Nederlandse norm

NEN-ISO/IEC 14496-16/A1

(en)

Information technology - Coding of audio-visual objects - Part 16: Animation Framework eXtension (AFX) - Morphing and textures (ISO/IEC 14496-16:2004/Amd 1:2006,IDT)

ICS 35.040

januari 2006

Als Nederlandse norm is aanvaard:

- ISO/IEC 14496-16:2004/Amd 1:2006,IDT

Normcommissie 381 029 "Multimedia/Hypermedia"

NEN

Postbus 5059, 2600 GB Delft
Telefoon (015) 2 690 390, Fax (015) 2 690 190

INTERNATIONAL STANDARD

ISO/IEC 14496-16

First edition
2004-02-01

AMENDMENT 1
2006-01-15

Information technology — Coding of audio-visual objects —

Part 16: Animation Framework eXtension (AFX)

AMENDMENT 1: Morphing and textures

Technologies de l'information — Codage des objets audiovisuels —

Partie 16: Extension du cadre d'animation (AFX)

AMENDEMENT 1: Morphage et textures

Reference number
ISO/IEC 14496-16:2004/Amd.1:2006(E)

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

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.

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

Amendment 1 to ISO/IEC 14496-16:2004 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 29, *Coding of audio, picture, multimedia and hypermedia information*.

Information technology — Coding of audio-visual objects —

Part 16: **Animation Framework eXtension (AFX)**

AMENDMENT 1: Morphing and textures

Add subclause 4.3.6 MorphSpace:

4.3.6 MorphSpace

4.3.6.1 Introduction

Morphing is mainly an interpolation technique used to create from two objects a series of intermediate objects that change continuously, in order to make a smooth transition from the source to the target. A straight extension of the morphing between two elements –the source and the target- consists in considering a collection of possible targets and compose a virtual object configuration by weighting those targets. This collection represents a basis of animation space and animation is performed by simply updating the weight vector. The following node allows the representation of a mesh as a combination of a base shape and a collection of target geometries.

4.3.6.2 Node interface

```
MorphShape{ #%NDT=SF3DNode,SF2DNode
    exposedField SFInt32 morphID
    exposedField SFShapeNode baseShape
    exposedField MFShapeNode targetShapes [ ]
    exposedField MFFloat weights [ ]
}
```

4.3.6.3 Semantics

morphID - a unique identifier between 0 and 1023 which allows that the morph to be addressed at animation run-time.

baseShape – a Shape node that represent the base mesh. The geometry field of the baseShape can be any geometry supported by ISOIEC 14496 (e.g. IndexedFaceSet, IndexedLineSet, SolidRep).

targetShapes – a vector of Shapes nodes representing the shape of the target meshes. The tool used for definig an appearance and a geometry of a target shape must be the same as the tool used for defining the appearance and the geometry of the base shape (e.g. if the baseShape is defined by using IndexedFaceSet, all the target shapes must be defined by using IndexedFaceSet).

weights – a vector of integers of the same size as the **targetShapes**. The morphed shape is obtained according to the following formula:

$$M = B + \sum_{i=1}^{n} (T_i - B) * w_i$$	(ADM1-1)

with

M –morphed shape,

B – base shape,

T_i – target shape *i*,

W_i – weight of the T_i.

The morphing is performed for all the components of the Shape (Appearance and Geometry) that have different values in the base shape and the target shapes [e.g. if the base shape and the target shapes are definined by using IndexedFaceSet and the *coord* field contains different values in the base shape and in the target geometries, the *coord* component of the morph shape is obtained by using Equation (ADM1-1)] applied to the *coord* field. Note that the size of the *coord* field must be the same for the base shapes and the target shapes).

If the shapes (base and targets) are defined by using IndexedFaceSet, a tipical decoder should support morphing of the following geometry components: *coord*, *normals*, *color*, *texCoord*.

