INTERNATIONAL STANDARD

ISO/IEC 14496-16

Second edition
2006-12-15

Information technology — Coding of audio-visual objects —

Part 16:
Animation Framework eXtension (AFX)

Technologies de l'information — Codage des objets audiovisuels —
Partie 16: Extension du cadre d'animation (AFX)

Reference number
ISO/IEC 14496-16:2006(E)

PDF disclaimer

This PDF file may contain embedded typefaces. In accordance with Adobe's licensing policy, this file may be printed or viewed but shall not be edited unless the typefaces which are embedded are licensed to and installed on the computer performing the editing. In downloading this file, parties accept therein the responsibility of not infringing Adobe's licensing policy. The ISO Central Secretariat accepts no liability in this area.

Adobe is a trademark of Adobe Systems Incorporated.

Details of the software products used to create this PDF file can be found in the General Info relative to the file; the PDF-creation parameters were optimized for printing. Every care has been taken to ensure that the file is suitable for use by ISO member bodies. In the unlikely event that a problem relating to it is found, please inform the Central Secretariat at the address given below.

iTeh STANDARD PREVIEW
(standards.iteh.ai)

ISO/IEC 14496-16:2006
https://standards.iteh.ai/catalog/standards/sist/af8296a4-4d8c-4210-b666-59d778efe4a4/iso-iec-14496-16-2006

© ISO/IEC 2006

All rights reserved. Unless otherwise specified, no part of this publication may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying and microfilm, without permission in writing from either ISO at the address below or ISO's member body in the country of the requester.

ISO copyright office
Case postale 56 • CH-1211 Geneva 20
Tel. + 41 22 749 01 11
Fax + 41 22 749 09 47
E-mail copyright@iso.org
Web www.iso.org

Published in Switzerland

Contents

Page

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work. In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1.

International Standards are drafted in accordance with the rules given in the ISO/IEC Directives, Part 2.

The main task of the joint technical committee is to prepare International Standards. Draft International Standards adopted by the joint technical committee are circulated to national bodies for voting. Publication as an International Standard requires approval by at least 75 % of the national bodies casting a vote.

ISO/IEC 14496-16 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 29, *Coding of audio, picture, multimedia and hypermedia information*.

This second edition cancels and replaces the first edition (ISO/IEC 14496-16:2004), which has been technically revised. It also incorporates the amendment ISO/IEC 14496-16:2004/Amd.1:2006 and the Technical Corrigenda ISO/IEC 14496-16:2004/Cor.1:2005 and ISO/IEC 14496-16:2004/Cor.2:2005.

ISO/IEC 14496 consists of the following parts, under the general title *Information technology — Coding of audio-visual objects*:

- *Part 1: Systems*
- *Part 2: Visual*
- *Part 3: Audio*
- *Part 4: Conformance testing*
- *Part 5: Reference software*
- *Part 6: Delivery Multimedia Integration Framework (DMIF)*
- *Part 7: Optimized reference software for coding of audio-visual objects* [Technical Report]
- *Part 8: Carriage of ISO/IEC 14496 contents over IP networks*
- *Part 9: Reference hardware description* [Technical Report]
- *Part 10: Advanced Video Coding*
- *Part 11: Scene description and application engine*
- *Part 12: ISO base media file format*
- *Part 13: Intellectual Property Management and Protection (IPMP) extensions*
- *Part 14: MP4 file format*
- *Part 15: Advanced Video Coding (AVC) file format*
- *Part 16: Animation Framework eXtension (AFX)*
- *Part 17: Streaming text format*
- *Part 18: Font compression and streaming*

- *Part 19: Synthesized texture stream*
- *Part 20: Lightweight Application Scene Representation (LASeR) and Simple Aggregation Format (SAF)*
- *Part 21: MPEG-J GFX*
- *Part 22: Open Font Format*

Introduction

The International Organization for Standardization (ISO) and International Electrotechnical Commission (IEC) draw attention to the fact that it is claimed that compliance with this document may involve the use of patents.

The ISO and IEC take no position concerning the evidence, validity and scope of these patent rights.

The holders of these patent rights have assured the ISO and IEC that they are willing to negotiate licences under reasonable and non-discriminatory terms and conditions with applicants throughout the world. In this respect, the statement of the holder of this patent right is registered with the ISO and IEC. Information may be obtained from the companies listed in Annex G.

