodes.x3d



Box Parsing

```
case "Box":
    X3DBox x3dBox = new X3DBox();
    x3dBox.SetSize(Parse_Vector3(Parse_AttributeValue(xnRoot, "size")));

    x3dBox.SetTranslation(m_vecParseTranslation);
    x3dBox.SetRotation(m_vecParseRotation);
    x3dBox.SetScale(m_vecParseScale);
    m_listX3DNode.Add(x3dBox);
    break;
```

Box Class

```
public class X3DBox : X3DNode
{
    protected Vector3 m_vecSize;

    참조 1개
    public void SetSize( Vector3 vec )
    {
        m_vecSize = vec;
    }
    참조 0개
    public void GetSize( ref Vector3 vec )
    {
        vec = m_vecSize;
    }

    참조 9개
    public override IEnumerator LoadEndAction()
    {
        Draw();
        yield return null;
    }

    참조 14개
    public override void Draw()
    {
        GameObject goBox = GameObject.Instantiate(Resources.Load("Box") as GameObject);
        goBox.transform.localPosition = m_vecTranslation;
        goBox.transform.localRotation = Quaternion.Euler(GetRotation());
        goBox.transform.localScale = m_vecSize;

        goBox.GetComponent<MeshRenderer>().material.SetColor("_Color", new Color(m_vecDiffuseColor.x, m_vecDiffuseColor.y, m_
    }
}
```

Box Unity

Cone Parsing

```
case "Cone":
    X3DCone x3dCone = new X3DCone();
    x3dCone.SetBottom(Parse_Bool(Parse_AttributeValue(xnRoot, "bottom")));
    x3dCone.SetBottomRadius(Parse_Float(Parse_AttributeValue(xnRoot, "bottomRadius")));
    x3dCone.SetHeight(Parse_Float(Parse_AttributeValue(xnRoot, "height")));
    x3dCone.SetSide(Parse_Bool(Parse_AttributeValue(xnRoot, "side")));

    x3dCone.SetTranslation(m_vecParseTranslation);
    x3dCone.SetRotation(m_vecParseRotation);
    x3dCone.SetScale(m_vecParseScale);
    m_listX3DNode.Add(x3dCone);
    break;
```

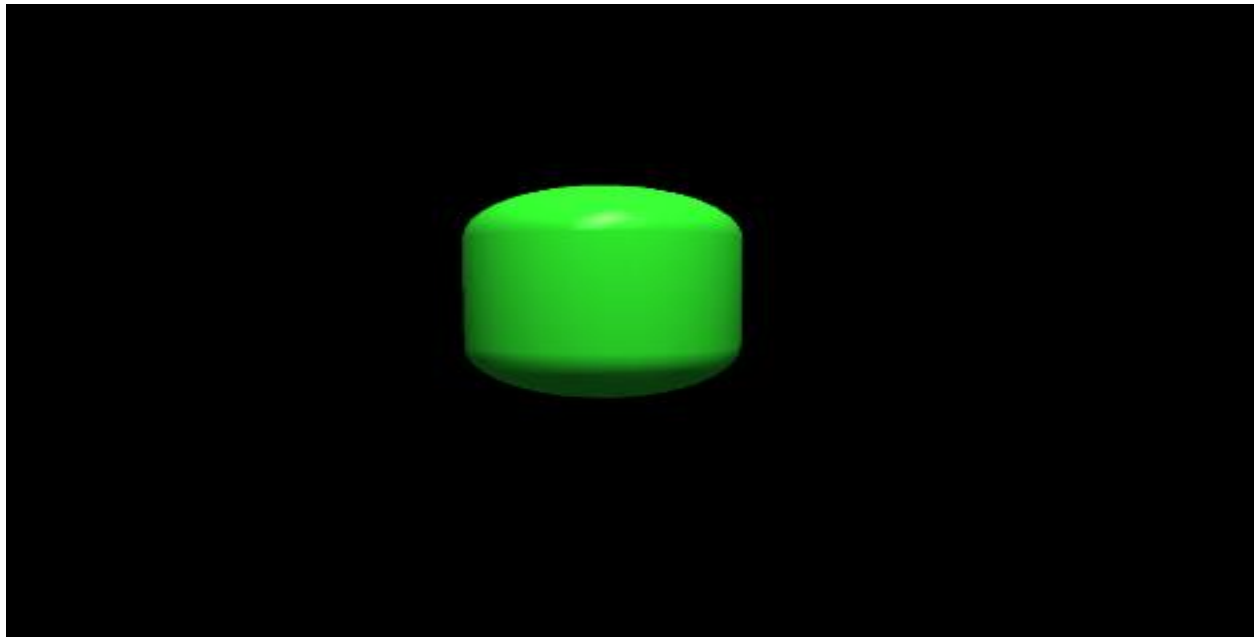
Cone Class

```
public class X3DCone : X3DNode
{
    protected bool m_bBottom;
    protected float m_fBottomRadius;
    protected float m_fHeight;
    protected bool m_bSide;

    참조 1개
    public void SetBottom(bool bBottom)
    {
        m_bBottom = bBottom;
    }
    참조 0개
    public void GetBottom(ref bool bBottom)
    {
        bBottom = m_bBottom;
    }
        2
    참조 1개
    public void SetBottomRadius(float fBottomRadius)
    {
        m_fBottomRadius = fBottomRadius;
    }
    참조 0개
    public void GetBottomRadius(ref float fBottomRadius)
    {
        fBottomRadius = m_fBottomRadius;
    }

    참조 1개
    public void SetHeight(float fHeight)
    {
        m_fHeight = fHeight;
    }
```

Cone Class

```
public void GetHeight(ref float fHeight)
{
    fHeight = m_fHeight;
}

참조 1개
public void SetSide(bool bSide)
{
    m_bSide = bSide;
}
참조 0개
public void GetSide(ref bool bSide)
{
    bSide = m_bSide;
}

참조 9개
public override IEnumerator LoadEndAction()
{
    Draw();
    yield return null;
}

참조 14개
public override void Draw()
{
    GameObject goCone = GameObject.Instantiate(Resources.Load("Cone") as GameObject);
    goCone.transform.localPosition = m_vecTranslation;
    goCone.transform.localRotation = Quaternion.Euler(GetRotation());
    goCone.transform.localScale = new Vector3(m_fBottomRadius * 2f, m_fHeight*0.5f, m_fBottomRadius * 2f);

    goCone.GetComponent<MeshRenderer>().material.SetColor("_Color", new Color(m_vecDiffuseColor.x, m_vecDiffuseColor.y, m_vecDiffuseCol
}
```

Cone Unity



Cylinder Parsing

```
case "Cylinder":
    X3DCylinder X3DCylinder = new X3DCylinder();
    X3DCylinder.SetBottom(Parse_Bool(Parse_AttributeValue(xnRoot, "bottom")));
    X3DCylinder.SetRadius(Parse_Float(Parse_AttributeValue(xnRoot, "radius")));
    X3DCylinder.SetHeight(Parse_Float(Parse_AttributeValue(xnRoot, "height")));
    X3DCylinder.SetSide(Parse_Bool(Parse_AttributeValue(xnRoot, "side")));
    X3DCylinder.SetTop(Parse_Bool(Parse_AttributeValue(xnRoot, "top")));

    X3DCylinder.SetTranslation(m_vecParseTranslation);
    X3DCylinder.SetRotation(m_vecParseRotation);
    X3DCylinder.SetScale(m_vecParseScale);
    m_listX3DNode.Add(X3DCylinder);
    break;
```

