ISO/IEC JTC 1/SC 29/WG 03 N0835

**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:** 2023-

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

**No. of pages:** 28 (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 N0853**

**April 2023, Antalya, TR**

|  |  |
| --- | --- |
| **Title** | **Exploration Experiments for MPEG-I Scene Description** |
| **Source** | **WG 03, MPEG Systems** |
| **Status** | **Approved** |
| **Serial Number** | **22577** |

[1 EE1: Carriage of Random Access Support in Scene Description (closed) 1](#_Toc125125378)

[2 EE2: Dynamic Scene Update (closed) 1](#_Toc125125379)

[3 EE3: Codec Support in MPEG-I SD (closed) 1](#_Toc125125380)

[4 EE4: Haptics Support (closed) 1](#_Toc125125381)

[5 EE5: Generic Interactivity Framework (closed) 2](#_Toc125125382)

[6 EE6: User Representation and Avatars (closed) 2](#_Toc125125383)

[7 EE7: AR Anchoring (closed) 2](#_Toc125126504)

[8 EE8: Lighting (ongoing) 2](#_Toc125126505)

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 (closed)**

EE closed at MPEG #140. Please see WG03 N00613 for the latest description of this EE.

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 (closed)**

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

1. **EE6: User Representation and Avatars (closed)**

EE closed at MPEG #141. Please see WG03 N0687 for the latest description of this EE.

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
* To study the compatibility of the existing scene updates mechanism and the use of the KHR\_animation\_pointer for updating properties of the lighting information in the scene
  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 4.

Table 4 - 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 5.

Table 5 - 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:

