

# ***Autodesk FBX SDK Overview***

***Version 200512***  
***March 2006***

<b>Document:</b>	Autodesk FBX SDK Overview
<b>Version:</b>	FBX SDK 200512
<b>Date:</b>	March 2006

This publication, ("Publication"), or parts thereof, may not be reproduced in any form, by any method, for any purpose.

AUTODESK MAKES AND YOU RECEIVE NO EXPRESS OR IMPLIED WARRANTIES. AUTODESK SPECIFICALLY DISCLAIMS ANY WARRANTY INCLUDING, WITHOUT LIMITATION, THE IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NON-INFRINGEMENT REGARDING THIS PUBLICATION, AND MAKES THIS PUBLICATION AVAILABLE SOLELY ON AN "AS-IS" BASIS.

IN NO EVENT SHALL AUTODESK OR ITS LICENSORS HAVE ANY LIABILITY FOR ANY INCIDENTAL, SPECIAL, DIRECT, INDIRECT, OR CONSEQUENTIAL DAMAGES, LOSS OF PROFITS, REVENUE, DATA, OR COST OF COVER IN CONNECTION WITH OR ARISING OUT OF THE PURCHASE OR USE OF THIS PUBLICATION. THIS LIMITATION OF LIABILITY SHALL APPLY TO ANY DAMAGES, HOWEVER CAUSED AND REGARDLESS OF THE THEORY OF LIABILITY, WHETHER DERIVED FROM CONTRACT, TORT (INCLUDING, BUT NOT LIMITED TO, NEGLIGENCE), OR OTHERWISE, EVEN IF AUTODESK HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

Autodesk reserves the right to revise and improve its products as it sees fit. This Publication describes the state of this product at the time of its publication, and may not reflect the product at all times in the future.

#### **AUTODESK TRADEMARKS**

The following are registered trademarks of Autodesk, Inc. in the USA and other countries: 3DEC (design/logo), 3December, 3December.com, 3D Studio, 3D Studio MAX, 3D Studio VIZ, 3ds Max, ActiveShapes, Actrix, ADI, AEC-X, Alias, Alias (swirl design/logo), Alias|Wavefront (design/logo), ATC, AUGI, AutoCAD, AutoCAD LT, Autodesk, Autodesk Envision, Autodesk Inventor, Autodesk Map, Autodesk MapGuide, Autodesk Streamline, Autodesk WalkThrough, Autodesk World, AutoLISP, AutoSketch, Backdraft, Bringing information down to earth, Buzzsaw, CAD Overlay, Can You Imagine, Character Studio, Cinepak, Cinepak (logo), Civil 3D, Cleaner, Combustion, Create>what's>Next (design/logo), DesignStudio, DesignStudio (design/logo), Design Your World, Design Your World (design/logo), EditDV, Education by Design, FBX, Filmbox, Gmax, Heidi, HOOPS, i-drop, IntroDV, Kaydara, Kaydara (design/logo), Lustre, Maya, Mechanical Desktop, ObjectARX, Open Reality, PortfolioWall, Powered with Autodesk Technology (logo), ProjectPoint, RadioRay, Reactor, Revit, SketchBook, Visual, Visual Construction, Visual Drainage, Visual Hydro, Visual Landscape, Visual Roads, Visual Survey, Visual Toolbox, Visual Tugboat, Visual LISP, Voice Reality, Volo, WHIP!, and WHIP! (logo).

The following are trademarks of Autodesk, Inc. in the USA and other countries: AutoCAD Learning Assistance, AutoCAD Simulator, AutoCAD SQL Extension, AutoCAD SQL Interface, AutoSnap, AutoTrack, Built with ObjectARX (logo), Burn, CAICE, Cinestream, Cleaner Central, ClearScale, Colour Warper, Content Explorer, Dancing Baby (image), DesignCenter, Design Doctor, Designer's Toolkit, DesignKids, DesignProf, DesignServer, Design Web Format, DWF, DWFit, DWG Linking, DWG TrueConvert, DWG TrueView, DXF, Extending the Design Team, GDx Driver, Gmax (logo), Gmax ready (logo), Heads-up Design, HumanIK, Incinerator, jobnet, LocationLogic, MotionBuilder, ObjectDBX, Plasma, PolarSnap, Productstream, RealDWG, Real-time Roto, Render Queue, StudioTools, Topobase, Toxik, Visual Bridge, Visual Syllabus, and Wiretap.

The following are registered trademarks of Autodesk Canada Co. in the USA and/or Canada and other countries: Discreet, Fire, Flame, Flint, Flint RT, Frost, Glass, Inferno, MountStone, Riot, River, Smoke, Sparks, Stone, Stream, Vapour, Wire.

