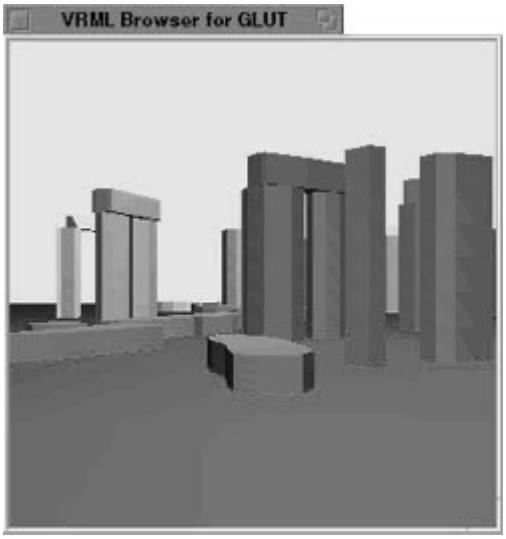



# CyberVRML97 for C++

# User's Guide

Table of Contents

# 1.Introduction

CyberVRML97 for C++ is a development library for building VRML97/2.0 applications quickly. Using the library, you can read and write the VRML files, set and get the scene graph information, draw the geometries, run the behaviors easily.

# 2.Setup

The CyberVRML97 library is consist of some C++ program source files and two parser syntax files. To use the library, you have to create two parser source files from the parser syntax, and you make the library using the source files or add the source files into your project.

## System Requirements

To use the CyberVRML97 library on your platform, you have to prepare the following software on your platform.

- C++ compiler
- Parser generators (lex and yacc)

If your C++ compiler supports STL package, you can get the more performance to load VRML files, and access to the scene graph with a compiler option.

## Building Parser Files

The CyberVRML97 has the following two parser syntax files, and you have to create the parser source files using lex/yacc generators.

| File Name | Detail |
|---|---|
| vrml.l | a lex syntax file of the CyberVRML97 |
| vrml.y | a yacc syntax file of the CyberVRML97 |

In the following example, the parser source files, “vrml.tab.h” and “vrml.yy.cpp” are created using GNU parser generators, Flex and Bison.

```
bison -ydv  vrml.y
mv y.tab.c vrml.tab.cpp
mv y.tab.h vrml.tab.h
flex -I vrml.l
mv lex.yy.c vrml.yy.cpp
```

## Compiler Flags

If you create the library with no compiler flags, you can only load VRML files, edit the scene graph information, and output the scene graph into a VRML file.

The CyberVRML97 has some the following compiler options. Depending on your purpose, set the flags into your compiler.

| Compiler Flag | Purpose | Requirements |
|---|---|---|
| SUPPORT_OPENGL | Drawing geometry nodes using OpenGL | OpenGL 1.x library |
| SUPPORT_GLUT | Drawing geometry nodes using GLUT with OpenGL | GLUT 3.x library |
| SUPPORT_JPEG | Loading JPEG image files in ImageTexture node | Independent JPEG Group's library |
| SUPPORT_PNG | Loading JPEG image files in ImageTexture node | libpng / zlib |
| SUPPORT_JSAI | Executing behaviors in Script nodes. | Sun JRE 1.1.x |

| | | CyberVRML97 for Java 1.0.2a |
|---|---|---|
| SUPPORT_OLDCPP | Supports some old c++ compilers. | None |
| SUPPORT_STL | Get good performance | STL package |

# Scene Graph

In the CyberVRML97, the scene graph is correction and hierarchical arrangement of VRML nodes.

There are two ways to build a scene graph dynamically, you can load it from a VRML file, or you can create new VRML node and add the node into the scene graph.

## Loading Scene Graph

Use SceneGraph::load() to load a scene graph from a VRML file. The load() clears all VRML nodes in the current scene graph.

```
SceneGraph *sceneGraph = new SceneGraph();
sceneGraph->load("world.wrl");
```

If your scene graph has some VRML nodes and you want to add a new scene graph from a VRML file into the current scene graph, use SceneGraph::add().

```
SceneGraph *sceneGraph = new SceneGraph();
        ..........
sceneGraph->add("world.wrl");
```

If ScneGraph::load() or SceneGraph::add() can't read your VRML file normally, the methods return false. Use the following methods to know the error in more detail.

```
int SceneGraph::getParserErrorLineNumber(void);
char *SceneGraph::getParserErrorMessage(void);
char *SceneGraph::getParserErrorToken(void);
char *SceneGraph::getParserErrorLineString(void);
```

The following example shows the parser error when the loading is failed.

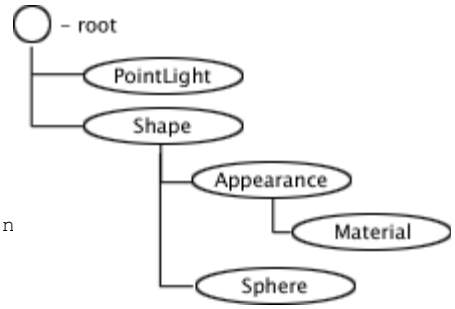
```
SceneGraph *sceneGraph = new SceneGraph();
bool result = sceneGraph->load("world.wrl");
if (result == false) {
    printf("$s (%d) : %s",
        sceneGraph.getParserErrorMessage(),
        sceneGraph.getParserErrorLineNumber(),
        sceneGraph.getParserErrorToken());
}
```

## Building Scene Graph

The CyberVRML97 has all C++ classes of VRML nodes, you can add and remove the all nodes dynamically. Use SceneGraph::addNode() to add a new node as a root node of the scene graph. Use Node::addChildNode() to add a new node as a child node of the other node.

The following example adds a PointLight and Shape node that has an appearance and a geometry node into an empty scene graph.

```
SceneGraph sg;
 // Add a PointLight node
 PointLightNode *plight = new PointLightNode();
  // Add a Shape node as a root node
 ShapeNode *shape = new ShapeNode();
 sg.addNode(shape);
 // Add an Appearance node as a child node of the Shape n
 Appearancenode *app = new AppearanceNode();
shape->addChildNode(app);
 // Add a Material node as a child node of the Appearance node
 MaterialNode *mat = new MaterialNode();
mat->setDiffuseColor(1.0f, 0.0f, 0.0f); // Red
app->addChildNode(mat);
// Add a Sphere node as a child node of the Shape node
SphereNode *sphere = new SphereNode();
 shape->addChildNode(sphere);
sphere->setRadius(10.0f);
```




Use Node::remove() to remove a node from the current scene graph. The following example uses Node::remove() to remove a PointLight node from a loaded scene graph.

```
SceneGraph sg;
sg.load("world.wrl");
// Remove first PointLight node
PointLightNode *plight = sg.findPointLightNode();
if (plight != NULL)
    plight->remove();
```

## Scene Graph Output

Use SceneGraph::save() to save a current scene graph into a VRML file.

```
sceneGraph->save("newworld.wrl");
```

Use SceneGraph::print() to output a current scene graph into a default console.

```
sceneGraph->print();
```

## Scene Graph Traversal

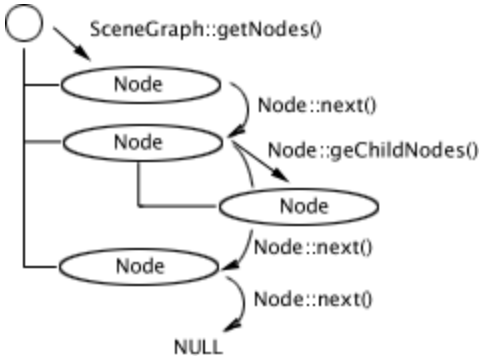
The scene graph has the VRML information as a collection of Node class instances. The Node class is a super class of all VRML node classes of CyberVRML97. There are two ways to access the nodes.

The first way is to use SceneGraph::getNodes() with Node::next() and Node::getChildNodes(). For example, if you want to get all viewpoint nodes in a scene graph …..

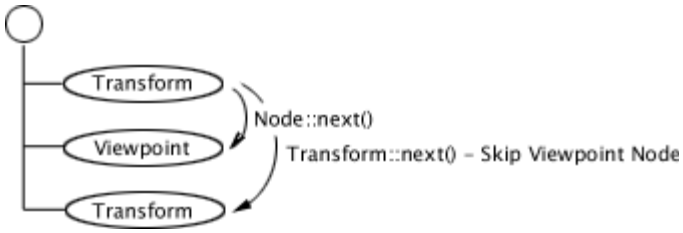
```
void GetViewpointInfomation(Node *node)
{
     if (node->isViewpointNode()) {
         ViewpointNode *view = (Viewpoint *)node;
         // Get a viewpoint information
         ..........
     }
     for (Node *cnode=node->getChildNodes(); cnode; cnode=cnode->next())
         GetViewpointInfomation(cnode);
}

void main()
{
     ..........
     SceneGraph *sg = new SceneGraph();
     sceneGraph->load("world.wrl");
     for (Node *node=sg->getNodes(); node; node=node->next())
         GetViewpointInfomation(node);
     ..........
}
```

SceneGraph::getNodes() returns a first node that are added into the scene graph root. Node::next() returns a next node in the same hierarchy, Node::getChildNodes() returns a first child node that are added into the parent node. The methods returns NULL if the next node does not exist.


SceneGraph::getNodes()
Node
Node::next()
Node
Node::geChildNodes()
Node
Node::next()
Node
Node::next()
NULL

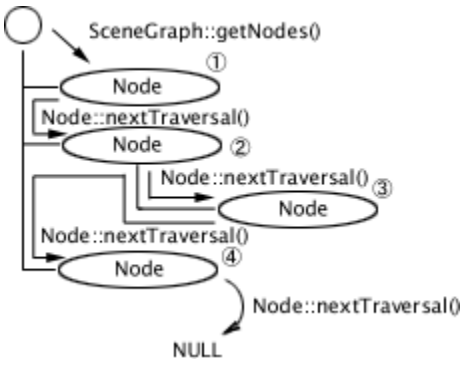
Node::next() is overridden in all sub classes, TransformNode class etc. The overridden method returns a next same class node. For example, Transform::next() returns a next Transform node.




The other way is to use SceneGraph::getNodes()with Node::nextTraversal(). The way is handier than the first one. For example, if you want to get all viewpoint nodes in the scene graph .....

```
void main()
{
    ..........
    SceneGraph *sg = new SceneGraph();
    sg->load("world.wrl");
    for (Node *node=sg->getNodes(); node; node=node->nextTraversal())
        if (node->isViewpoint()) {
            ViewpointNode *view = (ViewpointNode *)node;
            // Get a viewpoint information
            ..........
        }
    }
    ..........
}
```

Node::nextTraversal() is similar to Node::next(), but Node::nextTraversal tries to get a next node from the parent node when the next node does not exist. This Node::nextTraversal() is overridden in the sub classes to return a next same class node, too.