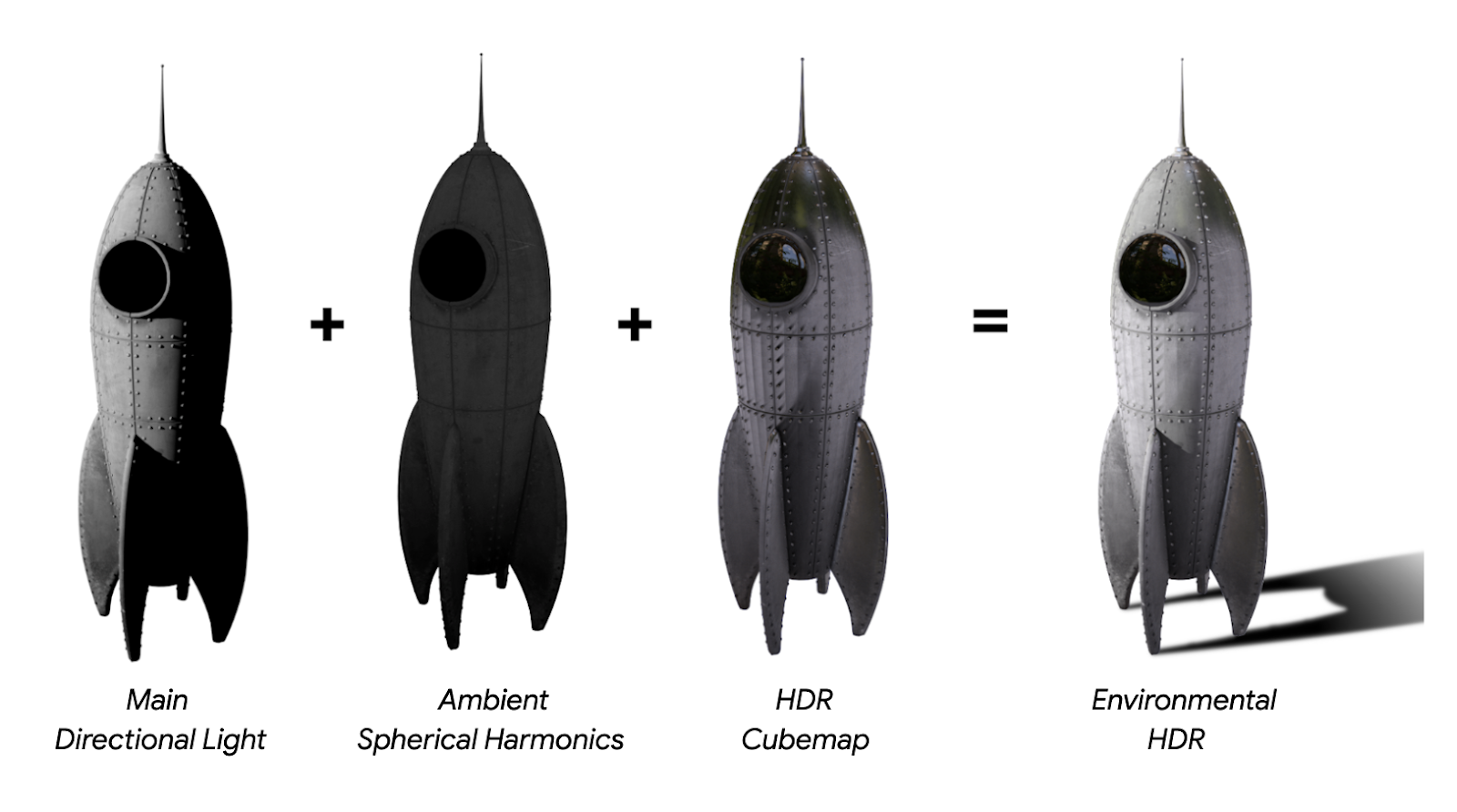


Figure 17 - Environmental lighting commonly used for AR **Error! Reference source not found.**

Chart, treemap chart

Description automatically generated

Figure 18 - 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. **m60868 [SD] [EE 8] – Timed lighting extensions**

**1 Introduction**

A lighting EE (EE8) has been created in the MPEG#138 of April 2022.

The goal of the EE, as mentioned in the document **Error! Reference source not found.**, is thus to specify the integration of lighting information existing in the glTF specification into the MPEG-I Scene Description standard. It addresses, amongst other light-related activities, the representation of time-dependent light sources.

MPEG-I SD framework already addresses time-evolving scenes (timed accessor, animation timing, scene update). It is thus important to do the same for lighting and address pre-defined time-evolving light properties (intensity, color, position, size…). For instance, a lighting animation for a set of spots in a virtual scene that change the colors and the intensity of the lights.

In the contributions **Error! Reference source not found.** and **Error! Reference source not found.**, use cases and test scenarios have been agreed for timed lighting support in scene description.

The study of these test scenarios revealed some missing features in the glTF format to describe time evolving lighting conditions:

* Lack of semantics to describe all the different types of light we may encounter in XR scene, except 2 extensions that describe statically punctual and image-based lights.
* No way to specify how light properties may vary over time.

Concerning the description of light sources, extensions are already available or in development:

* *KHR\_lights\_punctual* (Khronos ratified extension **Error! Reference source not found.**): this extension is used at the root of the glTF file, and at the node level to describe properties of punctual lights (spotlight, point light or directional light). These lights are referenced by nodes and inherit the transform of that node.
* *EXT\_lights\_image\_based* (vendor extension **Error! Reference source not found.**): this extension is used at the root of the glTF file, and at the scene level to describe properties of image-based lights. Each scene can reference only one. This extension specifies a cubemap and images from the gltf images array are referenced for the faces og the cube.
* Other extensions at the root level are in progress and may become available (*KHR\_lights\_ies*, *KHR\_ligths\_environment*, *KHR\_lights\_area* (**Error! Reference source not found.**)…). The *KHR\_lights\_area* extension is used at the root of the glTF file, and at the node level to describe properties of an area light (uniform surface). This light is referenced by a node and inherit the transform of that node.

A common characteristic of those extensions is that they define light properties at the glTF root level with an array of light elements for each extension. An example is given in Table 6.

