ISO/IEC JTC 1/SC 29/WG 03 N0613

**ISO/IEC JTC 1/SC 29/WG 03  
MPEG Systems   
Convenorship: KATS (Korea, Republic of)**

**Document type:** Output Document

**Title:** Exploration Experiments for MPEG-I Scene Description

**Status:** Approved

**Date of document:** 2022-08-12

**Source:** ISO/IEC JTC 1/SC 29/WG 03

**No. of pages:** 62 (with cover page)

**Email of Convenor:** young.L @ samsung . com

**Committee URL:** <https://isotc.iso.org/livelink/livelink/open/jtc1sc29wg3>

**INTERNATIONAL ORGANIZATION FOR STANDARDIZATION**

**ORGANISATION INTERNATIONALE DE NORMALISATION**

**ISO/IEC JTC 1/SC 29/WG 03 MPEG SYSTEMS**

**ISO/IEC JTC 1/SC 29/WG 03 N0613**

**August 2022, Virtual**

|  |  |
| --- | --- |
| **Title** | **Exploration Experiments for MPEG-I Scene Description** |
| **Source** | **WG 03, MPEG Systems** |
| **Status** | **Approved** |
| **Serial Number** | **21742** |

[1 EE1: Carriage of Random Access Support in Scene Description (closed) 1](#_Toc111215385)

[2 EE2: Dynamic Scene Update (closed) 1](#_Toc111215386)

[3 EE3: Codec Support in MPEG-I SD (ongoing) 1](#_Toc111215387)

[4 EE4: Haptics Support (closed) 23](#_Toc111215388)

[5 EE5: Generic Interactivity Framework (on-going) 23](#_Toc111215389)

[6 EE6: User Representation and Avatars (on-going) 40](#_Toc111215390)

[7 EE7: AR Anchoring (closed) 48](#_Toc111215391)

[8 EE8: Lighting (ongoing) 49](#_Toc111215954)

1. **EE1: Carriage of Random Access Support in Scene Description (closed)**

EE closed at MPEG #137. Please see WG03 N00383 for the latest description of this EE.

1. **EE2: Dynamic Scene Update (closed)**

EE closed at MPEG #137. Please see WG03 N00383 for the latest description of this EE.

1. **EE3: Codec Support in MPEG-I SD (ongoing)**
   1. **Background**

At the 136th MPEG meeting, WG7, WG4, and WG3 agreed jointly to establish a new EE as part of the MPEG-I Scene Description AHG to study and specify the necessary extensions to add support for the V3C codecs (V-PCC and MIV) in particular, and all immersive MPEG codecs in general.

The background of the discussion can be found in [1].

* 1. **Current Understanding**

It was established that when adding codec support, the following options are possible:

* Codec independent: in this option, all the necessary decoding and post-processing is performed to produce a primitive format that is natively supported by the Presentation Engine.
* Codec dependent: in this option, the Presentation Engine needs to have some level of support for the codec, in order to be able to render the object.
  + Variant a: in this variant, an intermediate uncompressed format is passed to the Presentation Engine for rendering. A Presentation Engine that supports this format may then load the appropriate shader programs to perform post-processing (e.g. 3D reconstruction) and rendering of the object. The Presentation Engine must support the intermediate format.
  + Variant b: in this variant, samples of the compressed stream are passed to the Presentation Engine for decompression, post-processing, and rendering. The Presentation Engine must support the compressed format.

These variants vary in the split of tasks between the media pipeline in the MAF and the Presentation Engine. The following diagram depicts example pipelines for these different options.



**Figure 1 Example Pipeline Options**

The following example show how the different options could be described in the MPEG-I SD glTF document:

**Table 1 Example glTF for Option 1**

|  |
| --- |
| *.*  *.*  *.*  {  *"name"*: ”vpcc\_longdress",  *"primitives"*: [  {  *"attributes"*: {  *"POSITION"*: 15,  *"COLOR\_0"*: 16  },  *"mode"*: 0  }  ]  }  *.*  *.*  *.*  *"extensions"*: {  *"MPEG\_media"*: {  *"media"*: [  {  *"name"*: "longdress",  *"timeOffset"*: 0.0,  *"alternatives"*: [  {  *"mimeType"*: "video/mp4;codec=v3e1.L2.0.0.1, avc1.4D401E",  *"uri"*: "https://example.com/vpcc\_longdress.mp4"  }  ]  }  ]  }  }, |

**Table 2 Example glTF for Option 2a**

|  |
| --- |
| {  *"name"*: "vpcc\_longdress",  *"primitives"*: [  {  *"attributes"*: {  *"\_MPEG\_V3C\_ATLAS\_0"*: 1136,  *"\_MPEG\_V3C\_GEOMETRY\_0"*: 1134,  *"\_MPEG\_V3C\_OCCUPANCY\_0"*: 1135,  *"\_MPEG\_V3C\_COLOR\_0"*: 1137  },  *"mode"*: 0  }  ]  }  *.*  *.*  *.*  *"extensions"*: {  *"MPEG\_media"*: {  *"media"*: [  {  *"alternatives"*: [  {  *"mimeType"*: "video/mp4;codec=v3e1.L2.0.0.1, avc1.4D401E",  *"tracks"*: [  {  *"track"*: "#track\_ID=1"  }  ],  *"uri"*: "https://example.com/vpcc\_longdress.mp4"  }  ],  *"loop"*: true,  *"timeOffset"*: 0  }  ]  }  }, |

**Table 3 Example glTF for Option 2b**

|  |
| --- |
| {  *"name"*: "vpcc\_longdress",  *"primitives"*: [  {  *"attributes"*: {  *"\_MPEG\_V3C\_POINTCLOUD"*: 165,  },  *"mode"*: 0  }  ]  }  *.*  *.*  *.*  *"extensions"*: {  *"MPEG\_media"*: {  *"media"*: [  {  *"alternatives"*: [  {  *"mimeType"*: "video/mp4;codec=v3e1.L2.0.0.1, avc1.4D401E ",  *"tracks"*: [  {  *"track"*: "#track\_ID=1"  }  ],  *"uri"*: " https://example.com/vpcc\_longdress.mp4"  }  ],  *"loop"*: true,  *"timeOffset"*: 0  }  ]  } |

* 1. **Mandates**

This EE will have the following mandates:

* Identify the MPEG codecs to be supported in MPEG-I SD
* Define the MIME type and any necessary signaling and extensions to enable options 1 and 2
* For codec dependent support:
  + Evaluate the codec-dependent options and decide on which ones to enable
  + Define the exact buffer formats and any necessary restrictions on the formats
  + Define any necessary glTF extensions and register any new \_MPEG attributes
  + Provide guidelines on how to implement the variant
* Provide test scenarios, assets, and implementation in the reference software
  1. **Support for V-PCC**
     1. **m59306 [SD][EE3] Supporting YUV textures**

**MPEG\_YUV sampler**

Modern graphics APIs provide a *sampler* structure that can be associated to a texture object such that sampling operations for the YCbCr color space can be natively supported on modern GPUs. A sampler-level extension is described to sample a video texture natively in parallel processing devices such as GPUs. The format of the video texture, such as “VK\_FORMAT\_G8\_B8\_R8\_3PLANE\_444\_UNORM” and other formats defined in [11] may be provided in the MPEG\_video\_texture.format property.

A texture object in the textures array may use a sampler with the “MPEG\_YUV” sampler extension to provide information to the Presentation Engine on how to sample the video texture when the texture format is a chroma format such as YCbCr.

"samplers": [

        {

"extensions" : {

                "MPEG\_YUV": {

                    "modelConvernsion": "#709",

                    "range": "#ITU range",

                    "chromaFilter": "#CHROMAFILTER",

                    “xchromaOffset": "#CHROMAOFFSET",

                    "yChromaOffset": "CHROMAOFFSET",

                }

            }

        }

    ],

Table MPEG YUV sampler semantic

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Type** | **Default** | **Usage** | **Description** |
| modelConversion | String | - | O | Describes the Color model component of a color space |
| range | String | - | O | Describes whether color components are encoded using the full range of numerical values or whether values are reserved for headroom and foot room. |
| chromaFilter | String | - | O | Describes filters used for texture lookups |
| components | String | - | O | Describes the order of the components |
| xChromaOffset | String | - | O | Describes the X location of downsampled chroma component samples relative to the luma samples. xChromaOffset has no effect for formats in which chroma components are not downsampled horizontally. |
| yChromaOffset | String | - | O | Describes the Y location of downsampled chroma component samples relative to the luma samples. yChromaOffset has no effect for formats in which chroma components are not downsampled vertically. |

The MPEG\_YUV sampler extension provides relevant configuration information for the shader compiler to read and sample a YUV texture.

**MPEG\_texture\_video**

Table MPEG Texture video semantic

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Type** | **Default** | **Usage** | **Description** |
| accessors | ~~Integer~~  Array | N/A | M | Provides a reference to the accessor(s), by specifying the accessor('s/s’) index in accessors array, that describes the buffer where the decoded timed texture will be made available.  In case of planar data alignment, there must be at most 4 accessors (e.g., RGBA) in the accessors property. Each accessor will refer to the data for a plane.  The accessor shall have the MPEG\_accessor\_timed extension.  The type, componentType, and count of the accessor depend on the width, height, and format. |
| width | integer | N/A | M | Provides the maximum width of the texture. |
| height | integer | N/A | M | Provides the maximum height of the texture. |
| format | string | RGB | O | Indicates the format of the pixel data for this video texture. The allowed values are: RED, GREEN, BLUE, RG, RGB, RGBA, BGR, BGRA, DEPTH\_COMPONENT. The semantics of these values are defined in Table 8.3 of OpenGL specification [OpenGL 4.6].  Note that the number of components shall match the type indicated by the referenced accessor. Normalization of the pixel data shall be indicated by the normalized attribute of the accessor. |
| subSampling | string | 444 | O | Describes the sub-sampling format of the source of the texture. The allowed values of the sub-sampling format are 444, 420, 422, 440 and 411. |
| alignment | string | INTERLEAVED | O | Specifies the data alignment of the source. Three allowed values for the data alignment property are INTERLEAVED, PLANAR and SEMI-PLANAR. |

In case the source of the video texture is a sub-sampling planar format, each accessor item in the MPEG\_video\_texture.accessors array property will refer to a plane. The size of each plane is determined by the sub-sampling format as well as the dimensions of the video texture. For instance, in case of 4:2:0 video texture, a video texture frame data will of bytes. The first accessor will point to data storing Y channel i.e., of bytes. The second and third accessor will refer to data storing U and V channel, each storing bytes. The default behavior for the MAF would be to provide interleaved 444 RGB textures.

* + 1. **m59656 [SD] V-PCC in Scene Description**

A mesh primitive that references a V3C formatted representation of a 3D object should have the following attributes:

* \_MPEG\_V3C\_GEOMETRY: this attribute shall reference a timed accessor that provides the decoded geometry video data. Exactly one geometry buffer shall be associated with an atlas data buffer.
* \_MPEG\_V3C\_OCCUPANCY: this attribute shall reference a timed accessor that provides the decoded occupancy video data. Exactly one occupancy map buffer shall be associated with an atlas data buffer.
* \_MPEG\_V3C\_ATTRIBUTE\_i: this attribute shall reference a timed accessor that provides the decoded attribute video data that corresponds to the ith attribute. The 0th attribute shall correspond to color data in YUV 4:2:0 format.
* \_MPEG\_V3C\_ATLAS\_V1: this attribute shall reference a timed accessor that provides the V3C atlas data buffer. The atlas buffer format is defined in section 3. Future specifications of the atlas data buffer format shall use a different attribute name, e.g. by incrementing the version number in \_MPEG\_V3C\_ATLAS\_V**1**.

Each mesh primitive shall reference exactly one atlas data buffer.