If you want to get only same class nodes, use SceneGraph::find<nodetype>Node() instead of SceneGraph::getNodes() with the <nodetype>::nextTraversal() that returns a next same class node. For example, if you want to get only all viewpoint nodes in the scene graph .....

```
SceneGraph *sg = new SceneGraph();
sg->load("world.wrl");
for (ViewpointNode *view=sceneGraph->findViewpointNode(); view; view=view->nextTraversal()) {
```

```
        // Get a viewpoint information
        ..........
    }
```

## Finding Node

Use SceneGraph::findNode() to find a named node by DEF keyword or Node::setName(). The following example loads a VRML file, “world.wrl”, and gets a node that is named as “MountFuji”.

```
SceneGraph sg;
sg.load(“world.wrl”)
Node *node = sg.findNode(“MountFuji”);
```

Use SceneGrahp::get<nodetype>Nodes() to get a specified first node from the root hierarchy. For example, you want to get a viewpoint that is a first viewpoint node in the scene graph root, use SceneGraph::getViewpointNodes();

```
SceneGraph sg;
sg.load(“world.wrl”)
ViewpointNode *defaultView = sg.getViewpointNodes();
```

# Node

The CyberVRML97 node name is identical to the VRML node name. For example, you would use the BoxNode class if you want to use the Box node.

## Creating Node

The node class has the default constructor only. Use the default constructor to create the new node, and set the field property. The following example creates a sphere and set the property.

```
SphereNode *spNode = new SphereNode();
spNode->setRadius(10.0);
```

## Node Type

If you want to know the node type, use Node::getType() or Node::is<nodetype>Node();

Node::getType() returns the node type as a string, Node::is<nodetype>Node() returns true when the node is the specified type. For example, you want to know whether a node is ViewpointNode.

```
SceneGraph *sg = new SceneGraph();
sg->load("world.wrl");
for (Node *node=sg->getNodes(); node; node=node->nextTraversal())
    char *nodeType = node->getType();
    if (node->isViewpoint() || strcmp(nodeType, "Viewpoint") == 0)
        printf("This node is ViewpointNode !!");
}
```

## Node Name

If you load a VRML file that has some named nodes by DEF keyword, you can get the name using Node::getName(); Use Node::setName() to name a node.

```
SceneGraph sg;
sg.load("world.wrl")
ShapeNode *mtNode = sg.getShapeNodes();
mtNode->setName("MtFuji");
```

The named nodes are output using DEF keyword when you save the scene graph.

```
#VRML V2.0 utf8
.........
DEF MtFuji Shape {
    .........
}
.........
```

## Accessing Fields

The node class has set<fieldname>() and get<fieldname>() to access to the VRML fields, and has getN<fieldname>s() if the field is multi field, MFString etc.. For example, the AnchorNode class has the following methods.

```
class AnchorNode {
        void      setDescription(char *value);
        char *     getDescription();
        void      addParameter(char *value);
        int        getNParameters();
        char *    getParameter(int index);
        void      addUrl(char *value);
        int       getNUrls();
        char *    getUrl(int index);
        void      setUrl(int index, char *urlString);
};
```

The Node classes has only basic field access methods. Use get<fieldname>Field() to access the field in more detail. The method returns the field class itself. You can operate the field in more detail to the field class. The following example gets a color field of a Color Node in an IndexedFaceSet node, and changes all the colors to red.

```
IndexedFaceSetNode *idxNode = ……
ColorNode *colNode = idxNode->getColorNodes();
MFColor *colField = colNode->getColorField();
int colCnt = colField->size();
for (int n=0; n<colCnt; n++)
     colField->set1Value(n, 0xff, 0x00, 0x00) // Red
```

## Adding to Scene Graph

Use SceneGraph::addNode() to add the node as a root node of the scene graph. The following example creates a transform node add the node to the scene graph root.

```
SceneGraph sg;
TransformNode *transNode = new TransformNode();
sg.addNode(transNode);
```

## Adding Child Node

Use Node::addChildNode() to add the node as a child node of the other node. The following example creates a shape and a box, add the shape to the scene graph root, and add the box to the shape.

```
SceneGraph sg;
ShapeNode *shapeNode = new ShapeNode();
BoxNode *boxNode = new BoxNode();
sg.addNode(shapeNode);
shapeNode.addChileNode(boxNode);
```

## Getting Child Node

There are two ways to get child nodes. The first way is to use Node::getNChildNodes() that returns a count of child nodes, and Node::getChileNode() that returns a selected child node. The following example shows the name and the type of all child nodes.

```
void PrintChildNodes(Node *node)
{
    int childNodeCnt = node->getNChildNodes();
    for (int n=0; n<childNodeCnt; n++) {
        Node *childNode = node->getChildNode(n);
        char *nodeType = childNode->getType();
        char *nodeName = childNode->getName();
        printf("[%d] = %s, %s", n, nodeType, nodeName);
    }
}
```

The other way is to use Node::getChildNodes() that returns a first child node with Node::next() returns a next child node in the same parent node. The Node::getChildNodes() and the Node::next() returns NULL if the node does not exist. The way is handier and faster than the first one. The following example shows the name and the type of all child nodes.

```
void PrintChildNodes(Node *node)
{
    Node *childNode = node->getChildNodes();
    while (childNode!= NULL) {
        char *nodeType = childNode->getType();
        char *nodeName = childNode->getName();
        printf("[%d] = %s, %s", n, nodeType, nodeName);
        childNode = childNode->next();
    }
}
```

## Removing from SceneGraph or Parent Node.

Use Node::remove() to remove a node from the scene graph root or the parent node. The following example removes a point light from the scene graph.

```
SceneGraph sg;
    ..........
PointLightNode *plight = sg.findPointLightNode();
if (plight != NULL)
    plight->remove();
```

To remove and delete a node, you should call the destructor simply because the destructor of Node classes removes the node from the scene graph root or the parent node if the node is added before the node is deleted.

```
SceneGraph sg;
    ..........
PointLightNode *plight = sg.findPointLightNode();
if (plight != NULL)
    delete plight;
```

## Instance Node

You can create an instance of a node like USE keyword of VRML97. Use Node::createInstanceNode() to create the instance node.

```
SceneGraph sg;
ShapeNode *shape = new ShapeNode();
shape->setName("BOX");
BoxNode *box = new BoxNode();
box->setSize(10.0, 20.0, 30.0);
sg.addNode(shape);
shape->addChildNode(box);
Node *shapeInstance = shape->createInstanceNode();
sg.addNode(shapeInstance);
```

The scene graph is output as the following when you save the scene graph.

```
DEF BOX Shape {
    geometry Box {
        size 10 20 30
    }
}
USE BOX
```

## Field

The node has several fields and the field is a property or attribute of a node. The field is identical to the VRML field name. For example, you would use the SFBool class if you want to use the SFBool field.

Use get<fieldname>Field() to get the field. The following example gets a color field of a Color Node in an IndexedFaceSet node, and changes all the colors to red.

```
IndexedFaceSetNode *idxNode = ……
 ColorNode *colNode = idxNode->getColorNodes();
MFColor *colField = colNode->getColorField();
int colCnt = colField->size();
for (int n=0; n<colCnt; n++)
     colField->set1Value(n, 0xff, 0x00, 0x00) // Red
```

# OpenGL

The CyberVRML97 supports the geometry rendering using OpenGL. All geometry classes are sub class of the GeometryNode class, and has a rendering methods, GeometryNode::draw(), if you compile the library with the SUPPORT_OPENGL option;

The geometry node has the display list of OpenGL, use ScenegGraph::initialize() or Node::initialize() to create the display list. Use GeometryNode::getDisplayList() to get the display list number. Because the display list has only the vertices information, you should set the transform and the appearance information before the geometry drawing.

The following function sets only the material and transform information for the geometry drawing. See the other samples, VRML browser using the CyberVRML97 etc., about the implementation in more detail.

```
void DrawShapeNode(ShapeNode *shape)
{
    glPushMatrix ();

    float color[4];
    color[3] = 1.0f;

    AppearanceNode *appearance = shape->getAppearanceNodes();
    MaterialNode           *material = NULL;

    if (appearance) {
        material = appearance->getMaterialNodes();
      if (material) {
            float ambientIntesity = material->getAmbientIntensity();
            material->getDiffuseColor(color);
            glMaterialfv(GL_FRONT_AND_BACK, GL_DIFFUSE, color);
            material->getSpecularColor(color);
            glMaterialfv(GL_FRONT_AND_BACK, GL_SPECULAR, color);
            material->getEmissiveColor(color);
            glMaterialfv(GL_FRONT_AND_BACK, GL_EMISSION, color);
            material->getDiffuseColor(color);
            color[0] *= ambientIntesity;
            color[1] *= ambientIntesity;
            color[2] *= ambientIntesity;
            glMaterialfv(GL_FRONT_AND_BACK, GL_AMBIENT, color);
            glMateriali(GL_FRONT, GL_SHININESS, (int)(material->getShininess()*128.0));
        }
    }

    float m4[4][4];
    shape->getTransformMatrix(m4);
    glMultMatrixf((float *)m4);

    GeometryNode *gnode = shape->getGeometry();
    if (gnode) {
        if (0 < gnode->getDisplayList())
```

```
            gnode->draw();
        }

        glPopMatrix();
    }
```

# Other

## PROTO

Only the SF<type> fields are supported, and the MF<type> fields are ignored. EXTERNPROTO is not supported.

## Script

Java of VRML 2.0 specification (JSAI) is supported as the script language. In current version, the processEvents() method and the eventsProcessed() method are not supported.

# License

CyberVRML97 for C++ is provided "AS IS". Licenser disclaims all warranties, including but not limited to, all express or implied warranties of merchant ability and fitness for a particular purpose.

Everyone can use the CyberVRML97 for commerce or personal purposes free. However, If you want to distribute your software using the CyberVRML97, you have to add state that "Portions of this software is based in part on the CyberVRML97 package written by Satoshi Konno" into the program or document.

