**ISO/IEC JTC 1/SC 29/WG 11  
Coding of moving pictures and audio  
Convenorship: UNI (Italy)**

**Document type: Approved WG 11 document**

**Title: Technologies under Considerations on Support of MPEG Media in Scene Descriptions**

**Status: Approved**

**Date of document: 2020-01-17**

**Source: Systems**

**Expected action:**

**No. of pages: 12**

**Email of convenor: leonardo@chiariglione.org**

**Committee URL: http://mpeg.chiariglione.org**

**INTERNATIONAL ORGANISATION FOR STANDARDISATION**

**ORGANISATION INTERNATIONALE DE NORMALISATION**

**ISO/IEC JTC 1/SC 29/WG 11**

**CODING OF MOVING PICTURES AND AUDIO**

**ISO/IEC JTC 1/SC 29/WG 11 N19071**

**Brussels, BE – January 2020**

**Source: Systems**

**Technologies under Considerations on Scene Description for MPEG Media**

**INTERNATIONAL ORGANISATION FOR STANDARDISATION**

**ORGANISATION INTERNATIONALE DE NORMALISATION**

**ISO/IEC JTC1/SC29/WG11**

**CODING OF MOVING PICTURES AND AUDIO**

**ISO/IEC JTC1/SC29/WG11 MPEG2018/N19071**

**Brussels, BE, January 2020**

|  |  |
| --- | --- |
| **Source** | **Systems** |
| **Status** | **Output** |
| **Title** | **Technologies under Considerations on Scene Description for MPEG Media** |
| **Authors** | Lukasz Kondrad, Imed Bouazizi |

# Introduction

This document contains technologies under consideration on support of MPEG Media in Scene Description. The listed contributions have been agreed to be included in WD N19070 [1] waiting the verification steps defined in Section 6.7 of N19072 [2]

The following contributions are covered

* Timed Accessors (m52463)
* MPEG media and MPEG video texture extensions for glTF2 (m52838)

# Timed Accessors (m52463)

In order to provide access to timed media and metadata in a scene, a new glTF extension is specified to define timed accessors. An accessor in glTF defines the types and layout of the data as stored in a buffer that is viewed through a bufferView.

When timed data is read from a buffer, the data in the buffer is expected to change dynamically with time. The buffer element is extended to add support for a circular buffer that is used with timed data.

## MPEG\_timed\_accessors extension

A scene that contains timed media and/or metadata shall use the timed accessor extension to access the data. The timed accessor is an extension to regular accessors to indicate that the underlying data buffer is dynamic.

The timed accessor extension is identified by "MPEG\_timed\_accessor", which shall be included in the extensionsUsed and extensionsRequired of the scene description document, whenever timed data is used in a scene.

The "MPEG\_timed\_accessor" extension shall be defined on "accessors" structures. It may contain the following properties:

**Table 1**

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Default** | **Description** |
| immutable | boolean | False | This flag indicates if the accessor information such as the componentType, bufferView, and type are allowed to change over time. Note that count, max, min, and byteOffset are expected to change and are always included as part of the timed accessor information header. |
| bufferView | integer | N/A | This provides the reference id of a bufferView that points to the timed accessor information header. |
| updateRate | number | 25.0 | The updateRate provides the frequency at which the underlying buffer data is expected to change. The rate is provided in number of changes per second. |

The timed accessor information header contains the dynamic metadata that is needed to access the timed data.

The following table describes the syntax and semantics of the timed accessor information header:

**Table 2**

|  |  |  |  |
| --- | --- | --- | --- |
| **Syntax** | **Length (bits)** | **type** | **Semantics** |
| timestamp\_delta | 32 | float | Provides a delta in seconds that is added to the timestamp of the referenced buffer to determine the timestamp of the referenced timed media. |
| if (!immutable) {  componentType  bufferView  type  normalized  reserved  } | 32  32  8  1  7 | integer  integer  integer  boolean | These fields correspond to the accessor properties as defined in [3]. The type differs from the definition in [3] in that it provides a 0-based index of the allowed types as defined in [3]. For example a type of 0 indicates that the data is a "SCALAR". |
| byteOffset  count  max  min | 32  32  32  32 | integer  integer  float  float | These fields correspond to the accessor properties as defined in [3]. |
| bufferViewByteOffset  bufferViewByteLength  bufferViewByteStride | 32  32  32 | integer  integer  integer | These fields correspond to the bufferView fields byteOffset, byteLength, and byteStride respectively. |

Note that the timed accessor information header is provided as binary data as part of the buffer data and is accessible through the bufferView of the timed accessor extension.