Add subclause 4.5.4 Depth Image-based Representation Version 2:

4.5.4 Depth Image-based Representation Version 2

4.5.4.1 Introduction

Version 1 of DIBR introduced depth image-based representations (DIBR) of still and animated 3D objects. Instead of a complex polygonal mesh, which is hard to construct and handle for realistic models, image- or point-based methods represent a 3D object (scene) as a set of reference images completely covering its visible surface. This data is usually accompanied by some kind of information about the object geometry. To that end, each reference image comes with a corresponding depth map, an array of distances from the pixels in the image plane to the object surface. Rendering is achieved by either forward warping or splat rendering. But with Version 1 of the specification of DIBR nodes no high-quality rendering can be achieved.

Version 2 nodes allow for high-quality rendering of depth image-based representations. High-quality rendering is based on the notion of point-sampled surfaces as non-uniformly sampled signals. Point-sampled surfaces can be easily constructed from the DIBR nodes by projecting the pixels with depth into 3D-space. The discrete signals are rendered by reconstructing and band-limiting a continuous signal in image space using so called resampling filters.

A point-based surface consists of a set of non-uniformly distributed samples of a surface; hence we interpret it as a non-uniformly sampled signal. To continuously reconstruct this signal, we have to associate a 2D reconstruction kernel $r_k(u)$ with each sample point p_k. The kernels are defined in a local tangent frame with coordinates $u = (u, v)$ at the point p_k, as illustrated on the left in Figure AMD1-1. The tangent frame is defined by the splat and normal extensions of the DIBR structures Version 2 [1].

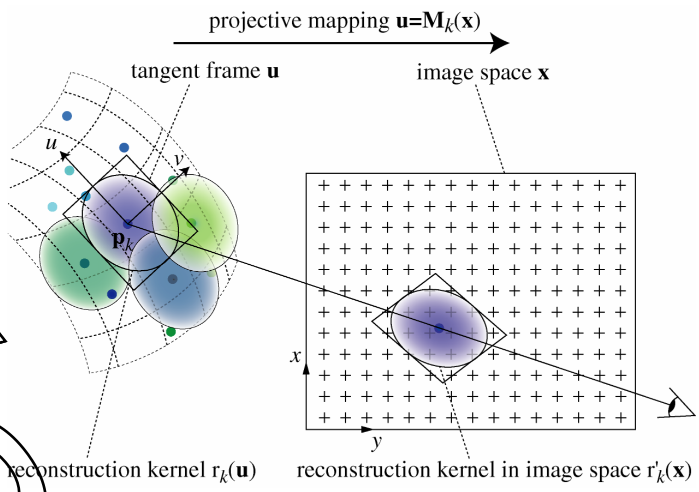

Figure AMD1-1 — Local tangent planes and reconstruction kernels

4.5.4.2 DepthImageV2 Node

4.5.4.2.1 Node interface

```
DepthImageV2 { #%NDT=SF3DNode
    exposedField   SFVec3f             position        0 0 10
    exposedField   SFRotation          orientation     0 0 1 0
    exposedField   SFVec2f             fieldOfView     π/4 π/4
    exposedField   SFFloat             nearPlane       10
    exposedField   SFFloat             farPlane        100
    field          SFVec2f             splatMinMax     0.1115 0.9875
    exposedField   SFBool              orthographic    TRUE
    field          SFDepthTextureNode  diTexture       NULL
}
```

4.5.4.2.2 Functionality and semantics

The **DepthImageV2** node defines a single IBR texture. When multiple **DepthImage** nodes are related to each other, they are processed as a group, and thus, should be placed under the same Transform node.

The **diTexture** field specifies the texture with depth, which shall be mapped into the region defined in the **DepthImageV2** node. It shall be one of the various types of depth image texture (**SimpleTextureV2** or **PointTextureV2**).

The **position** and **orientation** fields specify the relative location of the viewpoint of the IBR texture in the local coordinate system. **position** is relative to the coordinate system's origin (0, 0, 0), while **orientation** specifies a rotation relative to the default orientation. In the default position and orientation, the viewer is on the Z-axis looking down the –Z-axis toward the origin with +X to the right and +Y straight up. However, the transformation hierarchy affects the final position and orientation of the viewpoint.