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights other than those identified in Annex G. ISO and IEC shall not be held responsible for identifying any or all such patent rights.

Information technology — Coding of audio-visual objects —

Part 16:
Animation Framework eXtension (AFX)

1 Scope

This International Standard specifies MPEG-4 Animation Framework eXtension (AFX) model for creating interactive multimedia contents by composing natural and synthetic objects. Within this model, MPEG-4 is extended with higher-level synthetic objects for geometry, texture, and animation as well as dedicated compressed representations.

AFX also specifies a backchannel for progressive streaming of view-dependent information.

2 Normative references

The following referenced documents are indispensable for the application of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

ISO/IEC 14496-1:2004, *Information technology — Coding of audio-visual objects — Part 1: Systems*

ISO/IEC 14496-2:2004, *Information technology — Coding of audio-visual objects — Part 2: Visual*

ISO/IEC 14496-11:2005, *Information technology — Coding of audio-visual objects — Part 11: Scene description and application engine*

3 Symbols and abbreviated terms

List of symbols and abbreviated terms.

AFX	Animation Framework eXtension
BIFS	BInary Format for Scene
DIBR	Depth-Image Based Representation
ES	Elementary Stream
IBR	Image-Based Rendering
NDT	Node Data Type
OD	Object Descriptor
VRML	Virtual Reality Modelling Language

4 Animation Framework eXtension (AFX)

4.1 Introduction

> *"Most people think the word 'animation' means movement.*
> *But it doesn't. It comes from 'animus' which means 'life or to live'.*
> *Making it move is not animation, but just the mechanics of it."*
> Frank Thomas and Ollie Johnston
> *Disney Animation: the illusion of life*, 1981

MPEG-4, ISO/IEC 14496, is a multimedia standard that enables composition of multiple audio-visual objects on a terminal. Audio and visual objects can come from *natural* sources (e.g. a microphone, a camera) or from *synthetic* ones (i.e. made by a computer); each source is called a *media* or a *stream*. On their terminals, users can display, play, and interact with MPEG-4 audio-visual contents, which can be downloaded previously or streamed from remote servers. Moreover, each object may be protected to ensure a user has the right credentials before downloading and displaying it.

Unlike natural audio and video objects, computer graphics objects are purely synthetic. Mixing computer graphics objects with traditional audio and video enables augmented reality applications, i.e. applications mixing natural and synthetic objects. Examples of such contents range from DVD menus, and TV's Electronic Programming Guides to medical and training applications, games, and so on.

Like other computer graphics specifications, MPEG-4 synthetic objects are organized in a *scene graph* based on VRML97 [1], which is a direct acyclic tree where *nodes* represent objects and branches their properties, called *fields*. As each object can receive and emit events, two branches can be connected by the means of a *route*, which propagates events from one field of one node to another field of another node. As any other MPEG-4 media, scenes may receive updates from a server that modify the topology of the scene graph.

The Animation Framework eXtension (AFX) proposes a conceptual organization of synthetic models for computer animations as well as specific compression schemes; models defined in ISO/IEC 14496-11 and in this part fit in organization (see Figure 2).