The following is an example showing the new extension:

|  |
| --- |
| {  "accessors": [  {  "bufferView": 0,  "componentType": 5126,  "byteOffset": 0,  "count": 12323,  "type": "VEC4",  "extensions": {  "MPEG\_timed\_accessor": {  "immutable":1,  "bufferView":1,  "updateRate":25.0  }  }  }  ],  } |

# MPEG\_circular\_buffer extension

In order to support timed data access, the buffer element is extended to provide circular buffer functionality. The extension is named "MPEG\_circular\_buffer" and may be included as part of the "buffers" structures. Buffers that provide access to timed data shall include the "MPEG\_circular\_buffer" extension.

The following properties are defined for the "MPEG\_circular\_buffer":

**Table 3** Definitions oftop-level objects of MPEG\_circular\_buffer extension

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Default** | **Description** |
| count | integer | 5 | The count field provides the number of frames that are offered by this circular buffer. Each frame will hold data at a particular time instance and will be identified by an index in the range of [0,count-1]. The index, timestamp and length of the frame are signaled as the buffer header, which shall always be accessible at byte 0 of the frame data. |
| headerLength | integer | 12 | The headerLength provides the length of the buffer header and is the offset into the dynamic actual data. |
| updateRate | number | 25.0 | The updateRate provides the frequency at which the underlying buffer data is expected to change. The rate is provided in number of changes per second. |

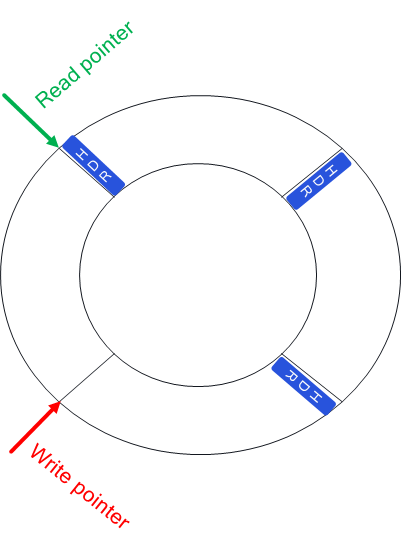
**Table 4** Syntax and semantics of the buffer header

|  |  |  |  |
| --- | --- | --- | --- |
| **Syntax** | **Length (bits)** | **type** | **Semantics** |
| index | 8 | integer | The index of the current buffer frame. The index is a value between 0 and count -1. |
| timestamp | 64 | integer | Provides the timestamp of the data that is contained in this buffer. The format of this field is in NTP Timestamp Format with 32 MSB for seconds and 32 LSB for fraction of seconds. Note that this timestamp field is not necessarily a wallclock time and the interpretation of this field is left to the rendering engine. |
| length | 32 | integer | The length of the data of this buffer frame, including the buffer header. |

Frames of the buffer may differ in length based on the amount of data for each frame. A read and a write pointer are maintained for each circular buffer. By default, read and write access to the buffer will be served from the frame that is referenced by the read or write pointer respectively. Access to a particular frame index or timestamp should be supported.

The frames are read at the read pointer for rendering. New incoming frames from the media decoder are inserted at the write pointer. Prior data in that frame will be overwritten and the frame buffer should be resized accordingly.

**Figure 1** depicts the buffer structure:



**Figure 1** Circular buffer operation

The renderer shall maintain that Timestamp(write\_pointer) > Timestamp(read\_pointer). When overwriting a frame in the buffer with new timed data, the renderer shall make sure that the read\_pointer is moved to the frame with the oldest timestamp. This would result in a frame drop but will ensure that no concurrent access to the same frame in the buffer is performed.

# MPEG media and MPEG video texture extensions for glTF2 (m52838)

During the MPEG 128 meeting in Geneva the scope of the scene description standardization work was agreed and described in N18869 [4]. As an essential future the support for timed media processing in glTF was listed. During the ad-hoc group for scene description in Brussels two contributions from Nokia Technologies m52179[5] and from Qualcomm Incorporated m52462[6] were presented that proposed solutions to enable this future. This document describes the new MPEG media and MPEG video texture extensions that originated from merging the above input contributions. Also prose some of the already existing KHR extensions to be mandatory whem If MPEG scene description is supported.