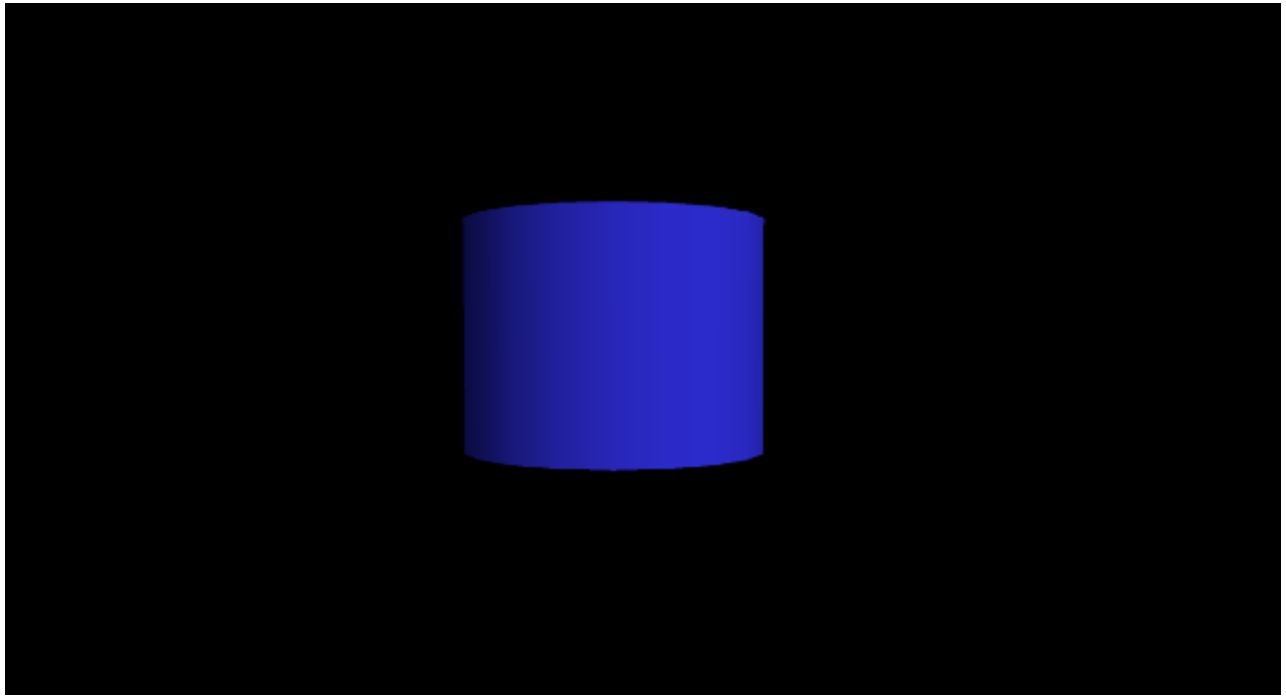
Cylinder Class

```
public class X3DCylinder : X3DNode
{
    protected bool m_bBottom;
    protected float m_fRadius;
    protected float m_fHeight;
    protected bool m_bSide;
    protected bool m_bTop;
        2
    참조 1개
    public void SetBottom(bool bBottom)
    {
        m_bBottom = bBottom;
    }
    참조 0개
    public void GetBottom(ref bool bBottom)
    {
        bBottom = m_bBottom;
    }

    참조 1개
    public void SetRadius(float fRadius)
    {
        m_fRadius = fRadius;
    }
    참조 0개
    public void GetRadius(ref float fRadius)
    {
        fRadius = m_fRadius;
    }
```

Cylinder Class

```
public void GetSide(ref bool bSide)
{
    bSide = m_bSide;
}

참조 1개
public void SetTop(bool bTop)
{
    m_bTop = bTop;
}
참조 0개
public void GetTop(ref bool bTop)
{
    bTop = m_bTop;
}

참조 9개
public override IEnumerator LoadEndAction()
{
    Draw();
    yield return null;
}
    2
참조 14개
public override void Draw()
{
    GameObject goCylinder = GameObject.Instantiate(Resources.Load("Cylinder") as GameObject);
    goCylinder.transform.localPosition = m_vecTranslation;
    goCylinder.transform.localRotation = Quaternion.Euler(GetRotation());
    goCylinder.transform.localScale = new Vector3(m_fRadius*2f, m_fHeight*0.5f, m_fRadius * 2f);

    goCylinder.GetComponent<MeshRenderer>().material.SetColor("_Color", new Color(m_vecDiffuseColor.x, m_vecDiffuseColor.y,
}
```

Cylinder Unity



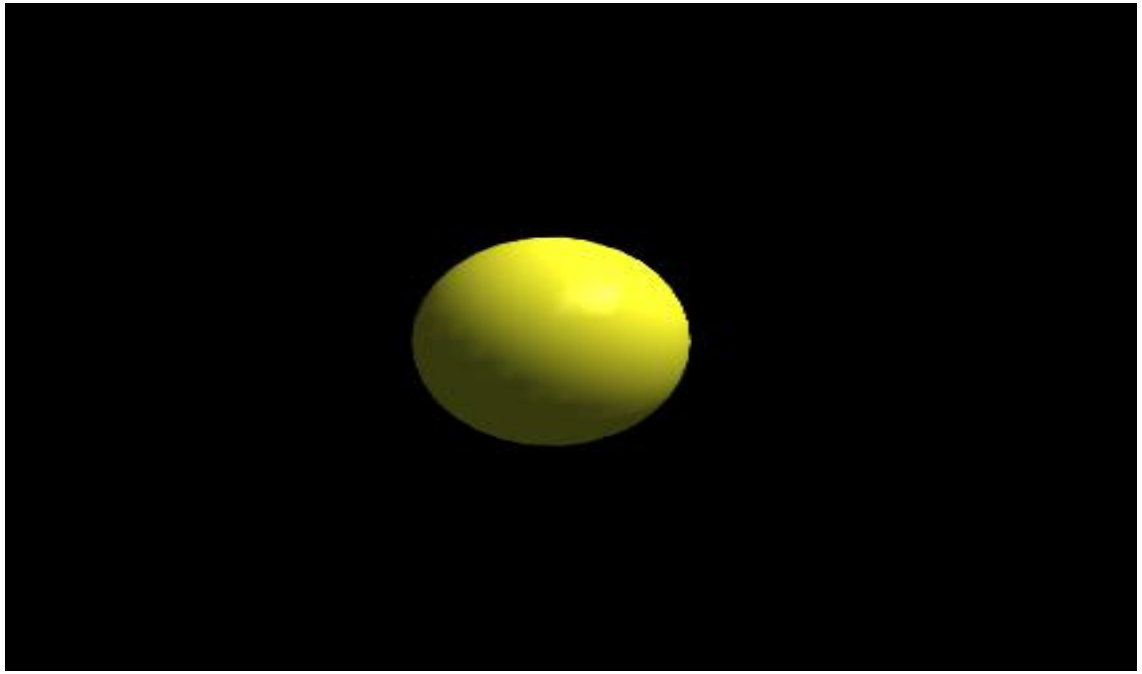
Sphere Parsing

```
case "Sphere":
    X3DSphere x3dSphere = new X3DSphere();
    x3dSphere.SetRadius(Parse_Float(Parse_AttributeValue(xnRoot, "radius")));

    x3dSphere.SetTranslation(m_vecParseTranslation);
    x3dSphere.SetRotation(m_vecParseRotation);
    x3dSphere.SetScale(m_vecParseScale);
    m_listX3DNode.Add(x3dSphere);
    break;
```

Sphere Class

```
public class X3DSphere : X3DNode
{
    protected float m_fRadius;

    참조 1개
    public void SetRadius(float fRadius)
    {
        m_fRadius = fRadius;
    }
    참조 0개
    public void GetRadius(ref float fRadius)
    {
        fRadius = m_fRadius;
    }
        2
    참조 9개
    public override IEnumerator LoadEndAction()
    {
        Draw();
        yield return null;
    }

    참조 14개
    public override void Draw()
    {
        GameObject goSphere = GameObject.Instantiate(Resources.Load("Sphere") as GameObject);
        goSphere.transform.localPosition = m_vecTranslation;
        goSphere.transform.localRotation = Quaternion.Euler(GetRotation());
        goSphere.transform.localScale = new Vector3(m_fRadius * 2f, m_fRadius * 2f, m_fRadius * 2f);

        goSphere.GetComponent<MeshRenderer>().material.SetColor("_Color", new Color(m_vecDiffuseColor.x, m_vecDiffuseColor.y
    }
}
```