The following are trademarks of Autodesk Canada Co. in the USA, Canada, and/or other countries: Backburner, Multi-Master Editing.

#### **THIRD-PARTY TRADEMARKS**

All other brand names, product names, or trademarks belong to their respective holders.

OpenGL is a registered trademark of Silicon Graphics, Inc. in the United States and/or other countries worldwide.

#### **THIRD-PARTY COPYRIGHT NOTICES**

Portions relating to JPEG © Copyright 1991-1998 Thomas G. Lane. All rights reserved. This software is based in part on the work of the Independent JPEG Group.

Portions relating to TIFF © Copyright 1997-1998 Sam Leffler. © Copyright 1991-1997 Silicon Graphics, Inc. All rights reserved.

The Tiff portions of this software is provided by the copyright holders and contributors "as is" and any express or implied warranties, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose are disclaimed. In no event shall the copyright owner or contributors of the TIFF portions be liable for any direct, indirect, incidental, special, exemplary, or consequential damages (including, but not limited to, procurement of substitute goods or services; loss of use, data, or profits; or business interruption) however caused and on any theory of liability, whether in contract, strict liability, or tort (including negligence or otherwise) arising in any way out of the use of the TIFF portions of this software, even if advised of the possibility of such damage.

Portions relating to libxml2 are provided under the MIT License. © Copyright 1998-2003 Daniel Veillard. All rights reserved.

#### **GOVERNMENT USE**

This Publication is a "Commercial Item," as that term is defined at 48 C.F.R. §2.101, consisting of "Commercial Computer Software Documentation," as such term is used in 48 C.F.R. §12.212 or 48 C.F.R. §227.7202, as applicable. Consistent with 48 C.F.R. §12.212 and 48 C.F.R. §227.7202-1 through 227.7202-4, as applicable, the Publication is provided to U.S. Government end users (1) only as a Commercial Item, and (2) with only those rights as are granted to all other end users pursuant to the terms and conditions hereof. Manufacturer is Autodesk, Inc. 111 McInnis Parkway, San Rafael, California 94903, USA.

# Contents

## 3 Introduction 1

Welcome to the Autodesk FBX SDK .....	1
FBX SDK Features .....	1
FBX SDK Examples .....	2
ExportScene01 .....	2
ExportScene02 .....	3
ExportScene03 .....	3
ExportScene04 .....	4
ExportScene05 .....	4
ImportScene .....	5
Layers .....	5
User Properties .....	5
ViewScene .....	5

## 4 The FBX SDK Architecture 7

Overview .....	7
Object Management .....	8
Scene Description .....	8
Classes KFCurveNode, KFCurve and KFCurveKey .....	8
Class KFbxConstraint .....	9
Class KFbxGenericNode .....	9
Classes KFbxGeometryBase and KFbxGeometry .....	9
Class KFbxImageConverter .....	11
Class KFbxLayer .....	11
Class KFbxLayerElement .....	11
Class KFbxLight .....	12

## Contents

Class KFbxNode	12
Class Hierarchy Based on KFbxNodeAttribute	13
Class Hierarchy Based on KFbxObject	15
Class KFbxPose	15
Class KFbxScene	16
Class KFbxSceneInfo	18
Class KFbxTakeInfo	18
Class KFbxTakeNode	18
Class KFbxThumbnail	23
Class KFbxUserProperty	23
Scene Import and Export	24
Tools	25
Utilities	25

# 3

## Introduction

### Welcome to the Autodesk FBX SDK

The Autodesk FBX SDK is a C++ software development kit (SDK) that lets you import and export scenes in the *.fbx* file format. Using this SDK, you can create, access, or modify various scene elements such as meshes, parametric surfaces, skeletons, lights, cameras, and animation data.

With the FBX SDK, you can:

- Convert between the *.fbx* format and other proprietary 3D file formats.
- Make a 3D software application FBX-compatible.
- Build tools to batch process scenes.

### FBX SDK Features

The FBX SDK lets you access, create, or modify the following elements of a scene in the *.fbx* file format:

- Mesh, nurb, and patch
- Texture mapping over a mesh, nurb, or patch
- Material mapping over a mesh, nurb, or patch
- Normals and color of vertex, mapping over a mesh, nurb, or patch
- Link constraints on the control points of a mesh, nurb, or patch
- Shape constraints on the control points of a mesh, nurb, or patch
- Position, Rotation, Scale, Parent, Single Chain IK, and Aim constraints.
- Multiple cameras and a camera switcher

## 3 | Introduction

### FBX SDK Examples

- Multiple lights and gobos
- Markers
- Skeleton segments (root, limb, and limb node)
- Multiple takes of animation
- Global camera, light, and time settings
- Bind pose for a list of nodes (bones, mesh, nurbs, patches)
- Rest pose for a list of nodes (bones, mesh, nurbs, patches)

In addition, the FBX SDK provides tools to:

- Process animation data
- Triangulate meshes, nurbs, and patches

The FBX SDK reads *.fbx* files produced by FiLMBOX version 2.5 and later. It writes *.fbx* files compatible with FiLMBOX (version 3.0 and later) and MotionBuilder (up to version 7.x).

