INTERNATIONAL STANDARD ISO/IEC 14496-16:2006/Amd.1:2007
TECHNICAL CORRIGENDUM 1

Published 2008-06-15

INTERNATIONAL ORGANIZATION FOR STANDARDIZATION • МЕЖДУНАРОДНАЯ ОРГАНИЗАЦИЯ ПО СТАНДАРТИЗАЦИИ • ORGANISATION INTERNATIONALE DE NORMALISATION
INTERNATIONAL ELECTROTECHNICAL COMMISSION • МЕЖДУНАРОДНАЯ ЭЛЕКТРОТЕХНИЧЕСКАЯ КОМИССИЯ • COMMISSION ÉLECTROTECHNIQUE INTERNATIONALE

Information technology — Coding of audio-visual objects —

Part 16:
Animation Framework eXtension (AFX)

AMENDMENT 1: Geometry and shadow

TECHNICAL CORRIGENDUM 1

Technologies de l'information — Codage des objets audiovisuels —

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

AMENDEMENT 1: Géométrie et ombre

RECTIFICATIF TECHNIQUE 1

Technical Corrigendum 1 to ISO/IEC 14496-16:2006/Amd.1:2007 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 29, *Coding of audio, picture, multimedia and hypermedia information.*

ICS 35.040 **Ref. No. ISO/IEC 14496-16:2006/Amd.1:2007/Cor.1:2008(E)**

© ISO/IEC 2008 – All rights reserved

Published in Switzerland

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS

Not for Resale

In 5.9.2.2.1.9, 3DMC_extension_header, replace:

3DMC_extension_header() {	**No. of bits**	**Mnemonic**
do {		
function_type	4	uimsbf
if (function_type == "Order_mode")		
Order_mode_header ()		
else if (function_type == "Adaptive_quant_texCoord_mode")		
Adaptive_quant_texCoord_mode_header()		
} while(function_type != "Escape_mode")		
}		

with the following (where changes are highlighted in grey):

3DMC_extension_header() {	**No. of bits**	**Mnemonic**
do {		
function_type	4	vlclbf
if (function_type == "Order_mode")		
Order_mode_header ()		
else if (function_type == "Adaptive_quant_texCoord_mode")		
Adaptive_quant_texCoord_mode_header()		
else if (function_type == "Multiple_attribute_mode")		
multiple_attribute_mode_header()		
} while(function_type != "Escape_mode")		
}		

Insert the following new subclauses, renumbering subsequent subclauses:

5.9.2.2.1.11 multiple_attribute_mode_header

multiple_attribute_mode_header(){	**No. of bits**	**Mnemonic**
number_of_texCoord	5	
for 2 to number_of_texCoord		vlclbf
texCoord_header()		
number_of_otherAttr	8	
for 1 to number_of_otherAttr		
otherAttr_header()		
}		

© ISO/IEC 2008 – All rights reserved
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

5.9.2.2.1.12 otherAttr_header

otherAttr_header() {	No. of bits	Mnemonic
otherAttr_binding	2	uimsbf
otherAttr_dimension	8	
if (otherAttr_binding != "not_bound") {		
otherAttr_bbox	1	
if (otherAttr_bbox == '1') {		
for (i= &; i < otherAttr_dimension; i++)		
otherAttr_min(i)	32	bslbf
otherAttr_size	32	bslbf
}		
otherAttr_quant	5	uimsbf
otherAttr_pred_type	2	uimsbf
if (otherAttr_pred_type=="tree_prediction" || otherAttr_pred_type=="parallelogram_prediction") {		
otherAttr_nlambda	2	uimsbf
for (i=1; i<otherAttr_nlambda; i++)		
otherAttr_lambda	4-19	simsbf
}		
}		
}		

Replace 5.9.2.2.1.18, root_triangle():

root_triangle() {	No. of bits	Mnemonic
if (marching_triangle)		
qf_decode(**marching_pattern**, marching_pattern_context[**marching_pattern**])		vlclbf
else {		
if (3D_MOBL_start_code == "partition_type_2")		
if (tt_leaf == '0' && depth==0)		
qf_decode(**td_orientation**, td_orientation_context)		vlclbf
if (tt_leaf == '0')		
depth++		
else		
depth--		
}		
if (3D_MOBL_start_code == "partition_type_2")		
if (triangulated == '0')		
qf_decode(**polygon_edge**, polygon_edge_context[**polygon_edge**])		vlclbf
root_coord()		
root_normal()		
root_color()		
root_texCoord()		
}		