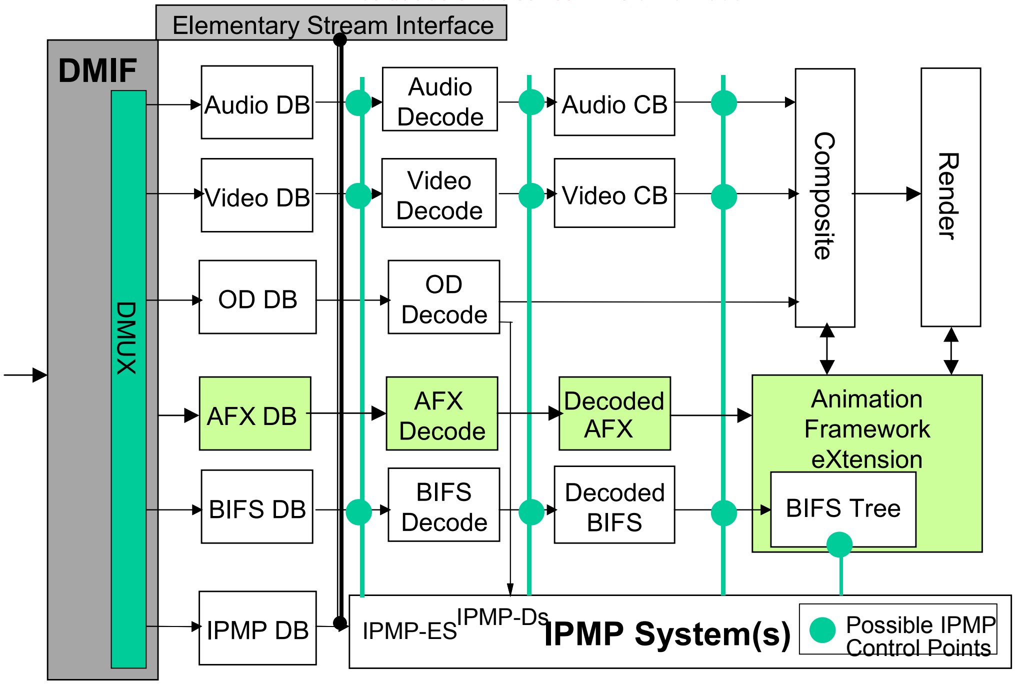

Figure 1 — Animation Framework and MPEG-4 Systems.

Figure 1 shows the position of the Animation Framework within MPEG-4 Terminal architecture. It extends the existing BIFS tree with new tools and define the AFX streams that carry dedicated compressed object representations and a backchannel for view-dependent features.

4.2 The AFX model

To understand the AFX model, let's take an example. Suppose one wants to build an avatar. The avatar consists of geometry elements that describe its legs, arms, head and so on. Simple geometric elements can be used and deformed to produce more physically realistic geometry. Then, skin, hair, cloths are added. These may be physic-based models attached to the geometry. Whenever the geometry is deformed, these models deform and thanks to their physics, they may produce wrinkles. Biomechanical models are used for motion, collision response, and so on. Finally, our avatar may exhibit special behaviors when it encounters objects in its world. It might also learn from experiences: for example, if it touches a hot surface and gets hurt, next time, it will avoid touching it.

This hierarchy also works in a top to bottom manner: if it touches a hot surface, its behavior may be to retract its hand. Retracting its hand follows a biomechanical pattern. The speed of the movement is based on the physical property of its hand linked to the rest of its body, which in turn modify geometric properties that define the hand.

AFX defines 6 groups of components, following [38].

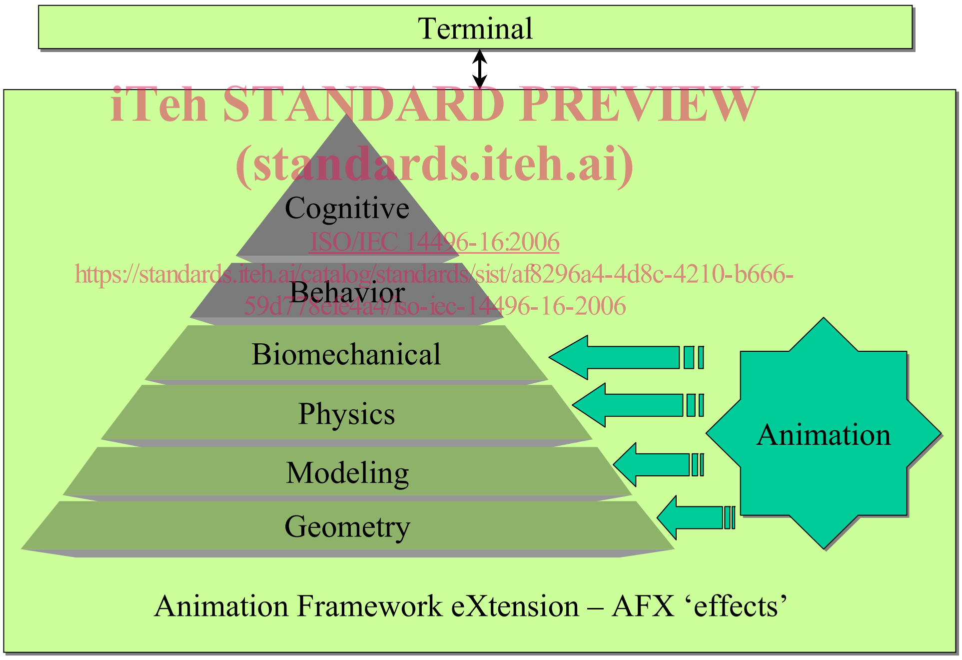

Figure 2 — Models in computer games and animation [38].