The atlas data buffer is binary formatted data that shall comply to the following format:

|  |  |  |
| --- | --- | --- |
| **Field** | **Type** | **Description** |
| **frame\_width** | uint32 | indicates the frame width in luma samples of the atlas and all other associated V3C components. |
| **frame\_height** | uint16 | indicates the frame height in luma samples of the atlas and all other associated V3C components. |
| **attribute\_count** | uint8 | the number of attributes of the V3C object. This number shall match the number of MPEG\_V3C\_ATTRIBUTE\_i elements in the mesh primitive. |
| for( i=0;i<attribute\_count;i++) { |  |  |
| **attribute\_type** | uint8 | the attribute type of the ith attribute. The types are defined in Table 3 of ISO/IEC 23090-5 [1]. |
| } |  |  |
| **scale** | float[3] | indicates the global scale factor that shall be applied to the reconstructed V3C object. |
| **offset** | float[3] | indicates the global translation vector that shall be applied to the reconstruction V3C object |
| **rotation** | float[4] | indicates the global rotation that shall be applied to the reconstructed V3C object as a quaternion. |
| **patch\_count** | uint16 |  |
| for( i=0;i<patch\_count;i++ ) { |  |  |
| **2d\_pos\_x** | float | specifies the x-coordinate of the top-left corner of the patch bounding box for the current patch. |
| **2d\_pos\_y** | float | specifies the y-coordinate of the top-left corner of the patch bounding box for the current patch. |
| **2d\_size\_x** | float | specifies the width of the current patch. |
| **2d\_size\_y** | float | specifies the height of the current patch. |
| **3d\_offset\_u** | float | specifies the shift to be applied to the reconstructed patch points in the current patch along the tangent axis. |
| **3d\_offset\_v** | float | specifies the shift to be applied to the reconstructed patch points in the current patch along the bi-tangent axis. |
| **3d\_offset\_d** | float | specifies the shift to be applied to the reconstructed patch points in the current patch along the normal axis. |
| **patch\_projection\_id** | uint8 | specifies the identifier of the projection mode and the index of the normal to the projection plane of the current patch. |
| **patch\_orientation** | uint8 | specifies the index of the patch orientation of the current patch. |
| } |  |  |

* 1. **Support for V3C**
     1. **m58918 - InterDigital Response to EE3**

An option for media pipeline is envisioned where the MAF performs the decoding, and any necessary processing and the presentation engine performs the 3D reconstruction. The option is informally called as pipeline option #2A.

In the V3C specification, V3C profiles follow a structured and flexible definition to allow for clearly identifying two distinct conformance points namely conformance point A and conformance point B (see Annex A in ISO/IEC 23090-5 [2]). The first conformance point, point A, covers the decoded video sub-bitstreams and atlas sub-bitstream. It also covers the derived block to patch map information in atlas sub-bitstream. The second conformance point, point B, covers the reconstruction process.

Following the definition in V3C and the design goals for pipeline option #2A, the MAF can be assumed to perform operations associated with conformance point A in the V3C specification. On the other hand, the Presentation Engine is responsible for performing the operations associated with conformance point B, as shown in Figure 1.

It is necessary to express the intermediatory formats for the different V3C components such that the Presentation Engine can use the information in relevant buffer/texture formats for 3D reconstruction. The MAF performs decoding and processes the decoded V3C components. The result of MAF processing is a representation of the different decoded and processed V3C components in formats that are consumable by the Presentation Engine.

In response to EE3, this contribution proposes a new extension to support V3C content in ISO/IEC 23090-14.

Diagram

Description automatically generated

Figure 2 An overview of the glTF document structure with MPEG extensions and MPEG\_V3C extensions

**MPEG\_V3C scene-level extension**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Type** | **Default** | **Usage** | **Description** |
| objects | Array | [] | - | Array of V3C objects |

**MPEG\_V3C node-level extension**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Type** | **Default** | **Usage** | **Description** |
| object | number | - | M | An index of a V3C object in the objects array in the scene-level MPEG\_V3C extension. |

**V3C\_ATLAS**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Type** | **Default** | **Usage** | **Description** |
| patchBlockSize | Number | 16 | M | Describes the patch block size of the atlas frame |
| blockToPatchInformation | Number | - | M | Index in the accessor array which refers to the block to patch information data |
| totalPatches | Number | - | M | Index in the accessor array which holds the information on number of patches |
| commonPatchParameters | Number | - | M | Index in the accessor array which holds the information on common patch parameters |
| patchInformation | Array | [] | M | Array of patch types and their respective information |

**V3C\_ATLAS.patchInformation**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Type** | **Default** | **Usage** | **Description** |
| type | Number | - | M | Describes the type of patch |
| PLRDLevel | Number | - | O | Index in the accessors array which holds the information whether the PLR is at block level or patch level.  Only applicable if patch type is PROJECTED. |
| PLRDPresentBlockFlag | Number | - | O | Index in the accessors array which holds the information on presence of block level PLR mode.  Only applicable if patch type is PROJECTED. |
| PLRDBlockModeMinus1 | Number | - | O | Index in the accessors array which holds the information on block level PLR mode.  Only applicable if patch type is PROJECTED. |
| PLRDPresentFlag | Number | - | O | Index in the accessors array which holds the information on presence of patch level PLR Mode.  Only applicable if patch type is PROJECTED. |
| PLRDModeMinus1 | Number | - | O | Index in the accessors array which holds the information on patch level PLR mode.  Only applicable if patch type is PROJECTED. |
| patchAssociatedPatchIndex | Number | - | O | Index in the accessors array which specifies the index of the patches associated EOM patches.  Only applicable if patch type is EOM. |
| patchEOMPoints | Number | - | O | Index in the accessors array which holds the information on the number of EOM coded points.  Only applicable if patch type is EOM. |
| numberRAWPoints | Number | - | O | Index in the accessors array which holds the information on the number of raw coded points.  Only applicable if patch type is RAW. |

**V3C\_OCCUPANCY and V3C\_GEOMETRY**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Type** | **Default** | **Usage** | **Description** |
| index | Number | - | M | The index of a texture object in the textures array that is associated with the V3C component |

**V3C\_ATTRIBUTE**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Type** | **Default** | **Usage** | **Description** |
| index | Number | - | M | The index of a texture object in the textures array that is associated with the V3C attribute component |
| type | Number | - | M | Key identifier for the V3C attribute type, as defined in Table 3 of ISO/IEC 23090-5. |

**V3C\_ATTRIBUTE.type**

|  |  |
| --- | --- |
| **Attribute values** | **Attribute type** |
| 0 | Texture |
| 1 | Material ID |
| 2 | Transparency |
| 3 | Reflectance |
| 4 | Normals |
| 5..14 | Reserved |
| 15 | Unspecified |

* + 1. **m59287 [SD] InterDigital's EE3 response**

The V3C atlas buffer format consists of two main structures:

1. *Block-to-patch map:* Maps each block in an atlas frame to a patch index. In case the block is not covered by any patch, the patch index value assigned to that block is -1.
2. *Patch list:* The data for each patch is contained in a patch list. A patch consists of two sets of information.
   1. Common Patch Information data
   2. Application-Specific Patch information data. There may be additional information depending on the type of patch e.g., Point Local Reconstruction, EOM points etc., which needs to be stored in the atlas buffer.

**Atlas frame format**

An atlas frame is defined as arrays of V3C atlas syntax structure which form the complete atlas frame for a V3C content. The syntax elements in an V3C atlas frame are defined as references to accessors containing corresponding data. `BlockToPatchMap` property corresponds to patch index per-block. `CommonPatchParameters` property corresponds to common patch parameters (please see section 9.2.7.3.2 ISO/IEC 23090-5). These patch parameters are stored in the following order.

Table X. Ordering of common patch parameters

|  |
| --- |
| Common patch parameters |
| PatchInAuxVideo |
| PatchType |
| Patch2dSizeX |
| Patch2dSizeY |
| Patch2DPosX |
| Patch2DPosY |
| Patch3dOffsetU |
| Patch3dOffsetV |
| Patch3dOffsetD |
| Patch3dRangeD |
| PatchProjectionID |
| PatchOrientationIndex |
| PatchLODScaleX |
| PatchLODScaleY |
| PatchRAWPoints |
| PatchEOMPatchCount |

Depending on the type of patch i.e., PROJECTED, EOM or RAW, additional information may be provided. For instance, an atlas frame consists of PROJECTED patches, each projected patch may have point-local reconstruction (PLR) information. Whether the PLR information for a patch is available on a block-level or patch-level is provided through `PLRLevel` and corresponding PLR data is provided in `BlockPLRD` and/or `PatchPLRD`. `BlockSize` corresponds to a value of the patch packing blocking size.

The following example defines properties for the V3C atlas component of a V3C content containing PROJECTED patches as well as EOM patches.



Valid accessor type and component type for each property of a V3C atlas frame are defined in Table X.

Table X. V3C atlas properties

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | Usage | Accessor type(s) | Component type(s) | Description |
| BlockToPatchMap | M | SCALAR | Unsigned int | Store stores patch index for every block in an atlas frame. In case, a block is not assigned with a patch index, the block is assigned with value 0. |
| NumberOfPatches | M | SCALAR | Unsigned int | Stores the information on total of number of patches as well as total number of different patch types. |
| CommonPatchParameters | M | VEC2 SCALAR | Unsigned int | Stores common patch parameters per patch in an atlas frame. |
| PLRLevel | O | VEC2 | Unsigned int | Stores the PLRD level information for each PROJECTED patch type. In case, the PLR level is 0, the PLR information is available on per block level. Else if PLR level is 1, the PLR information is available on the patch level. |
| BlockPLRD | O | VEC3 | Unsigned int | Stores block-level PLRD information for PROJECTED patch type. |
| PatchPLRD | O | VEC3 | Unsigned int | Stores patch-level PLRD information for PROJECTED patch type. |
| EOMPatchInfo | O | VEC3 | Unsigned int | Stores application-specific information related to EOM patches. |

**Atlas buffer reader**

A set of accessors provide the means to access specific information for the patches in an atlas frame whilst referring to a single binary buffer element. A single buffer will be referenced by a set of bufferViews and each bufferView will have its own accessor element. The use of accessors provides the convenience to the presentation engine of referring to specific information for the patches contained in a decoded atlas frame.

A picture containing timeline

Description automatically generated

Figure . An example of reading from a decoded atlas frame

* + 1. **m59305 [SD][EE3] On V3C Support in MAF Pipeline Option 2a**

It is proposed to introduce a new MPEG extension MPEG\_objects that declares the media objects and associates them to nodes in the scene graph. Information pertaining to the specific codec used by a media object may be described by defining codec-specific extensions to this extension such as the MPEG\_V3C extension described in our previous contribution.

Similar to meshes, the media objects listed by the *MPEG\_objects* extension must be attached to nodes by defining a node.extensions.MPEG\_objects property that provides an index into the objects array using an object property. A media object that is associated with a particular node will inherit the transform of that node.

In a glTF file, *MPEG\_objects* extension shall be added to the extensionRequired and extensionUsed top-level glTF arrays when MPEG media objects are present in the scene.



Figure – An overview of the glTF document structure with the current MPEG extensions and the MPEG\_objects and MPEG\_V3C extensions

An example of how this association is signalled in the glTF file is given below, where the nodes array contains one node and the objects array of the *MPEG\_objects* extension contains three media objects (two V3C objects and one G-PCC object). The sole node in the example is associated with the first media object in the objects array via the index assigned to the object property of node.extensions.MPEG\_objects.

"extensionsRequired": [

    "MPEG\_objects",

    "MPEG\_V3C"

],

"extensionsUsed": [

    "MPEG\_objects,

    "MPEG\_V3C"

],

"nodes": [

    {

        "matrix": {},

        "extensions": {

            "MPEG\_objects": {

                "object": 1

             }

        }

    }

],

"extensions":

{

    "MPEG\_objects": {

        "objects": [

        {

            "extensions": {

               "MPEG\_V3C": {

                 // describes the V3C components

                }

            }

        },

        {

            "extensions": {

               "MPEG\_V3C": {

                 // describes the V3C components

               }

            }

        },

        {

            "extensions": {

               "MPEG\_GPCC": {

                 // describes the GPCC components

               }

            }

        }

    ]

    }

}

### Semantics

The definition of all objects within the *MPEG\_objects* extension is provided in Table 1.

Table 6 – Definition of the MPEG\_objects extension

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Type** | **Default** | **Usage** | **Description** |
| objects | array | [] | - | An array of media objects which are coded using MPEG codecs. |
| object | number |  | - | Index to an object in the objects array of the *MPEG\_objects* extension. |

### Processing Model

A glTF file that includes at least one MPEG media object shall include the MPEG\_objects property in the top-level extensionsRequired array. If the MPEG\_objects property is present in the top-level extensionsUsed array, a Presentation Engine supporting the *MPEG\_objects* extension reads the corresponding property in the top-level extensions array to identify the media objects that may be present in the scene. Processing of the *MPEG\_objects* extension depends on the codec used for the media object. When rendering the scene, the Presentation Engine identifies the media object codec through the codec-specific extension associated with the object in the *MPEG\_objects* extension and reads the MPEG object data using accessors and textures referenced by the codec-specific extension.