The features supported by the FBX SDK are only a subset of the information generated by FiLMBOX and MotionBuilder. Among the unsupported features are FiLMBOX and MotionBuilder tool settings, animation layers, timewarps, and constraints.

## FBX SDK Examples

Examples are a good way to understand the FBX SDK. This distribution includes nine examples which cover most of the features of the FBX SDK.

### ExportScene01

The scene created in this example is a cylinder linked to a skeleton made of three segments. Two takes of animation show the influence of the skeleton segments over the cylinder. ExportScene01 illustrates how to:

- Create a patch

- Create a skeleton segment
- Create a link
- Store the Bind pose
- Store one arbitrary Rest pose
- Create multiple takes of animation
- Create meta-data for a scene
- Export a scene in an *.fbx* file (in ASCII mode)

## **ExportScene02**

The scene created in this example is a sphere morphed by two shapes. A take of animation shows the influence of the shapes over the sphere. ExportScene02 illustrates how to:

- Create a nurb
- Map a shape over a nurb
- Map a texture over a nurb
- Map a material over a nurb
- Create a take of animation
- Export a scene in an *.fbx* file

## **ExportScene03**

The scene created in this example is a textured cube and a pyramid with materials mapped on its faces. A take of animation displays six different angles of both models. ExportScene03 illustrates how to:

- Create a mesh
- Map a texture over a mesh
- Map a material over a mesh
- Map a shape over a mesh
- Create a take of animation

- Export a scene in an *.fbx* file

## **ExportScene04**

The scene created in this example is a group of lights, a marker and two cameras. A take of animation rotates the lights and moves the cameras around. ExportScene04 illustrates how to:

- Create lights and assign a gobo
- Set global light settings
- Create a marker
- Create a camera and link it to a point of interest
- Create a take of animation
- Create camera switcher animation
- Export a scene in an *.fbx* file

## **ExportScene05**

The scene created in this example is a skeleton made of three segments. The position of a node in an *.fbx* file is expressed in coordinates relative to its parent. This example shows how to convert to and from a global position. ExportScene05 illustrates how to:

- Create a skeleton segment
- Get the global default position of a node
- Set the global default position of a node
- Set limits, rotation order, and pre/post pivots
- Use a custom memory allocator
- Export a scene in an *.fbx* file



## ImportScene

This example illustrates how to get the version number of an *.fbx* file, how to detect if an *.fbx* file is password protected, how to import a scene, and how to browse a scene to access node and animation information. The `ImportScene` example displays the version number and the content of the *.fbx* file whose name is passed as the program argument. You can try it with the various *.fbx* files created from the export examples.

## Layers

The scene created in this example is a cube with layered textures and with materials mapped on its faces. The example illustrates how to use the layers.

## User Properties

This example illustrates how to create customized properties, attach them to a simple cube, and how to create a constraint using these customized properties.

## ViewScene

This example illustrates how to display the content of an *.fbx* file in a graphical window. This program is based on the OpenGL® Utility Toolkit (GLUT). A menu is provided to select the current camera and the current take. Among other things, the `ViewScene` example illustrates how to:

- Import a scene from *.fbx*, *.obj*, *.dxf*, *.3ds*, and *.dae* (Collada) files
- Convert the nurbs and patch attribute types of a scene into mesh node attributes
- Get the list of all the cameras in the scene
- Find the current camera

### 3 | Introduction

#### FBX SDK Examples

- Get the relevant settings of a camera depending on its projection type and aperture mode
- Compute the local and global positions of a node
- Compute the orientation of a camera
- Compute the orientation of a light
- Compute the shape deformation of mesh vertices
- Compute the link deformation of mesh vertices
- Get a list of all poses in the scene
- Display the scene with a specific pose

# 4

## The FBX SDK Architecture

### Overview

The FBX SDK set of classes can be divided into five categories:

- **Object management:**  
Refers to object instantiation and garbage collecting. See "Object Management" on page 8.
- **Scene description:**  
These classes are the central part of the FBX SDK. They cover both the static description of the elements of a scene and the animation data modifying these elements over time. See "Scene Description" on page 8.
- **Scene import and export:**  
These classes are the bridge between scene description classes and the *.fbx* file format. See "Scene Import and Export" on page 24.
- **Tools:**  
These classes provide services to process animation data and triangulate meshes, nurbs, and patches. See "Tools" on page 25.
- **Utilities:**  
These classes comprise array templates, string, time, error handling, color, and various match-oriented classes. See "Utilities" on page 25.