The **fieldOfView** field specifies a viewing angle from the camera viewpoint defined by **position** and **orientation** fields. The first value denotes the angle to the horizontal side and the second value denotes the angle to the vertical side. The default values are 45 degrees in radians. However, when **orthographic** field is set to TRUE, the **fieldOfView** field denotes the width and height of the near plane and far plane.

The **nearPlane** and **farPlane** fields specify the distances from the viewpoint to the near plane and far plane of the visibility area. The texture and depth data shows the area closed by the near plane, far plane and the **fieldOfView**. The depth data are scaled to the distance from **nearPlane** to **farPlane**.

The **splatMinMax** field specifies the minimum and maximum splat vector lengths. The splatU and splatV data of SimpleTextureV2 is scaled to the interval defined by the splatMinMax field.

The **orthographic** field specifies the view type of the IBR texture. When set to TRUE, the IBR texture is based on orthographic view. Otherwise, the IBR texture is based on perspective view.

The **position, orientation, fieldOfView, nearPlane, farPlane,** and **orthographic** fields are exposedField types, which are for extrinsic parameters. The DepthImage node supports the camera movement and changeable view frustum corresponding to movement or deformation of a DIBR object.

Reference images that are suitable to the characteristic of a DIBR model are obtained in the modeling stage. Therefore, the fields that reflect the camera movement and the changeable view frustum and the reference images in the modeling stage are used to create a view frustum and a DIBR object in the rendering stage.

4.5.4.3 SimpleTextureV2 node

4.5.4.3.1 Node interface

```
SimpleTextureV2 { #%NDT=SFDepthTextureNode
	field	SFTextureNode	texture	NULL
	field	SFTextureNode	depth	NULL
	field	SFTextureNode	normal	NULL
	field	SFTextureNode	splatU	NULL
	field	SFTextureNode	splatV	NULL
}
```

4.5.4.3.2 Functionality and semantics

The **SimpleTextureV2** node defines a single layer of IBR texture.

The **texture** field specifies the flat image that contains color for each pixel. It shall be one of the various types of texture nodes (**ImageTexture**, **MovieTexture** or **PixelTexture**).

The **depth** field specifies the depth for each pixel in the **texture** field. The size of the depth map shall be the same size as the image or movie in the **texture** field. Depth field shall be one of the various types of texture nodes (ImageTexture, MovieTexture or PixelTexture), where only the nodes representing gray scale images are allowed. If the depth field is unspecified, the alpha channel in the texture field shall be used as the depth map. If the depth map is not specified through depth field or alpha channel, the result is undefined.

Depth field allows to compute the actual distance of the 3D points of the model to the plane which passes through the viewpoint and parallel to the near plane and far plane:

$$dist = nearPlane + \left(1 - \frac{d-1}{d_{\max}-1}\right)(farPlane - nearPlane). \qquad \text{(AMD1-2)}$$

where d is depth value and d_{max} is maximum allowed depth value. It is assumed that for the points of the model, $d>0$, where $d=1$ corresponds to far plane, $d=d_{max}$ corresponds to near plane.

This formula is valid for both perspective and orthographic case, since d is distance between the point and the plane. max d is the largest d value that can be represented by the bits used for each pixel:

(1) If the depth is specified through **depth** field, then depth value d equals to the gray scale.

(2) If the depth is specified through alpha channel in the image defined via **texture** field, then the depth value d is equal to alpha channel value.

The depth value is also used to indicate which points belong to the model: only the point for which d is nonzero belong to the model.

For animated DepthImage-based model, only DepthImage with SimpleTextures as diTextures are used.