© ISO/IEC 2008 - All rights reserved
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

with the following (where changes are highlighted in grey):

root_triangle() {	No. of bits	Mnemonic
if (marching_triangle)		
qf_decode(**marching_pattern**, marching_pattern_context[**marching_pattern**])		vlclbf
else {		
if (3D_MOBL_start_code == "partition_type_2")		
if (tt_leaf == '0' && depth==0)		
qf_decode(**td_orientation**, td_orientation_context)		vlclbf
if (tt_leaf == '0')		
depth++		
else		
depth--		
}		
if (3D_MOBL_start_code == "partition_type_2")		
if (triangulated == '0')		
qf_decode(**polygon_edge**, polygon_edge_context[**polygon_edge**])		vlclbf
root_coord()		
root_normal()		
root_color()		
for (i =1; i<= number_of_texCoord; i++)		
root_texCoord(i)		
for (i = 1; i<= number_of_otherAttr; i++)		
root_otherAttr(i)		
}		

At the end of 5.9.2.2.1.18, add the following tables:

root_otherAttr() {	No. of bits
if (otherAttr_binding != "not_bound")	
if (3D_MOBL_start_code == "partition_type_2") {	
if (otherAttr_binding != "bound_per_vertex" || visited[vertex_index] == 0) {	
root_otherAttr_sample()	
if (otherAttr_binding!= "bound_per_vertex" || visited[vertex_index] == 0) {	
otherAttr_sample()	
otherAttr_sample()	
}	
}	
else {	
root_otherAttr_sample()	
otherAttr_sample()	
otherAttr_sample()	
}	
}	

--`,,```,,,,````-`-`,,`,,`,`,,`---
© ISO/IEC 2008 – All rights reserved
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

'root_otherAttr_sample() {	No. of bits
for (i=0; i<otherAttr_order; i++)	
for (j=0; j<otherAttr_quant; j++)	
qf_decode(**otherAttr_bit**, zero_context)	
}	

otherAttr_sample() {	No. of bits
for (i=0; i< otherAttr_order; i++) {	
j=0	
do {	
qf_decode(**otherAttr_leading_bit**, otherAttr_leading_bit_context[2*j+i])	
j++	
} while (j<otherAttr_quant && otherAttr_leading_bit == '0')	
if (otherAttr_leading_bit == '1') {	
qf_decode(**otherAttr_sign_bit**, zero_context)	
do {	
qf_decode(**otherAttr_trailing_bit**, zero_context)	
} while (j<otherAttr_quant)	
}	
}	
}	

Replace 5.9.2.2.1.19, triangle():

triangle(i) {	No. of bits	Mnemonic
if (marching_triangle)		
qf_decode(**marching_edge**, marching_edge_context[**marching_edge**])		vlclbf
else {		
if (3D_MOBL_start_code == "partition_type_2")		
if (tt_leaf == '0' && depth==0)		
qf_decode(**td_orientation**, td_orientation_context)		vlclbf
if (tt_leaf == '0')		
depth++		
else		
depth--		
if (triangulated == '0')		
qf_decode(**polygon_edge**, polygon_edge_context[**polygon_edge**])		vlclbf
coord()		
normal()		
color()		
texCoord()		
}		

--`,,```,,,,````-`-`,,`,,`,`,,`---
© ISO/IEC 2008 - All rights reserved
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

with the following (where changes are highlighted in grey):

triangle(i) {	No. of bits	Mnemonic
if (marching_triangle)		
qf_decode(**marching_edge**, marching_edge_context[**marching_edge**])		vlclbf
else {		
if (3D_MOBL_start_code == "partition_type_2")		
if (tt_leaf == '0' && depth==0)		
qf_decode(**td_orientation**, td_orientation_context)		vlclbf
if (tt_leaf == '0')		
depth++		
else		
depth--		
}		
if (triangulated == '0')		vlclbf
qf_decode(**polygon_edge**, polygon_edge_context[**polygon_edge**])		
coord()		
normal()		
color()		
for (i=1; i<= number_of_texCoord; i++)		
texCoord(i)		
for (i=1; i<= number_of_otherAttr; i++)		
otherAttr(i)		
}		