a) *Geometric component*. These models capture the form and appearance of an object. Many characters in animations and games can be quite efficiently controlled at this low-level. Due to the predictable nature of motion, building higher-level models for characters that are controlled at the geometric level is generally much simpler.

b) *Modeling component*. These models are an extension of geometric models and add linear and non-linear deformations to them. They capture transformation of the models without changing its original shape. Animations can be made on changing the deformation parameters independently of the geometric models.

c) *Physical component*. These models capture additional aspects of the world such as an object’s mass inertia, and how it responds to forces such as gravity. The use of physical models allows many motions to be created automatically and with unparallel realism.

d) *Biomechanical component*. Real animals have muscles that they use to exert forces and torques on their own bodies. If we already have built physical models of characters, they can use virtual muscles to move themselves around. These models have their roots in control theory.

e) *Behavioral component*. A character may expose a *reactive behavior* when its behavior is solely based on its perception of the current situation (i.e. no memory of previous situations). *Goal-directed behaviors* can be used to define a cognitive character's goals. They can also be used to model *flocking behaviors*.

f) *Cognitive component*. If the character is able to learn from stimuli from the world, it may be able to adapt its behavior. These models are related to artificial intelligence techniques.

AFX specification currently deals with the first four categories. Models of the last two categories are typically application-specific and often designed programmatically. Similarly, while the first four categories can be animated using existing tools such as interpolators, the last two categories have their own logic and cannot be animated the same way.

In each category, one can find many models for any applications: from simple models that require little processing power (low-level models) to more complex models (high-level models) that require more computations by the terminal. VRML [1] and BIFS specifications provide low-level models that belong to the Geometry component with the exception of Face and Body Animation tools that belong to the Biomechanical component.

Higher-level components can be defined as providing a compact representation of functionality in a more abstract manner. Typically, this abstraction leads to mathematical models that need few parameters. These models cannot be rendered directly by a graphic card: internally, they are converted to low-level primitives a graphic card can render. Besides a more compact representation, this abstraction often provides other functionalities such as but not limited to compact representation, view-dependent subdivision, automatic level-of-details, smoother graphical representation , scalability across terminals, and progressive local refinements.

For example, a subdivision surface can be subdivided based on the area viewed by the user. For animations, piecewise-linear interpolators require few computations but require lots of data in order to represent a curved path. Higher-level animation models represent animation using piecewise-spline interpolators with less values and provide more control over the animation path and timeline.

In the remaining of this document, this conceptual organization of tools is followed in the same spirit an author will create content: the geometry is first defined with or without solid modeling tools, then texture is added to it. Objects can be deformed and animated using modeling and animation tools. Finally, avatars need biomechanical tools. Behavioral and Cognitive models can be programmatically implemented using JavaScript or MPEG-J defined in ISO/IEC 14496-11.

NOTE

Some generic tools developed originally within the Animation Framework eXtension have been relocated in ISO/IEC 14496-11/AMD1 along with other generic tools. This includes:

- Spline-based generic animation tools, called Animator nodes;
- Optimized interpolator compression tools;
- BitWrapper node that enables compressed representation of exisiting nodes;
- Procedural textures based on fractal plasma.

4.3 Geometry tools

4.3.1 Introduction

Geometry tools consist of the following technologies:

- NURBS, which consists of the following nodes: **NurbsSurface**, **NurbsCurve**, and **NurbsCurve2D**;
- Subdivision surfaces consisting of the following nodes: **SubdivisionSurface**, **SubdivSurfaceSector** and **WaveletSubdivisionSurface** of which the latter enables to add wavelet-encoded details at different resolutions to subdivision surfaces;
- MeshGrid, which consists of the **MeshGrid** node;

4.3.2 Non-Uniform Rational B-Spline (NURBS)

4.3.2.1 Introduction

A Non-Uniform Rational B-Spline (NURBS) curve of degree $p > 0$ (and hence of order $p+1$) and control points $\{\mathbf{P}_i\}$ $(0 \le i \le n-1;\ n \ge p+1)$ has for Equation 1 [37], [63], [88]:

$$\mathbf{C}(u) = \sum_{i=0}^{n-1} R_{i,p}(u)\mathbf{P}_i = \frac{\sum_{i=0}^{n-1} N_{i,p}(u) w_i \mathbf{P}_i}{\sum_{i=0}^{n-1} N_{i,p}(u) w_i}$$

Equation 1 — NURBS curve definition.