# SceneGraph Class Reference

## SceneGraph

```
class SceneGraph {

        SceneGraph();
        ~SceneGraph();

        ////////////////////////////////////////////////
        // Load
        ////////////////////////////////////////////////

        void load(char *filename)
        void add(char *filename)
        void clear();

        int getParserErrorLineNumber(void);
        char *getParserErrorMessage(void);
        char *getParserErrorToken(void);
        char *getParserErrorLineString(void);

        ////////////////////////////////////////////////
        // Output node infomations
        ////////////////////////////////////////////////

        bool save(char *filename);
        void print();

        ////////////////////////////////////////////////
        // Child Node
        ////////////////////////////////////////////////
```

```
int getNNodes();
Node *getNodes();

GroupingNode *getGroupingNodes();
AnchorNode *getAnchorNodes();
AppearanceNode *getAppearanceNodes();
AudioClipNode *getAudioClipNodes();
BackgroundNode *getBackgroundNodes();
BillboardNode *getBillboardNodes();
BoxNode *getBoxeNodes();
CollisionNode *getCollisionNodes();
ColorNode *getColorNodes();
ColorInterpolatorNode *getColorInterpolatorNodes();
ConeNode *getConeNodes();
CoordinateNode *getCoordinateNodes();
CoordinateInterpolatorNode *getCoordinateInterpolatorNodes();
CylinderNode *getCylinderNodes();
CylinderSensorNode *getCylinderSensorNodes();
DirectionalLightNode *getDirectionalLightNodes();
ElevationGridNode *getElevationGridNodes();
ExtrusionNode *getExtrusionNodes();
FogNode *getFogNodes();
FontStyleNode *getFontStyleNodes();
GroupNode *getGroupNodes();
ImageTextureNode *getImageTextureNodes();
IndexedFaceSetNode *getIndexedFaceSetNodes();
IndexedLineSetNode *getIndexedLineSetNodes();
InlineNode *getInlineNodes();
LodNode *getLodNodes();
MaterialNode *getMaterialNodes();
MovieTextureNode *getMovieTextureNodes();
NavigationInfoNode *getNavigationInfoNodes();
```

```
NormalNode *getNormalNodes();
NormalInterpolatorNode *getNormalInterpolatorNodes();
OrientationInterpolatorNode *getOrientationInterpolatorNodes();
PixelTextureNode *getPixelTextureNodes();
PlaneSensorNode *getPlaneSensorNodes();
PointLightNode *getPointLightNodes();
PointSetNode *getPointSetNodes();
PositionInterpolatorNode *getPositionInterpolatorNodes();
ProximitySensorNode *getProximitySensorNodes();
ScalarInterpolatorNode *getScalarInterpolatorNodes();
ScriptNode *getScriptNodes();
ShapeNode *getShapeNodes();
SoundNode *getSoundNodes();
SphereNode *getSphereNodes();
SphereSensorNode *getSphereSensorNodes();
SpotLightNode *getSpotLightNodes();
SwitchNode *getSwitchNodes();
TextNode *getTextNodes();
TextureCoordinateNode *getTextureCoordinateNodes();
TextureTransformNode *getTextureTransformNodes();
TimeSensorNode *getTimeSensorNodes();
TouchSensorNode *getTouchSensorNodes();
TransformNode *getTransformNodes();
ViewpointNode *getViewpointNodes();
VisibilitySensorNode *getVisibilitySensorNodes();
WorldInfoNode *getWorldInfoNodes();

////////////////////////////////////////////////
//       Find Node
////////////////////////////////////////////////

GroupingNode *findGroupingNode();
AnchorNode *findAnchorNode();
```

```
AppearanceNode *findAppearanceNode();
AudioClipNode *findAudioClipNode();
BackgroundNode *findBackgroundNode();
BillboardNode *findBillboardNode();
BoxNode *findBoxNode();
CollisionNode *findCollisionNode();
ColorNode *findColorNode();
ColorInterpolatorNode *findColorInterpolatorNode();
ConeNode *findConeNode();
CoordinateNode *findCoordinateNode();
CoordinateInterpolatorNode *findCoordinateInterpolatorNode();
CylinderNode *findCylinderNode();
CylinderSensorNode *findCylinderSensorNode();
DirectionalLightNode *findDirectionalLightNode();
ElevationGridNode *findElevationGridNode();
ExtrusionNode *findExtrusionNode();
FogNode *findFogNode();
FontStyleNode *findFontStyleNode();
GroupNode *findGroupNode();
ImageTextureNode *findImageTextureNode();
IndexedFaceSetNode *findIndexedFaceSetNode();
IndexedLineSetNode *findIndexedLineSetNode();
InlineNode *findInlineNode();
LodNode *findLodNode();
MaterialNode *findMaterialNode();
MovieTextureNode *findMovieTextureNode();
NavigationInfoNode *findNavigationInfoNode();
NormalNode *findNormalNode();
NormalInterpolatorNode *findNormalInterpolatorNode();
OrientationInterpolatorNode *findOrientationInterpolatorNode();
PixelTextureNode *findPixelTextureNode();
PlaneSensorNode *findPlaneSensorNode();
PointLightNode *findPointLightNode();
```

```
PointSetNode *findPointSetNode();
PositionInterpolatorNode *findPositionInterpolatorNode();
ProximitySensorNode *findProximitySensorNode();
ScalarInterpolatorNode *findScalarInterpolatorNode();
ScriptNode *findScriptNode();
ShapeNode *findShapeNode();
SoundNode *findSoundNode();
SphereNode *findSphereNode();
SphereSensorNode *findSphereSensorNode();
SpotLightNode *findSpotLightNode();
SwitchNode *findSwitchNode();
TextNode *findTextNode();
TextureCoordinateNode *findTextureCoordinateNode();
TextureTransformNode *findTextureTransformNode();
TimeSensorNode *findTimeSensorNode();
TouchSensorNode *findTouchSensorNode();
TransformNode *findTransformNode();
ViewpointNode *findViewpointNode();
VisibilitySensorNode *findVisibilitySensorNode();
WorldInfoNode *findWorldInfoNode();

////////////////////////////////////////////////
//       find*(char *name)
////////////////////////////////////////////////

Node *findNode(char *name);

AnchorNode *findAnchorNode(char *name);
AppearanceNode *findAppearanceNode(char *name);
AudioClipNode *findAudioClipNode(char *name);
BackgroundNode *findBackgroundNode(char *name);
BillboardNode *findBillboardNode(char *name);
BoxNode *findBoxNode(char *name);
```

```
CollisionNode *findCollisionNode(char *name);
ColorNode *findColorNode(char *name);
ColorInterpolatorNode *findColorInterpolatorNode(char *name);
ConeNode *findConeNode(char *name);
CoordinateNode *findCoordinateNode(char *name);
CoordinateInterpolatorNode *findCoordinateInterpolatorNode(char *name);
CylinderNode *findCylinderNode(char *name);
CylinderSensorNode *findCylinderSensorNode(char *name);
DirectionalLightNode *findDirectionalLightNode(char *name);
ElevationGridNode *findElevationGridNode(char *name);
ExtrusionNode *findExtrusionNode(char *name);
FogNode *findFogNode(char *name);
FontStyleNode *findFontStyleNode(char *name);
GroupNode *findGroupNode(char *name);
ImageTextureNode *findImageTextureNode(char *name);
IndexedFaceSetNode *findIndexedFaceSetNode(char *name);
IndexedLineSetNode *findIndexedLineSetNode(char *name);
InlineNode *findInlineNode(char *name);
LodNode *findLodNode(char *name);
MaterialNode *findMaterialNode(char *name);
MovieTextureNode *findMovieTextureNode(char *name);
NavigationInfoNode *findNavigationInfoNode(char *name);
NormalNode *findNormalNode(char *name);
NormalInterpolatorNode *findNormalInterpolatorNode(char *name);
OrientationInterpolatorNode *findOrientationInterpolatorNode(char *name);
PixelTextureNode *findPixelTextureNode(char *name);
PlaneSensorNode *findPlaneSensorNode(char *name);
PointLightNode *findPointLightNode(char *name);
PointSetNode *findPointSetNode(char *name);
PositionInterpolatorNode *findPositionInterpolatorNode(char *name);
ProximitySensorNode *findProximitySensorNode(char *name);
ScalarInterpolatorNode *findScalarInterpolatorNode(char *name);
ScriptNode *findScriptNode(char *name);
```

```
ShapeNode *findShapeNode(char *name);
SoundNode *findSoundNode(char *name);
SphereNode *findSphereNode(char *name);
SphereSensorNode *findSphereSensorNode(char *name);
SpotLightNode *findSpotLightNode(char *name);
SwitchNode *findSwitchNode(char *name);
TextNode *findTextNode(char *name);
TextureCoordinateNode *findTextureCoordinateNode(char *name);
TextureTransformNode *findTextureTransformNode(char *name);
TimeSensorNode *findTimeSensorNode(char *name);
TouchSensorNode *findTouchSensorNode(char *name);
TransformNode *findTransformNode(char *name);
ViewpointNode *findViewpointNode(char *name);
VisibilitySensorNode *findVisibilitySensorNode(char *name);
WorldInfoNode *findWorldInfoNode(char *name);

////////////////////////////////////////////////
// initialize
////////////////////////////////////////////////

void initialize();

////////////////////////////////////////////////
// update
////////////////////////////////////////////////

void update();

////////////////////////////////////////////////
// Bindable Nodes
////////////////////////////////////////////////

void setBindableNode(BindableNode *node, bool bind);
```

```
        void setBackgroundNode(BackgroundNode *bg, bool bind);
        void setFogNode(FogNode *fog, bool bind);
        void setNavigationInfoNode(NavigationInfoNode *navInfo, bool bind);
        void setViewpoboolNode(ViewpoboolNode *view, bool bind);

        BackgroundNode *getBackgroundNode();
        FogNode *getFogNode();
        NavigationInfoNode *getNavigationInfoNode();
        ViewpointNode *getViewpointNode();

        ////////////////////////////////////////////////
        // BoundingBoxSize
        ////////////////////////////////////////////////

        void setBoundingBoxSize(float value[]);
        void setBoundingBoxSize(float x, float y, float z);
        void getBoundingBoxSize(float value[]);

        ////////////////////////////////////////////////
        // BoundingBoxCenter
        ////////////////////////////////////////////////

        void setBoundingBoxCenter(float value[]);
        void setBoundingBoxCenter(float x, float y, float z);
        void getBoundingBoxCenter(float value[]);
};
```