## MPEG\_media extension

MPEG media extension, identified by MPEG\_media, provides an array of MPEG media items used in the scene.

If MPEG scene description is supported, then the MPEG\_media extension shall be supported. The MPEG media extension shall be included in the extensionsUsed and extensionsRequired of the scene description document for scene descriptions that require the use of mpeg media support.

The extension shall be declared at the top-level as follows:

{

"extensionsRequired": [

" MPEG\_media"

]

"extensionsUsed": [

" MPEG\_media"

]

}

The definition of all objects within MPEG\_media extension is provided in the tables below.

**Table 1** Definitions oftop-level objects of MPEG\_media extension

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Default** | **Description** |
| media | array | N/A | An array of items that list the media referenced by other object in a scene. (e.g reference by MPEG\_video\_texture) |

**Table 2** Definitions ofitem in the media array of MPEG\_media extension

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Default** | **Description** |
| name | string | N/A | Label of the media. |
| renderingRate | number | 25.0 | The renderingRate attribute is used to indicate the frequency at which the timed texture is expected to be updated as frames per second. |
| startTime | number | 0 | The startTime gives the time at which the rendering of the timed texture will be in seconds. By default, the referenced image will be rendered as a static texture until the startTime. A startTime of 0 means the presentation time of the current scene.  Either startTime or autoplay shall be present in glTF description. |
| timeOffset | number | 0 | The timeOffset indicates the time offset into the source, starting from which the timed texture shall be generated. The value is provided in seconds, where 0 corresponds to the start of the source. |
| autoplay | boolean | N/A | Specifies that the video will start playing as soon as it is ready.  Either startTime or autoplay shall be present in glTF description. |
| loop | boolean | N/A | Specifies that the video will start over again, every time it is finished. |
| controls | boolean | N/A | Specifies that video controls should be displayed (such as a play/pause button etc). |
| alternatives | array | N/A | An array of items that indicate alternatives of the same media (e.g. different video code used)" |

**Table 3** Definitions ofitems in the alternatives array of MPEG\_media extension

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Default** | **Description** |
| mimeType | string | N/A | The MPEG media's MIME type. |
| uri | string | N/A | The uri of the media. Relative paths are relative to the .gltf file.  Either uri or bufferView shall be present in glTF description. |
| bufferView | number | N/A | The index of the bufferView that contains the MPEG media. This is used instead of the uri property.  Either uri or bufferView shall be present in glTF description. |
| tracks | array | N/A | An array of items that list of all tracks in MPEG media container (e.g. mp4 file or DASH manifest) |

**Table 4** Definitions ofitems in the tracks array of MPEG\_media extension

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Default** | **Description** |
| track | string | N/A | URL fragments to access the track within MPEG media.  e.g.  DASH : Using MPD Anchors (URL fragments) as defined in Annex C of 23009-1 (Table C.1).  MP4: URL fragments as specified in Annex L of 14496-12. |

In the example below, two media items are listed by MPEG\_media object. The first media item contains only one item within alternatives which is a DASH manifest that contains one track. Even though there are no alternatives at the MPEG media level, DASH manifest may still have different Representation within Adaptation Set (but this is outside of the scope of the extension). The second media item contains two items within alternatives. The first one list an mp4 file that contains data compress using AVC codec while the second one list an mp4 file that contains data compress using HEVC codec. Each item within alternatives array has to have the same amount of track items within tracks object. However, each track item may contain different information, which depends on the structure of the MP4 file.