## Object Management

Object management refers to object instantiation and garbage collecting. Class `KFbxSdkManager` is used to instantiate and destroy most object types. It performs garbage collecting when deleted.

## Scene Description

The FBX SDK holds a scene description in a tree structure. The position of each node is expressed in coordinates relative to its parent.

Every node has an attribute and an array of takes of animation data. The attribute describes the content of each node. Each take of animation data describes modifications to the state of the node over time.

## Classes `KFCurveNode`, `KFCurve` and `KFCurveKey`

Class `KFCurveNode` is the basic element of a set of animation channels. Below, figure 4-1 shows the composition of class `KFCurveNode`.

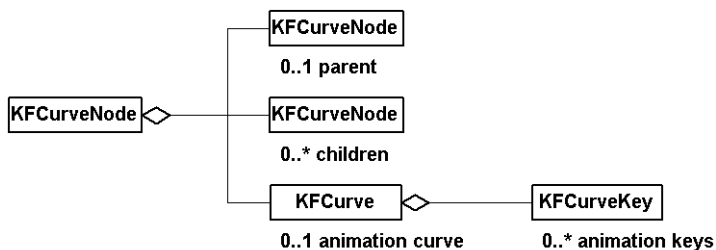


figure 4-1: Composition of class `KFCurveNode`

Class `KFCurveNode` has the interface to support a tree structure through a parent and an array of children. An associated animation curve (`KFCurve` object) defines the animation data. This animation data is made of multiple keys (`KFCurveKey` objects), each one made of a time and value pair.

## Class `KFbxConstraint`

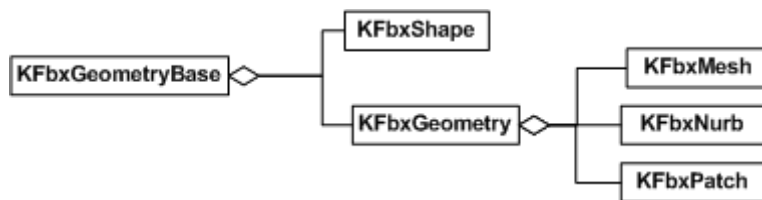
Class `KFbxConstraint` defines a constraint. This class is composed of `KFbxConstraintAim`, `KFbxConstraintParent`, `KFbxConstraintPosition`, `KFbxConstraintRotation`, `KFbxConstraintScale`, and `KFbxConstraintSingleChainIK`.

## Class `KFbxGenericNode`

Use class `KFbxGenericNode` to extend the FBX SDK with user-defined nodes. The generic node is an empty node that users can configure by adding user-defined properties (`KFbxUserProperty`).

## Classes `KFbxGeometryBase` and `KFbxGeometry`

Class `KFbxGeometryBase` is the base class for all types of node attributes made of control points. Below, figure 4-2 shows the composition of class `KFbxGeometryBase`.



*figure 4-2: Composition of class `KFbxGeometryBase`*

## Class `KFbxTexture`

Class `KFbxTexture` defines the mapping of an image over a surface.

## 4 | The FBX SDK Architecture

### Scene Description

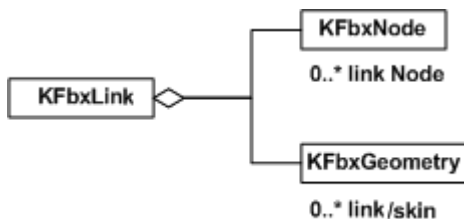
Each texture within a scene must have a unique name. See "Class Hierarchy Based on KFbxObject" on page 15. See "Scene Import and Export" on page 24 for details on the file referenced by a texture.

### Class KFbxMaterial

Class KFbxMaterial defines the various color rendering parameters of a surface. Each material within a scene must have a unique name. See "Class Hierarchy Based on KFbxObject" on page 15.

### Class KFbxLink

Class KFbxLink defines the geometrical link between nodes. A link consists of a link node (also called cluster or bone) and is set to instances of KFbxGeometry.



*figure 4-3: Composition of class KFbxLink*

### Class KFbxShape

Class KFbxShape defines a deformation on a set of control points. Shapes should not be shared between many node attributes of type KFbxGeometry because they are closely related to the set of control points with which they are associated.

### Class KFbxImageConverter

Class KFbxImageConverter is used to provide the SDK with image conversion routines. Conversion routines are used to fill a raw data buffer from a file of a specific format (FileToBuffer), or to write a raw data buffer to a specific file format (BufferToFile). Use the KFbxImageConverter to embed texture media in a portable manner.