Each of the Simple Textures can be animated in one of the following ways:

(1) **depth** field is still image satisfying the above condition, **texture** field is arbitrary MovieTexture

(2) **depth** field is arbitrary MovieTexture satisfying the above condition on the **depth** field, **texture** field is still image

(3) both **depth** and **texture** are MovieTextures, and **depth** field satisfies the above condition

(4) **depth** field is not used, and the depth information is retrieved from the alpha channel of the MovieTexture that animates the **texture** field

The **normal** field specifies the normal vector for each pixel in the **texture** field. The normal vector should be assigned to the object-space point sample derived from extruding the pixel with depth to 3-space. The normal map shall be the same size as the image or movie in the **texture** field. Normal field shall be one of the various types of texture nodes (ImageTexture, MovieTexture, or PixelTexture), where only the nodes representing color images are allowed. If the normal map is not specified through the normal field, the decoder can calculate a normal field by evaluating the cross-product of the splatU and splatV fields. If neither the normal map nor the splatU and splatV fields are specified, only basic rendering is possible.

The **splatU** and **splatV** fields specify the tangent plane and reconstruction kernel needed for high-quality point-based rendering. Both splatU and splatV fields have to be scaled to the interval defined by the splatMinMax field.

The **splatU** field specifies the splatU vector for each pixel in the **texture** field. The splatU vector should be assigned to the object-space point sample derived from extruding the pixel with depth to 3-space. The splatU map shall be the same size as the image or movie in the **texture** field. splatU field shall be one of the various types of texture nodes (ImageTexture, MovieTexture, or PixelTexture), where the nodes either representing color or gray scale images are allowed. If the splatU map is specified as gray scale image the decoder can calculate a circular splat by using the normal map to produce a tangent plane and the splatU map as radius. In this case, if the normal map is not specified, the result is undefined. If the splatU map is specified as color image, it can be used in conjunction with the splatV map to calculate a tangent frame and reconstruction kernel for high-quality point-based rendering. If neither the normal map nor the splatV map is specified, the result is undefined.

The **splatV** field specifies the splatV vector for each pixel in the **texture** field. The splatV vector should be assigned to the object-space point sample derived from extruding the pixel with depth to 3-space. The splatV map shall be the same size as the image or movie in the **texture** field. splatV field shall be one of the various types of texture nodes (ImageTexture, MovieTexture, or PixelTexture), where only the nodes representing color images are allowed. If the splatU map is not specified as well, the result is undefined.

4.5.4.4 PointTextureV2 node

4.5.4.4.1 Node interface

```
PointTextureV2 { #%NDT=SFDepthTextureNode
    field    SFInt32         width        256
    field    SFInt32         height       256
    field    SFInt32         depthNbBits  7
    field    MFInt32         depth        []
    field    MFColor         color        []
    field    SFNormalNode    normal
    field    MFVec3f         splatU       []
    field    MFVec3f         splatV       []
}
```

4.5.4.4.2 Functionality and semantics

The **PointTextureV2** node defines multiple layers of IBR points.

The **width** and **height** field specify the width and height of the texture.

Geometrical meaning of the depth values, and all the conventions on their interpretation adopted for the SimpleTexture, apply here as well.

The **depth** field specifies a multiple depths of each point in the projection plane, which is assumed to be farPlane (see above) in the order of traversal, which starts from the point in the lower left corner and traverses to the right to finish the horizontal line before moving to the upper line. For each point, the number of depths (pixels) is first stored and that number of depth values shall follow.

The **color** field specifies color of current pixel. The order shall be the same as the **depth** field except that number of depths (pixels) for each point is not included.

The **depthNbBits** field specifies the number of bits used for the original depth data. The value of depthNbBits ranges from 0 to 31, and the actual number of bits used in the original data is depthNbBits+1. The d_{max} used in the distance equation is derived as follows:

$d_{\max} = 2^{(depthNbBits+1)} - 1.$	(AMD1-3)

The **normal** field specifies normals for each specified depth of each point in the projection plane in the same order. The normal vector should be assigned to the object-space point sample derived from extruding the pixel with depth to 3-space. If the normals are not specified through the normal field, the decoder can calculate a normal field by evaluating the cross-product of the splatU and splatV fields. If neither the normals nor the splatU and splatV fields are specified, only basic point rendering is possible. Normals can be quantized by using the SFNormalNode functionality.