"extensions": {

"**EXT\_lights\_image\_based**": {

"lights": [

{

"intensity": 1,

"irradianceCoefficients": [

…

],

"name": "imageBasedLight",

"rotation": [

0,

0.7071067094802856,

0,

0.7071068286895752

],

"specularImageSize": 256,

"specularImages": [

[

0,

1,

2,

3,

4,

5

]

]

}

]

},

"**KHR\_lights\_punctual**": {

"lights": [

{

"color": [

1,

1,

1

],

"intensity": 0.5,

"type": "directional",

"name": "Sun"

},

{

"color": [

1,

0,

0

],

"intensity": 1000,

"spot": {

"innerConeAngle": 0.3337942263919169,

"outerConeAngle": 0.39269909262657166

},

"type": "spot",

"name": "Spot"

},

{

"color": [

0,

0,

1

],

"intensity": 1000,

"spot": {

"innerConeAngle": 0.3337942263919169,

"outerConeAngle": 0.39269909262657166

},

"type": "spot",

"name": "Spot.001"

}

]

},

"**KHR\_lights\_area**": {

"lights": [

{

"intensity": 1.0,

"color": [

1.0,

1.0,

1.0

],

"shape": "rect",

"width": 1.0,

"height": 1.0,

},

]

}

}

Table 6: light sources definition in a glTF file

Concerning the time evolving light properties, a new Khronos extension is in discussion that may fill the gap. The KHR\_animation\_pointer **Error! Reference source not found.** is used at the “animations/channel” level to specify a JSON pointer that indicates a glTF element to animate. Any element in the glTF file (including elements from an extension) may be specified, provided that this element is “*Animatable*”. The keyframes data and their timing are provided in a separate binary file or in the gltf file as a base64 coded buffer. They are accessed via accessors, bufferview and buffer as for any other glTF animations (**Error! Reference source not found.**). The *input* value of the animation sampler points to the list of times and the *output* value points to a list of values for each keyframe.

"animations": [

{

"channels": [

{

"sampler": 0,

"target": {

"path": "pointer",

"extensions": {

"KHR\_animation\_pointer": {

"pointer": "/materials/3/pbrMetallicRoughness/baseColorFactor"

}

}

}

}

],

"samplers": [

{

"input": 24,

"interpolation": "STEP",

"output": 25

}

]

}

]

Table 7: KHR\_animation\_pointer example (glTF file extract)

The current document reviews use cases from **Error! Reference source not found.Error! Reference source not found.** and focuses on the spotlights and image-based lights scenarios. It proposes an implementation using the above available or in progress extensions, when applicable. Otherwise, new solutions/extensions are introduced.

**2 Test case virtual-1: virtual spots in virtual scene**

This use case may be implemented by the *KHR\_animation\_pointe*r extension.

The times of each color change and the new color values for each spot are set in a binary animation file. The color property of each spot is animated thanks to this extension:

"animations": [

{

"channels": [

{

"sampler": 0,

"target": {

"path": "pointer",

"extensions": {

"KHR\_animation\_pointer": {

"pointer": "/extensions/KHR\_lights\_punctual/lights/0/color"

}

}

}

},

{

"sampler": 1,

"target": {

"path": "pointer",

"extensions": {

"KHR\_animation\_pointer": {

"pointer": "/extensions/KHR\_lights\_punctual/lights/1/color"

}

}

}

}

],

"samplers": [

{

"input": 24,

"interpolation": "STEP",

"output": 25

},

{

"input": 24,

"interpolation": "STEP",

"output": 27

}

]

}

]

Table 8: lights color animation with KHR\_animation\_pointer (glTF file extract)

No interpolation is performed between two keyframes.

The BabylonJS and ThreeJS engine implement the *KHR\_animation\_pointer* and can play this test case. **The glTF file and the bin file are delivered with this contribution**.

Graphical user interface

Description automatically generatedGraphical user interface

Description automatically generatedGraphical user interface, application

Description automatically generated

Figure 19: lights color animation in BabylonJS

The following table shows the times and the keyframes values of the animated colors, stored in the bin file:

|  |  |  |
| --- | --- | --- |
| **Time** | **Spot1 color** | **Spot2 color** |
| 0.0 | [1.0, 1.0, 1.0] | [1.0, 1.0, 1.0] |
| 5.0 | [1.0, 1.0, 0.0] | [0.0, 1.0, 0.0] |
| 10.0 | [1.0, 0.0, 0.0] | [0.0, 0.0, 1.0] |
| 15.0 | [1.0, 1.0, 1.0] | [1.0, 1.0, 1.0] |

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

This test case may be implemented with the *EXT\_lights\_image\_based* extension and can be played by BabylonJS and ThreeJS engine.