Sphere Unity



Text Parsing

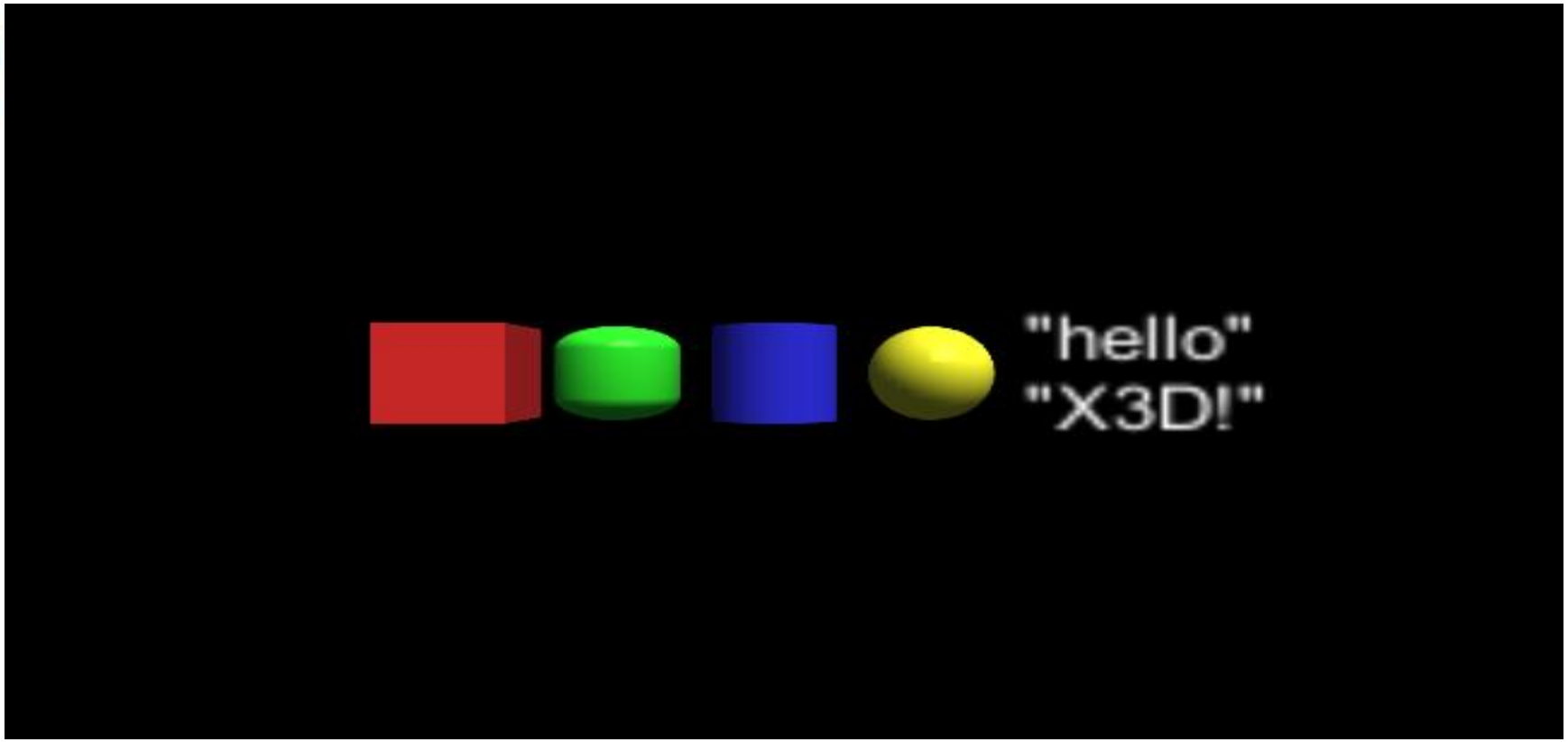
```
case "Text":
    X3DText x3dText = new X3DText();
    List<string> listString = Parse_ListString(Parse_AttributeValue(xnRoot, "string"));
    string strText = "";
    for(int i = 0; i < listString.Count; ++i)
    {
        if(i != 0)
        {
            strText += "₩n";
        }

        strText += listString[i];
    }
    strText.Replace("₩"", "");
    x3dText.SetString(strText);

    x3dText.SetTranslation(m_vecParseTranslation);
    x3dText.SetRotation(m_vecParseRotation);
    x3dText.SetScale(m_vecParseScale);
    m_listX3DNode.Add(x3dText);
    break;
```

Text Class

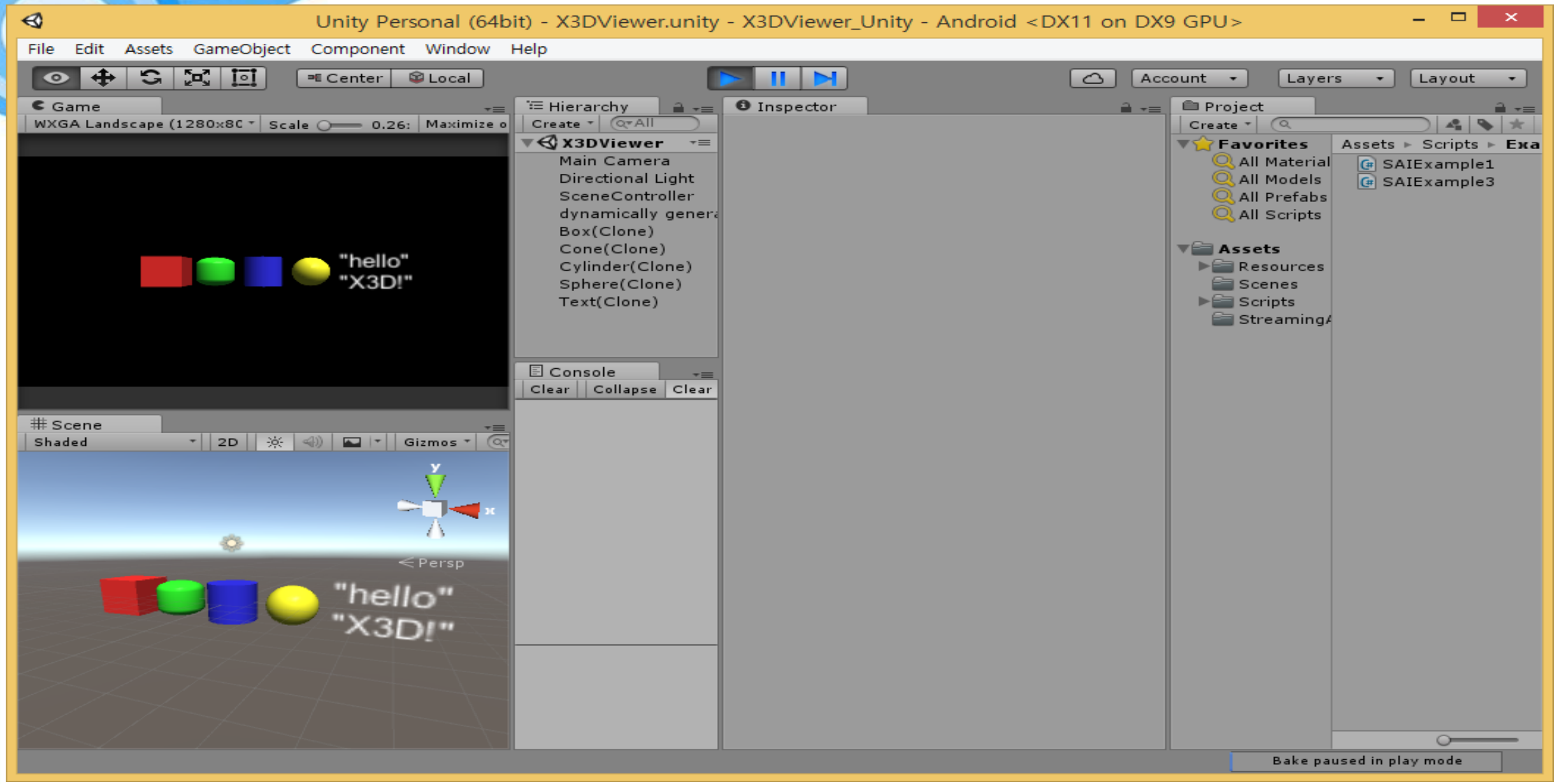
```
public class X3DText : X3DNode
{
    protected string m_strString;

    참조 1개
    public void SetString(string strString)
    {
        m_strString = strString;
    }
    참조 0개
    public void GetString(ref string strString)
    {
        strString = m_strString;
    }

    참조 9개
    public override IEnumerator LoadEndAction()
    {
        Draw();
        yield return null;
    }
        2
    참조 14개
    public override void Draw()
    {
        GameObject goText = GameObject.Instantiate(Resources.Load("Text") as GameObject);
        goText.transform.localPosition = m_vecTranslation;
        goText.transform.localRotation = Quaternion.Euler(GetRotation());
        goText.transform.localScale = GetScale();

        goText.GetComponent<TextMesh>().text = m_strString;
    }
}
```