### Class KFbxLayer

Class KFbxLayer provides the base for the layering mechanism. A layer can contain one or more of the following layer elements:

- Normals
- UVs
- Textures
- Materials
- Polygon Groups
- Vertex Colors

In most cases, a single layer is sufficient to describe a geometry. Many applications will only support what is defined on the first layer (Layer 0).

### Class KFbxLayerElement

Class KFbxLayerElement is the base class for Layer Elements. It describes how a Layer Element is mapped on a geometry surface and how the mapping information is arranged in memory. Layer Elements are then added to a Layer, starting at Layer 0. Layer Elements exist for the following types:

- KFbxLayerElementNormal
- KFbxLayerElementUV
- KFbxLayerElementTexture

## 4 | The FBX SDK Architecture

### Scene Description

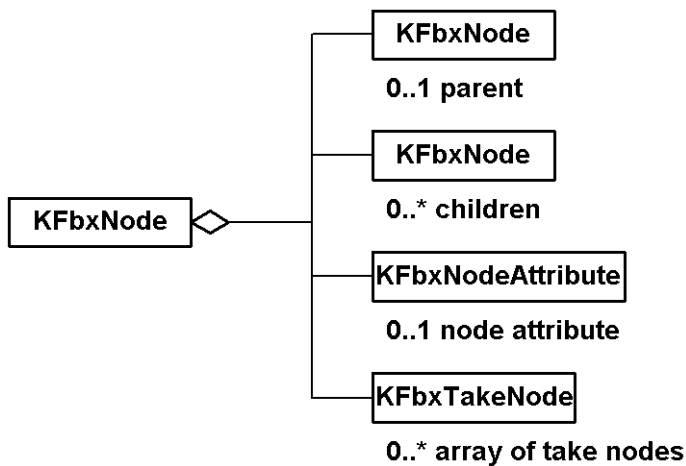
- KFbxLayerElementMaterial
- KFbxLayerElementPolygonGroup
- KFbxLayerElementVertexColor

### Class KFbxLight

Class KFbxLight defines the parameters of a light source. If the light source is a spotlight, you can specify a gobo. See "Scene Import and Export" on page 24 for details on the file referenced by a gobo.

### Class KFbxNode

Class KFbxNode is the basic element of a scene description. See figure 4-4, which shows the composition of class KFbxNode.



**figure 4-4:** Composition of class KFbxNode



Class `KFbxNode` has the interface to support a tree structure through a parent and an array of children. Duplicate node names are supported, but they are not clean. The user should provide a renaming strategy using `kFbxRenamingStrategy`.

An associated node attribute defines the content of a node. A node attribute can't be shared between nodes. If no node attribute is set, the node attribute is of type null. A null node attribute can be used as a reference point in a scene. Node attribute types are defined under "Class Hierarchy Based on `KFbxNodeAttribute`" on page 13.

The array of take nodes contains animation data associated with the node. See "Take Management" on page 23 for more information.

## Class Hierarchy Based on `KFbxNodeAttribute`

Class `KFbxNodeAttribute` is the base class for all types of node attributes. See figure 4-5, which shows the class hierarchy based on `KFbxNodeAttribute`.

The types of node attributes are the following:

- Class `KFbxCamera` defines the parameters of a camera.