The **splatU** field specifies splatU vectors for each specified depth of each point in the projection plane in the same order. The splatU vector should be assigned to the object-space point sample derived from extruding the pixel with depth to 3-space. If the splatV vectors are not specified the decoder can calculate a circular splat by using the normals to produce a tangent plane and the length of the splatU vectors as radius. In this case, if the normals are not specified, the result is undefined. If the splatU vectors are specified, it can be used in conjunction with the splatV vectors to calculate a tangent frame and reconstruction kernel for high-quality point-based rendering. If neither the normals nor the splatV vectors are specified, the result is undefined.

The **splatV** field specifies splatV vectors for each specified depth of each point in the projection plane in the same order. The splatV vector should be assigned to the object-space point sample derived from extruding the pixel with depth to 3-space. If the splatU vectors are not specified as well, the result is undefined.

Add subclause 4.5.5 Multitexturing:

4.5.5 Multitexturing

4.5.5.1 MultiTexture Node

4.5.5.1.1 Node interface

```
MultiTexture { #%NDT=SFTextureNode
  exposedField SFFloat        alpha       1      #[0,1]
  exposedField SFColor        color       1 1 1  #[0,1]
  exposedField MFInt          function    []
  exposedField MFInt          mode        []
  exposedField MFInt          source      []
```

Bestelformulier

Stuur naar:

NEN Standards Products & Services
t.a.v. afdeling Klantenservice
Antwoordnummer 10214
2600 WB Delft

Ja, ik bestel

__ ex. NEN-ISO/IEC 14496-16:2004/A1:2006 en Information technology - Coding of audio-visual objects - Part 16: Animation Framework eXtension (AFX) - Morphing and textures — € 143.10

Wilt u deze norm in PDF-formaat? Deze bestelt u eenvoudig via www.nen.nl/normshop

Gratis e-mailnieuwsbrieven

Wilt u op de hoogte blijven van de laatste ontwikkelingen op het gebied van normen, normalisatie en regelgeving? Neem dan een gratis abonnement op een van onze e-mailnieuwsbrieven. www.nen.nl/nieuwsbrieven

Gegevens

Bedrijf / Instelling ______________________

T.a.v. ______________________ O M O V

E-mail ______________________

Klantnummer NEN ______________________

Uw ordernummer ____________ BTW nummer ____________

Postbus / Adres ______________________

Postcode ____________ Plaats ____________

Telefoon ____________ Fax ____________

Factuuradres (indien dit afwijkt van bovenstaand adres)

Postbus / Adres ______________________

Postcode ____________ Plaats ____________

Datum ____________ Handtekening ____________

LEREN, WERKEN EN GROEIEN MET NEN

NEN Standards Products & Services

Postbus 5059
2600 GB Delft

Vlinderweg 6
2623 AX Delft

T (015) 2 690 390
F (015) 2 690 271

www.nen.nl/normshop

Retourneren

Fax: 015 2 690 271
E-mail: klantenservice@nen.nl
Post: NEN Standards Products & Services,
t.a.v. afdeling Klantenservice
Antwoordnummer 10214,
2600 WB Delft
(geen postzegel nodig).

Voorwaarden

- De prijzen zijn geldig tot 31 december 2018, tenzij anders aangegeven.
- Alle prijzen zijn excl. btw, verzend- en handelingskosten en onder voorbehoud bij o.m. ISO- en IEC-normen.
- Bestelt u via de normshop een pdf, dan betaalt u geen handeling en verzendkosten.
- Meer informatie: telefoon 015 2 690 391, dagelijks van 8.30 tot 17.00 uur.
- Wijzigingen en typefouten in teksten en prijsinformatie voorbehouden.
- U kunt onze algemene voorwaarden terugvinden op: www.nen.nl/leveringsvoorwaarden.