Text Unity

GeometryPrimitiveNodes.x3d Unity



GeometryPrimitiveNodes.x3d Unity

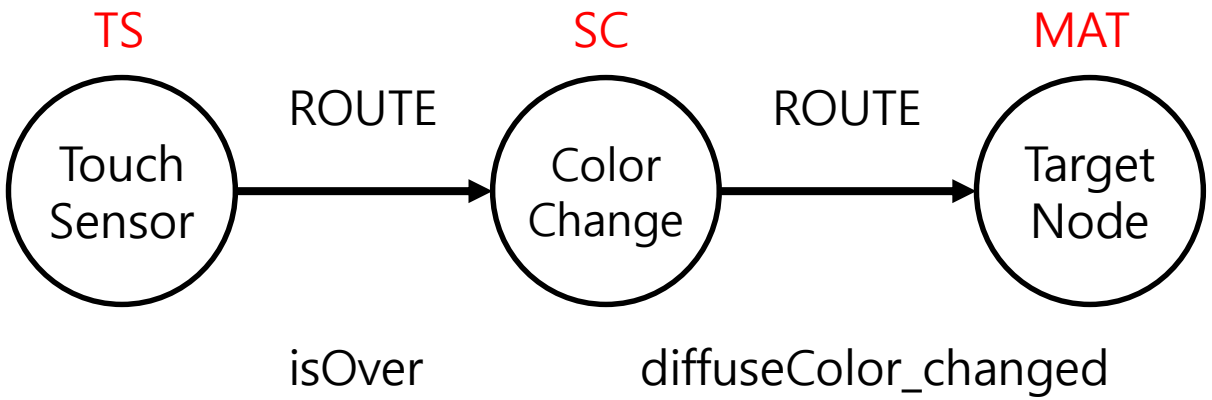


Example 1. TouchSensor isOver event

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE X3D PUBLIC "ISO//Web3D//DTD X3D 3.0//EN" "http://www.web3d.org/specifications/x3d-3.0.dtd">
<X3D profile="Immersive">
  <head>
    <meta content="TouchSensorIsOverEvent.x3d" name="filename"/>
    <meta content="Xeena VRML importer" name="translator"/>
    <meta content="23 February 2005" name="imported"/>
    <meta content="23 February 2005" name="revised"/>
    <meta
content="X3D-Edit, http://www.web3D.org/TaskGroups/x3d/translation/README.X3D-Edit.html" name="generator"/>
    <meta content="Vrml97ToX3dNist, http://ovrt.nist.gov/v2_x3d.html" name="generator"/>
  </head>
  <Scene>
    <Group>
      <Shape>
        <Appearance>
          <Material DEF="MAT" diffuseColor="0 0 1"/>
        </Appearance>
        <Box/>
      </Shape>
      <TouchSensor DEF="TS"/>
    </Group>
    <Script DEF="SC" url="&quot;SAIExample1.class&quot; ">
      <field accessType="inputOnly" name="isOver" type="SFBool"/>
      <field accessType="outputOnly" name="diffuseColor_changed" type="SFColor"/>
    </Script>
    <ROUTE fromField="isOver" fromNode="TS" toField="isOver" toNode="SC"/>
    <ROUTE fromField="diffuseColor_changed" fromNode="SC"
     toField="set_diffuseColor" toNode="MAT"/>
```

Example 1. TouchSensor isOver event (Implementation Overview)

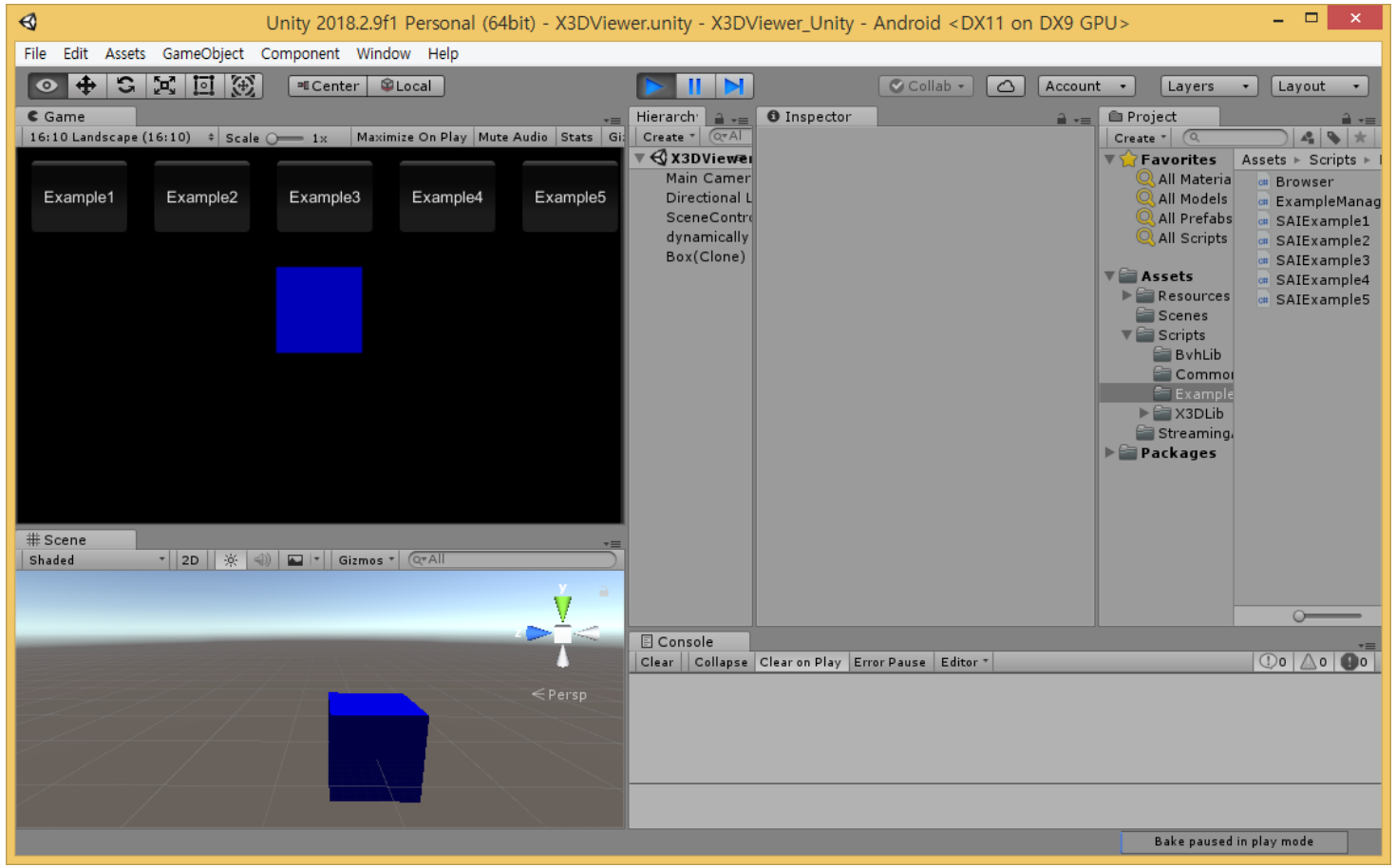
```
<Material DEF="MAT" diffuseColor="0 0 1"/>
<TouchSensor DEF="TS"/>
<Script DEF="SC" url="&quot;SAIExample1.class&quot; ">
    <field accessType="inputOnly" name="isOver" type="SFBool"/>
    <field accessType="outputOnly" name="diffuseColor_changed" type="SFColor"/>
</Script>
<ROUTE fromField="isOver" fromNode="TS" toField="isOver" toNode="SC"/>
<ROUTE fromField="diffuseColor_changed" fromNode="SC" toField="set_diffuseColor" tooNode="MAT"/>
```



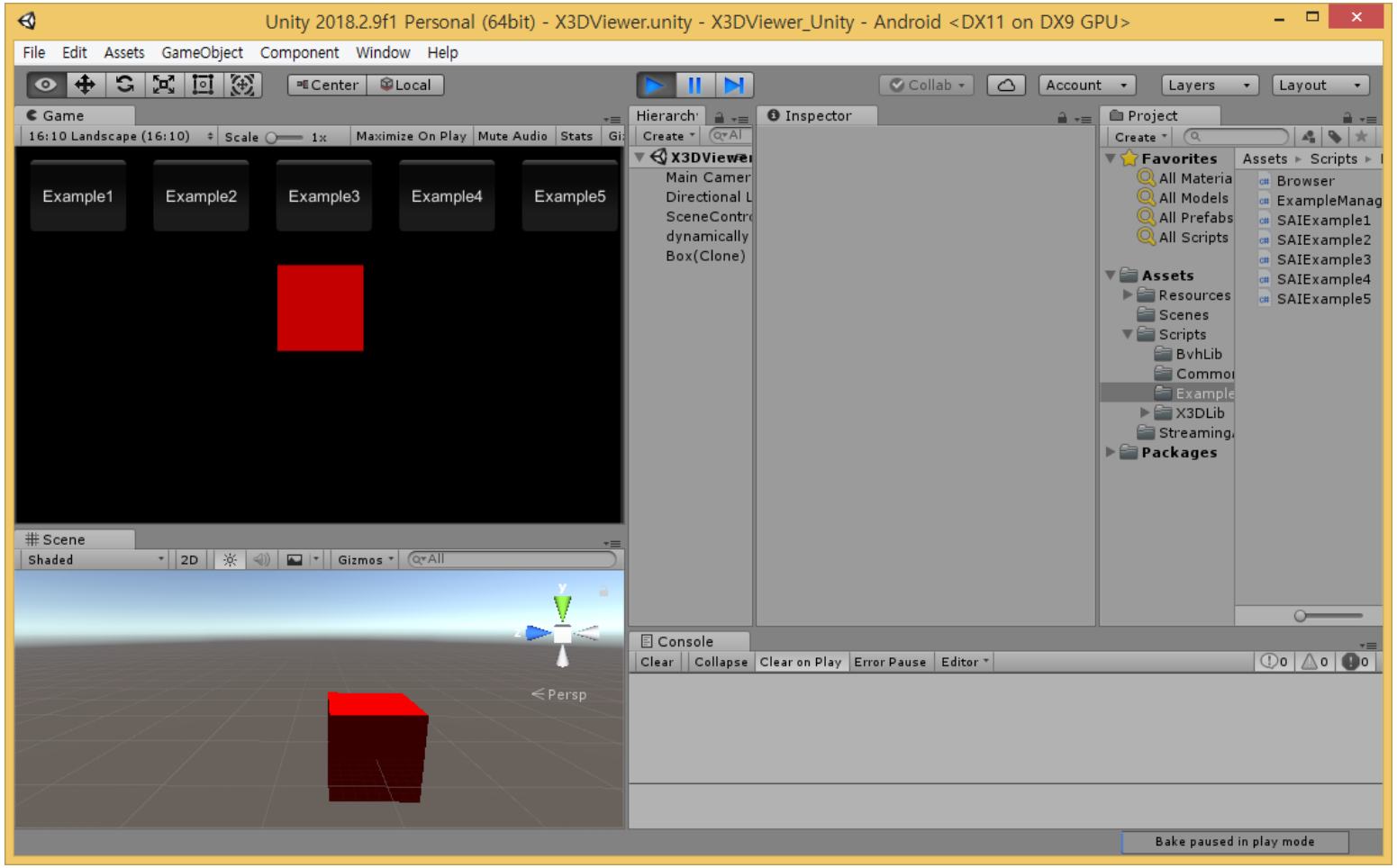
Example 1. TouchSensor isOver event (Implemtation) (1)

On click, an event is generated

- On click, the color of the box is changed from blue to red.
- On click in the outside of the box, the color of the box is changed to blue.



Example 1. TouchSensor isOver event (Implementation) (2)



Example 1. TouchSensor isOver event (Implementation) (3)

- Define ad create Map, Browser, SFBool classes
- Defined similarly as Java examples
- Unity 3D was used for rendering geometries such as Box

Example 1. TouchSensor isOver event (Implementation) (4)