…

"extensionsUsed" : [

"KHR\_lights\_punctual",

"EXT\_lights\_image\_based",

"KHR\_animation\_pointer"

],

"extensionsRequired" : [

"KHR\_lights\_punctual"

],

"extensions" : {

"EXT\_lights\_image\_based": {

"lights": [

{

"intensity": 1,

"irradianceCoefficients": [

[1.8839140747279469,1.2336689528140037,1.6815759445875259],

[1.0005113784288705,0.8691400255493019,1.4887876533795357],

[0.5603794677467341,0.2578132145126057,0.19374826573501498],

[1.3072342827477733,0.6636485650699964,0.6695344061570127],

[0.5640030294080713,0.37938937249123669,0.49194331732327276],

[0.27256774141207748,0.143343904079048,0.1155890697070088],

[-0.1381991414602802,-0.057096853570897488,-0.04879314267934546],

[0.5350810043540868,0.263230477756704,0.24531039907656564],

[0.43283339060831907,0.12637845128810608,-0.0041528480118368589]

],

"name": "imageBasedLight",

"rotation": [0,0.7071067094802856,0,0.7071068286895752],

"specularImageSize": 256,

"specularImages": [

[0,1,2,3,4,5]

]

}

]

}

},

"scene" : 0,

"scenes" : [

{

"name" : "Scene",

"nodes" : [0,2,3,5],

"extensions": {

"EXT\_lights\_image\_based": {

"light": 0

}

}

}

],

"images": [

{

"uri": "IBL/grey\_sky/+X.png",

"mimeType" : "image/png",

"name" : "pz"

},

{

"uri": "IBL/grey\_sky/-X.png",

"mimeType" : "image/png",

"name" : "nz"

},

{

"uri": "IBL/grey\_sky/-Z.png",

"mimeType" : "image/png",

"name" : "py"

},

{

"uri": "IBL/grey\_sky/+Z.png",

"mimeType" : "image/png",

"name" : "ny"

},

{

"uri": "IBL/grey\_sky/+Y.png",

"mimeType" : "image/png",

"name" : "nx"

},

{

"uri": "IBL/grey\_sky/-Y.png",

"mimeType" : "image/png",

"name" : "px"

}

]

**…**

Table 9: ambient light with EXT\_lights\_image\_based (glTF file extract)

The glTF asset for this scenario has already been provided in the contribution m60669 **Error! Reference source not found.**.

Graphical user interface, chat or text message

Description automatically generated

Figure 20: Ambient light with EXT\_lights\_image\_based in BabylonJS

**4 Test case env-2: dynamic picture-based environment cubemap**

Although no implementation is available yet to animate this kind of element, it would be possible to address this scenario with the *KHR\_animation\_pointer*, by animating the glTF array element *“/extensions/EXT\_lights\_image\_based/lights/0/specularImages/0”*: It would be similar to the way an animation is described for the “*weights*” parameter of a node or a mesh element (**Error! Reference source not found.**).

For this test case, the scene description document specifies an array of 12 images. The indexes array of the extension is *[0,1,2,3,4,5]* at time 0s and *[6,7,8,9,10,11]* at time 5s. No interpolation is performed between the two keyframes.

The following table shows how it could be implemented in a scene description document.

**…**

"animations": [

{

"channels": [

{

"sampler": 0,

"target": {

"path": "pointer",

"extensions": {

"KHR\_animation\_pointer": {

"pointer":/extensions/EXT\_lights\_image\_based/lights/0/specularImages/0"

}

}

}

}

],

"samplers": [

{

"input": 8,

"interpolation": "STEP",

"output": 9

}

]