# Common Node Reference

## Node

```
class Node {

        ////////////////////////////////////////////////
        // Name
        ////////////////////////////////////////////////

        void setName(char * name);
        char *getName();
        bool hasName();

        ////////////////////////////////////////////////
        // Child Node
        ////////////////////////////////////////////////

        void addChildNode(Node *node)

        int getNChildNodes();
        Node *getChildNodes();
        Node *getChildNode(int n);

        ////////////////////////////////////////////////
        // Remove
        ////////////////////////////////////////////////

        void remove();

        ////////////////////////////////////////////////
        // Parent Node
```

```
////////////////////////////////////////////////

Node *getParentNode();

////////////////////////////////////////////////
// Traversal Node List
////////////////////////////////////////////////

Node *next();
Node *nextTraversal();

////////////////////////////////////////////////
// Child Node
////////////////////////////////////////////////

GroupingNode *getGroupingNodes();
Node *getGeometryNode();
TextureNode *getTextureNode();

AnchorNode *getAnchorNodes();
AppearanceNode *getAppearanceNodes();
AudioClipNode *getAudioClipNodes();
BackgroundNode *getBackgroundNodes();
BillboardNode *getBillboardNodes();
BoxNode *getBoxeNodes();
CollisionNode *getCollisionNodes();
ColorNode *getColorNodes();
ColorInterpolatorNode *getColorInterpolatorNodes();
ConeNode *getConeNodes();
CoordinateNode *getCoordinateNodes();
CoordinateInterpolatorNode *getCoordinateInterpolatorNodes();
CylinderNode *getCylinderNodes();
CylinderSensorNode *getCylinderSensorNodes();
```

```
DirectionalLightNode *getDirectionalLightNodes();
ElevationGridNode *getElevationGridNodes();
ExtrusionNode *getExtrusionNodes();
FogNode *getFogNodes();
FontStyleNode *getFontStyleNodes();
GroupNode *getGroupNodes();
ImageTextureNode *getImageTextureNodes();
IndexedFaceSetNode *getIndexedFaceSetNodes();
IndexedLineSetNode *getIndexedLineSetNodes();
InlineNode *getInlineNodes();
LodNode *getLodNodes();
MaterialNode *getMaterialNodes();
MovieTextureNode *getMovieTextureNodes();
NavigationInfoNode *getNavigationInfoNodes();
NormalNode *getNormalNodes();
NormalInterpolatorNode *getNormalInterpolatorNodes();
OrientationInterpolatorNode *getOrientationInterpolatorNodes();
PixelTextureNode *getPixelTextureNodes();
PlaneSensorNode *getPlaneSensorNodes();
PointLightNode *getPointLightNodes();
PointSetNode *getPointSetNodes();
PositionInterpolatorNode *getPositionInterpolatorNodes();
ProximitySensorNode *getProximitySensorNodes();
ScalarInterpolatorNode *getScalarInterpolatorNodes();
ScriptNode *getScriptNodes();
ShapeNode *getShapeNodes();
SoundNode *getSoundNodes();
SphereNode *getSphereNodes();
SphereSensorNode *getSphereSensorNodes();
SpotLightNode *getSpotLightNodes();
SwitchNode *getSwitchNodes();
TextNode *getTextNodes();
TextureCoordinateNode *getTextureCoordinateNodes();
```

```
TextureTransformNode *getTextureTransformNodes();
TimeSensorNode *getTimeSensorNodes();
TouchSensorNode *getTouchSensorNodes();
TransformNode *getTransformNodes();
ViewpointNode *getViewpointNodes();
VisibilitySensorNode *getVisibilitySensorNodes();
WorldInfoNode *getWorldInfoNodes();

////////////////////////////////////////////
// Type
////////////////////////////////////////////

char *getType();

bool isGroupingNode();
bool isSpecialGroupNode();
bool isCommonNode();
bool isLightNode();
bool isGeometryNode();
bool isGeometryPropertyNode();
bool isTextureNode();
bool isSensorNode();

bool isAnchorNode();
bool isAppearanceNode();
bool isAudioClipNode();
bool isBackgroundNode();
bool isBillboardNode();
bool isBoxNode();
bool isCollisionNode();
bool isColorNode();
bool isColorInterpolatorNode();
bool isConeNode();
```

```
bool isCoordinateNode();

bool isCoordinateInterpolatorNode()

bool isCylinderNode();

bool isCylinderSensorNode();

bool isDirectionalLightNode();

bool isElevationGridNode();

bool isExtrusionNode();

bool isFogNode();

bool isFontStyleNode();

bool isGroupNode();

bool isImageTextureNode();

bool isIndexedFaceSetNode();

bool isIndexedLineSetNode();

bool isInlineNode();

bool isLodNode();

bool isMaterialNode();

bool isMovieTextureNode();

bool isNavigationInfoNode();

bool isNormalNode();

bool isNormalInterpolatorNode();

bool isOrientationInterpolatorNode();

bool isPixelTextureNode();

bool isPlaneSensorNode();

bool isPointLightNode();

bool isPointSetNode();

bool isPositionInterpolatorNode();

bool isProximitySensorNode();

bool isScalarInterpolatorNode();

bool isScriptNode();

bool isShapeNode();

bool isSoundNode();

bool isSphereNode();

bool isSphereSensorNode();
```

```
        bool isSpotLightNode();
        bool isSwitchNode();
        bool isTextNode();
        bool isTextureCoordinateNode();
        bool isTextureTransformNode();
        bool isTimeSensorNode();
        bool isTouchSensorNode();
        bool isTransformNode();
        bool isViewpointNode();
        bool isVisibilitySensorNode();
        bool isWorldInfoNode();

        ////////////////////////////////////////////////
        // Instance node
        ////////////////////////////////////////////////

        bool isInstanceNode();
        Node *createInstanceNode();
};
```

## BindableNode

```
class BindableNode : public Node {
        BindableNode();
        virtual ~BindableNode();

        SFBool *getBindField();
        void setBind(bool value);
        bool  getBind();
        bool  isBind();

        SFTime *getBindTimeField();
        void setBindTime(double value);
        double getBindTime();
```

```
        SFBool *getIsBoundField();
        void setIsBound(bool  value);
        bool  getIsBound();
        bool  isBound();
};
```

## GeometryNode

```
class GeometryNode : public Node
{
        GeometryNode();
        virtual ~GeometryNode();

        SFVec3f *getBoundingBoxSizeField();
        void setBoundingBoxSize(float value[]);
        void setBoundingBoxSize(float x, float y, float z);
        void getBoundingBoxSize(float value[]);

        SFVec3f *getBoundingBoxCenterField();
        void setBoundingBoxCenter(float value[]);
        void setBoundingBoxCenter(float x, float y, float z);
        void getBoundingBoxCenter(float value[]);

#ifdef SUPPORT_OPENGL
        SFInt32 *getDisplayListField();
        void setDisplayList(unsigned int n);
        unsigned int getDisplayList();
        virtual void draw();
#endif
};
```

## GroupingNode

```
class GroupingNode : public Node {
```

```
        GroupingNode();
        virtual ~GroupingNode();

        SFVec3f *getBoundingBoxSizeField();
        void setBoundingBoxSize(float value[]);
        void setBoundingBoxSize(float x, float y, float z);
        void getBoundingBoxSize(float value[]);

        SFVec3f *getBoundingBoxCenterField();
        void setBoundingBoxCenter(float value[]);
        void setBoundingBoxCenter(float x, float y, float z);
        void getBoundingBoxCenter(float value[]);

        void setBoundingBox(BoundingBox *bbox);
        void recomputeBoundingBox();

        GroupingNode *next();
        GroupingNode *nextTraversal();
};
```

## InterpolatorNode

```
class InterpolatorNode : public Node {
        InterpolatorNode();
        ~InterpolatorNode();

        MFFloat *getKeyField();
        void addKey(float value);
        int getNKeys();
        float getKey(int index);

        SFFloat *getFractionField();
        void setFraction(float value);
        float getFraction();
```

```
};
```

## LightNode

```
class LightNode : public Node {
       LightNode();
       virtual ~LightNode();

       SFBool *getOnField();
       void setOn(bool on);
       void setOn(int value);
       bool isOn();

       SFFloat *getIntensityField();
       void setIntensity(float value);
       float getIntensity();

       SFColor *getColorField();
       void setColor(float value[]);
       void setColor(float r, float g, float b);
       void getColor(float value[]);
};
```

## SensorNode

```
class SensorNode : public Node {
       SensorNode();
       ~SensorNode();

       SFBool *getEnabledField();
       void setEnabled(bool  value);
       void setEnabled(int value);
       bool  getEnabled();
       bool  isEnabled();
```

```
        SFBool *getIsActiveField();
        void setIsActive(bool  value);
        void setIsActive(int value);
        bool  getIsActive();
        bool isActive();
};
```

## TextureNode

```
class TextureNode : public Node {
        TextureNode();
        ~TextureNode();

        SFBool *getRepeatSField();
        void setRepeatS(bool value);
        void setRepeatS(int value);
        bool getRepeatS();

        SFBool *getRepeatTField();
        void setRepeatT(bool value);
        void setRepeatT(int value);
        bool getRepeatT();
};
```

# Node Reference

## AnchorNode

```
class AnchorNode : public GroupingNode {
        AnchorNode();
        ~AnchorNode();

        SFString *getDescriptionField();
        void setDescription(char *value);
        char *getDescription();

        MFString *getParameterField();
        void addParameter(char *value);
        int getNParameters();
        char *getParameter(int index);

        MFString *getUrlField();
        void addUrl(char *value);
        int getNUrls();
        char *getUrl(int index);

        AnchorNode *next();
        AnchorNode *nextTraversal();
};
```

## AppearanceNode

```
class AppearanceNode : public Node {
        AppearanceNode();
        ~AppearanceNode();

        AppearanceNode *next();
```

```
        AppearanceNode *nextTraversal();
};
```

## AudioClipNode

```
class AudioClipNode : public Node {
        AudioClipNode();
        ~AudioClipNode();

        SFString *getDescriptionField();
        void setDescription(char * value);
        char *getDescription();

        SFBool *getLoopField();
        void setLoop(bool value);
        void setLoop(int value);
        bool getLoop();
        bool isLoop();

        SFFloat *getPitchField();
        void setPitch(float value);
        float getPitch();

        SFTime *getStartTimeField();
        void setStartTime(double value);
        double getStartTime();

        SFTime *getStopTimeField();
        void setStopTime(double value);
        double getStopTime();

        SFBool *getIsActiveField();
        void setIsActive(bool  value);
        bool getIsActive();
```

```
        bool isActive();

        SFTime *getDurationChangedField();
        void setDurationChanged(double value);
        double getDurationChanged();
        MFString *getUrlField();

        void addUrl(char * value);
        int getNUrls();
        char *getUrl(int index);

        AudioClipNode *next();
        AudioClipNode *nextTraversal();
};
```

## BackgroundNode

```
class BackgroundNode : public BindableNode {
        BackgroundNode();
        ~BackgroundNode();

        MFColor *getGroundColorField();
        void addGroundColor(float value[]);
        int getNGroundColors();
        void getGroundColor(int index, float value[]);

        MFColor *getSkyColorField();
        void addSkyColor(float value[]);
        int getNSkyColors();
        void getSkyColor(int index, float value[]);

        MFFloat *getGroundAngleField();
        void addGroundAngle(float value);
        int getNGroundAngles();
```

```
float getGroundAngle(int index);

MFFloat *getSkyAngleField();
void addSkyAngle(float value);
int getNSkyAngles();
float getSkyAngle(int index);

MFString *getFrontUrlField();
void addFrontUrl(char *value);
int getNFrontUrls();
char * getFrontUrl(int index);

MFString *getBackUrlField();
void addBackUrl(char *value);
int getNBackUrls();
char * getBackUrl(int index);

MFString *getLeftUrlField();
void addLeftUrl(char *value);
int getNLeftUrls();
char * getLeftUrl(int index);

MFString *getRightUrlField();
void addRightUrl(char *value);
int getNRightUrls();
char * getRightUrl(int index);

MFString *getTopUrlField();
void addTopUrl(char *value);
int getNTopUrls();
char * getTopUrl(int index);

MFString *getBottomUrlField();
```

```
        void addBottomUrl(char *value);
        int getNBottomUrls();
        char * getBottomUrl(int index);

        BackgroundNode *next();
        BackgroundNode *nextTraversal();
};
```

## BillboardNode

```
class BillboardNode : public GroupingNode {
        BillboardNode();
        ~BillboardNode();

        SFVec3f *getAxisOfRotationField();
        void setAxisOfRotation(float value[]);
        void setAxisOfRotation(float x, float y, float z);
        void getAxisOfRotation(float value[]);

        BillboardNode *next();
        BillboardNode *nextTraversal();
};
```

## BoxNode

```
class BoxNode : public GeometryNode {
        BoxNode();
        ~BoxNode();

        SFVec3f *getSizeField();
        void setSize(float value[]);
        void setSize(float x, float y, float z);
        void getSize(float value[]);
        float getX();
        float getY();
```

```
        float getZ();

        BoxNode *next();
        BoxNode *nextTraversal();
};
```