If MPEG\_objects is in the extensionsRequired array and the Presentation Engine does not support the *MPEG\_objects* extension, the Presentation Engine must fail loading the scene.

### Relation to MPEG\_media extension

Within the ISO/IEC 23090-14 architecture, the *MPEG\_objects* extension would serve a different but complementary function to the *MPEG\_media* extension. The *MPEG\_objects* extension enables describing media objects within a scene that require further processing by the Presentation Engine after any initial decoding and processing done by the MAF pipeline. Similar to how meshes in glTF 2.0 point to accessors from which the mesh data can be read when rendering, the media objects listed by the *MPEG\_objects* extension also point to accessors and textures that the Presentation Engine can read the resulting intermediate media data from. The *MPEG\_media* extension is still needed for identifying the sources from which the MAF can fetch the input to the MAF processing pipeline.

* + 1. **m59596 About Pipeline #2a reflecting design principle of 23090-10**

V3C content is composed of two types of components, V3C atlas and V3C components, where V3C atlas is the entry point of decoding. Therefore, scene description must list all the components while distinguishing V3C atlas from V3C component and indicate V3C components must not be processed unless a V3C atlas they belong to are not selected.

Current MPEG\_media extension does not allow distinction of V3C component from V3C atlas item. If V3C components are listed in MPEG\_media then it would be considered as media items individually referenceable. So, new extension specifically listing the components items which shall not be directly referenced and only be processed together with other media items separately from the media items independently referenceable, MPEG\_media\_compound is proposed in 4.1.

In addition, current MPEG\_buffer\_circular can only reference items in the media array of MPEG\_media. So, it needs to be also extended to reference the items in the components array when MPEG\_media\_compound is used. So, modification in 4.2 is also proposed.

## Definition of MPEG\_media\_compound extension

1. * 1. **MPEG\_media\_compound extension**
        1. **General**

The compound MPEG media extension, identified by MPEG\_media\_compound provides two arrays of the media items. The media array provides the list of media which can be directly referenced. The components array provides the list of media to be used as a component of one of the items in the media array. Definition of media array is exactly same as MPEG\_media extension.

The items in the components array shall have one reference to the item in the media array.

* + - 1. **Semantics**

The definition of all objects within MPEG\_media\_compound extension is provided in Table X and Table Y.

**Table X – Definitions of top-level objects of MPEG\_media extension**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Type** | **Default** | **Usage** | **Description** |
| media | array | N/A | M | identical with media property of MPEG\_media extension |
| components | array | N/A | M | An array of items that describe the external media used as a component of an item in the media array. |

**Table Y – Definitions of item in the components array of MPEG\_media\_compound extension**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Type** | **Default** | **Usage** | **Description** |
| reference\_media | integer | N/A | M | Index of the media entry which the current item is used as a component of |
| alternatives | array | N/A | M | identical with MPEG\_media.media.alternatives |

* + - 1. **Processing Model**

Processing of the MPEG\_media\_compound extension is identical with MPEG\_media extension except the processing of the items in components array. In general, items in the components array may be referenced by a circular buffer and processed synchronously with the items in the media array that is referenced by it.

## Amending definition of MPEG\_buffer\_circular.media

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| media | integer | N/A | M | Index of the media entry in the MPEG\_media extension, which is used as the source for the input data to the buffer.  If the MPEG\_media\_compound extension is used, the items in the components array is indexed continuously after the items in the media array without any gap. For example, if there are 4 items in the media array then the index of the first item in the component array becomes 4. |

**Proposed text for Annex G**

Annex G of ISO/IEC 23090-14 supposed to describe how MPEG-I media is supported. The draft text of that section does not clearly explain which type of Pipeline should be created. With the proposal in section 4, a way to select type of Pipeline can be clearly defined. Following is a paragraph proposed to be added to G.1.1

Various type of Pipelines for processing of V3C contents can be established depending on the location of decoding and 3D reconstruction. Selection of the type of Pipelines is indicated by how the external references for a V3C content are provided in scene description. When a scene description contains only one reference to a V3C atlas then decoding and 3D reconstruction of volumetric frames are done by MAF and volumetric frames are delivered to Presentation Engine through a single Buffer. When a scene description contains the references to all external media resources comprising a V3C content, both V3C atlas and all V3C components, then MAF instantiates the media decoders for each external media resource and the Buffers for each media decoders are also established. Each external media resource is individually decoded and decoded raw media data are individually delivered to Presentation Engine through the Buffers. Presentation Engine reconstruct volumetric frames by its own 3D reconstruction process.

* + 1. **m59597 About Pipeline #2b reflecting design principle of 23090-10**

To show that V3C content stored in multiple tracks can also be processed with Pipeline #2b, following description and figure are proposed to be added.

When a V3C content is stored in multiple tracks and compressed bitstreams should be delivered to the Presentation Engine for decompression, 3D reconstruction, and rendering, the samples from all tracks with same CTS are multiplexed into a single unit and delivered to Presentation Engine through a single Buffer as shown in Figure 2.

Shape

Description automatically generated with medium confidence

Figure 2 — Pipeline #2b with multiple track case

## Scene description extension for Pipeline #2b

As there will be only one Buffer delivering data from MAF to Presentation Engine, scene description must list only one external media resource. However, there should be an indication that whether decoding and reconstruction must be done by MAF or not. Therefore, it is proposed to add following property to MPEG\_media extensions

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| decoding | boolean | True | O | Specifies whether decoding is performed by MAF or not. If this property is set to False then compressed bitstream is delivered to Presentation Engine without decoding. |

* 1. **Support for MIV**
     1. **MIV player from m58430**

As part of the scene description EE on codec support[[1]](#footnote-1), Philips has provided a software example for a MIV renderer.

The provided real-time renderer is a simple example to "get going" and does not have sufficient quality in general, but as this EE is more of an implementation effort, we believe that this example is suitable.

The OpenGL ES 3.2 shaders are included as raw string literals within vr.scene.zmin.cpp, and the C++ code is only provided to create a running example.

* + 1. **Build instructions**

1. Unpack the archive at any location.
2. Add pre-built or self-built external libraries:
   * GLEW 2.1 to C:\X\ext\glew
   * GLFW 3.3 or newer to C:\X\ext\glfw
   * OpenCV 3.4.13 to C:\X\ext\opencv-3.4.13\_install
   * In case of confusion, please study the file C:\X\files.txt. It has the output of Get-ChildItem -Recurse. We prefer not to provide binaries to avoid software licensing problems.
3. For the purpose of this description, this document is located at C:\X\README.md
4. Open the folder C:\X\ee\_on\_miv\_support\_in\_sd in a terminal
5. mkdir build
6. cd build
7. cmake -DOpenCV\_DIR=C:\X\ext\opencv-3.4.13\_install ..
8. Open Visual Studio 2019 to build in Release mode with platform x64
9. Copy C:\X\ext\glew\bin\win64\glew32.dll to C:\X\ee\_on\_miv\_support\_in\_sd\build\AppsGL\Release\
10. Copy "C:\X\ext\opencv-3.4.13\_install\x64\vc16\bin\opencv\_world3413.dll" to C:\X\ee\_on\_miv\_support\_in\_sd\build\AppsGL\Release\
    * 1. **Run instructions**
11. On a multi-GPU system, make sure that the right GPU is selected. For NVIDIA:
    1. Control panel
    2. 3D Settings
    3. Manage 3D Settings
    4. Preferred graphics processor
    5. High-Performance NVIDIA processor
12. Start run.bat
    * 1. **Location**

* Software: <http://mpegx.int-evry.fr/software/MPEG/MIV/other/miv-player-example>
* Documentation: <http://mpegx.int-evry.fr/software/MPEG/MIV/other/miv-player-example/README.md>
* Zip-file with software and example data on the MPEG content server:  
  /MPEG-I/Part12-ImmersiveVideo/test\_material/m58999 SD-EE on Codec Support
  + 1. **Test data**

The example data includes one frame of Museum (ERP, 3DoF+) and one of Kitchen (PSP, 6DoF window) each with three pose traces.

* + 1. **License**

The software has the typical ISO/IEC modified BSD license.

External libraries are not included to avoid software license issues, but the documentation includes build instructions.

* 1. **Support for G-PCC**
  2. **Participants**

The following EE participants are identified:

|  |  |  |
| --- | --- | --- |
| **Participant** | **Affiliation** | **NB** |
| Imed Bouazizi | Qualcomm | US |
| Basel Salahieh | Vimmerse | US |
| Lauri Illola | Nokia | DE |
| Lukasz Kondrad | Nokia | DE |
| Ahmed Hamza | Interdigital | CA |
| Gurdeep Bhullar | Interdigital | CA |
| Bart Kroon | Philips | NL |

* 1. **Timeline**
* 2022-01-12: MPEG document upload deadline
* 2022-01-17: MPEG #137(online) meeting starts
* 2022-04-25: MPEG #138(online) meeting starts
* 2022-07-18: MPEG #139(online) meeting starts
  1. **References**

[1] m58329, Codec Support in SD

[2] MIV Test Model, <https://dms.mpeg.expert/doc_end_user/documents/135_OnLine/wg11/MDS20596_WG04_N00112.zip>

1. **EE4: Haptics Support** **(closed)**



























EE closed at MPEG #139. Please see WG03 N0540 for the latest description of this EE.

1. **EE5: Generic Interactivity Framework (on-going)** 
   1. **Introduction**

The MPEG Scene Description solution has added support for timed media to glTF 2.0. A Media Access Function (MAF) offers an API to the Presentation Engine, through which timed media can be requested. The current scene description solution allows the user to consume the scene in 6DoF, thus, moving freely in the 3D scene. To offer a realistic experience, the viewer should be able to interact with objects in the scene in different ways.

* 1. **Problem statement**

The following aspects of interactivity are identified:

* the user cannot walk through obstacles in the scene (such as walls, chairs, tables, …)
* the user is able to interact with objects in the scene in a way that results in changes to the scene (e.g. turn on a TV, open a door, push objects, …)
* the user will perceive the changes caused by the interaction (e.g. visual, audio, and haptics feedback)

This EE will focus on developing the necessary extensions to support basic interactivity in scene description.

The following simplified architecture is identified as the baseline for a generic interactivity framework.

Shape

Description automatically generated with medium confidence

Triggers are events that will trigger some form of interactivity. Actions are the interactivity feedback. The TuC [1] currently contains a collision model that defines one form of interactivity trigger. Objects provide a simplified mesh that will allow for detection of collision with the viewer. A detected collision will trigger some interactivity actions, such as starting an animation, haptics and/or audio feedback.

* 1. **Use cases relevant for the EE**
  2. **Related (WG2) and Extracted (new) Requirements**

The following requirements are relevant and addressed by this EE:

* It shall be possible to discover user interactivity modules (requirement #85)
* It shall be possible to define custom interactivity procedures based on input from the user or from the user’s devices and sensors (requirement #86)
* Support of user interactivity with objects within a virtual environment (requirement #90)
* Support of interaction between multiple users within an immersive environment (requirement #95)
* The specification shall support interactivity models related to avatar position and orientation (requirement #129)
* The specification shall support coding and presentation of interactivity models related to avatar-scene or avatar-avatar interactions (requirement #130)
* The specification shall support different media types and various haptic feedback paradigms (requirement #131)
  1. **Relation to other activities (EE, requirements, etc…)**

A relationship to the Haptics phase 2 activity has been identified.