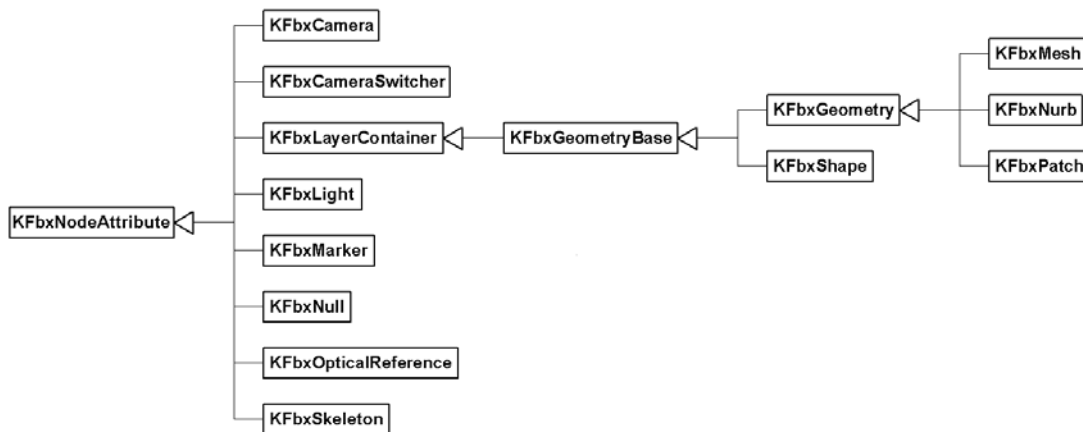


figure 4-5: Class hierarchy based on `KFbxNodeAttribute`

## 4 | The FBX SDK Architecture

### Scene Description

- Class `KFbxCameraSwitcher` defines the parameters of a camera switcher. Only one node attribute of this type is allowed per scene. It is owned by the camera switcher node contained in class `KFbxScene`. See "Class `KFbxScene`" on page 16.
- Class `KFbxLayerContainer` is the base class that contains layers of materials, textures, normals, and UV coordinates. Class `KFbxLayerContainer` defines an element that manages layers and contains a `KFbxGeometryBase` that holds geometry and shapes. The geometry (`kFbxGeometry`) is split into three classes. Class `KFbxMesh` defines an element made of polygons. Classes `KFbxNurb` and `KFbxPatch` define types of parametric surfaces. See "Classes `KFbxGeometryBase` and `KFbxGeometry`" on page 9 for more information.
- Class `KFbxLight` defines the parameters of a light source. See "Class `KFbxLight`" on page 12 for more on lights.
- Class `KFbxMarker` defines a reference point as a cube.
- Class `KFbxNull` defines a reference point as a null.
- Class `KFbxOpticalReference` defines a reference for optical segments.
- Class `KFbxSkeleton` defines the following skeleton segments: root, limb, and limb node.

Some of the above node attributes add additional animated parameters to a node. See "Animation Channels" on page 18 for more information.

## Class Hierarchy Based on KFbxObject

Classes derived from KFbxObject are checked for name duplicates within a scene. See figure 4-6, which shows the class hierarchy based on KFbxObject.

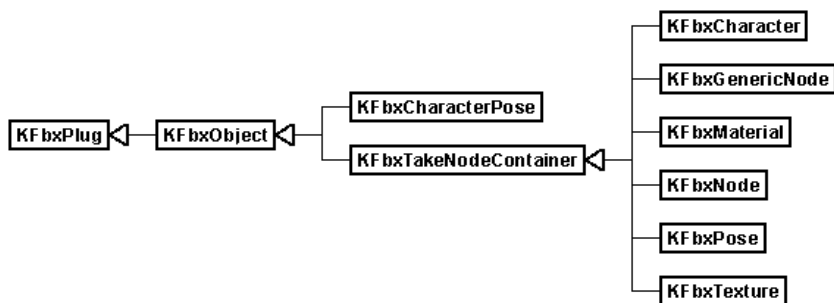


figure 4-6: Class hierarchy based on KFbxObject

## Class KFbxPose

Class KFbxPose defines the nodes that are part of a pose and their transformation matrix.

There are two types of poses:

- Bind poses
- Rest poses

### Bind Pose

A Bind pose holds the transformation matrix of all nodes implied in a link deformation. This includes the geometry being deformed, the links deforming the geometry, and recursively all the link's parent nodes.

The Bind pose gives you the transformation of the nodes at the moment of the binding operation when no deformations occur. This is useful when you are re-creating deformable models in third party applications.

## Rest Pose

A Rest pose is a snapshot of node transformation. Use a Rest pose to store the position of every node of a character at a certain point in time.

You can then use this pose as a reference position for animation tasks, for example, editing walk cycles.

## Class KFbxScene

Class KFbxScene contains a complete scene description. The composition of class KFbxScene is shown in figure 4-7.

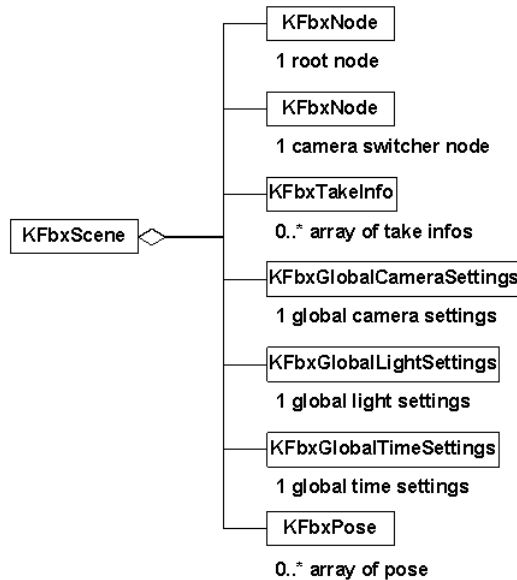


figure 4-7: Composition of class KFbxScene

The root node is the node under which the hierarchy of 3D elements is built. This node is not saved. So, if a global transformation must be applied, insert a node between the provided root node and its children.

Cyclic graphs are not allowed. A node cannot be shared between scenes.