The parameter $u \in [0, 1]$ allows to travel along the curve and $\{R_{i,p}\}$ are its rational basis functions. The latter can in turn be expressed in terms of some positive (and not all null) weights $\{w_i\}$, and $\{N_{i,p}\}$, the p^{th}-degree B-Spline basis functions defined on the possibly non-uniform, but always non-decreasing knot sequence/vector of length $m = n + p+1$: $U = \{u_0, u_1, \ldots, u_{m-1}\}$, where $0 \le u_i \le u_{i+1} \le 1\ \forall\ 0 \le i \le m-2$.

The B-Spline basis functions are defined recursively using the Cox de Boor formula:

$$\begin{aligned} N_{i,0}(u) &= \begin{cases} 1 & \text{if } u_i \le u < u_{i+1}; \\ 0 & \text{otherwise;} \end{cases} \\ N_{i,p}(u) &= \frac{u - u_i}{u_{i+p} - u_i} N_{i,p-1}(u) + \frac{u_{i+p+1} - u}{u_{i+p+1} - u_{i+1}} N_{i+1,p-1}(u). \end{aligned}$$

If $u_i = u_{i+1} = \ldots = u_{i+k-1}$, it is said that u_i has multiplicity k. $\mathbf{C}(u)$ is infinitely differentiable inside knot spans, *i.e.*, for $u_i < u < u_{i+1}$, but only $p-k$ times differentiable at the parameter value corresponding to a knot of multiplicity k, so setting several consecutive knots to the same value u_j decreases the smoothness of the curve at u_j. In general, knots cannot have multiplicity greater than p, but the first and/or last knot of U can have multiplicity $p+1$, *i.e.*, $u_0 = \ldots = u_p = 0$ and/or $u_{m-p-1} = \ldots = u_{m-1} = 1$, which causes $\mathbf{C}$ to interpolate the corresponding endpoint(s) of the control polygon defined by $\{\mathbf{P}_i\}$, *i.e.*, $\mathbf{C}(0) = \mathbf{P}_0$ and/or $\mathbf{C}(1) = \mathbf{P}_{n-1}$. Therefore, a knot vector of the kind $U = \left\{ \underbrace{u_0 = 0, \ldots, 0}_{p+1}, u_{p+1}, \ldots, u_{m-p-2}, \underbrace{1, \ldots, 1 = u_{m-1}}_{p+1} \right\}$ causes the curve to be endpoint interpolating, *i.e.*, to interpolate both endpoints of its control polygon. Extreme knots, multiple or not, may enclose any non-decreasing subsequence of interior knots: $0 < u_i \le u_{i+1} < 1$. An endpoint interpolating curve with no interior knots, *i.e.*, one with U consisting of $p+1$ zeroes followed by $p+1$ ones, with no other values in between, is a p^{th}-

degree Bézier curve: *e.g.*, a cubic Bézier curve can be described with four control points (of which the first and last will lie on the curve) and a knot vector $U = \{0, 0, 0, 0, 1, 1, 1, 1\}$.

It is possible to represent all types of curves with NURBS and, in particular, all conic curves (including parabolas, hyperbolas, ellipses, etc.) can be represented using rational functions, unlike when using merely polynomial functions.

Other interesting properties of NURBS curves are the following:

- Affine invariance: rotations, translations, scalings, and shears can be applied to the curve by applying them to $\{\mathbf{P}_i\}$.
- Convex hull property: the curve lies within the convex hull defined by $\{\mathbf{P}_i\}$. The control polygon defined by $\{\mathbf{P}_i\}$ represents a piecewise approximation to the curve. As a general rule, the lower the degree, the closer the NURBS curve follows its control polygon.
- Local control: if the control point $\mathbf{P}_i$ is moved or the weight w_i is changed, only the portion of the curve swept when $u_i < u < u_{i+p+1}$ is affected by the change.

NURBS surfaces are defined as tensor products of two NURBS curves, of possibly different degrees and/or numbers of control points:

$$\mathbf{C}(u,v) = \frac{\sum_{i=0}^{n-1}\sum_{j=0}^{l-1} N_{i,p}(u)N_{j,q}(v)w_{i,j}\mathbf{P}_{i,j}}{\sum_{i=0}^{n-1}\sum_{j=0}^{l-1} N_{i,p}(u)N_{j,q}(v)w_{i,j}}$$

Equation 2 — NURBS surface definition.