* 1. **Mandates**

The mandates for this EE are as follows:

* refine the generic interactivity framework reference architecture
* define a basic set of interactivity triggers
* define a basic set of interactivity actions that covers different media types
* define test scenarios and collect test assets
* evaluate proposed solutions
* develop the reference software integration and validate against the test scenarios
  1. **Participants**

|  |  |  |  |
| --- | --- | --- | --- |
| Participant | Contact | Email | Type |
|  |  |  |  |
| Qualcomm | Imed Bouazizi | bouazizi@qti.qualcomm.com | L |
| Immerse | Yeshwant Muthusamy | ymuthusamy@immersion.com | P |
| Interdigital | Fabien Danieau | fabien.danieau@interdigital.com | P |
| Interdigital | Patrice Hirtzlin | patrice.hirtzlin@interdigital.com | P |
| Interdigital | Gurdeep Bhullar | gurdeep.bhullar@interdigital.com | P |
| Xiaomi | Emmanuel Thomas | thomase@xiaomi.com | P |

(P = proponent, L = leader)

* 1. **Information about proposed technologies**

The following contributions on Interactivity have been submitted:

**Meeting #134**

[m56337](https://dms.mpeg.expert/doc_end_user/current_document.php?id=78240&id_meeting=0) [SD] Interactivity in Scene Description

**Meeting #135**[m57409 [SD] Interactivity support in scene description](https://dms.mpeg.expert/doc_end_user/current_document.php?id=79601&id_meeting=187)

**Meeting #136**

[m58104](https://dms.mpeg.expert/doc_end_user/current_document.php?id=80564&id_meeting=0) [SD] On scene interactivity

[m58146](https://dms.mpeg.expert/doc_end_user/current_document.php?id=80606&id_meeting=0) [SD] Describing camera paths for interactivity

**Meeting #137**

[m58486 [SD] Collision model for Interactivity](https://dms.mpeg.expert/doc_end_user/current_document.php?id=81216&id_meeting=189)

[m58794](https://dms.mpeg.expert/doc_end_user/current_document.php?id=81524&id_meeting=0) [SD] On interactivity support

**Meeting #138**

[m59773](https://dms.mpeg.expert/doc_end_user/current_document.php?id=82774&id_meeting=190) [SD] EE Interactivity – framework reference architecture

[m59774](https://dms.mpeg.expert/doc_end_user/current_document.php?id=82775&id_meeting=190) [SD] EE Interactivity – Use case proposal

**Meeting #139**

[m59898](https://dms.mpeg.expert/doc_end_user/documents/138_OnLine/wg11/m59898-v1-m59898.zip) [SD][EE5] On reference software implementation for interactivity support

[m60569](https://dms.mpeg.expert/doc_end_user/documents/139_OnLine/wg11/m60569-v2-m60569.zip) [SD] Interactivity technologies for AMD2

[m59961](https://dms.mpeg.expert/doc_end_user/documents/139_OnLine/wg11/m59961-v1-m59961.zip) [SD] EE8 – Interactive lighting use case

[m59896](https://dms.mpeg.expert/doc_end_user/documents/138_OnLine/wg11/m59896-v1-m59896.zip) [SD] EE Interactivity – Assets for the Scene Description Test Scenarios

* 1. **Extracted from TuC**
     1. **General**

In order to provide an immersive experience to the viewer, it is important that the viewer interacts properly with objects in the scene. The viewer should not be able to walk through solid objects in the scene, such as walls, chairs, and tables.

The following figure depicts a 3D mesh representation of a chair, together with its collision boundaries, defined as a set of cuboids.

Graphical user interface

Description automatically generated

* + 1. **Semantics**

The “MPEG\_mesh\_collision” extension is defined to provide a description of the collision boundaries of a mesh. The extension shall be defined on mesh objects as a set of cuboids around the mesh geometry.

It contains the following properties.

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Default** | **Description** |
| boundaries | Array(object) | N/A | Array of boundary shapes that are used to define the collision boundaries of the mesh object. The boundaries may be spheroids or cuboids, as defined in the MPEG\_camera\_control extension. |
| static | boolean | True | Determines if the object is affected by collisions or not. An object that is static will not be affected by collisions, which means that when the viewer or another object collides with this object, its position will not be altered. |
| material | number | N/A | The index of a collision material that defines how colliding objects or viewers will interact with this object. This may include bounciness, friction, etc. |
| animations | Array(object) | N/A | Defines animations that are triggered by a collision or action on this object. The animation may be limited to a subset of other objects, e.g. only the viewer may trigger this animation. It also contains a pointer to the animation that is to be executed when triggered. |
|  |  |  |  |

The mesh collision information consists of the cuboid vertex coordinates (x,y,z) for cuboid boundaries or the sphere center and radius for spherical boundaries. The values are provided as float numbers.

* + 1. **Processing Model**

The Presentation Engine shall support the MPEG\_mesh\_collision extension. The camera position (x,y,z) shall not be contained within one of the defined mesh cuboids at any point of time. Collision may be signaled to the viewer through visual, acoustic, and/or haptic feedback.

This information on the boundaries for the nodes may be used to initialize and configure a 3D physics engine that will detect collisions.

* 1. **Contribution m58794**
     1. **MPEG scene interactivity glTF extension**

A MPEG interactivity glTF extension, called MPEG\_scene\_interactivity, is introduced at the scene level as shown in Figure 5.



Figure 5 : MPEG interactivity glTF extension at scene level

MPEG\_scene\_interactivity glTF scene-level extension adopts a semantic approach based on the definition of behaviors, triggers and actions.

* + 1. **MPEG\_scene\_interactivity definition**

A behavior defines which kind of interactivity is allowed at runtime for dedicated virtual objects, corresponding to glTF nodes.

A behavior corresponds to a unique association of triggers and actions:

* the triggers define the runtime conditions to be met before executing the behavior actions
* the actions define how the behavior affects the scene
  + 1. **MPEG\_scene\_interactivity semantic**

The semantic of the MPEG\_scene\_interactivity glTF extension is shown in Table 3.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Type** | **Usage** | **Default** | **Description** |
| triggers | Array | M | [] | Contains the definition of all the triggers used in that scene |
| actions | Array | M | [] | Contains the definition of all the actions used in that scene |
| behaviors | Array | M | [] | Contains the definition of all the behaviors used in that scene. A behavior is composed of a pair of (triggers, actions), control parameters of triggers and actions, a priority weight and an optional interrupt action as detailed in 3.3 |

Table 7 : Semantic of the MPEG\_scene\_interactivity extension

* + 1. **Trigger semantic**

The semantic of a trigger is provided in Table 4.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Type** | **Usage** | **Default** | **Description** |
| type | enumeration | M | VISIBILITY | Defines the type of the trigger by taking one of the following values:  VISIBILITY = 0,  PROXIMITY = 1,  USER\_INPUT = 2,  TIMED = 3,  COLLIDER = 4 |
| activateOnce | Boolean | M | FALSE | If FALSE: the trigger is activated each time its conditions are met.  If TRUE: the trigger is activated once when its conditions are met.  Refer to Figure 6 |
| If(type== VISIBILITY){ |  |  |  |  |
| cameraNode | Number | M |  | Index to the node containing a camera in the nodes array for which the visibilities are determined |
| nodes | Array | M |  | Indices of the nodes in the nodes array to be considered. All the nodes shall be visible by the camera to activate the trigger |
| } |  |  |  |  |
| If(type == PROXIMITY){ |  |  |  |  |
| distanceLowerLimit | Number | M | 0 | Threshold min in meters for the node proximity calculation |
| distanceUpperLimit | Number | O |  | Threshold max in meters for the node proximity calculation |
| nodes | Array | M | [] | Indices of the nodes in the nodes array to be considered. All the nodes shall have a distance from the user camera above the distanceLowerLimit and below the distanceUpperLimit to activate the trigger |
| } |  |  |  |  |
| If(type == USER\_INPUT){ |  |  |  |  |
| userInputDescription | String | M |  | Describe the user body part and gesture related to the input. E.g. “/user/hand/left/grip” |
| nodes | Array | O |  | Indices of the nodes in the nodes array to be considered for this user input |
| } |  |  |  |  |
| If(type == TIMED){ |  |  |  |  |
| media | Number | M | 0 | Index of the media in the MPEG media array used to retrieve the media playback timeline |
| timeLowerLimit | Number | M | 0 | Indicates the start time offset into the media playback timeline at which the trigger is activated, in second. The default value of 0 means the activation of the trigger at the start of the media playback. |
| timeUpperLimit | Number | O |  | Indicates the end time offset into the media playback timeline at which the trigger is deactivated, in second. If not present, the trigger is active until the end of the media timeline. |
| } |  |  |  |  |
| If(type == COLLIDER){ |  |  |  |  |
| nodes | Array | M |  | Indices of the nodes in the nodes array to be considered for collision determination. Any detection of collision shall activate the trigger |
| } |  |  |  |  |

Table 8 : Semantic of a trigger

* + 1. **Action semantic**

The semantic of an action is provided in Table 5.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Type** | **Usage** | **Default** | **Description** |
| type | enumeration | M | ACTIVATE | Defines the type of the action by taking one of the following values:  ACTIVATE = 0,  TRANSFORM = 1,  ANIMATE = 2,  CONTROL\_MEDIA = 3,  PLACE\_AT = 4,  MANIPULATE = 5,  SET\_MATERIAL = 6 |
| delay | number | O |  | Duration of delay in second before executing the action |
| If(type == ACTIVATE){ |  |  |  |  |
| activationStatus | enum | M | ENABLED | ENABLED=0: the node shall be considered by the application  DISABLED =1: the node shall not be considered by the application |
| nodes | array | M | [] | Indices of the nodes in the nodes array to set the activation status |
| } |  |  |  |  |
| If(type== TRANSFORM){ |  |  |  |  |
| transform |  | M |  | 4x4 transformation matrix to apply to the nodes |
| nodes | array | M |  | Indices of the nodes in the nodes array to be transformed |
| } |  |  |  |  |
| If(type == ANIMATE){ |  |  |  |  |
| animation | number | M |  | index of the animation in the animations array to be considered |
| animationControl | enum | M | PLAY | PLAY = 0,  PAUSE = 1,  RESUME = 2,  STOP = 3 |
| } |  |  |  |  |
| If(type == CONTROL\_MEDIA){ |  |  |  |  |
| media | number | M |  | Index of the media in the MPEG media array to be considered |
| mediaControl | enum | M | PLAY | PLAY = 0,  PAUSE = 1,  RESUME = 2,  STOP = 3 |
| } |  |  |  |  |
| If(type == PLACE\_AT){ |  |  |  |  |
| placeDescription | string | M |  | Describe the place position. E.g. “/user/hand/left/pose” |
| nodes | array | M |  | Indices of the nodes in the nodes array to be placed. |
| } |  |  |  |  |
| If(type== MANIPULATE){ |  |  |  |  |
| action | enum | M | FREE | FREE= 0: the nodes follow the user pointing device and its rotation,  FREE\_FIXED\_ROTATION=1: the nodes follow the user pointing device but without rotation,  SLIDE=2: the nodes move linearly along the provided axis by following the user pointing device  TRANSLATE=3: the nodes translate by following the user pointing device,  ROTATE=4: the nodes rotate around the provided axis by following the user pointing device,  SCALE=5: performs a central scaling of the nodes by following the user pointing device |
| axis | array | O |  | (x,y,z) coordinates of the axis used for rotation and sliding. These coordinates are relative to the local space created by the USER\_INPUT trigger activation. E.g. a “/user/hand/left/pose” user input trigger creates a local space attached to the user left hand |
| nodes | array | M |  | Indices of the nodes in the nodes array to be manipulated |
| } |  |  |  |  |
| If(type == SET\_MATERIAL){ |  |  |  |  |
| material | number | M |  | Index of the material in the materials array to apply to the nodes |
| nodes | array | M |  | Indices of the nodes in the nodes array to set their material |
| } |  |  |  |  |

Table 9 : Semantic of an action

* + 1. **Behavior semantic**

The semantic of a behavior is provided in Table 6.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Type** | **Usage** | **Default** | **Description** |
| triggers | array | M |  | Indices of the triggers in the triggers array considered for this behavior |
| actions | array | M |  | Indices of the actions in the actions array considered for this behavior |
| triggersControl | enum | M | LOGICAL\_OR | LOGICAL\_OR = 0: an activation of any of the defined triggers shall execute the defined actions,  LOGICAL\_AND=1: all the defined triggers shall be activated to execute the defined actions |
| actionsControl | enum | M | SEQUENTIAL | Defines the way to execute the defined actions.  SEQUENTIAL=0: each defined action is executed sequentially in the order of the actions array,  PARALLEL=1: the defined actions are executed concurrently |
| interruptAction | number | O |  | Index of the action in the actions array to be executed if the behavior is still on-going and is no more defined in a newly received scene update |
| priority | number | M |  | Weight associated to the behavior. Used to select a behavior when several behaviors are active at same time for one node |

Table 10 : Semantic of a behavior

* + 1. **Processing model**

During runtime, the application iterates on each defined behavior and checks the realization of the related triggers following the procedure detailed in Figure 6.



Figure 6 : Processing model to activate a trigger

When the defined triggers of a behavior are activated, then the corresponding actions are launched.

A behavior has an “*on-going*” status between the launch and the completion of its defined actions.

When several behaviors are in concurrence to affect the same node at the same time, the behavior having the highest priority is processed for this affected node. The other concurrent behaviors are then not processed.