Each node inserted in the tree below the root node must have a unique name. See "Class Hierarchy Based on KFBXObject" on page 15 for more details on this restriction.

The camera switcher node has a node attribute of type KFBxCameraSwitcher. See "Class Hierarchy Based on KFBxNodeAttribute" on page 13 for more details on this type of node attribute. The camera switcher is used to set camera switching animation data. This node is not saved if the scene contains no camera. Nodes inserted below are never saved.

The array of take information contains global data for all imported takes. By default, a newly created take does not have take information. It must be set explicitly. An array of take information is available in class KFBxImporter before importing a scene to allow selection of imported takes. See "Scene Import and Export" on page 24.

For more information on take management services supported by class KFBxScene, see "Take Management" on page 23.

Global camera settings comprise a default camera, a viewing mode and producer cameras. Global light settings comprise an ambient color, fog options and shadow planes. Global time settings comprise various time format options and time markers.

Poses can be stored in the scene for a group of nodes. Rest poses and Bind poses are supported.

See "Class KFBxPose" on page 15 for a description of poses.

## Class KFbxSceneInfo

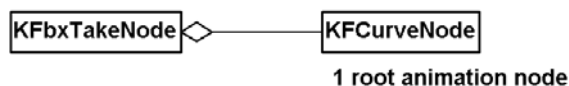
Class KFbxSceneInfo holds the user-defined summary information for a scene. This includes the scene title, the author, comments, search keywords, and so on. This information can then be searched to classify the file using user-defined criteria.

## Class KFbxTakeInfo

Class KFbxTakeInfo contains information about a take. This information includes take statistics like the animation interval and reference time, along with user-defined information like the take name, a take description, and a thumbnail image for visual reference.

## Class KFbxTakeNode

Class KFbxTakeNode contains animation data associated with a take. See figure 4-8, which shows the composition of class KFbxTakeNode.



**figure 4-8:** Composition of class KFbxTakeNode

## Animation Channels

Class KFbxTakeNode is a wrapper to a KFbxCurveNode object. It provides an easy way to directly access the animation channels found in the KFbxCurveNode object internal tree structure. See "Classes KFbxCurveNode, KFbxCurve and KFbxCurveKey" on page 8 for more details.

This section lists the default animation channels and additional node attribute types that have special animation channels. For more information on the various node attribute types, see "Class Hierarchy Based on KFbxCurveNodeAttribute" on page 13.

### Default Channels

By default, a take node has the following animation channels:

- Translation X
- Translation Y
- Translation Z
- Rotation X
- Rotation Y
- Rotation Z
- Scaling X
- Scaling Y
- Scaling Z
- Visibility

### KFbxCamera Channels

If the take node is part of a node with a node attribute of type KFbxCamera, the following additional channels are available:

- Field of view X
- Field of view Y
- Focal length
- Optical center X
- Optical center Y
- Roll
- Turntable
- Background color red
- Background color green
- Background color blue

## 4 | The FBX SDK Architecture

### Scene Description

#### **KFbxCameraSwitcher Channels**

If the take node is part of the camera switcher node found in class KFbxScene, a camera index channel is available.

Camera indices start at 1. Out of range indices are clamped between 1 and the number of cameras in the scene. The index of a camera refers to its order of appearance when searching the scene's node tree depth first. See "Class KFbxScene" on page 16 for more details on the camera switcher node.

#### **KFbxConstraint Channels**

If the take node is part of KFbxTexture, the following additional channels are available:

- Object Weight
- Position Offset (X,Y,Z)
- Rotation Offset (X,Y,Z)
- Scale Offset (X,Y,Z)
- Parent Rotation Offset (X,Y,Z)
- Parent Translation Offset (X,Y,Z)
- Aim Offset (X,Y,Z)
- Aim WorldUp Vector (X,Y,Z)
- Aim Vector (X,Y,Z)
- Aim Up Vector (X,Y,Z)
- SCik weight
- SCik twist
- SCik Pole Vector (X,Y,Z)



### **KFbxGeometry Channels**

If the take node is part of a node with a node attribute of type KFbxGeometry, one additional channel per shape is available. Shape channels can also be accessed through the interface of class KFbxGeometry. See "Class KFbxShape" on page 10 for more details.