```
using UnityEngine;
using System.Collections;
using System.Collections.Generic;

public class SAIExample1 : X3DScriptImplementation, X3DFieldEventListener
{
    /** Color Constant, RED */
    private static float[] RED = new float[] { 1.0f, 0, 0 };

    /** Color Constant, BLUE */
    private static float[] BLUE = new float[] { 0, 0, 1.0f };

    /** A mapping for fieldName(String) to an X3DField object */
    private Map fields;

    /** The isOver field */
    private SFBool isOver;

    /** The diffuseColor_changed field */
    private Color diffuseColor;

    private GameObject goBox;
```

Example 1. TouchSensor isOver event (Implementation) (5)

```
    //-------------------------------------------------------
        // Methods from the X3DScriptImplementation interface.
        //-------------------------------------------------------
        /**
         * Set the browser instance to be used by this script implementation.
         *
         * @param browser The browser reference to keep
         */
        public void setBrowser(Browser browser)
        {
        }

        /**
         * Set the listing of fields that have been declared in the file for
         * this node. .
         *
         * @param The external view of ourselves, so you can add routes to yourself
         *     using the standard API calls
         * @param fields The mapping of field names to instances
         */
        public void setFields(X3DScriptNode externalView, Map fields)
        {
            this.fields = fields;
        }
```

Example 1. TouchSensor isOver event (Implementation) (6)

```
    /**
     * Notification that the script has completed the setup and should go
     * about its own internal initialization.
     */
    public void initialize()
    {
        isOver = (SFBool)fields.get("isOver");
        diffuseColor = (Color)fields.get("diffuseColor_changed");

        // Listen to events on isOver
        isOver.addX3DEventListener(this);
    }

    /**
    * Notification that this script instance is no longer in use by the
    * scene graph and should now release all resources.
    */
    public void shutdown()
    {
    }
```

Example 1. TouchSensor isOver event (Implementation) (7)

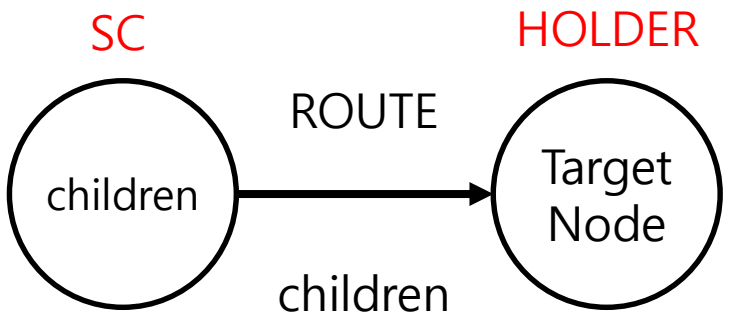
```
    //-----------------------------------------------------------
    // Methods from the X3DFieldEventListener interface.
    //-----------------------------------------------------------

    /**
     * Handle field changes.
     *
     * @param evt The field event
     */
    public void readableFieldChanged(X3DFieldEvent evt)
    {
        if (evt.getSource() == isOver)
        {
            if (isOver.getValue() == true)
                diffuseColor = new Color(RED[0], RED[1], RED[2]);
            else
                diffuseColor = new Color(BLUE[0], BLUE[1], BLUE[2]);
        }
        else
        {
            Debug.Log("Unhandled event: " + evt);
        }
    }
```

Example 2. Create Nodes

```
<?xml version="1.0" encoding="UTF-8"?>
<X3D profile="Immersive">
  <head>
    <meta content="CreateNodes.x3d" name="filename"/>
    <meta content="Xeena VRML importer" name="translator"/>
    <meta content="*enter date of initial version here*" name="created"/>
    <meta content="23 February 2005" name="imported"/>
    <meta content="23 February 2005" name="revised"/>
    <meta content="X3D-Edit, http://www.web3D.org/TaskGroups/x3d/translation/README.X3D-Edit.html" name="generator"/>
    <meta content="Vrml97ToX3dNist, http://ovrt.nist.gov/v2_x3d.html" name="generator"/>
  </head>
  <Scene>
    <Transform DEF="HOLDER" translation="-2.0 0.0 0.0"/>
    <Script DEF="SC" url="SAIExample.class">
      <field accessType="outputOnly" name="children" type="MFNode"/>
    </Script>
    <ROUTE fromField="children" fromNode="SC" toField="children" toNode="HOLDER"/>
  </Scene>
</X3D>
```

Example 2. Create Nodes (Implementation Overview)

```
<Transform DEF="HOLDER" translation="-2.0 0.0 0.0"/>
    <Script DEF="SC" url="SAIExample.class">
      <field accessType="outputOnly" name="children" type="MFNode"/>
    </Script>
    <ROUTE fromField="children" fromNode="SC"  toField="children"
    toNode="HOLDER"/>
```



Example 2. Create Nodes (Implementation) (1)

```
using UnityEngine;
using System.Collections;
using System.Collections.Generic;

public class SAIExample2 : X3DScriptImplementation
{

    /** Color Constant, RED */
    private static float[] RED = new float[] { 1.0f, 0, 0 };

    /** A mapping for fieldName(String) to an X3DField object */
    private Map fields;

    /** A reference to the browser */
    private Browser browser;

    /** The field to place the generated nodes via createX3DFromString */
    private MFNode children;
```

Example 2. Create Nodes (Implementation) (2)

```
    //-------------------------------------------------------
    // Methods from the X3DScriptImplementation interface.
    //-------------------------------------------------------
    /**
     * Set the browser instance to be used by this script implementation.
     *
     * @param browser The browser reference to keep
     */
    public void setBrowser(Browser browser)
    {
        this.browser = browser;
    }

    /**
     * Set the listing of fields that have been declared in the file for
     * this node. .
     *
     * @param The external view of ourselves, so you can add routes to yourself
     *    using the standard API calls
     * @param fields The mapping of field names to instances
     */
    public void setFields(X3DScriptNode externalView, Map fields)
    {
        this.fields = fields;
    }
```

Example 2. Create Nodes (Implementation) (3)

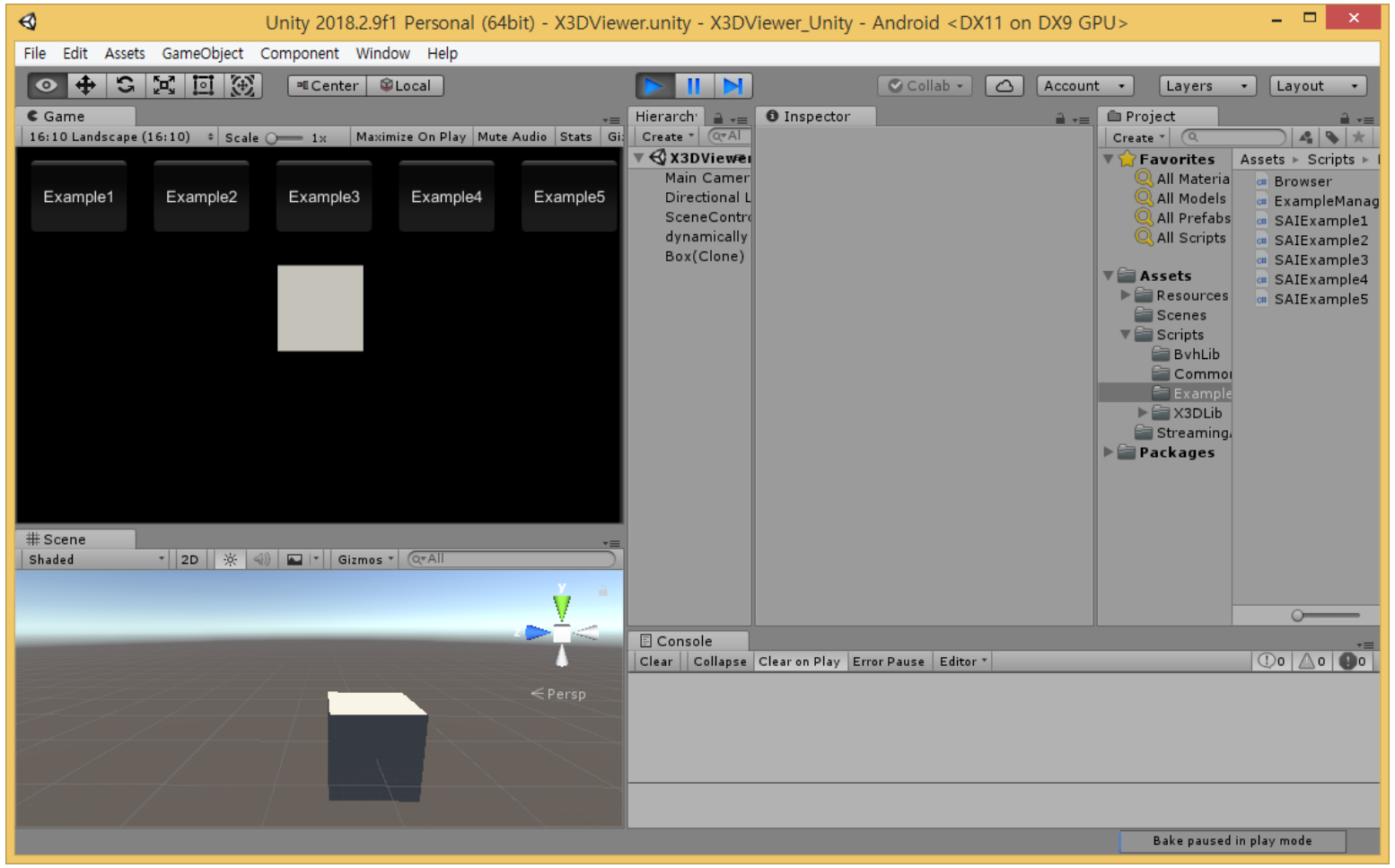
```
    /**
     * Notification that the script has completed the setup and should go
     * about its own internal initialization.
     */
    public void initialize()
    {
        children = (MFNode)fields.get("children");