Once achieved, the application iterates on each behavior as defined in Figure 6.

If a node is affected by concurrent behaviors with a same priority value, then the application shall manage the potential conflict.

When a new scene description update is received, the application shall follow the procedure detailed in Figure 7.

A behavior is considered on-going when the related action(s) is(are) currently being executed when the scene update is processed.

A behavior is considered “*still defined*” if its unique association of (triggers, actions) is still described in the scene update.

If a behavior is no more “*still defined*”, its interrupt action is executed.

When all the interrupt actions (if any) are achieved, then the application removes any obsolete scene data and considers any new data to match the updated scene description.



Figure 7 : Processing model when a new scene description is received

* 1. **m59773 [SD] EE Interactivity – framework reference architecture**

*Triggers* and *actions* are the elementary elements that should be defined to build an interactivity framework

**trigger**

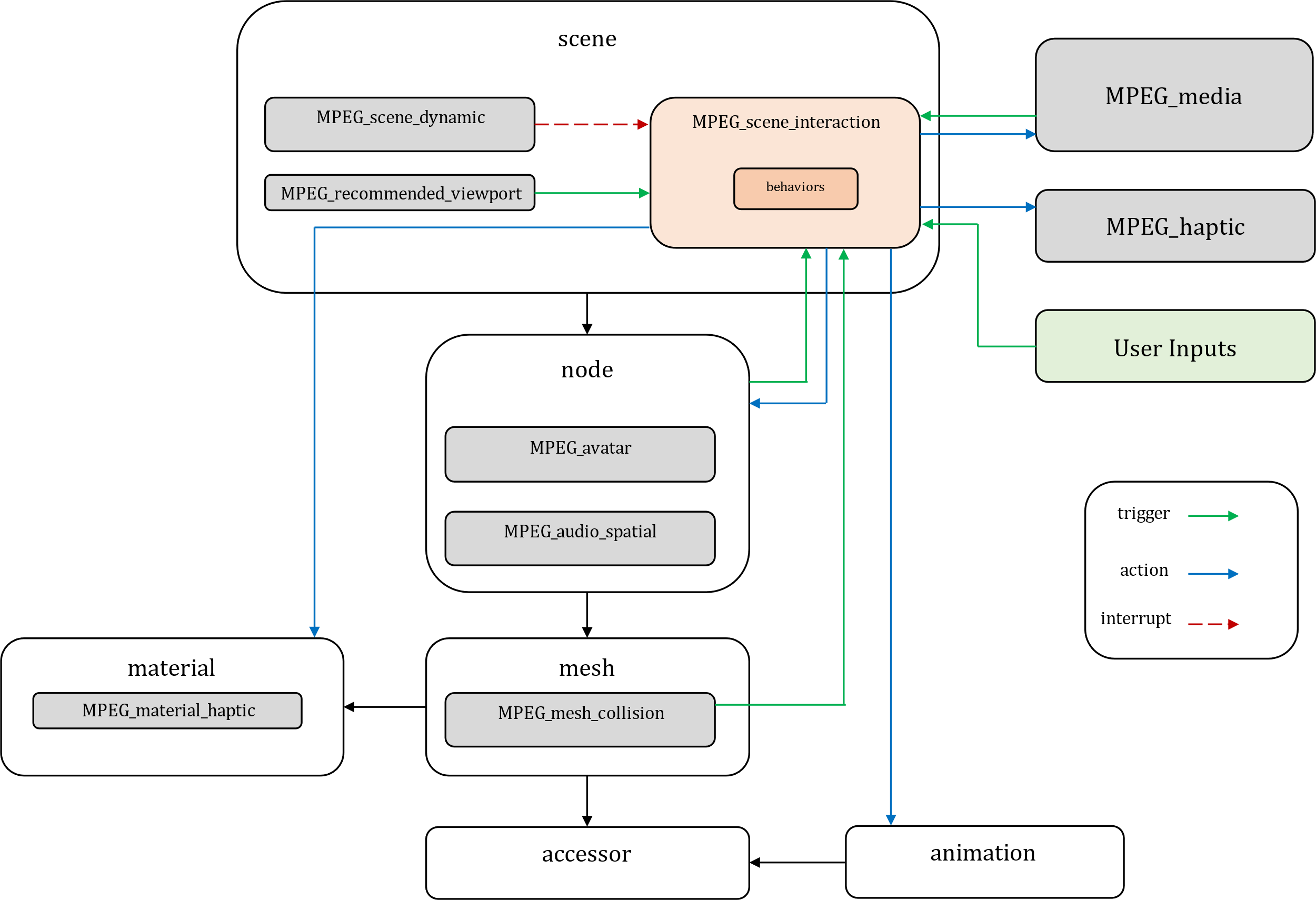
condition to be met at runtime, such as collision detection, visibility, proximity, timing or user input condition. A trigger is said activated when its defined condition is met.

**action**

action affecting the virtual scene, such as activating, transforming, animating a scene element, setting a material or controlling a media asset

**behavior**

unique mapping of triggers and actions



* 1. **Test cases**
     1. **Collision**
        1. **Description**

In this use case, yellow, red and gray balls are rolling on a surface. Only when yellow and red balls collide, a trigger is fired for actions execution. When the yellow and the gray balls or the red and gray balls collide, nothing happens.

In a first variant, the resulting action corresponds to change the color of the gray ball to blue.

In a second variant, there are two resulting actions, the material of the gray ball changes to blue color and simultaneously a sound is played.

In a third variant, there are two resulting actions, the material of the gray ball changes to blue color and after 5 seconds a sound is played.

A picture containing shape

Description automatically generated

Figure 1: Use case1

* + - 1. **Test Scenario**

|  |  |
| --- | --- |
| Item | Description |
| Title | Interactivity Use Case 1 (Collision) |
| Description | The use case is described in 2.1.1. It relates to the following requirements:   * It shall be possible to discover user interactivity modules (requirement #85) * Support of user interactivity with objects within a virtual environment (requirement #90) |
| Required test assets | * Scene with the following 3D objects:   + Plane   + 3 balls * Animation of balls (rolling on the plane) * Audio track for the sound |
| Current Support | The following features are supported:   * Support for 3D scenes, * Partial support for timed animations * Support for audio   Support for interactivity is missing  Support for animation is missing |
| Criteria | The test scenario is validated if upon collision detection, the following actions defined in use case are executed:   * color changing * or color changing and simultaneous sound playing * or color changing and delayed sound playing   When the yellow and the gray balls collide, nothing happens.  When the red and gray balls collide, nothing happens. |

* + 1. **Visibility** 
       1. **Description**

In this use case, a user is moving inside a virtual scene. The camera is associated to the user.

When a set of objects (3 spheres, 3 cones, 1 cylinder) are in the camera viewing frustum, a trigger is fired for actions execution.

In a first variant, the resulting actions are the following: spheres bump and simultaneously cones produce a specific sound. Nothing happens on the cylinder.

In a second variant the resulting actions are the following: spheres bump and cones produce a specific sound in a sequential way. Nothing happens on the cylinder.

Diagram

Description automatically generated

Figure 2:Use Case 2

* + - 1. **Test Scenario**

|  |  |
| --- | --- |
| Item | Description |
| Title | Interactivity Use Case 2 (Visibility) |
| Description | The use case is described in 2.2.1. It relates to the following requirements:   * It shall be possible to discover user interactivity modules (requirement #85) * The specification shall support interactivity models related to avatar position and orientation (requirement #129) |
| Required test assets | Scene with the following 3D objects:   * Plane * 3 Spheres * 3 Cones * 1 Cylinder   Animation of sphere (bump)  3 audio tracks for the sound |
| Current Support | The following features are supported:   * Support for 3D scenes, * Partial support for timed animations * Support for audio   Support for interactivity is missing  Support for animation is missing |
| Criteria | The test scenario is validated if upon visibility detection, the following actions defined in use case are executed:   * spheres bump and different sound are produced simultaneously by cones * or spheres bump and different sound are produced in a sequential way by cones   Nothing changes for the cylinder. |

* + 1. **Proximity**
       1. **Description**

An avatar moves closer to objects, then back away. When the distance of the avatar from the object group is under a certain distance (threshold), a trigger is fired for actions execution. When the avatar is closer than 5m to the group of objects, some actions are executed. Once the avatar goes back away and is further than 5m, actions are stopped.

In a first variant, the resulting actions are the following: spheres bump and simultaneously cones produce a specific sound. Nothing happens on the cylinder.

In a second variant a second trigger related to the visibility of the group of objects is added. An avatar moves closer to objects. When the distance of the avatar from the object group is under a certain distance (threshold) and the group of objects are in the camera frustum the actions describe in the first variant are executed. There is a logical combination of two triggers (AND combination).

A picture containing text, businesscard, vector graphics

Description automatically generated

Figure 3: Use Case 3 variant 1

* + - 1. **Test Scenario**

|  |  |
| --- | --- |
| Item | Description |
| Title | Interactivity Use Case 3 |
| Description | The use case is described in.2.3.1. It relates to the following requirements:   * It shall be possible to discover user interactivity modules (requirement #85) * The specification shall support interactivity models related to avatar position and orientation (requirement #129) |
| Required test assets | Scene with the following 3D objects:   * Table * 3 Spheres * 3 Cones * 1 Cylinder * Avatar (camera)   Animation of sphere (bump)  3 audio tracks for the sound |
| Current Support | The following features are supported:   * Support for 3D scenes, * Partial support for timed animations * Support for audio   Support for interactivity is missing  Support for animation is missing |
| Criteria | The test scenario is validated by successful trigger activation either proximity detection (variant 1) or proximity AND visibility detection (variant 2). The following actions defined in Use case are executed:   * spheres bump and different sound are produced simultaneously by cones   Nothing changes for the cylinder. |

* 1. **Evaluation criteria**

List of criteria that will allow to compare the different technical solutions and converge to a unique solutions. Criteria can be objective like memory efficiency, bitrate or subjective flexibility, compatibility with legacy solution, etc..

|  |  |  |
| --- | --- | --- |
| **Criteria** | **Description** | **Evaluation** |
| Criteria #1 | Description | The technical solution should minimize/optimise … |

* 1. **Timeline**
* 2022-01-17: MPEG #138: Refine architecture and define basic triggers and actions
  1. **References**

[1] “Technologies under Consideration on Scene Description for MPEG Media”, N00367, MPEG2021, Online, October 2021

1. **EE6: User Representation and Avatars (on-going)**
   1. **Introduction**

The MPEG Scene Description group relies on the glTF2.0 technology to enable the support of 3D scene in MPEG media. Using the extension mechanism, the solution allows synchronization between traditional MPEG media within 3D content.

As defined in the requirements, one goal is to permit a user to navigate the content and interact with the surrounding objects and characters [1].

* 1. **Problem statement**

In order to interact within the 3D scene, the user must be represented in the scene. This representation is called an avatar and reinforces the user’s feeling of presence in the virtual world. An avatar is not mandatory if the user is simply walking through and watching some content, but as soon as there is interactivity and collision, the user must be able to visualize or detect the boundaries of the avatar. As of today, the representation of the user within the scene is not formally defined. Requirements for MPEG-I Phase 2 only mention the user as “the listener whose position and orientation are used for rendering” [2].

This EE will focus on developing the necessary glTF extensions to support user representations in scene description.

* 1. **Use cases relevant for the EE**

Basically, all use cases listed by the haptic group require an avatar so the user can touch virtual objects [3]. The audio use cases do not explicitly mention the user appearance although the objects in the scene impact the sound rendering (#4) [5]. Besides the social VR scenario implicitly means that users can see each other's.

* 1. **Related (WG2) and Extracted (new) Requirements**

The following requirements are thus relevant and addressed by this EE:

* Support of user interactivity with objects within a virtual environment (requirement #90)
* Support of interaction between multiple users within an immersive environment (requirement #95)
* The specification shall support interactivity models related to avatar position and orientation (requirement #129)
* The specification shall support coding and presentation of interactivity models related to avatar-scene or avatar-avatar interactions (requirement #130)
* The specification shall support different media types and various haptic feedback paradigms (requirement #131)
  1. **Relation to other activities (EE, requirements, etc…)**

A relationship to the Haptics phase 2 activity has been identified. The user could touch virtual objects, hence a visual representation is mandatory.

This EE relates with the MPEG Systems EE on Interactivity [4] and Haptics [6].