{

"extensions": {

"MPEG\_media": {

media: [

{

"name": "source 0",

"renderingRate": 25.0,

"timeOffset": 0.0,

"autoplay": "true",

"loop": "true",

"controls": "false",

"alternatives": [

{

"mimeType": "application/dash+xml",

"uri": "manifest.mpd",

"tracks": [

{

"track": "#track=1"

}

]

}

]

},

{

"name": "source 1",

"renderingRate": 30.0,

"startTime": 9.0,

"timeOffset": 10.0,

"loop": "true",

"controls": "false",

"alternatives": [

{

"mimeType": "video/mp4;codecs=\"avc1.42E01E\"",

"uri": "video1.mp4",

"tracks": [

{

"track": "#track\_ID=1"

},

{

"track": "#track\_ID=2"

}

]

},

{

"mimeType": "video/mp4;codecs=\"hev1.1.6.L93.B0\"",

"uri": "video2.mp4",

"tracks": [

{

"track": "#track\_ID=3"

},

{

"track": "#track\_ID=1"

}

]

}

]

}

]

}

}

}

## MPEG\_video\_texture extension

MPEG video texture extension, identified by MPEG\_video\_texture, provide the possibility to link a glTF texture object to a media and it's respective track listed by MPEG\_media object. MPEG video texture extension provides also a reference to the timed accessor, using timedAccessor object (see Section 3 of this document), where the decoded timed texture will be made available.

If MPEG scene description is supported, then the MPEG\_video\_texture extension shall be supported. The MPEG video texture extension shall be included in the extensionsUsed and extensionsRequired of the scene description document for scene descriptions that require the use of timed textures.

The extension shall be declared at the top-level as follows:

{

"extensionsRequired": [

" MPEG\_texture\_video"

]

"extensionsUsed": [

" MPEG\_texture\_video"

]

}

When the MPEG\_video\_texture extension is not supported, a texture buffer will be filled by data described by the standard glTF source object.

**Table 5** Definition oftop-level objects of MPEG\_video\_texture extension

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Default** | **Description** |
| source | number | N/A | Provides the index of the media listed by MPEG\_media extension used by this texture. |
| track | number | N/A | Provides the index of a track of a media object, indicated by source object and listed by MPEG\_media extension, used by this texture. |
| timedAccessor | number | N/A | Provides a reference to the timed accessor where the decoded timed texture will be made available. |

In the example below, two texture items are listed. Each texture item use MPEG\_video\_texture extension. The first texture item is linked with source 1 listed by MPEG\_media and track 0, and it is expected that decoded texture will be available in buffer indicated by timed accessor 2. The second texture item is linked with the same source 1 listed by MPEG\_media but with a different track, track 0, and it will be available in buffer indicated by timed accessor 3.

{

"textures": [

{

"sampler": 0,

"source": 1,

"extensions": {

"MPEG\_video\_texture": {

"source": 1,

"track": 0,

"timedAccessor": 2,

}

}

},

{

"sampler": 1,

"source": 0,

"extensions": {

"MPEG\_video\_texture": {

"source": 1,

"track": 1

"timedAccessor": 3,

}

}

}

]

}

## Support for dynamic texture coordinates

UV coordinates may vary with every new frame of the timed texture. The texture coordinates for a primitive as referenced by TEXCOORD\_0 and TEXCOORD\_1 will be expected to change. Furthermore, the geometry of the primitive itself and by consequence the vertex indices may change as well. The dynamic changes are described through references to timed accessors.

## Texture transforms and compression

A single timed texture may be used to provide texture for multiple primitives. This operation is typical when a texture atlas is used. The transform is performed prior to the texture coordinate mapping. In order to support texture transforms, the KHR\_texture\_transform shall be supported.

The texture shall be provided in an internal format that is supported by the GPU at rendering time. A compressed texture format is preferable to reduce the memory usage on the GPU.

# References

[1] “WD on Support of MPEG Media in Scene Descriptions”, ISO/IEC JTC1/SC29/WG11 output document N19070, January 2020, Brussels, BE.

[2] “Procedures for standard development and reference software of ISO/IEC 23090-14”, ISO/IEC JTC1/SC29/WG11 output document N19072, January 2020, Brussels, BE.

[3] Khronos Group, The GL Transmission Format (glTF), version 2.0, https://github.com/KhronosGroup/glTF/tree/master/specification/2.0#specifying-extensions

[4] “Considerations for a WD on MPEG-I Scene Descriptions”, ISO/IEC JTC1/SC29/WG11 output document N18869, October 2019, Geneva, CH.

[5] “[SD] MPEG media extensions for glTF2”, ISO/IEC JTC1/SC29/WG11 input document m52179, January 2020, Brussels, BE.

[6] “Timed Video Textures”, ISO/IEC JTC1/SC29/WG11 input document m52462, January 2020, Brussels, BE.