The two independent parameters $u, v \in [0, 1]$ allow to travel across the surface. The B-Spline basis functions are defined as previously, and the resulting surface has the same interesting properties that NURBS curves have. Multiplicity of knots may be used to introduce sharp features (corners, creases, etc.) in an otherwise smooth surface, or to have it interpolate the perimeter of its control polyhedron.

4.3.2.2 NurbsCurve

4.3.2.2.1 Node interface

```
NurbsCurve { #%NDT=SFGeometryNode
    eventIn        MFInt32        set_colorIndex
    exposedField   SFColorNode    color           NULL
    exposedField   MFVec4f        controlPoint    []
    exposedField   SFInt32        tessellation    0    # [0, ∞)
    field          MFInt32        colorIndex      [] # [-1, ∞)
    field          SFBool         colorPerVertex  TRUE
    field          MFFloat        knot            []   # (-∞, ∞)
    field          SFInt32        order           4    # [3, 34]
}
```

4.3.2.2.2 Functionality and semantics

The **NurbsCurve** node describes a 3D NURBS curve, which is displayed as a curved line, similarly to what is done with the **IndexedLineSet** primitive.

The **order** field defines the order of the NURBS curve, which is its degree plus one.

The **controlPoint** field defines a set of control points in a coordinate system where the weight is the last component. The number of control points must be greater than or equal to the order of the curve. All weight values must be greater than or equal to 0, and at least one weight must be strictly greater than 0. If the weight of a control point is increased above 1, that point is more closely approximated by the surface. However the surface is not changed if all weights are multiplied by a common factor.

The **knot** field defines the knot vector. The number of knots must be equal to the number of control points plus the order of the curve, and they must be ordered non-decreasingly. By setting consecutive knots to the same value, the degree of continuity of the curve at that parameter value is decreased. If *o* is the value of the field **order**, *o* consecutive knots with the same value at the beginning (resp. end) of the knot vector cause the curve to interpolate the first (resp. last) control point. Other than at its extremes, there may not be more than *o*–1 consecutive knots of equal value within the knot vector. If the length of the knot vector is 0, a default knot vector consisting of *o* 0's followed by *o* 1's, with no other values in between, will be used, and a Bézier curve of degree *o*–1 will be obtained. A closed curve may be specified by repeating the starting control point at the end and specifying a periodic knot vector.

The **tessellation** field gives hints to the curve tessellator as to the number of subdivision steps that must be used to approximate the curve with linear segments: for instance, if the value *t* of this field is greater than or equal to that of the **order** field, *t* can be interpreted as the absolute number of tesselation steps, whereas $t = 0$ lets the browser choose a suitable tessellation.

Fields **color**, **colorIndex**, **colorPerVertex**, and **set_colorIndex** have the same semantic as for **IndexedLineSet** applied to the control points.

4.3.2.3 NurbsCurve2D

4.3.2.3.1 Node interface

```
NurbsCurve2D { #%NDT=SFGeometryNode
    eventIn         MFInt32          set_colorIndex
    exposedField    SFColorNode      color             NULL
    exposedField    MFVec3f          controlPoint      []
    exposedField    SFInt32          tessellation      0    # [0, ∞)
    field           MFInt32          colorIndex        [] # [-1, ∞)
    field           SFBool           colorPerVertex    TRUE
    field           MFFloat          knot              []   # (-∞, ∞)
    field           SFInt32          order             4    # [3, 34]
}
```

4.3.2.3.2 Functionality and semantics

The **NurbsCurve2D** is the 2D version of **NurbsCurve**; it follows the same semantic as **NurbsCurve** with 2D control points.

4.3.2.4 NurbsSurface

4.3.2.4.1 Node interface

```
NurbsSurface { #%NDT=SFGeometryNode
    eventIn         MFInt32                    set_colorIndex
    eventIn         MFInt32                    set_texCoordIndex
    exposedField    SFColorNode                color             NULL
    exposedField    MFVec4f                    controlPoint      []
    exposedField    SFTextureCoordinateNode    texCoord          NULL
    exposedField    SFInt32                    uTessellation     0    # [0, ∞)
    exposedField    SFInt32                    vTessellation     0    # [0 ,∞)
    field           MFInt32                    colorIndex        [] # [-1, ∞)
```