* 1. **Mandates**

The mandates for this EE are as follows:

* Define the term avatar in the MPEG-I Phase 2 requirements
* Identify the existing glTF-based solutions to describe avatars
* Define the scope of the glTF extension for avatars within the scene description architecture
* Define test scenarios and collect test assets
* Define the evaluation criteria
* Evaluate proposed solutions
* Develop the reference software integration and validate against the test scenarios
  1. **Participants**

|  |  |  |  |
| --- | --- | --- | --- |
| Participant | Contact | Email | Type |
|  |  |  |  |
| InterDigital | Fabien Danieau | [fabien.danieau@interdigital.com](mailto:fabien.danieau@interdigital.com) | L |
| Qualcomm | Imed Bouazizi | bouazizi@qti.qualcomm.com | P |
| Immersion | Yeshwant Muthusamy | ymuthusamy@immersion.com | P |

(P = proponent, L = leader)

* 1. **Information about proposed technologies**

List of already submitted contributions on this topic.

* + 1. **m56337 [SD] Interactivity in Scene Description**

This contribution introduced the user as the camera controller which is his/her only representation in the scene.

* + 1. **m58104 [SD] On scene interactivity**

This contribution presented the camera as the user avatar. It also presents a need for a collider so the user cannot walk beyond the limited space of the experience.

* + 1. **m58146 [SD] Describing camera paths for interactivity**

This contribution also considers the user as the camera and limits his/her movement to a camera path.

* + 1. **m58487 [SD] MPEG-I SD Revised Haptic Schema and Processing Model**

This contribution discusses the problem of the user representation and mentions a potential solution VRM to be evaluated. it is also indicated that the haptic needs are more focused on the collision (i.e. bounding box) than the visual appearance.

* + 1. **m59269 [SD] EE on User Representation and Avatars**

Two separate extensions are proposed MPEG\_avatar and MPEG\_collider. Indeed, colliders can be used for other purposes than for the avatar only. It is meant to replace the extension MPEG\_mesh\_collision [7] with additional features.

Diagram

Description automatically generated

### MPEG\_avatar

Since glTF allows to define mesh, joints and skinning there is not much to extend to enable avatars. This extension will simply indicate what node is used to describe this mesh .

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Default** | **Description** |
| is\_avatar | boolean | True | indicates that the nodes contains an avatar |

glTF does not provide any kind of specification regarding a **humanoid** body rig, so we propose the following conventions so the presentation engine will know how to interpret the mesh as an avatar:

**Initial pose**

Humanoid avatars (the mesh) are assumed to have a T-pose.

**Skeleton**

Joints in the glTF format can be identified by names and their hierarchy. We propose the following structure and name convention to enable an easy identification of the avatar skeleton. It is based on the Unity recommendation and Mixamo skeleton that should make it compatible with most game engines.

|  |
| --- |
| * Hips   + Spine     - Chest       * UpperChest         + Shoulder\_Left   UpperArm\_Left  LowerArm\_Left  Hand\_Left  *See below*   * + - * + Shoulder\_Right   UpperArm\_Right  LowerArm\_Right  Hand\_Right  *See below*   * + - * + Neck   Head  Eye\_Left  Eye\_Right  Jaw   * + UpperLeg\_Left     - LowerLeg\_Left       * Foot\_Left         + Toes\_Left   + UpperLeg\_Right     - LowerLeg\_Right       * Foot\_Right         + Toes\_Right |

|  |
| --- |
| * Hand\_Left   + ProximalThumb\_Left     - IntermediateThumb\_Left       * DistalThumb\_Left   + ProximalIndex\_Left     - IntermediateIndex\_Left       * DistalIndex\_Left   + ProximalMiddle\_Left     - IntermediateMiddle\_Left       * DistalMiddle\_Left   + ProximalRing\_Left     - IntermediateRing\_Left       * DistalRing\_Left   + ProximalLittle\_Left     - IntermediateLittle\_Left       * DistalLittle\_Left * Hand\_Right   + ProximalThumb\_Right     - IntermediateThumb\_Right       * DistalThumb\_Right   + ProximalIndex\_Right     - IntermediateIndex\_Right       * DistalIndex\_Right   + ProximalMiddle\_Right     - IntermediateMiddle\_Right       * DistalMiddle\_Right   + ProximalRing\_Right     - IntermediateRing\_Right       * DistalRing\_Right   + ProximalLittle\_Right     - IntermediateLittle\_Right       * DistalLittle\_Right |

A visual mesh is not mandatory. The avatar could be the camera only and a collider for instance. So, the user cannot go through walls or beyond a specified space although it is not visually represented.

**Partial avatars**

Avatar may partially represent the user (only the hands or the upper body). In this case a sub part of the skeleton is used to describe the joints.

|  |
| --- |
| {  "$schema": "http://json-schema.org/draft-04/schema",  "title": "MPEG\_avatar",  "type": "object",  "description": "An avatar.",  "allOf": [ { "$ref": "glTFChildOfRootProperty.schema.json" } ],  "properties": {  "is\_avatar": {  "type": "boolean",  "description": "List of haptic avatars",  “default” : True,  "gltf\_detailedDescription": "Indicates that the sub mesh should be considered as an avatar."  }  } |

### Colliders

In order to detect collision between the avatar and 3D objects, colliders are necessary (i.e. rough shape surrounding the object that decreases the computing cost of the collision detection). This question has been partially answered in m58486 [7]. The extension MPEG\_mesh\_collision allows defining a mesh (boundaries) as a collider to be linked to another mesh. We propose to improve this extension with the following features.

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Default** | **Description** |
| shape | Integer | 0 | Shape of the collider: 0 – sphere, 1 – box, 2 – cylinder, 3 – capsule, 4 – mesh (defined below) |
| custom\_shape | Integer | -1 | Index of the mesh that will be used to describe the collision boundaries for this node. The collision mesh shall not be referenced by any other node in the scene description. |
| static | boolean | True | Determines if the object is affected by collisions or not. An object that is static will not be affected by collisions, which means that when the viewer or another object collides with this object, its position will not be altered. |

**Shapes**

Game engines usually rely on primitive shapes to define colliders (spheres, cubes, etc). Collision computing is much simpler with these shapes than with complex ones. For instance, Unity supports boxes, spheres, capsules [8]; godot supports boxes, spheres, capsules and cylinders [9]. Of course, mesh colliders are also supported (colliders that follow the exact shape of the displayed mesh).

We propose to support these primitive shapes in addition to the custom one (a.k.a. boundaries in the previous extension).

Also, we propose to define the extension at the node level since a collider may be used by multiple meshes.

**Material and animations**

These properties of MPEG\_mesh\_collision are not kept since they will be discussed in the EE on haptics and interactivity. They may have their own extension.

|  |
| --- |
| {  "$schema": "http://json-schema.org/draft-04/schema",  "title": "MPEG\_collider",  "type": "object",  "description": "A 3D collider.",  "allOf": [ { "$ref": "glTFChildOfRootProperty.schema.json" } ],  "properties": {  "shape": {  "type": "integer",  "description": "Specifies the shape of the collider.",  "gltf\_detailedDescription": "Bounding box surrounding a node. Shape is defined by the transform properties of the node."  "anyOf": [  {  "enum": [ 0 ],  "Sphere": "Sphere-like volume. Diameter of the sphere is given by the scale of the parent node."  },  {  "enum": [ 1 ],  "Cube": "Cube-like volume. Shape of the cube is given by the scale of the parent node."  },  {  "enum": [ 2 ],  "Capsule": "Capsule-like volume. Diameter of the capsule is given by the x-scale of the parent node, and height by the y-scale."  },  {  "enum": [ 2 ],  "Custom": "Custom mesh volume. The mesh ID is provided in the next field."  },  {  "type": "integer"  }  ],  default : "0";  },  "custom\_shape": {  "allOf": [ { "$ref": " mesh.schema.json" } ],  "description": "Mesh used to defined the bounding box.",  "gltf\_detailedDescription": "Index of the mesh that will be used to describe the collision boundaries for this node. The collision mesh shall not be referenced by any other node in the scene description."  },  "static": {  "type": "boolean",  "description": "Static object.",  "gltf\_detailedDescription": "Determines if the object is affected by collisions or not. An object that is static will not be affected by collisions, which means that when the viewer or another object collides with this object, its position will not be altered.",  "default" : "False",  },  "name": { },  "extensions": { },  "extras": { }  }  } |

* + 1. **m60296 [SD] EE6 - User Representation and Avatars for ISO/IEC 23090-14:2021**

A reference glTF content file has been created for the full body and for the partial avatars (upper body and hand).

For the full body content (Figure 1), two different bodies are defined (a male and a female one). In both cases, the skeleton hierarchy is exactly following the one described in chapter 2.1. These two models have been designed in the DCC software Autodesk Maya © and asset files were exported using a glTF export plugin.

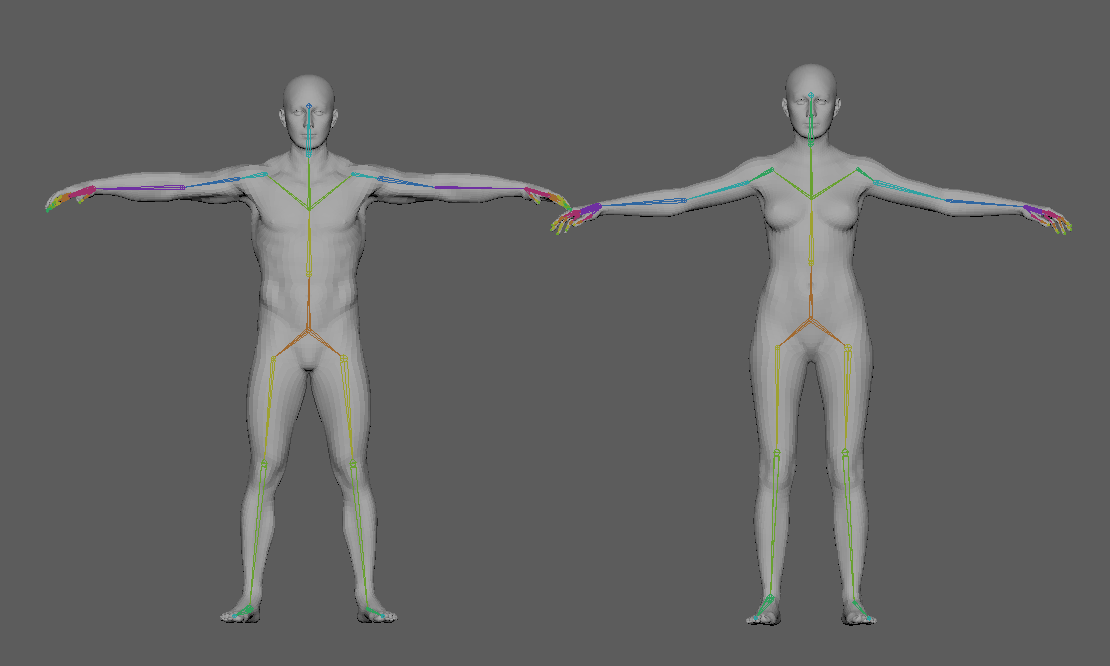
****

Figure 2 - Male and Female body models

A reference glTF content file has been created for the upper body representation. The skeleton used starts from the “Hips' ' node and only goes through the “Spine” sub-node.

# 

Figure 2 - Male upper body model

A reference glTF content file has been created for the hand representation. The skeleton used starts from the “Hips' ' node and only goes through the “Spine” sub-node.

# 

Figure 3 - hand model

* 1. **Test cases**

The EE may define test cases for which the evaluation criteria will be analyzed. For instance, a first test case can be with live content while another in the on-demand content.

* + 1. **Camera representation**

This test case corresponds to the use case proposed by m56337 and m58104. It has no visual appearance but only controls the camera viewport. A collider is associated to this “body representation” so the user cannot go beyond a limited area. This collider will also be used to detect was body part collides with an object and thus trigger a haptic effect.

* + 1. **Limited representation**

Same as above, with the display of the user’s head and hands (most current VR avatars). Tracking is performed by a VR headset and associated controllers. User can interact with objects in the scene.

* + 1. **Full body avatar**

Same as above, with the display of the full body avatar. Full body tracking is required or simulated. Multiple users can see each other.