        // Create nodes directly in the parent scene
        X3DScene scene = (X3DScene)browser.getExecutionContext();

        X3DShapeNode shape = (X3DShapeNode)scene.createNode("Shape");
        X3DGeometryNode box = (X3DGeometryNode)scene.createNode("Box");

        shape.setGeometry(box);
        scene.addRootNode(shape);



        // Create children using the createX3DFromString service
        string vrmlCmd = "PROFILE Interchange  Shape { geometry Sphere{} }";

        X3DScene tmpScene = browser.createX3DFromString(vrmlCmd);
        X3DNode[] nodes = tmpScene.getRootNodes();
```

Example 2. Create Nodes (Implementation) (4)

```
    /**
     * Notification that this script instance is no longer in use by the
     * scene graph and should now release all resources.
     */
    public void shutdown()
    {
    }

    /**
     * Notification that all the events in the current cascade have finished
     * processing.
     */
    public void eventsProcessed()
    {
    }
```

Example 2. Create Nodes (Implementation) (5)

프로그램 기동시 초기화 처리에서
Scene에 Shape>Box생성

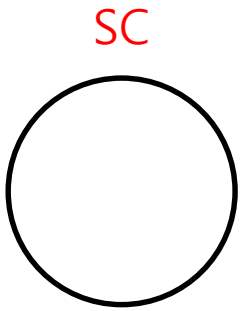


Example 3. Per frame notification

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE X3D PUBLIC "http://www.web3d.org/specifications/x3d-3.0.dtd"
                     "file:///www.web3d.org/TaskGroups/x3d/translation/x3d-3.0.dtd">
<X3D profile="Immersive">
  <head>
    <meta content="PerFrameNotification.x3d" name="filename"/>
    <meta content="Xeena VRML importer" name="translator"/>
    <meta content="23 February 2005" name="imported"/>
    <meta content="23 February 2005" name="revised"/>
    <meta
content="X3D-Edit, http://www.web3D.org/TaskGroups/x3d/translation/README.X3D-Edit.html" name="generator"/>
    <meta content="Vrml97ToX3dNist, http://ovrt.nist.gov/v2_x3d.html" name="generator"/>
  </head>
  <Scene>
    <Script DEF="SC" url="&quot;SAIExample3.class&quot; "/>
  </Scene>
</X3D>
```

Example 3. Per frame notification (Implementation Overview)

<Script DEF="SC" url=""SAIExample3.class" "/>



Initialize() : Initialize Time value
prepareEvents() : FPS is calculated and displayed

Example 3. Per frame notification (Implementation) (1)

```
using UnityEngine;
using System.Collections;
using System.Collections.Generic;

public class SAIExample3 : X3DPerFrameObserverScript
{
    /** When did the last frame start */
    private float lastStartTime;

    //----------------------------------------------------------
    // Methods from the X3DScriptImplementation interface.
    //----------------------------------------------------------
    /**
     * Set the browser instance to be used by this script implementation.
     *
     * @param browser The browser reference to keep
     */
    public void setBrowser(Browser browser)
    {
    }
```

Example 3. Per frame notification (Implementation) (2)

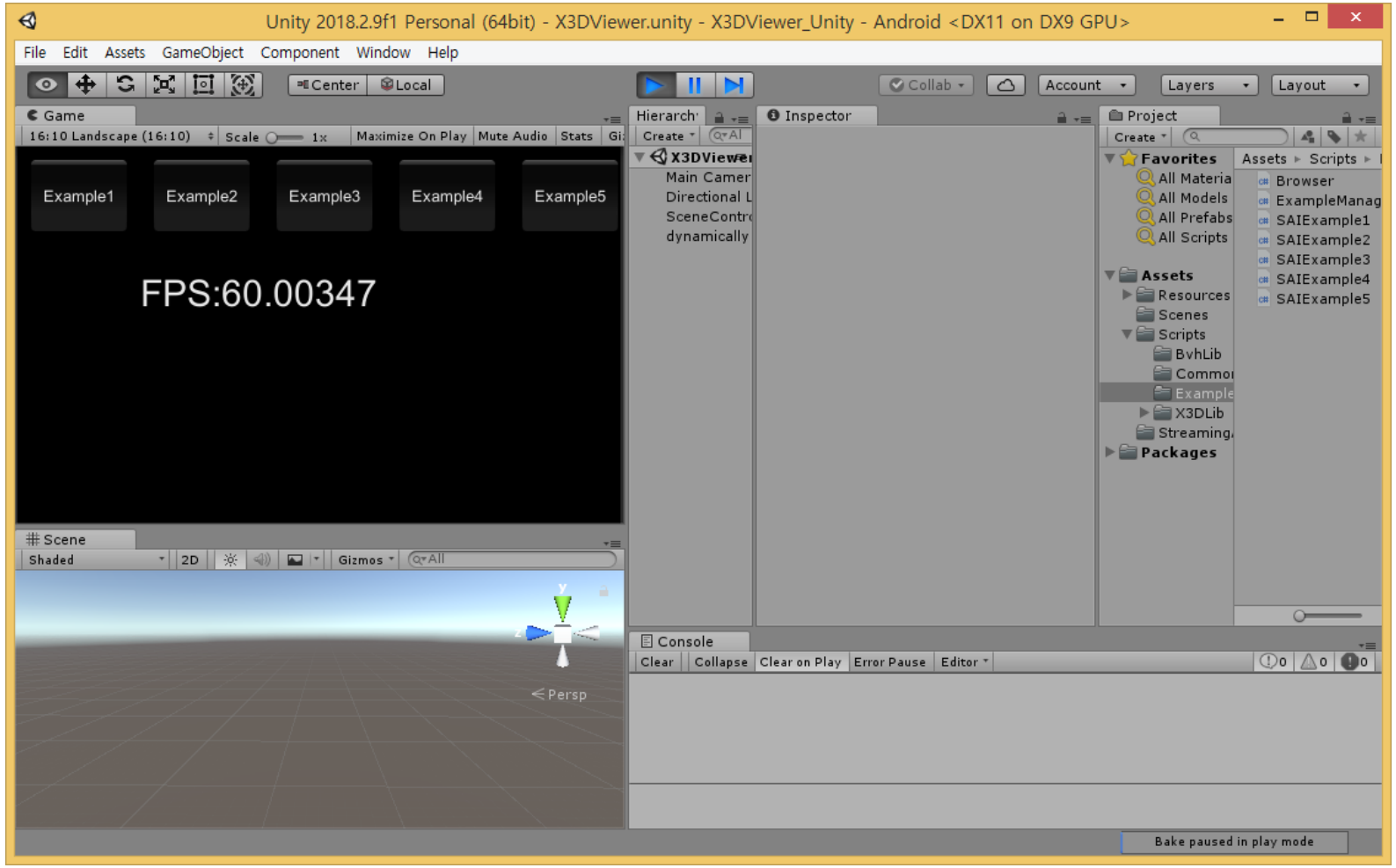
```
    /**
     * Set the listing of fields that have been declared in the file for
     * this node. .
     *
     * @param The external view of ourselves, so you can add routes to yourself
     *     using the standard API calls
     * @param fields The mapping of field names to instances
     */
    public void setFields(X3DScriptNode externalView, Map fields)
    {
    }

    /**
     * Notification that the script has completed the setup and should go
     * about its own internal initialization.
     */
    public void initialize()
    {
        lastStartTime = Time.deltaTime;
    }
```