### **KFbxMarker Channels**

If the take node is part of a node with a node attribute of type KFbxMarker, the following additional channels are available:

- Occlusion
- IK Reach Translation
- IK Reach Rotation

### **KFbxMaterial Channels**

If the take node is part of KFbxMaterial, the following additional channels are available:

- Emissive color red
- Emissive color green
- Emissive color blue
- Ambient color red
- Ambient color green
- Ambient color blue
- Diffuse color red
- Diffuse color green
- Diffuse color blue
- Specular color red
- Specular color green
- Specular color blue
- Opacity

## 4 | The FBX SDK Architecture

### Scene Description

- Reflectivity
- Shininess

#### **KFbxLight Channels**

If the take node is part of a node with a node attribute of type KFbxLight, the following additional channels are available:

- Color red
- Color green
- Color blue
- Intensity
- Cone angle
- Fog

#### **KFbxTexture Channels**

If the take node is part of KFbxTexture, the following additional channels are available:

- Texture Translation X
- Texture Translation Y
- Texture Translation Z
- Texture Rotation X
- Texture Rotation Y
- Texture Rotation Z
- Texture Scale X
- Texture Scale Y
- Texture Scale Z
- Alpha

## KFbxUserProperty Channels

If the take node is part of KFbxUserProperty, animation channels can be accessed using the property index.

## Take Management

Take management is divided between the interfaces of classes KFbxScene and KFbxNode as follows:

- All take-related functions in the class KFbxScene interface are applied to all nodes in the node tree under the scene's root node. Take-related functions in class KFbxNode interface are applied to the node and, if specified, recursively to its children.
- Class KFbxNode has an array of take nodes. The first element contains the default values for the animated parameters. No other animation data should be added to this element. A take node is not necessary for every existing take. The default take node is used when there is no take node associated with the current take.

## Class KFbxThumbnail

The thumbnail is a small bit map icon representing the content of the FBX file and is helpful when browsing the files.

## Class KFbxUserProperty

The FBX SDK allows the management of user properties, which can be used to extend objects according to the needs of the user. They are limited to the following basic types:

eBOOL	A Boolean value (true or false).
eREAL	A floating point value (double).
eCOLOR	An R,G,B,A structure.
eINTEGER	An integer number.

## 4 | The FBX SDK Architecture

### Scene Import and Export

eVECTOR	An X,Y,Z,W structure of double values.
eLIST	A list of strings.

## Scene Import and Export

Scene import and export classes are the bridge between scene description classes and the *.fbx* file format.

Class `KFbxIO` is the virtual base class of `KFbxImporter` and `KFbxExporter`.

Class `KFbxImporter` loads a scene from an *.fbx* file. Options are available to select the scene elements to import. It is possible to access an array of take information between initialization and import to select imported takes. See "Class `KFbxScene`" on page 16 for more information.

Class `KFbxExporter` saves a scene to an *.fbx* file. Options are available to select the scene elements to export and whether or not files referenced by textures are embedded. By default, files referenced by textures are not embedded in the *.fbx* file. Instead, their absolute and relative paths are saved. The referenced files must be provided along with the *.fbx* file to keep the scene complete.

Both the `KFbxImporter` and `KFbxExporter` classes support password protection.

File versioning was new in the FBX SDK 3.6.1. Function `KFbxIO::GetCurrentVersion()` has been added to get the FBX version number written in files created by the FBX SDK.

Function `KFbxImporter::GetFileVersion()` has been added to get the FBX version number of *.fbx* files read by the FBX SDK. All *.fbx* files created before the introduction of this feature are assigned a version number of 5.0.0.

## Tools

Tool classes provide services to process animation data and triangulate meshes, nurbs, and patches. These transformations are often required to convert between different methods of describing a scene.

Class `KFCurveUtils` provides services related to tangent mode, interpolation, resampling, merging, and time offset.

Class `KFbxGeometryConverter` triangulates meshes, nurbs, and patches. It also converts patches into nurbs.

## Utilities

Utility classes are made available to FBX SDK users to ease the integration of the FBX SDK with other APIs.

Classes `KArrayUL`, `KArrayTemplate`, and `KStructArrayTemplate` are dynamically growing arrays. `KArrayUL` contains unsigned long elements, `KArrayTemplate` is designed to contain typed pointers, and `KStructArrayTemplate` is designed to contain structures.

Class `KString` encapsulates a dynamically allocated character array.

Classes `KTime` and `KTimeSpan` provide an accurate time format.

Class `KError` eases error handling.

Class `KFbxColor` holds RGBA values.

Class `KFbxQuaternion` contains a rotation expressed in quaternion coordinates.

Class `KFbxMatrix` contains a 4x4 matrix.

Class `KFbxMemoryAllocator` provides services for memory management.

## 4 | The FBX SDK Architecture

### Utilities

Class `KFbxRenamingStrategy` provides utilities for renaming all objects in a scene, resolving name clashing and illegal characters, managing namespaces, and keeping backward compatibility.

Classes `KFbxVector2` and `KFbxVector4` hold an array of 2 or 4 elements respectively.

Class `KFbxXMatrix` contains a 4x4 affine matrix.