In 5.9.2.3.1.9 3, DMC_extension_header, replace:

function_type: This 4-bit unsigned integer indicates the function type supported in 3DMC extension. Table AMD1-17shows the admissible values for **function_type.**

Table AMD1-17 — Admissible values for function type

function_type_code	function_type
0000	Order_mode
0001	Adaptive_quant_texCoord_mode
0011~1110	Reserved
1111	Escape code

with the following (where changes are highlighted in grey)::

function_type: This 4-bit unsigned integer indicates the function type supported in 3DMC extension. Table AMD1-17 shows the admissible values for **function_type**

© ISO/IEC 2008 – All rights reserved
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Table AMD1-17 — Admissible values for function type

function_type_code	function_type
0000	Order_mode
0001	Adaptive_quant_texCoord_mode
0010	Multiple_attribute_mode
0011~1110	Reserved
1111	Escape code

Insert the following new subclauses, renumbering subsequent subclauses:

5.9.2.3.1.12 multiple_attribute_mode_header
number_of_texCoord: This 5 bit unsigned integer gives the number of texture coordinates per vertex.
number_of_otherAttr: This 8 bit unsigned integer gives the number of additional attributes per vertex

5.9.2.3.1.13 otherAttr_header
otherAttr_binding: This 2 bit unsigned integer indicates the binding of texture coordinates to the 3D mesh. Table AMD1COR1.1 describes the admissible values.

Table AMD1COR1.1 — Admissible values for otherAttr_binding

otherAttr_binding	Binding
00	not_bound
01	bound_per_vertex
10	forbidden
11	bound_per_corner

otherAttr_order: This 8-bit unsigned integer gives the order of the other attribute.
otherAttr_bbox: This boolean value indicates if a bounding box for texture coordinates is given. If no bounding box is provided, a default bounding box is used. The default bounding box is defined as **otherAttr_min(i)**=0 **otherAttr_size**=1.
otherAttr_umin, **otherAttr_vmin**: These floating point values give the position of the lower left corner of the bounding box in 2D space.
otherAttr_size: This floating point value gives the size of the texture coordinate bounding box.
otherAttr_quant: This 5-bit unsigned integer indicates the quantisation step used for texture coordinates. The minimum value of otherAttr_quant is 1 and the maximum is 16.
otherAttr_pred_type: This 2-bit unsigned integer indicates how colors are predicted. Table AMD1COR1.2 shows the admissible values, and Table 4 shows admissible values as a function of otherAttr_binding.

Table AMD1COR1.2 — Admissible values for otherAttr_pred_type

otherAttr_pred_type	prediction type
00	no_prediction
01	Forbidden
10	parallelogram_prediction
11	Reserved

© ISO/IEC 2008 - All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Table AMD1COR1.3 — Admissible combinations of otherAttr_binding and otherAttr_pred_type

otherAttr_binding	otherAttr_pred_type
not_bound	not coded
bound_per_vertex	no_prediction, parallelogram_prediction
bound_per_corner	no_prediction, tree_prediction

otherAttr_nlambda: This 2-bit unsigned integer indicates the number of ancestors used to predict normals. Admissible values of **otherAttr_nlambda** are 1, 2, and 3. Table 5shows admissible values as a function of **otherAttr_pred_type.**

Table AMD1COR1.4 — Admissible values for otherAttr_nlambda as a function of otherAttr_prediction type

otherAttr_pred_type	otherAttr_nlambda
not_prediction	not coded
tree_prediction	1, 2, 3
parallelogram_prediction	3

otherAttr_lambda: This signed fixed-point indicates the weight given to an ancestor for prediction. The number of bits used for this field is equal to **otherAttr_quant** + 3. The 3 leading bits represent the integer part, and the **otherAttr_quant** remaining bits the fractional part.

At the end of 5.9.2.3.1.17, triangle, add the following sentences:

otherAttr _bit: This boolean value indicates the value of a otherAttr bit. This field is arithmetic coded.
otherAttr_leading_bit: This boolean value indicates the value of a leading otherAttr bit. This field is arithmetic coded.
otherAttr_sign_bit: This boolean value indicates the sign of a otherAttr sample. This field is arithmetic coded.
otherAttr_trailing_bit: This boolean value indicates the value of a trailing otherAttr bit. This field is arithmetic coded.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
© ISO/IEC 2008 – All rights reserved
Not for Resale