## CollisionNode

```
class CollisionNode : public GroupingNode {
        CollisionNode();
        ~CollisionNode();

        SFBool *getCollideField();
        void setCollide(bool  value);
        void setCollide(int value);
        bool getCollide();

        SFTime *getCollideTimeField();
        void setCollideTime(double value);
        double getCollideTime();

        CollisionNode *next();
        CollisionNode *nextTraversal();
};
```

## ColorInterpolatorNode

```
class ColorInterpolatorNode : public InterpolatorNode {
        ColorInterpolatorNode();
        ~ColorInterpolatorNode();

        MFColor *getKeyValueField();
        void addKeyValue(float color[]);
        int getNKeyValues();
        void getKeyValue(int index, float color[]);
```

```
        SFColor *getValueField();
        void setValue(float color[]);
        void getValue(float color[]);

        ColorInterpolatorNode *next();
        ColorInterpolatorNode *nextTraversal();
};
```

## ColorNode

```
class ColorNode : public Node {
        ColorNode();
        ~ColorNode();

        MFColor *getColorField();
        void addColor(float color[]);
        int getNColors();
        void getColor(int index, float color[]);

        ColorNode *next();
        ColorNode *nextTraversal();
};
```

## ConeNode

```
class ConeNode : public GeometryNode {
        ConeNode();
        ~ConeNode();

        SFFloat *getBottomRadiusField();
        void setBottomRadius(float value);
        float getBottomRadius();

        SFFloat *getHeightField();
```

```
        void setHeight(float value);
        float getHeight();

        SFBool *getSideField();
        void setSide(bool value);
        void setSide(int value);
        bool getSide();

        SFBool *getBottomField();
        void setBottom(bool value);
        void setBottom(int value);
        bool getBottom();

        ConeNode *next();
        ConeNode *nextTraversal();
};
```

## CoordinateInterpolatorNode

```
class CoordinateInterpolatorNode : public InterpolatorNode {
        CoordinateInterpolatorNode();
        ~CoordinateInterpolatorNode();

        MFVec3f *getKeyValueField();
        void addKeyValue(float vector[]);
        int getNKeyValues();
        void getKeyValue(int index, float vector[]);

        SFVec3f *getValueField();
        void setValue(float vector[]);
        void getValue(float vector[]);

        CoordinateInterpolatorNode *next();
        CoordinateInterpolatorNode *nextTraversal();
```

```
};
```

## CoordinateNode

```
class CoordinateNode : public Node {
        CoordinateNode();
        ~CoordinateNode();

        MFVec3f *getPointField();
        void addPoint(float point[]);
        void addPoint(float x, float y, float z);
        int getNPoints();
        void getPoint(int index, float point[]);
        void setPoint(int index, float point[]);
        void setPoint(int index, float x, float y, float z);
        void removePoint(int index);
        void removeLastPoint();
        void removeFirstPoint();
        void removeAllPoints();

        CoordinateNode *next();
        CoordinateNode *nextTraversal();
};
```

## CylinderNode

```
class CylinderNode : public GeometryNode {
        CylinderNode();
        ~CylinderNode();

        SFFloat *getRadiusField();
        void setRadius(float value);
        float getRadius();

        SFFloat *getHeightField();
```

```
        void setHeight(float value);
        float getHeight();

        SFBool *getTopField();
        void setTop(bool value);
        void setTop(int value);
        bool getTop();

        SFBool *getSideField();
        void setSide(bool value);
        void setSide(int value);
        bool getSide();

        SFBool *getBottomField();
        void setBottom(bool  value);
        void setBottom(int value);
        bool  getBottom();

        CylinderNode *next();
        CylinderNode *nextTraversal();
};
```

## CylinderSensorNode

```
class CylinderSensorNode : public SensorNode {
        CylinderSensorNode();
        ~CylinderSensorNode();

        SFBool *getAutoOffsetField();
        void setAutoOffset(bool  value);
        void setAutoOffset(int value);
        bool  getAutoOffset();
        bool  isAutoOffset();
```

```
        SFFloat *getDiskAngleField();
        void setDiskAngle(float value);
        float getDiskAngle();

        SFFloat *getMinAngleField();
        void setMinAngle(float value);
        float getMinAngle();

        SFFloat *getMaxAngleField();
        void setMaxAngle(float value);
        float getMaxAngle();

        SFFloat *getOffsetField();
        void setOffset(float value);
        float getOffset();

        SFRotation *getRotationChangedField();
        void setRotationChanged(float value[]);
        void setRotationChanged(float x, float y, float z, float rot);
        void getRotationChanged(float value[]);

        SFVec3f *getTrackPointChangedField();
        void setTrackPointChanged(float value[]);
        void setTrackPointChanged(float x, float y, float z);
        void getTrackPointChanged(float value[]);

        CylinderSensorNode *next();
        CylinderSensorNode *nextTraversal();
};
```

## DirectionalLightNode

```
class DirectionalLightNode : public LightNode {
        DirectionalLightNode();
```

```
        ~DirectionalLightNode();

        SFFloat *getAmbientIntensityField();
        void setAmbientIntensity(float value);
        float getAmbientIntensity();

        SFVec3f *getDirectionField();
        void setDirection(float value[]);
        void setDirection(float x, float y, float z);
        void getDirection(float value[]);

        DirectionalLightNode *next();
        DirectionalLightNode *nextTraversal();
};
```

## ElevationGridNode

```
class ElevationGridNode : public GeometryNode {
        ElevationGridNode();
        ~ElevationGridNode();

        SFFloat *getXSpacingField();
        void setXSpacing(float value);
        float getXSpacing();

        SFFloat *getZSpacingField();
        void setZSpacing(float value);
        float getZSpacing();

        SFInt32 *getXDimensionField();
        void setXDimension(int value);
        int getXDimension();

        SFInt32 *getZDimensionField();
```

```
void setZDimension(int value);
int getZDimension();

SFBool *getColorPerVertexField();
void setColorPerVertex(bool  value);
void setColorPerVertex(int value);
bool  getColorPerVertex();

SFBool *getNormalPerVertexField();
void setNormalPerVertex(bool  value);
void setNormalPerVertex(int value);
bool  getNormalPerVertex();

SFBool *getCCWField();
void setCCW(bool  value);
void setCCW(int value);
bool  getCCW();

SFBool *getSolidField();
void setSolid(bool  value);
void setSolid(int value);
bool  getSolid();

SFFloat *getCreaseAngleField();
void setCreaseAngle(float value);
float getCreaseAngle();

MFFloat *getHeightField();
void addHeight(float value);
int getNHeights();
float getHeight(int index);

ElevationGridNode *next();
```

```
        ElevationGridNode *nextTraversal();
};
```

## ExtrusionNode

```
class ExtrusionNode : public GeometryNode {
        ExtrusionNode();
        ~ExtrusionNode();

        SFBool *getBeginCapField();
        void setBeginCap(bool value);
        void setBeginCap(int value);
        bool getBeginCap();

        SFBool *getEndCapField();
        void setEndCap(bool value);
        void setEndCap(int value);
        bool getEndCap();

        SFBool *getCCWField();
        void setCCW(bool value);
        void setCCW(int value);
        bool getCCW();

        SFBool *getConvexField();
        void setConvex(bool value);
        void setConvex(int value);
        bool getConvex();

        SFFloat *getCreaseAngleField();
        void setCreaseAngle(float value);
        float getCreaseAngle();

        SFBool *getSolidField();
```

```
        void setSolid(bool value);
        void setSolid(int value);
        bool getSolid();

        MFRotation *getOrientationField();
        void addOrientation(float value[]);
        void addOrientation(float x, float y, float z, float angle);
        int getNOrientations();
        void getOrientation(int index, float value[]);

        MFVec2f *getScaleField();
        void addScale(float value[]);
        void addScale(float x, float z);
        int getNScales();
        void getScale(int index, float value[]);

        MFVec2f *getCrossSectionField();
        void addCrossSection(float value[]);
        void addCrossSection(float x, float z);
        int getNCrossSections();
        void getCrossSection(int index, float value[]);

        MFVec3f *getSpineField();
        void addSpine(float value[]);
        void addSpine(float x, float y, float z);
        int getNSpines();
        void getSpine(int index, float value[]);

        ExtrusionNode *next();
        ExtrusionNode *nextTraversal();
};
```

## FogNode

```
class FogNode : public BindableNode {
        FogNode();
        ~FogNode();

        SFColor *getColorField();
        void setColor(float value[]);
        void setColor(float r, float g, float b);
        void getColor(float value[]);

        SFString *getFogTypeField();
        void setFogType(char *value);
        char *getFogType();

        SFFloat *getVisibilityRangeField();
        void setVisibilityRange(float value);
        float getVisibilityRange();

        FogNode *next();
        FogNode *nextTraversal();
};
```

## FontStyleNode

```
class FontStyleNode : public Node {
        FontStyleNode();
        ~FontStyleNode();

        SFFloat *getSizeField();
        void setSize(float value);
        float getSize();

        SFString *getFamilyField();
        void setFamily(char *value);
```

```
char *getFamily();
int getFamilyNumber();

SFString *getStyleField();
void setStyle(char *value);
char *getStyle();
int getStyleNumber();

SFString *getLanguageField();
void setLanguage(char *value);
char *getLanguage();

SFBool *getHorizontalField();
void setHorizontal(bool value);
void setHorizontal(int value);
bool getHorizontal();

SFBool *getLeftToRightField();
void setLeftToRight(bool value);
void setLeftToRight(int value);
bool getLeftToRight();

SFBool *getTopToBottomField();
void setTopToBottom(bool value);
void setTopToBottom(int value);
bool getTopToBottom();

MFString *getJustifyField();
void addJustify(char *value);
int getNJustifys();
char *getJustify(int index);

SFFloat *getSpacingField();
```

```
        void setSpacing(float value);
        float getSpacing();

        FontStyleNode *next();
        FontStyleNode *nextTraversal();
};
```

## GroupNode

```
class GroupNode : public GroupingNode {
        GroupNode();
        ~GroupNode();

        GroupNode *next();
        GroupNode *nextTraversal();
};
```

## ImageTextureNode

```
class ImageTextureNode : public TextureNode {
        ImageTextureNode();
        ~ImageTextureNode();

        MFString *getUrlField();
        void addUrl(char * value);
        int getNUrls();
        char *getUrl(int index);

        ImageTextureNode *next();
        ImageTextureNode *nextTraversal();
};
```

## IndexedFaceSetNode

```
class IndexedFaceSetNode : public GeometryNode {
        IndexedFaceSetNode();
```

```
~IndexedFaceSetNode();

SFBool *getCCWField();
void setCCW(bool value);
void setCCW(int value);
bool getCCW();

SFBool *getColorPerVertexField();
void setColorPerVertex(bool value);
void setColorPerVertex(int value);
bool getColorPerVertex();

SFBool *getNormalPerVertexField();
void setNormalPerVertex(bool value);
void setNormalPerVertex(int value);
bool getNormalPerVertex();

SFBool *getSolidField();
void setSolid(bool value);
void setSolid(int value);
bool getSolid();

SFBool *getConvexField();
void setConvex(bool value);
void setConvex(int value);
bool getConvex();

SFFloat *getCreaseAngleField();
void setCreaseAngle(float value);
float getCreaseAngle();

MFInt32 *getCoordIndexField();
void addCoordIndex(int value);
```

```
        int getNCoordIndexes();
        int getCoordIndex(int index);

        MFInt32 *getTexCoordIndexField();
        void addTexCoordIndex(int value);
        int getNTexCoordIndexes();
        int getTexCoordIndex(int index);

        MFInt32 *getColorIndexField();
        void addColorIndex(int value);
        int getNColorIndexes();
        int getColorIndex(int index);

        MFInt32 *getNormalIndexField();
        void addNormalIndex(int value);
        int getNNormalIndexes();
        int getNormalIndex(int index);

        IndexedFaceSetNode *next();
        IndexedFaceSetNode *nextTraversal();
};
```