Example 3. Per frame notification (Implementation) (3)

```
    //-------------------------------------------------------
    // Methods from the X3DPerFrameObserver interface.
    //-------------------------------------------------------

    /**
     * Start of frame notification.
     */
    public void prepareEvents()
    {
        float frameTime = Time.deltaTime;
        float fps = 1.0f / frameTime;

        GUI.skin.label.fontSize = 30;
        GUI.Label(new Rect(100f, 100f, 1000f, 1000f), "FPS:" + fps);
    }
```

Example 3. Per frame notification (Implementation Results)

prepareEvents()
FPS is calculated and displayed

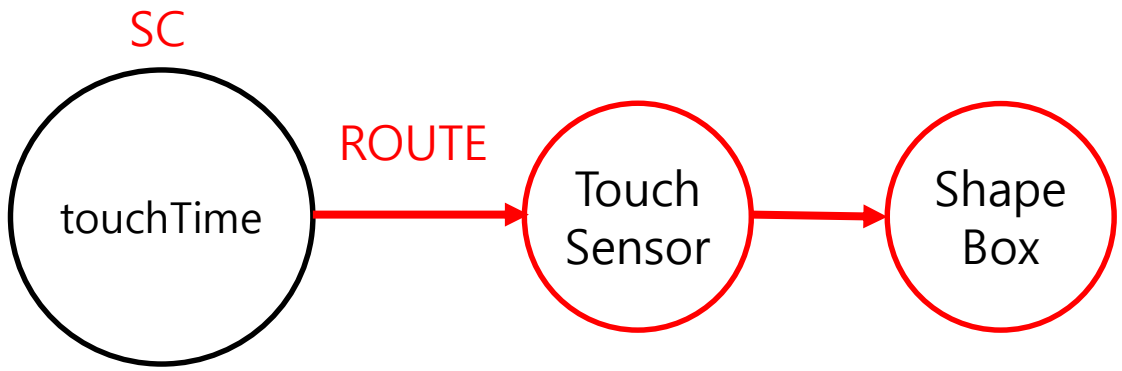


Example 4. Add dynamic routes

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE X3D PUBLIC "ISO//Web3D//DTD X3D 3.0//EN" "http://www.web3d.org/specifications/x3d-3.0.dtd">
<X3D profile="Immersive">
  <Scene>
    <Script DEF="SC" url="SAIExample4.class">
      <field accessType="inputOnly" name="touchTime" type="SFTime"/>
    </Script>
  </Scene>
</X3D>
```

Example 4. Add dynamic routes (Implementation Overview)

```
<Script DEF="SC" url="SAIExample4.class">
      <field accessType="inputOnly" name="touchTime" type="SFTime"/>
</Script>
```



Initialize() :
  Read “touchTime” field.
  Create Shape(Box), TouchSensor under Group in the scene
  Create “touchTime” ROUTE
  Connect TouchSensor to the scene
readableFieldChanged()
  When TouchSensor is generated, “Poke!” is displayed.

Example 4. Add dynamic routes (Implementation) (1)

```
using UnityEngine;
using System.Collections;
using System.Collections.Generic;

public class SAIExample4 : X3DScriptImplementation, X3DFieldEventListener
{
    /** A mapping for fieldName(String) to an X3DField object */
    private Map fields;

    /** A reference to the browser */
    private Browser browser = new Browser();

    /** inputOnly touchTime */
    private SFTime touchTime;

    /** initializeOnly selfRef */
    private X3DScriptNode selfRef;

    private GameObject goBox;
```

Example 4. Add dynamic routes (Implementation) (2)

```
    //-------------------------------------------------------
        // Methods from the X3DScriptImplementation interface.
        //-------------------------------------------------------
        /**
         * Set the browser instance to be used by this script implementation.
         *
         * @param browser The browser reference to keep
         */
        public void setBrowser(Browser browser)
        {
            this.browser = browser;
        }

        /**
         * Set the listing of fields that have been declared in the file for
         * this node. .
         *
         * @param The external view of ourselves, so you can add routes to yourself
         *     using the standard API calls
         * @param fields The mapping of field names to instances
         */
        public void setFields(X3DScriptNode externalView, Map fields)
        {
            this.fields = fields;
            selfRef = externalView;

```

Example 4. Add dynamic routes (Implementation) (3)

```
    /**
     * Notification that the script has completed the setup and should go
     * about its own internal initialization.
     */
    public void initialize()
    {
        touchTime = (SFTime)fields.get("touchTime");

        // Listen to events on touchTime
        touchTime.addX3DEventListener(this);

        // Create nodes directly in the parent scene
        X3DScene scene = (X3DScene)browser.getExecutionContext();

        X3DShapeNode shape = (X3DShapeNode)scene.createNode("Shape");
        X3DGeometryNode box = (X3DGeometryNode)scene.createNode("Box");
        X3DNode touchSensor = scene.createNode("TouchSensor");

        shape.setGeometry(box);

        // Create a Group to hold the nodes
        X3DGroupingNode group = (X3DGroupingNode)scene.createNode("Group");
```

Example 4. Add dynamic routes (Implementation) (4)

```
    //-------------------------------------------------------
    // Methods from the X3DFieldEventListener interface.
    //-------------------------------------------------------

    /**
     * Handle field changes.
     *
     * @param evt The field event
     */
    public void readableFieldChanged(X3DFieldEvent evt)
    {
        if (evt.getSource() == touchTime)
        {
            GUI.skin.label.fontSize = 30;
            GUI.Label(new Rect(100f, 100f, 1000f, 1000f), "Poke!");
        }
        else
        {
            Debug.Log("Unhandled event: " + evt);
        }
    }
```

Create nodes from a prototype

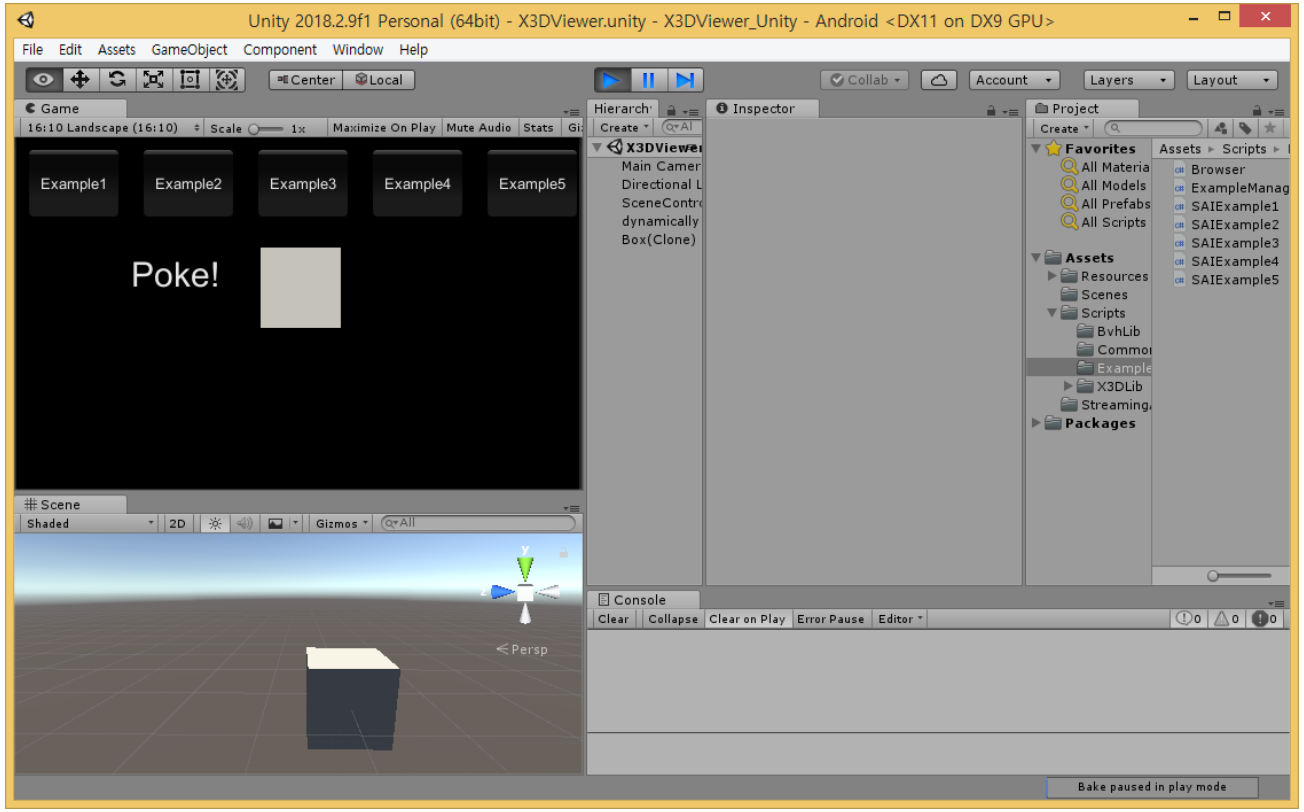
```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE X3D PUBLIC "ISO//Web3D//DTD X3D 3.0//EN" "http://www.web3d.org/specifications/x3d-3.0.dtd">
<X3D profile="Immersive">
  <Scene>
    <ProtoDeclare name="ColoredSphere">
      <ProtoInterface>
        <field accessType="initializeOnly" name="color" type="SFColor" value="0.0 0.0 0.0"/>
      </ProtoInterface>
      <ProtoBody>
        <Shape>
          <Appearance>
            <Material>
              <IS>
                <connect nodeField="diffuseColor" protoField="color"/>
              </IS>
            </Material>
          </Appearance>
          <Sphere/>
        </Shape>
      </ProtoBody>
    </ProtoDeclare>
    <Script DEF="SC" url="SAIExample5.class"/>
  </Scene>
</X3D>
```

Example 4. Add dynamic routes (Implementation Results)

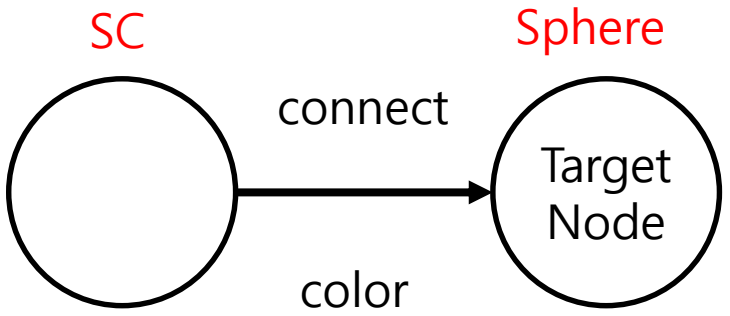
In the initialization

Create Scene > Group > Shape > Box > TouchSensor

On clicking Box, “Poke!” is displayed



Example 5. Create nodes from a prototype (Implementation Overview)



Initialize() :
Read “color” field under ProtoDeclare and ProtoInterface
Set RED value to diffuseColor of Material

Example 5. Create nodes from a prototype (Implementatio) (1)