* 1. **Evaluation criteria**

List of criteria that will allow to compare the different technical solutions and converge to a unique solutions. Criteria can be objective like memory efficiency, bitrate or subjective flexibility, compatibility with legacy solution, etc..

|  |  |  |
| --- | --- | --- |
| **Criteria** | **Description** | **Evaluation** |
| #1 mesh | A mesh represents the avatar | The mesh is compliant with glTF format |
| #2 collider | A collider surrounds the avatar | The collider allows to detect precise collision on the user’s body |
| #3 bones | The mesh is rigged | The set of bones is formally defined and allows motion tracking |
| #4 overhead | Size of the metadata / glTF extension | the smaller overhead to glTF the better (Size) |

* 1. **Timeline**
* 2022-01-17: MPEG #138
  + Define the term avatar
  + Define the scope of the avatar extension
  + Analyze existing solutions
  + Propose test cases and criteria
* 2022-07-18: MPEG #139
  + completion of EE
  1. **References**

1. “Requirements Coverage of MPEG-I Scene Description”, N00369, MPEG2021, Online, October 2021
2. “Requirements for MPEG-I Phase 2 WG 2, MPEG Technical requirements”, m57684, MPEG2021, Online, July 2021
3. “[Haptics] Updated MPEG-I Phase 2 Haptics Use Cases”, m57952, MPEG2021, Online, October 2021
4. “[SD] Description of EE on Generic Interactivity Framework”, m59166, MPEG2021, Online, January 2022
5. “Thoughts on MPEG-I Audio requirements”, m46062, MPEG2019,Marrakesh, MA, January 2019

“[SD] Description of EE on Haptics Support in SD”, m59210, MPEG2021, Online, January 202

1. **EE7: AR Anchoring (closed)**

EE closed at MPEG #139. Please see WG03 N0540 for the latest description of this EE.



































1. **EE8: Lighting (ongoing)**
   1. **Introduction**

When it comes to inserting visual information in a captured real-world environment, lighting is a fundamental cue to take into account to provide a realistic experience to the user. Indeed, a virtual object overlaid on an real-world environment with inappropriate lighting and shadows can break the immersive illusion. In a VR context, accurate lighting models allow to achieve a high-level of realism which is also key for many VR applications that offer the illusion of “being there” for the user.

The goal of the EE is thus to specify the integration of lighting information existing in the glTF specification into the MPEG-I Scene Description standard. In particular, the EE will study the integration of lighting estimation operation with the MPEG-I SD architecture and the representation of light sources in scene description documents. Based on this integration and the possible identified technical gaps, the EE aims at defining the necessary extensions to glTF specification and to MPEG-I SD specification, as well as possible implementation guidelines.

* 1. **Problem statement**

A presentation engine is responsible for rendering a view of a scene to the user based on the scene description document. Among other things (physics engine, object drawing, etc..), the presentation engine renders the effect of light propagation in the scene, reflections, shadows, object illumination etc… In order to render those effects, the presentation engine needs a model of the lighting conditions at the time of the rendered frame. The light sources can be of two natures: real or virtual. In the AR context, there exists by definition a set of real light sources (sun, lamps, etc.). In addition, the scene may also contain virtual light sources. In the VR context, there are by definition only virtual light sources. In both cases, real and virtual, the light sources may be represented by the same model (punctual light, ambient light, etc.) and sometimes based on textures for the so-called environment cubemap. Those texture stored in buffer will impact the overall buffer management of the application and cannot ne omitted by MPEG-I SD architecture. One additional challenge in the AR context is to estimate the representation of the current lighting condition in which the AR application runs. This is called lighting estimation function and is provided by existing AR framework via API calls.

All those light-related operations are thus part of the presentation engine and requires:

* Representation model of light sources
* Spatialization of light sources representation (user can move in the scene)
* Time-dependent light source representation (the light sources can change over time)
* Integration with lighting estimation APIs (only for AR)
  1. **Use cases relevant for the EE**
  2. **Relation to other activities (EE, requirements, etc…)**

From WG 02 N00130, “MPEG-I Phase 2 Requirements”, MPEG136, October 2021.

*4.3.1 Reference Scene Description Selection*

77. The scene description should support nodes and attributes in order to implement natural laws of light, energy propagation and physical kinematic operations.

*4.4 Descriptions for Content Interactivity*

89. The specification shall enable realistic composition of a 6DoF scene depending on the user-selected location and orientation.

*Note: Such composition may, e.g., include delivering proper lighting information and some form of geometry information of the scene so the view is rendered with realistic lighting and shadows.*

* 1. **Mandates**

The mandates for this EE are as follows:

* To study the integration of AR lighting estimation API call with the MPEG-I SD reference architecture
* To specify the light source representation model for both real and virtual light sources, possibly based on commonly used models
* To specify the signaling of spatialized and time-dependent light sources in the scene description document based on existing glTF light-related extension
  1. **Participants**

|  |  |  |  |
| --- | --- | --- | --- |
| Participant | Contact | Email | Type |
|  |  |  |  |
| Xiaomi | Emmanuel Thomas | thomase@xiaomi.com | L |
| Qualcomm | Imed Bouazizi | bouazizi@qti.qualcomm.com | P |
| InterDigital | Patrice Hirtzlin | Patrice.hirtzlin@interdigital.com | P |

(P = proponent, L = leader)

* 1. **Information about proposed technologies**

List of already submitted contributions on this topic.

* + 1. **m59520 – Scene description and lighting information**

This contribution describes the background on light theory and its representation in AR frameworks.

* + - 1. **Light source inventory**

As explained in [1], there exist several types of light whether it is emitted or reflected light and different types of light sources which are primarily determined by the physical properties of the light such as the distance of the source, the direction of the ray of lights, etc..

The different types of light can be categorized as listed in Table 1.

Table 13 - Categorization of light types

|  |  |  |
| --- | --- | --- |
| **Type** | **Description** | **Example** |
| Ambient light | Ambient light is light that doesn't come from a defined source, but is just present throughout the scene. This light reaches every surface in the scene at the same intensity from every direction, and is then reflected equally in every direction. As a result, the effect applied by ambient light is universally equal all through the scene. | A sphere which only has ambient lighting. Note the total lack of any shading to indicate the depth of the sphere. |
| Diffuse light | Diffuse light is light which is evenly and directionally emitted from or reflected off a surface. This is the majority of the light we usually see. Diffuse light comes from a particular position or direction and casts shadows. Due to its directionality, the faces of an object facing a diffuse light source will be brighter than the other faces. | Saturn's fifth-largest moon, Tethys, is lit primarily by the sun, with some light reflected from Saturn. This is diffuse lighting. |
| Specular light | Specular light is the light that makes up the highlights on reflective objects, such as gems, eyes, shiny cups and plates, and the like. Specular lights tend to appear as bright spots or squares on a surface at the point where a light source strikes the surface most directly. | A photo taken by NASA's Cassini spacecraft showing specular reflection of light from a lake of liquid methane on the surface of Saturn's moon Titan. |

Another categorization pertains to the light sources as described in Table 2.

Table 14 - Categorization of light source types

|  |  |  |
| --- | --- | --- |
| **Type** | **Description** | **Example** |
| Ambient light sources | An ambient light source is a light source describing the level and color of ambient light in a scene. While there may be more than one of these in a scene, you can probably slightly improve performance by combining them into one on your own, since each one will always affect every pixel evenly anyway. |  |
| Directional light sources | A directional light source is a light source that comes from a specific direction, but not from a specific source, so its emitted light rays are parallel to one another. In addition, the intensity of the light doesn't change over distance. This means that shadows cast by directional lights are very sharp, with an essentially instant transition between lighted and shadowed. | A photo taken by the Galileo spacecraft from about 6.3 million kilometers away, with Earth and moon both half-lit by the sun. |
| Point light sources | A point light source is a light source located at a specific location, radiating outward equally in every direction. Light bulbs, candles, and the like are examples of point light sources. The closer an object is to a point light source, the brighter the light it casts onto that object. The rate at which the brightness of a point light falls off is called attenuation, and is a configurable feature of the light source in WebGL and other lighting systems. |  |
| Spot light sources | A spot light source (or spotlight) is a light source which is located at a specific position, emitting a cone of light in the direction of its orientation vector. A tapering rate parameter defines how quickly the brightness of the light falls off at the edges of the cone of light, and, as with point lights, an attenuation parameter controls how the light fades over distance. | Photo of a spotlight shining upon a stucco wall at night. |

* + - 1. **Lighting model**

As explained by the documentation of Google ARCore [2], the common model to reconstruct realistic lighting is composed of three elements:

1. **Main directional light**. Represents the main light source. Can be used to cast shadows.
2. **Ambient spherical harmonics**. Represents the remaining ambient light energy in the scene.
3. **An HDR cubemap**. Can be used to render reflections in shiny metallic objects.

The effect of each component and the final combined result are illustrated below:

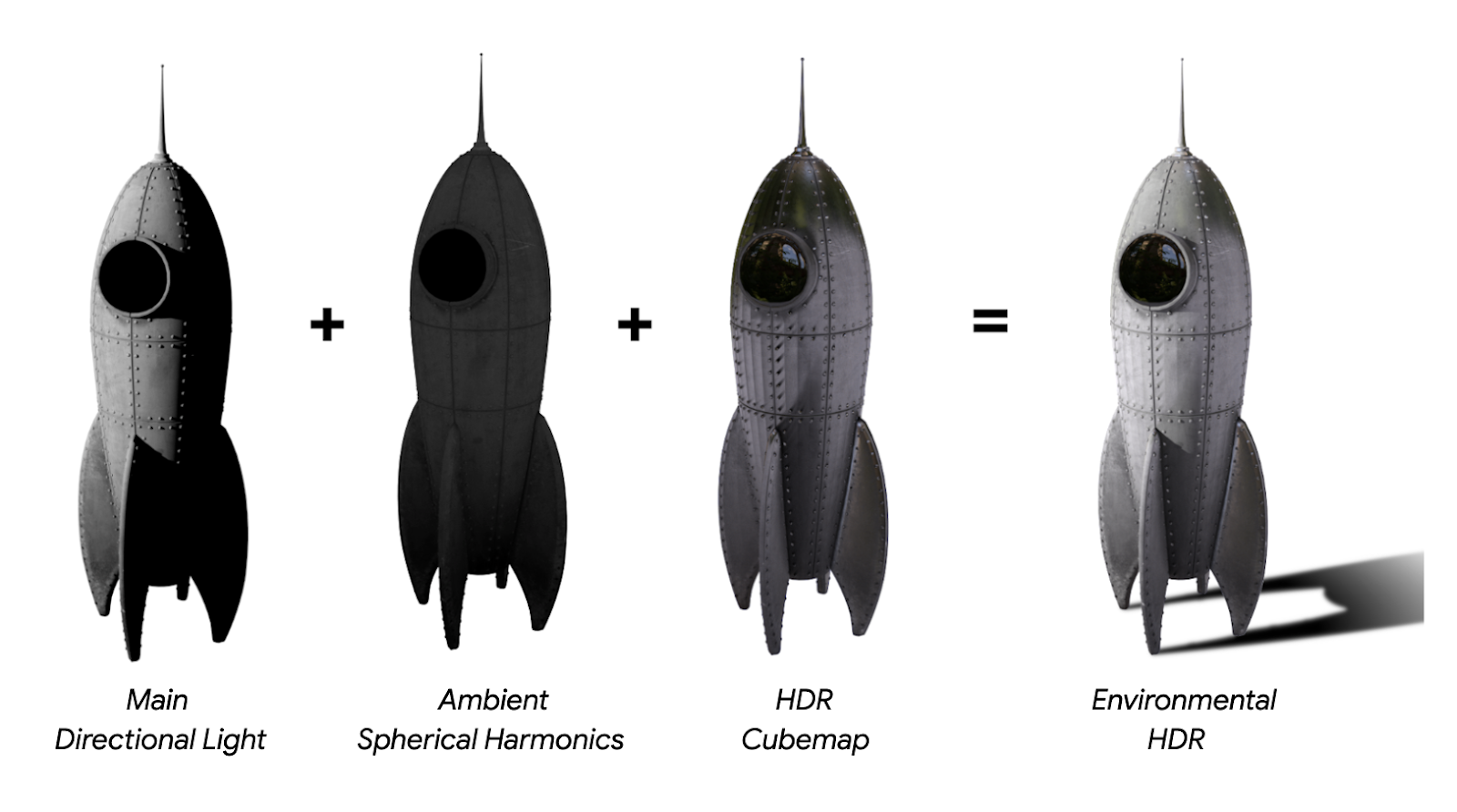


Figure 1 - Environmental lighting commonly used for AR [4]

Chart, treemap chart

Description automatically generated

Figure - Specular Radiance Cubemaps in EXT\_lights\_image\_based

For an optimal rendering quality, these three elements need to be refreshed every time a new frame is rendered by the AR engine. As a result, the light direction information and the ambient spherical harmonics become a timed sequences of metadata and the HDR cubemap becomes a timed sequence of textures, i.e. a video sequence.