## IndexedLineSetNode

```
class IndexedLineSetNode : public GeometryNode {
        IndexedLineSetNode();
        ~IndexedLineSetNode();

        SFBool *getColorPerVertexField();
        void setColorPerVertex(bool value);
        void setColorPerVertex(int value);
        bool getColorPerVertex();

        MFInt32 *getCoordIndexField();
```

```
        void addCoordIndex(int value);
        int getNCoordIndexes();
        int getCoordIndex(int index);

        MFInt32 *getColorIndexField();
        void addColorIndex(int value);
        int getNColorIndexes();
        int getColorIndex(int index);

        IndexedLineSetNode *next();
        IndexedLineSetNode *nextTraversal();
};
```

## InlineNode

```
class InlineNode : public GroupingNode {
        InlineNode();
        ~InlineNode();

        MFString *getUrlField();
        void addUrl(char *value);
        int getNUrls();
        char *getUrl(int index);
        void setUrl(int index, char *urlString);

        InlineNode *next();
        InlineNode *nextTraversal();
};
```

## LodNode

```
class LodNode : public Node {
        LodNode();
        ~LodNode();
```

```
        SFVec3f *getCenterField();
        void setCenter(float value[]);
        void setCenter(float x, float y, float z);
        void getCenter(float value[]);

        MFFloat *getRangeField();
        void addRange(float value);
        int getNRanges();
        float getRange(int index);

        LodNode *next();
        LodNode *nextTraversal();
};
```

## MaterialNode

```
class MaterialNode : public Node {
        MaterialNode();
        ~MaterialNode();

        SFFloat *getTransparencyField();
        void setTransparency(float value);
        float getTransparency();

        SFFloat *getAmbientIntensityField();
        void setAmbientIntensity(float intensity);
        float getAmbientIntensity();

        SFFloat *getShininessField();
        void setShininess(float value);
        float getShininess();

        SFColor *getDiffuseColorField();
        void setDiffuseColor(float value[]);
```

```
        void setDiffuseColor(float r, float g, float b);
        void getDiffuseColor(float value[]);

        SFColor *getSpecularColorField();
        void setSpecularColor(float value[]);
        void setSpecularColor(float r, float g, float b);
        void getSpecularColor(float value[]);

        SFColor *getEmissiveColorField();
        void setEmissiveColor(float value[]);
        void setEmissiveColor(float r, float g, float b);
        void getEmissiveColor(float value[]);

        MaterialNode *next();
        MaterialNode *nextTraversal();
};
```

## MovieTextureNode

```
class MovieTextureNode : public TextureNode {
        MovieTextureNode();
        ~MovieTextureNode();

        MFString *getUrlField();
        void addUrl(char *value);
        int getNUrls();
        char *getUrl(int index);

        SFBool *getLoopField();
        void setLoop(bool value);
        void setLoop(int value);
        bool getLoop();
        bool isLoop();
```

```
        SFFloat *getSpeedField();
        void setSpeed(float value);
        float getSpeed();

        SFTime *getStartTimeField();
        void setStartTime(double value);
        double getStartTime();

        SFTime *getStopTimeField();
        void setStopTime(double value);
        double getStopTime();

        SFBool *getIsActiveField();
        void setIsActive(bool value);
        bool getIsActive();
        bool isActive();

        SFTime *getDurationChangedField();
        void setDurationChanged(double value);
        double getDurationChanged();

        MovieTextureNode *next();
        MovieTextureNode *nextTraversal();
};
```

## NavigationInfoNode

```
class NavigationInfoNode : public BindableNode {
        NavigationInfoNode();
        ~NavigationInfoNode();

        MFString *getTypeField();
        void addType(char *value);
        int getNTypes();
```

```
        char *getType(int index);

        MFFloat *getAvatarSizeField();
        void addAvatarSize(float value);
        int getNAvatarSizes();
        float getAvatarSize(int index);

        SFBool *getHeadlightField();
        void setHeadlight(bool value);
        void setHeadlight(int value);
        bool getHeadlight();

        SFFloat *getVisibilityLimitField();
        void setVisibilityLimit(float value);
        float getVisibilityLimit();

        SFFloat *getSpeedField();
        void setSpeed(float value);
        float getSpeed();

        NavigationInfoNode *next();
        NavigationInfoNode *nextTraversal();
};
```

## NormalInterpolatorNode

```
class NormalInterpolatorNode : public InterpolatorNode {
        NormalInterpolatorNode();
        ~NormalInterpolatorNode();

        MFVec3f *getKeyValueField();
        void addKeyValue(float vector[]);
        int getNKeyValues();
        void getKeyValue(int index, float vector[]);
```

```
        SFVec3f *getValueField();
        void setValue(float vector[]);
        void getValue(float vector[]);

        NormalInterpolatorNode *next();
        NormalInterpolatorNode *nextTraversal();
};
```

## NormalNode

```
class NormalNode : public Node {
        NormalNode();
        ~NormalNode();

        MFVec3f *getVectorField();
        void addVector(float value[]);
        int getNVectors();
        void getVector(int index, float value[]);

        NormalNode *next();
        NormalNode *nextTraversal();
};
```

## OrientationInterpolatorNode

```
class OrientationInterpolatorNode : public InterpolatorNode {
        OrientationInterpolatorNode();
        ~OrientationInterpolatorNode();

        MFRotation *getKeyValueField();
        void addKeyValue(float rotation[]);
        int getNKeyValues();
        void getKeyValue(int index, float rotation[]);
```

```
        SFRotation *getValueField();
        void setValue(float rotation[]);
        void getValue(float rotation[]);

        OrientationInterpolatorNode *next();
        OrientationInterpolatorNode *nextTraversal();
};
```

## PixelTextureNode

```
class PixelTextureNode : public TextureNode {
        PixelTextureNode();
        ~PixelTextureNode();



        SFImage *getImageField();
        void addImage(int value);
        int getNImages();
        int getImage(int index);

        PixelTextureNode *next();
        PixelTextureNode *nextTraversal();
};
```

## PlaneSensorNode

```
class PlaneSensorNode : public SensorNode {
        PlaneSensorNode();
        ~PlaneSensorNode();

        SFBool *getAutoOffsetField();
        void setAutoOffset(bool value);
        void setAutoOffset(int value);
        bool getAutoOffset();
```

```
        bool  isAutoOffset();

        SFVec2f *getMinPositionField();
        void setMinPosition(float value[]);
        void setMinPosition(float x, float y);
        void getMinPosition(float value[]);
        void getMinPosition(float *x, float *y);

        SFVec2f *getMaxPositionField();
        void setMaxPosition(float value[]);
        void setMaxPosition(float x, float y);
        void getMaxPosition(float value[]);
        void getMaxPosition(float *x, float *y);

        SFVec3f *getOffsetField();
        void setOffset(float value[]);
        void getOffset(float value[]);

        SFVec3f *getTranslationChangedField();
        void setTranslationChanged(float value[]);
        void setTranslationChanged(float x, float y, float z);
        void getTranslationChanged(float value[]);

        SFVec3f *getTrackPointChangedField();
        void setTrackPointChanged(float value[]);
        void setTrackPointChanged(float x, float y, float z);
        void getTrackPointChanged(float value[]);

        PlaneSensorNode *next();
        PlaneSensorNode *nextTraversal();
};
```

## PointLightNode

```
class PointLightNode : public LightNode {
        PointLightNode();
        ~PointLightNode();

        SFFloat *getAmbientIntensityField();
        void setAmbientIntensity(float value);
        float getAmbientIntensity();

        SFVec3f *getLocationField();
        void setLocation(float value[]);
        void setLocation(float x, float y, float z);
        void getLocation(float value[]);

        SFFloat *getRadiusField();
        void setRadius(float value);
        float getRadius();

        SFVec3f *getAttenuationField();
        void setAttenuation(float value[]);
        void setAttenuation(float x, float y, float z);
        void getAttenuation(float value[]);

        PointLightNode *next();
        PointLightNode *nextTraversal();
};
```

## PointSetNode

```
class PointSetNode : public GeometryNode {
        PointSetNode();
        ~PointSetNode();

        PointSetNode *next();
```

```
        PointSetNode *nextTraversal();
};
```

## PositionInterpolatorNode

```
class PositionInterpolatorNode : public InterpolatorNode {
        PositionInterpolatorNode();
        ~PositionInterpolatorNode();

        MFVec3f *getKeyValueField();
        void addKeyValue(float vector[]);
        int getNKeyValues();
        void getKeyValue(int index, float vector[]);

        SFVec3f *getValueField();
        void setValue(float vector[]);
        void getValue(float vector[]);
        PositionInterpolatorNode *next();
        PositionInterpolatorNode *nextTraversal();
};
```

## ProximitySensorNode

```
class ProximitySensorNode : public SensorNode {
        ProximitySensorNode();
        ~ProximitySensorNode();

        SFVec3f *getCenterField();
        void setCenter(float value[]);
        void setCenter(float x, float y, float z);
        void getCenter(float value[]);

        SFVec3f *getSizeField();
        void setSize(float value[]);
        void setSize(float x, float y, float z);
```

```
        void getSize(float value[]);

        SFVec3f *getPositionChangedField();
        void setPositionChanged(float value[]);
        void setPositionChanged(float x, float y, float z);
        void getPositionChanged(float value[]);

        SFRotation *getOrientationChangedField();
        void setOrientationChanged(float value[]);
        void setOrientationChanged(float x, float y, float z, float rot);
        void getOrientationChanged(float value[]);

        SFTime *getEnterTimeField();
        void setEnterTime(double value);
        double getEnterTime();

        SFTime *getExitTimeField();
        void setExitTime(double value);
        double getExitTime();

        SFBool *getInRegionField();
        void setInRegion(bool value);
        bool inRegion();

        ProximitySensorNode *next();
        ProximitySensorNode *nextTraversal();
};
```

## ScalarInterpolatorNode

```
class ScalarInterpolatorNode : public InterpolatorNode {
        ScalarInterpolatorNode();
        ~ScalarInterpolatorNode();
```

```
        MFFloat *getKeyValueField();
        void addKeyValue(float value);
        int getNKeyValues();
        float getKeyValue(int index);

        SFFloat *getValueField();
        void setValue(float vector);
        float getValue();

        ScalarInterpolatorNode *next();
        ScalarInterpolatorNode *nextTraversal();
};
```