```
using UnityEngine;
using System.Collections;
using System.Collections.Generic;

public class SAIExample5 : X3DScriptImplementation
{
    /** Color Constant, RED */
    private static float[] RED = new float[] { 1.0f, 0, 0 };

    /** A mapping for fieldName(String) to an X3DField object */
    private Map fields;

    /** A reference to the browser */
    private Browser browser;

    /** The field to place the generated nodes via createX3DFromString */
    private MFNode children;
```

Example 5. Create nodes from a prototype (Implementation) (2)

```
    //-------------------------------------------------------
        // Methods from the X3DScriptImplementation interface.
        //-------------------------------------------------------
        /**
         * Set the browser instance to be used by this script implementation.
         *
         * @param browser The browser reference to keep
         */
        public void setBrowser(Browser browser)
        {
            this.browser = browser;
        }

        /**
         * Set the listing of fields that have been declared in the file for
         * this node. .
         *
         * @param The external view of ourselves, so you can add routes to yourself
         *     using the standard API calls
         * @param fields The mapping of field names to instances
         */
        public void setFields(X3DScriptNode externalView, Map fields)
        {
            this.fields = fields;
        }
```

Example 5. Create nodes from a prototype (Implementation) (3)

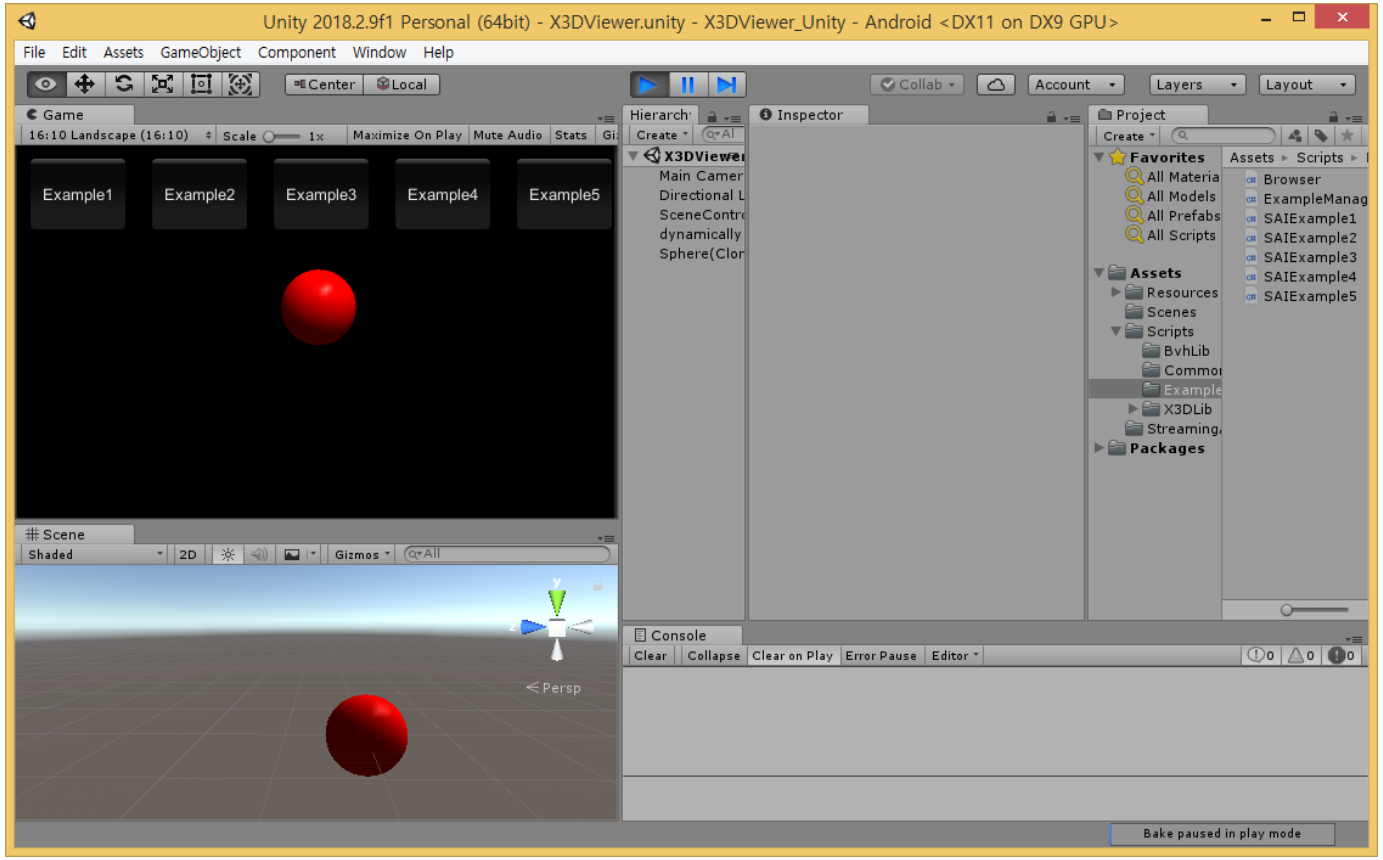
```
    /**
     * Notification that the script has completed the setup and should go
     * about its own internal initialization.
     */
    public void initialize()
    {
        // Create nodes directly in the parent scene
        X3DScene scene = (X3DScene)browser.getExecutionContext();

        // Create protoInstance nodes

        // Get the proto declaration declared in the main scene
        X3DProtoDeclaration protoDecl = scene.getProtoDeclaration("ColoredSphere");

        // Create a new instance of this proto
        X3DProtoInstance instance = protoDecl.createInstance();

        // Get the color field and set it to red
        SFColor color = (SFColor)instance.getField("color");
        color.setValue(RED);

        // Add the created proto instance to the scene
        scene.addRootNode(instance);
    }
```

Example 5. Create nodes from a prototype (Implementation) (4)

```
    /**
     * Notification that this script instance is no longer in use by the
     * scene graph and should now release all resources.
     */
    public void shutdown()
    {
    }

    /**
     * Notification that all the events in the current cascade have finished
     * processing.
     */
    public void eventsProcessed()
    {
    }
```

Example 5. Create nodes from a prototype (Implementation Results)

In the initialization, read a ColoredSphere in the scene.
Read “color” field.
Set RED to “color”



Work in Progress

- 19777-3, 19777-4, 19777-5 NWIPs submission (2018.7)
- 19777-5 NWIP submission (2018.1), but non pass for insufficient participation ➔ re-voting
- Implementation of C, C++ and C# language bindings
  - 19777-3 X3D scene access interface definition using C
    - Visual C++ and OpenGL
  - 19777-4 X3D scene access interface definition using C++
    - Visual C++ and OpenGL
  - 19777-5 X3D scene access interface definition using C#
    - Unity 3D and C#
- Developing X3D Binding viewer programs with C, C++ and C# binding capability