* + - 1. **AR Lighting estimation**

In the context AR, the sources of light in the scene are dynamically estimated by the application. The existing AR frameworks provide such API functions:

|  |  |  |
| --- | --- | --- |
| **Framework** | **Function** | **Reference** |
| ARcore | class **LightEstimate** returned Frame.getLightEstimate() | [LightEstimate  |  ARCore  |  Google Developers](https://developers.google.com/ar/reference/java/com/google/ar/core/LightEstimate) |
| ARKit | class ARLightEstimate : NSObject | [ARLightEstimate | Apple Developer Documentation](https://developer.apple.com/documentation/arkit/arlightestimate) |
| WEbXR | interface **XRLightEstimate** | [WebXR Lighting Estimation API Level 1 (w3.org)](https://www.w3.org/TR/webxr-lighting-estimation-1/#xrlightestimate) |

* 1. **Test cases**
     1. **Test case virtual-1: virtual spots in virtual scene**

|  |  |
| --- | --- |
| **Test case identifier** | virtual-1 |
| **Description** | A set of two spotlights or point lights with white color illuminates a surface with 2 objects (2 spheres). A light animation changes the color of the two spotlights on a time bases: after 5 s, the spot one change to green and the second change to yellow. After another 5s, the spot one change to blue and the second change to red. |
| **Test assets** | Scene with the following 3D objects:   * + Plane + 2 spheres   + 2 spotlights |
| **Current support** | The following features are supported:   * + Punctual light (spotlight) |
| **Criteria** | A renderer needs to be able to render the spotlights with the starting color and then change the color at the predefined times. |
| **Location** | <https://mpegfs.int-evry.fr/mpegcontent/ws-mpegcontent/MPEG-I/Part14-SceneDescriptions/content/EE8-Lighting/virtual-1> |

* + 1. **Test case virtual-2: various virtual lights in virtual scene**

|  |  |
| --- | --- |
| **Test case identifier** | virtual-2 |
| **Description** | In this test case, the scene is virtual and the lighting are also virtual. To generate this test case, it is possible to create a glTF scene using the KH\_punctual\_light extension.  It contains three punctual lights of type spot above the living room. Those ceiling spots have respectively a red, blue and green colour. The red spot has a larger light cone than the two others. In addition, four punctual lights of type “point” are on the wall in the back. Their colour is white. |
| **Test assets** | The scene contains:   * + A carpet   + A coffee table   + A sofa   + A shelve   + Planes making an indoor room |
| **Current support** | Light source extension:   * + KH\_punctual\_light |
| **Criteria** | A renderer needs to be able to render the virtual lights according to their description in terms of size, intensity, direction and color. |
| **Location** | <https://mpegfs.int-evry.fr/mpegcontent/ws-mpegcontent/MPEG-I/Part14-SceneDescriptions/content/EE8-Lighting/virtual-2> |

* + 1. **Test case virtual-3: picture-based illuminating surface in virtual scene**

|  |  |
| --- | --- |
| **Test case identifier** | virtual-3 |
| **Description** | A TV screen displays an image and illuminates a surface with 2 spheres (for instance with an area light).  An audio source is played by the TV and at some predefined times, a new image is displayed on the TV screen.  The lighting on the objects changes over time as the surface light intensity changes. |
| **Test assets** | Scene with the following 3D objects:   * + For the TV, a vertical plane with an image texture   + Plane + 2 spheres |
| **Current support** | The following features are supported:   * Image texture * The support for area light is missing and may be difficult to implement in mpegtrimesh. |
| **Criteria** | A renderer needs to be able to render the TV screen with the displayed images and render the correct appearances of object accordingly. |
| **Location** | <https://mpegfs.int-evry.fr/mpegcontent/ws-mpegcontent/MPEG-I/Part14-SceneDescriptions/content/EE8-Lighting/virtual-3> |

* + 1. **Test case virtual-4: video-based illuminating surface in virtual scene**

|  |  |
| --- | --- |
| **Test case identifier** | virtual-4 |
| **Description** | A TV screen displays an image and illuminates a surface with 2 spheres (for instance with an area light).  An audio source is played by the TV and at some predefined times, a new image is displayed on the TV screen.  The lighting on the objects changes over time as the surface light intensity changes. |
| **Test assets** | Scene with the following 3D objects:   * + For the TV, a vertical plane with an video texture   + Plane + 2 spheres |
| **Current support** | The following features are supported:   * MPEG video texture for the variant * The support for area light is missing and may be difficult to implement in mpegtrimesh. |
| **Criteria** | A renderer needs to be able to render the TV screen with the displayed video and render the correct appearances of objects accordingly. |
| **Location** | <https://mpegfs.int-evry.fr/mpegcontent/ws-mpegcontent/MPEG-I/Part14-SceneDescriptions/content/EE8-Lighting/virtual-4> |

* + 1. **Test case env-1: static picture-based environment cubemap**

|  |  |
| --- | --- |
| **Test case identifier** | env-1 |
| **Description** | A sphere on a surface is inside an environment that illuminates the 2 objects (for instance with a lighting cubemap). |
| **Test assets** | Scene with the following 3D objects:   * + Plane + 1 sphere   + One 2D images as environment cubemap texture |
| **Current support** | The following features are supported:   * + The support for lighting cube map (for instance Khronos EXT\_lights\_image\_based extension) is missing. |
| **Criteria** | A renderer needs to be able to render the objects appearances based on the environment texture. |
| **Location** | <https://mpegfs.int-evry.fr/mpegcontent/ws-mpegcontent/MPEG-I/Part14-SceneDescriptions/content/EE8-Lighting/env-1> |

* + 1. **Test case env-2: dynamic picture-based environment cubemap**

|  |  |
| --- | --- |
| **Test case identifier** | env-2 |
| **Description** | A sphere on a surface is inside an environment that illuminates the 2 objects (for instance with a lighting cubemap).  At some predefined times, the environment changes and the lighting on the objects changes accordingly. |
| **Test assets** | Scene with the following 3D objects:   * + Plane + 1 sphere   + Two 2D images as environment cubemap texture |
| **Current support** | The following features are supported:   * + The support for lighting cube map (for instance Khronos EXT\_lights\_image\_based extension) is missing. |
| **Criteria** | A renderer needs to be able to render the objects appearances based on the environment texture as the map changes, the rendering of the objects has to change. |
| **Location** | <https://mpegfs.int-evry.fr/mpegcontent/ws-mpegcontent/MPEG-I/Part14-SceneDescriptions/content/EE8-Lighting/env-2> |

* + 1. **Test case env-3: video-based environment cubemap**

|  |  |
| --- | --- |
| **Test case identifier** | env-3 |
| **Description** | A sphere on a surface is inside an environment that illuminates the 2 objects (for instance with a lighting cubemap).  The environment texture is based on a coded video. |
| **Test assets** | Scene with the following 3D objects:   * + Plane + 1 sphere   + A 2D video as environment cubemap texture |
| **Current support** | The following features are supported:   * + The support for lighting cube map (for instance Khronos EXT\_lights\_image\_based extension) is missing. |
| **Criteria** | A renderer needs to be able to render the objects appearances based on the environment texture as the map changes over time, possibly at every frame of the video. |
| **Location** | <https://mpegfs.int-evry.fr/mpegcontent/ws-mpegcontent/MPEG-I/Part14-SceneDescriptions/content/EE8-Lighting/env-3> |

* + 1. **Test case real-1: Virtual objects with shadows from real light**

|  |  |
| --- | --- |
| **Test case identifier** | real-1 |
| **Description** | A real light illuminates a real table on which are placed a real object (for example, a speaker) and a virtual object (for example a teddy bear).  The real light is simulated by a virtual light (spotlight). |
| **Test assets** | Scene with the following 3D objects:   * + Plane + Speaker (real scan: obj format for example)   + 1 Teddy Bear (virtual object) |
| **Current support** | The following features are supported:   * + Support for shadows   + Support for spot light |
| **Criteria** | A renderer needs to be able to:   * + Render the shadow associated to the virtual object   + Render the direction of shadow associated to the virtual object   + Render the contour of shadow associated to the virtual object |
| **Location** | <https://mpegfs.int-evry.fr/mpegcontent/ws-mpegcontent/MPEG-I/Part14-SceneDescriptions/content/EE8-Lighting/real-1> |

* 1. **AR scene recording format**

In order to record AR scene in the context of this EE, the EE defines mp4 formats to be observed by the submitted mp4 files containing the assets. This way, they can be unambiguously consumed by participants and reference software such as mpegtrimesh.

Note that currently ony an ARcore-based recoding format is defined but there may be more in the future.

* + 1. **ARCore-based recording format**

For recording made using AR Android device, the format defined in Table 15 is expected.

Table 15 - ARCore-based recording format

|  |  |  |  |
| --- | --- | --- | --- |
| **Track #** | **Codec type** | **Track type** | **Sample format** |
| 1 | avc1 | Visual track | AVC NAL format |
| 2 | mett (application/arcore-video-0) | Metadata track | Mysterious |
| 3 | mett (application/ arcore-gyro) | Metadata track | Need to check |
| 4 | mett (application/ arcore-accel) | Metadata track | Need to check |
| 5 | mett (application/ arcore-custom-event) | Metadata track | Need to check |
| 6 | mett (application/ hello-recording-playback-anchor) | Metadata track | Need to check |
| 7 | mett (application/ mpeg-sd-spherical-harmonics) | Metadata track | Vector of 27 floats coded on 32 bits.  ARCore API: getEnvironmentalHdr-AmbientSphericalHarmonics() |
| 8 | mett (application/ mpeg-sd-main-light) | Metadata track | Vector of 3 floats coded on 32 bits (direction), vector of 3 floats coded on 32 bits (intensity)  ARCore API: getEnvironmentalHdr-MainLightDirection() and getEnvironmentalHdr-MainLightIntensity() |
| 9 | mett (application/ mpeg-sd-environment-cubemap) | Metadata track | PNG-compressed cubemap in 3x2 layout  Width : 48 pixels  Height : 32 pixels  Color space: RGBA  Bit depth : 16 bits  Compression mode: Lossless  Projection:    See 5.2.3 Cubemap projection for one sample location in MPEG-I OMAF  ARCore API: acquireEnvironmentalHdrCubeMap() |

Tracks from 1 to 6 are provided as is by the mp4 recording API of ARCore. The tracks from 7 to 9 are defined for the specific purpose of the EE8.

* 1. **Evaluation criteria**

List of criteria that will allow to compare the different technical solutions and converge to a unique solutions. Criteria can be objective like memory efficiency, bitrate or subjective flexibility, compatibility with legacy solution, etc..

|  |  |  |
| --- | --- | --- |
| **Criteria** | **Description** | **Evaluation** |
| Crieria #1 | Description | The technical solution should minimize/optimise … |

* 1. **Timeline**
* 2022-04-30: post MPEG#138 AHG
  + Collection of use test cases
  + Collection of evaluation criteria
  + Initial thoughts on reference software implication (e.g. light rendering in trimesh)
* 2022-07-18: MPEG #139(online) meeting starts
  + Agree on test cases and their possible prioritisation
  + Agree on evaluation criteria
* 2022-07-22: post MPEG#139 AHG
  + Provide and collect all test assets for the agreed test cases
  + Collect possible additional test cases
  + Collect input on initial technical solutions for light source representation and signalling (preferably with assets and implementations)
  + Collect input on MPEG-I SD architecture lighting rendering integration
* 2022-10-24: MPEG #140(online) meeting starts
  + Progress work
* 2023-01-16: MPEG #141(online) meeting starts
  + Agree on final technical solution (assets and implementations needed for agreement)
  + Agree on modified MPEG-I SD reference architecture (if modifications needed)
  1. **References**

1. Lighting a WebXR setting - Web APIs | MDN (mozilla.org), <https://developer.mozilla.org/en-US/docs/Web/API/WebXR_Device_API/Lighting>
2. Introduction to Lighting Estimation | ARCore | Google Developers, <https://developers.google.com/ar/develop/java/lighting-estimation/introduction?hl=en>

1. [MPEG/Systems/SceneDescription/MPEG-Contributions#222](file://code1/software/MPEG/Systems/SceneDescription/MPEG-Contributions/-/issues/222) [↑](#footnote-ref-1)