## ScriptNode

```
class ScriptNode : public Node {
        ScriptNode();
        ~ScriptNode();

        SFBool *getDirectOutputField();
        void setDirectOutput(bool  value);
        void setDirectOutput(int value);
        bool  getDirectOutput();

        SFBool *getMustEvaluateField();
        void setMustEvaluate(bool  value);
        void setMustEvaluate(int value);
        bool  getMustEvaluate();

        MFString *getUrlField();
        void addUrl(char * value);
        int getNUrls();
        char *getUrl(int index);
```

```
        ScriptNode *next();
        ScriptNode *nextTraversal();
};
```

## ShapeNode

```
class ShapeNode : public Node {
        ShapeNode();
        ~ShapeNode();

        ShapeNode *next();
        ShapeNode *nextTraversal();
};
```

## SoundNode

```
class SoundNode : public Node {
        SoundNode();
        ~SoundNode();

        SFVec3f *getDirectionField();
        void setDirection(float value[]);
        void setDirection(float x, float y, float z);
        void getDirection(float value[]);

        SFVec3f *getLocationField();
        void setLocation(float value[]);
        void setLocation(float x, float y, float z);
        void getLocation(float value[]);

        SFFloat *getMinFrontField();
        void setMinFront(float value);
        float getMinFront();

        SFFloat *getMaxFrontField();
```

```
        void setMaxFront(float value);
        float getMaxFront();

        SFFloat *getMinBackField();
        void setMinBack(float value);
        float getMinBack();

        SFFloat *getMaxBackField();
        void setMaxBack(float value);
        float getMaxBack();

        SFFloat *getIntensityField();
        void setIntensity(float value);
        float getIntensity();

        SFFloat *getPriorityField();
        void setPriority(float value);
        float getPriority();

        SFBool *getSpatializeField();
        void setSpatialize(bool value);
        void setSpatialize(int value);
        bool getSpatialize();

        SoundNode *next();
        SoundNode *nextTraversal();
};
```

## SphereNode

```
class SphereNode : public GeometryNode {
        SphereNode();
        ~SphereNode();
```

```
        SFFloat *getRadiusField();
        void setRadius(float value);
        float getRadius();

        SphereNode *next();
        SphereNode *nextTraversal();
};
```

## SphereSensorNode

```
class SphereSensorNode : public SensorNode {
        SphereSensorNode();
        ~SphereSensorNode();

        SFBool *getAutoOffsetField();
        void setAutoOffset(bool value);
        void setAutoOffset(int value);
        bool getAutoOffset();
        bool  isAutoOffset();

        SFRotation *getOffsetField();
        void setOffset(float value[]);
        void getOffset(float value[]);

        SFRotation *getRotationChangedField();
        void setRotationChanged(float value[]);
        void setRotationChanged(float x, float y, float z, float rot);
        void getRotationChanged(float value[]);

        SFVec3f *getTrackPointChangedField();
        void setTrackPointChanged(float value[]);
        void setTrackPointChanged(float x, float y, float z);
        void getTrackPointChanged(float value[]);
```

```
        SphereSensorNode *next();
        SphereSensorNode *nextTraversal();
};
```

## SpotLightNode

```
class SpotLightNode : public LightNode {
        SpotLightNode();
        ~SpotLightNode();

        SFFloat *getAmbientIntensityField();
        void setAmbientIntensity(float value);
        float getAmbientIntensity();

        SFVec3f *getLocationField();
        void setLocation(float value[]);
        void setLocation(float x, float y, float z);
        void getLocation(float value[]);

        SFVec3f *getDirectionField();
        void setDirection(float value[]);
        void setDirection(float x, float y, float z);
        void getDirection(float value[]);

        SFFloat *getRadiusField();
        void setRadius(float value);
        float getRadius();

        SFVec3f *getAttenuationField();
        void setAttenuation(float value[]);
        void setAttenuation(float x, float y, float z);
        void getAttenuation(float value[]);

        SFFloat *getBeamWidthField();
```

```
        void setBeamWidth(float value);
        float getBeamWidth();

        SFFloat *getCutOffAngleField();
        void setCutOffAngle(float value);
        float getCutOffAngle();

        SpotLightNode *next();
        SpotLightNode *nextTraversal();
};
```

## SwitchNode

```
class SwitchNode : public Node {
        SwitchNode();
        ~SwitchNode();

        SFInt32 *getWhichChoiceField();
        void setWhichChoice(int value);
        int getWhichChoice();

        SwitchNode *next();
        SwitchNode *nextTraversal();
};
```

## TextNode

```
class TextNode : public GeometryNode {
        TextNode();
        ~TextNode();

        SFFloat *getMaxExtentField();
        void setMaxExtent(float value);
        float getMaxExtent();
```

```
        MFString *getStringField();
        void addString(char *value);
        int getNStrings();
        char *getString(int index);

        MFFloat *getLengthField();
        void addLength(float value);
        int getNLengths();
        float getLength(int index);

        TextNode *next();
        TextNode *nextTraversal();
};
```

## TextureCoordinateNode

```
class TextureCoordinateNode : public Node {
        TextureCoordinateNode();
        ~TextureCoordinateNode();

        MFVec2f *getPointField();
        void addPoint(float point[]);
        int getNPoints();
        void getPoint(int index, float point[]);

        TextureCoordinateNode *next();
        TextureCoordinateNode *nextTraversal();
};
```

## TextureTransformNode

```
class TextureTransformNode : public Node {
        TextureTransformNode();
        ~TextureTransformNode();
```

```
        SFVec2f *getTranslationField();
        void setTranslation(float value[]);
        void setTranslation(float x, float y);
        void getTranslation(float value[]);

        SFVec2f *getScaleField();
        void setScale(float value[]);
        void setScale(float x, float y);
        void getScale(float value[]);

        SFVec2f *getCenterField();
        void setCenter(float value[]);
        void setCenter(float x, float y);
        void getCenter(float value[]);

        SFFloat *getRotationField();
        void setRotation(float value);
        float getRotation();

        TextureTransformNode *next();
        TextureTransformNode *nextTraversal();
};
```

## TimeSensorNode

```
class TimeSensorNode : public SensorNode {
        TimeSensorNode();
        ~TimeSensorNode();

        SFBool *getLoopField();
        void setLoop(bool value);
        void setLoop(int value);
        bool getLoop();
        bool isLoop();
```

```
        SFTime *getCycleIntervalField();
        void setCycleInterval(double value);
        double getCycleInterval();

        SFTime *getStartTimeField();
        void setStartTime(double value);
        double getStartTime();

        SFTime *getStopTimeField();
        void setStopTime(double value);
        double getStopTime();

        SFFloat *getFractionChangedField();
        void setFractionChanged(float value);
        float getFractionChanged();

        SFTime *getCycleTimeField();
        void setCycleTime(double value);
        double getCycleTime();

        SFTime *getTimeField();
        void setTime(double value);
        double getTime();

        TimeSensorNode *next();
        TimeSensorNode *nextTraversal();
};
```

## TouchSensorNode

```
class TouchSensorNode : public SensorNode {
        TouchSensorNode();
        ~TouchSensorNode();
```

```
        SFBool *getIsOverField();
        void setIsOver(bool  value);
        void setIsOver(int value);
        bool  getIsOver();
        bool  isOver();

        SFVec3f *getHitNormalChangedField();
        void setHitNormalChanged(float value[]);
        void setHitNormalChanged(float x, float y, float z);
        void getHitNormalChanged(float value[]);

        SFVec3f *getHitPointChangedField();
        void setHitPointChanged(float value[]);
        void setHitPointChanged(float x, float y, float z);
        void getHitPointChanged(float value[]);

        SFVec2f *getHitTexCoordField();
        void setHitTexCoord(float value[]);
        void setHitTexCoord(float x, float y);
        void getHitTexCoord(float value[]);

        SFTime *getTouchTimeField();
        void setTouchTime(double value);
        double getTouchTime();

        TouchSensorNode *next();
        TouchSensorNode *nextTraversal();
};
```

## TransformNode

```
class TransformNode : public GroupingNode {
        TransformNode();
```

```
        ~TransformNode();

        SFVec3f *getTranslationField();
        void setTranslation(float value[]);
        void setTranslation(float x, float y, float z);
        void getTranslation(float value[]);

        SFVec3f *getScaleField();
        void setScale(float value[]);
        void setScale(float x, float y, float z);
        void getScale(float value[]);

        SFVec3f *getCenterField();
        void setCenter(float value[]);
        void setCenter(float x, float y, float z);
        void getCenter(float value[]);

        SFRotation *getRotationField();
        void setRotation(float value[]);
        void setRotation(float x, float y, float z, float w);
        void getRotation(float value[]);

        SFRotation *getScaleOrientationField();
        void setScaleOrientation(float value[]);
        void setScaleOrientation(float x, float y, float z, float w);
        void getScaleOrientation(float value[]);

        void getSFMatrix(SFMatrix *mOut);

        TransformNode *next();
        TransformNode *nextTraversal();
};
```

## ViewpointNode

```
class ViewpointNode : public BindableNode {
        ViewpointNode();
        ~ViewpointNode();

        SFBool *getJumpField();
        void setJump(bool value);
        void setJump(int value);
        bool getJump();

        SFFloat *getFieldOfViewField();
        void setFieldOfView(float value);
        float getFieldOfView();

        SFString *getDescriptionField();
        void setDescription(char *value);
        char *getDescription();

        SFVec3f *getPositionField();
        void setPosition(float value[]);
        void setPosition(float x, float y, float z);
        void getPosition(float value[]);

        SFRotation *getOrientationField();
        void setOrientation(float value[]);
        void setOrientation(float x, float y, float z, float w);
        void getOrientation(float value[]);

        void addPosition(float worldTranslation[3]);
        void addPosition(float worldx, float worldy, float worldz);
        void addPosition(float localTranslation[3], float frame[3][3]);
        void addPosition(float x, float y, float z, float frame[3][3]);
```

```
        void addOrientation(SFRotation *rot);
        void addOrientation(float rotationValue[4]);
        void addOrientation(float x, float y, float z, float rot);

        ViewpointNode *next();
        ViewpointNode *nextTraversal();

        void getFrame(float frame[3][3]);
        void translate(float vector[3]);
        void translate(SFVec3f vec);
        void rotate(float rotation[4]);
        void rotate(SFRotation rot);

        void getMatrix(SFMatrix *matrix);
        void getMatrix(float value[4][4]);
        void getTranslationMatrix(SFMatrix *matrix);
        void getTranslationMatrix(float value[4][4]);
};
```

## VisibilitySensorNode

```
class VisibilitySensorNode : public SensorNode {
        VisibilitySensorNode();
        ~VisibilitySensorNode();

        SFVec3f *getCenterField();
        void     setCenter(float value[]);
        void     setCenter(float x, float y, float z);
        void     getCenter(float value[]);

        SFVec3f *getSizeField();
        void     setSize(float value[]);
        void     setSize(float x, float y, float z);
        void     getSize(float value[]);
```

```
        SFTime *getEnterTimeField();
        void     setEnterTime(double value);
        double   getEnterTime();

        SFTime *getExitTimeField();
        void     setExitTime(double value);
        double   getExitTime();

        VisibilitySensorNode *next();
        VisibilitySensorNode *nextTraversal();
};
```

## WorldInfoNode

```
class WorldInfoNode : public Node {
        WorldInfoNode();
        ~WorldInfoNode();

        SFString *getTitleField();
        void setTitle(char *value);
        char *getTitle();

        MFString *getInfoField();
        void addInfo(char *value);
        int getNInfos();
        char *getInfo(int index);

        WorldInfoNode *next();
        WorldInfoNode *nextTraversal();
};
```

# Field Class Reference

## Field

```
class Field {

        Field();

        virtual ~Field();

        char *getTypeName();

        char *getName();

};
```

## MField

```
class MField : public Field {

        MField();

        ~MField();


        int getSize();

        int size();


        void add(Field *object);

        void insert(int index, Field *object);

        void replace (int index, Field *object);

        void clear();


        void remove(int index);

        void removeLastObject();

        void removeFirstObject();


        Field *lastObject();

        Field *firstObject();

        Field *getObject(int index);

};
```

## SFBool

```
class SFBool : public Field {
        SFBool();
        SFBool(bool value);
        SFBool(SFBool *value);

        void setValue(bool value);
        void setValue(SFBool *value);
        bool getValue();
};
```

## SFColor

```
class SFColor : public Field {
        SFColor();
        SFColor(float r, float g, float b);
        SFColor(float value[]);
        SFColor(SFColor *color);

        void getValue(float value[]);
        float *getValue();
        float getRed();
        float getGreen();
        float getBlue();

        void setValue(float r, float g, float b);
        void setValue(float value[]);
        void setValue(SFColor *color);

        void add(float x, float y, float z);
        void add(float value[]);
        void add(SFColor value);

        void sub(float x, float y, float z);
```

```
        void sub(float value[]);
        void sub(SFColor value);
};
```

## SFFloat

```
class SFFloat : public Field {
        SFFloat();
        SFFloat(float value);
        SFFloat(SFFloat *value);

        void setValue(float value);
        void setValue(SFFloat *fvalue);
        float getValue();
};
```

## SFImage

```
class SFImage : public MField {
        SFImage();

        void addValue(int value);
        void addValue(SFInt32 *sfvalue);

        void insertValue(int index, int value);
        int get1Value(int index);

        void set1Value(int index, int value);
        void setValue(MField *mfield);
};
```

## SFInt32

```
class SFInt32 : public Field {
        SFInt32();
        SFInt32(int value);
```

```
        SFInt32(SFInt32 *value);

        void setValue(int value);
        void setValue(SFInt32 *value);
        int getValue();
};
```

## SFMatrix

```
class SFMatrix : public Field {
        SFMatrix();
        SFMatrix(float value[4][4]);
        SFMatrix(SFMatrix *value);
        SFMatrix(SFRotation *rot);
        SFMatrix(float x, float y, float z, float angle);
        SFMatrix(float x, float y, float z);

        void setValue(float value[4][4]);
        void setValue(SFMatrix *matrix);

        void setScaling(SFVec3f *vector);
        void setScaling(float value[]);
        void setScaling(float x, float y, float z);

        void setTranslation(SFVec3f *vector);
        void setTranslation(float value[]);
        void setTranslation(float x, float y, float z);

        void setDirection(SFVec3f *vector);
        void setDirection(float value[]);
        void setDirection(float x, float y, float z);

        void setRotation(SFRotation *rotation);
        void setRotation(float value[]);
```

```
        void setRotation(float x, float y, float z, float rot);

        void getValue(float value[4][4]);
        void getTranslation(float value[]);

        void add(SFMatrix *matrix);
        void multi(float vector[]);
        void multi(float *x, float *y, float *z);
        void multi(SFVec3f *vector);

        void getSFRotation(SFRotation *rotation);
};
```

## SFNode

```
class SFNode : public Field {
        SFNode();
        SFNode(Node *value);
        SFNode(SFNode *value);

        void setValue(Node *value);
        void setValue(SFNode *value);
        Node *getValue();
};
```

## SFRotation

```
class SFRotation : public Field {
        SFRotation();
        SFRotation(float x, float y, float z, float rot);
        SFRotation(float vector[3], float rot);
        SFRotation(float value[]);
        SFRotation(SFRotation *value);
        SFRotation(SFMatrix *matrix);
```

```
        void setValue(float x, float y, float z, float rot);
        void setValue(float value[]);
        void setValue(float value[], float angle);
        void setValue(SFRotation *rotation);
        void setValue(SFMatrix *matrix);

        void getValue(float value[]);
        void getVector(float vector[]);
        float getX();
        float getY();
        float getZ();
        float getAngle();

        void add(SFRotation *rot);
        void add(float rotationValue[]);
        void add(float x, float y, float z, float rot);

        void multi(float vector[]);
        void multi(float *x, float *y, float *z);
        void multi(SFVec3f *vector);

        void getSFMatrix(SFMatrix *matrix);

        void invert();
};
```

## SFString

```
class SFString : public Field {
        SFString();
        SFString(char *value);
        SFString(SFString *value);

        void setValue(char *value);
```

```
        void setValue(SFString *value);
        char *getValue();
};
```

## SFTime

```
class SFTime : public Field {
        SFTime();
        SFTime(double value);
        SFTime(SFTime *value);

        void setValue(double value);
        void setValue(SFTime *value);
        double getValue();
};
```

## SFVec2f

```
class SFVec2f : public Field {
        SFVec2f();
        SFVec2f(float x, float y);
        SFVec2f(float value[]);
        SFVec2f(SFVec2f *value);

        void getValue(float value[]);
        float *getValue();
        float getX();
        float getY();

        void setValue(float x, float y);
        void setValue(float value[]);
        void setValue(SFVec2f *vector);
        void setX(float x);
        void setY(float y);
```

```
        void add(float x, float y);
        void add(float value[]);
        void add(SFVec2f value);
        void translate(float x, float y);
        void translate(float value[]);
        void translate(SFVec2f value);

        void sub(float x, float y);
        void sub(float value[]);
        void sub(SFVec2f value);

        void scale(float value);
        void scale(float xscale, float yscale);
        void scale(float value[2]);

        void invert();

        float getScalar();

        void normalize();

        bool equals(Field *field);
        bool equals(float value[2]);
        bool equals(float x, float y);
};
```

## SFVec3f

```
class SFVec3f : public Field {
        SFVec3f();
        SFVec3f(float x, float y, float z);
        SFVec3f(float value[]);
        SFVec3f(SFVec3f *value);
```

```
void getValue(float value[]);
float *getValue();
float getX();
float getY();
float getZ();

void setValue(float x, float y, float z);
void setValue(float value[]);
void setValue(SFVec3f *vector);
void setX(float x);
void setY(float y);
void setZ(float z);

void add(float x, float y, float z);
void add(float value[]);
void add(SFVec3f value);
void translate(float x, float y, float z);
void translate(float value[]);
void translate(SFVec3f value);

void sub(float x, float y, float z);
void sub(float value[]);
void sub(SFVec3f value);

void scale(float value);
void scale(float xscale, float yscale, float zscale);
void scale(float value[3]);

void rotate(SFRotation *rotation);
void rotate(float x, float y, float z, float angle);
void rotate(float value[3]);

void invert();
```

```
        float getScalar();

        void normalize();

        bool equals(Field *field);
        bool equals(float value[3]);
        bool equals(float x, float y, float z);
};
```

## MFColor

```
class MFColor : public MField {
        MFColor();

        void addValue(float r, float g, float b);
        void addValue(float value[]);
        void addValue(SFColor *color);

        void insertValue(int index, float r, float g, float b);
        void insertValue(int index, float value[]);
        void insertValue(int index, SFColor *color);

        void get1Value(int index, float value[]);

        void set1Value(int index, float value[]);
        void set1Value(int index, float r, float g, float b);

        void setValue(MField *mfield);
        void setValue(MFColor *colors);
        void setValue(int size, float colors[][3]);
};
```

## MFFloat

```
class MFFloat : public MField {
        MFFloat();

        void addValue(float value);
        void addValue(SFFloat *sfvalue);
        void insertValue(int index, float value);
        float get1Value(int index);
        void set1Value(int index, float value);

        void setValue(MField *mfield);
        void setValue(MFFloat *values);
        void setValue(int size, float values[]);
};
```

## MFInt32

```
class MFInt32 : public MField {
        MFInt32();

        void addValue(int value);
        void addValue(SFInt32 *sfvalue);

        void insertValue(int index, int value);

        int get1Value(int index);
        void set1Value(int index, int value);

        void setValue(MField *mfield);
        void setValue(MFInt32 *values);
        void setValue(int size, int values[]);
};
```

## MFRotation

```
class MFRotation : public MField {
        MFRotation();

        void addValue(float x, float y, float z, float rot);
        void addValue(float value[]);
        void addValue(SFRotation *rotation);

        void insertValue(int index, float x, float y, float z, float rot);
        void insertValue(int index, float value[]);
        void insertValue(int index, SFRotation *rotation);

        void get1Value(int index, float value[]);
        void set1Value(int index, float value[]);
        void set1Value(int index, float x, float y, float z, float angle);

        void setValue(MField *mfield);
        void setValue(MFRotation *rotations);
        void setValue(int size, float rotations[][4]);
};
```

## MFString

```
class MFString : public MField {
        MFString();

        void addValue(char *value);
        void addValue(SFString *sfvalue);

        void insertValue(int index, char *value);

        char *get1Value(int index);

        void set1Value(int index, char *value);
```

```
        void setValue(MField *mfield);
        void setValue(MFString *strings);
        void setValue(int size, char *strings[]);
};
```

## MFTime

```
class MFTime : public MField {
        MFTime();

        void addValue(double value);
        void addValue(SFTime *sfvalue);
        void insertValue(int index, double value);

        double get1Value(int index);
        void set1Value(int index, double value);

        void setValue(MField *mfield);
        void setValue(MFTime *times);
        void setValue(int size, double times[]);
};
```

## MFVec2f

```
class MFVec2f : public MField {
        MFVec2f();

        void addValue(float x, float y);
        void addValue(float value[]);
        void addValue(SFVec2f *vector);

        void insertValue(int index, float x, float y);
        void insertValue(int index, float value[]);
        void insertValue(int index, SFVec2f *vector);
```

```
        void get1Value(int index, float value[]);
        void set1Value(int index, float value[]);
        void set1Value(int index, float x, float y);

        void setValue(MField *mfield);
        void setValue(MFVec2f *vectors);
        void setValue(int size, float vectors[][2]);
};
```

## MFVec3f

```
class MFVec3f : public MField {
        MFVec3f();

        void addValue(float x, float y, float z);
        void addValue(float value[]);
        void addValue(SFVec3f *vector);

        void insertValue(int index, float x, float y, float z);
        void insertValue(int index, float value[]);
        void insertValue(int index, SFVec3f *vector);

        void get1Value(int index, float value[]);
        void set1Value(int index, float value[]);
        void set1Value(int index, float x, float y, float z);

        void setValue(MField *mfield);
        void setValue(MFVec3f *vectors);
        void setValue(int size, float vectors[][3]);
};
```