}

],

"extensions": {

"EXT\_lights\_image\_based": {

"lights": [

{

"intensity": 1,

"irradianceCoefficients": [

[1.8839140747279469,1.2336689528140037,1.6815759445875259],

[1.0005113784288705,0.8691400255493019,1.4887876533795357],

[0.5603794677467341,0.2578132145126057,0.19374826573501498],

[1.3072342827477733,0.6636485650699964,0.6695344061570127],

[0.5640030294080713,0.37938937249123669,0.49194331732327276],

[0.27256774141207748,0.143343904079048,0.1155890697070088],

[-0.1381991414602802,-0.057096853570897488,-0.04879314267934546],

[0.5350810043540868,0.263230477756704,0.24531039907656564],

[0.43283339060831907,0.12637845128810608,-0.0041528480118368589]

],

"name": "imageBasedLight",

"rotation": [0,0.7071067094802856,0,0.7071068286895752],

"specularImageSize": 256,

"specularImages": [

[0,1,2,3,4,5]

]

}

]

}

},

"images": [

{

"uri": "IBL/grey\_sky/+X.png",

"mimeType" : "image/png",

"name" : "pz"

},

{

"uri": "IBL/grey\_sky/-X.png",

"mimeType" : "image/png",

"name" : "nz"

},

{

"uri": "IBL/grey\_sky/-Z.png",

"mimeType" : "image/png",

"name" : "py"

},

{

"uri": "IBL/grey\_sky/+Z.png",

"mimeType" : "image/png",

"name" : "ny"

},

{

"uri": "IBL/grey\_sky/+Y.png",

"mimeType" : "image/png",

"name" : "nx"

},

{

"uri": "IBL/grey\_sky/-Y.png",

"mimeType" : "image/png",

"name" : "px"

},

{

"uri": "IBL/blue\_sky/+X.png",

"mimeType" : "image/png",

"name" : "pz"

},

{

"uri": "IBL/blue\_sky/-X.png",

"mimeType" : "image/png",

"name" : "nz"

},

{

"uri": "IBL/blue\_sky/-Z.png",

"mimeType" : "image/png",

"name" : "py"

},

{

"uri": "IBL/blue\_sky/+Z.png",

"mimeType" : "image/png",

"name" : "ny"

},

{

"uri": "IBL/blue\_sky/+Y.png",

"mimeType" : "image/png",

"name" : "nx"

},

{

"uri": "IBL/blue\_sky/-Y.png",

"mimeType" : "image/png",

"name" : "px"

}

]

**…**

Table 10:image based light animation with KHR\_animation\_pointer

**The glTF file and the bin file for this test case are delivered with this contribution**.The following table shows the times and the keyframes values of the animated images array, stored in the bin file:

|  |  |
| --- | --- |
| **Time** | **Indexes array** |
| 0.0 | [0, 1, 2, 3, 4, 5] |
| 5.0 | [6, 7, 8, 9, 10, 11] |

**5 Test case env-3: video-based picture-based environment cubemap**

This test case is not addressed by an existing extension. The *EXT\_lights\_image\_based* extensions would be a solution if we could reference video content instead of images in the *specularImages* array.

We propose to introduce a new extension *MPEG\_lights\_texture\_based* where the *specularImages* array references texture instead of images.

This new extension is based on the *EXT\_lights\_image\_based* extension. It is used at the root of the glTF file, and at the scene level to describe properties of image-based lights.

**Since we are referencing a texture, this new extension could address image-based lights as well as video-based lights when associated with the *MPEG\_texture\_video* extension.**

The following table gives the semantics of this new extension. The properties that are highlighted are the ones that are changed compared to the *EXT\_lights\_image\_based* extension. This extension declares only one level of texture. The renderer will generate other levels if needed.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Type** | **Usage** | **Default** | **Description** |
| name | string | O |  | Name of the light |
| rotation | Array | O | [0.0,0.0,0.0] | Quaternion that represents the rotation of the IBL environment. |
| intensity | Number | O | 1.0 | Brightness multiplier for environment. |
| irradianceCoefficients | Array | M |  | Declares spherical harmonic coefficients for irradiance up to l=2. This is a 9x3 array. |
| specularImages | Array | M |  | Declares an array of textures for the cubemap, i.e. an array of 6 texture indexes, one for each cube face. |
| specularImageSize | Number | M |  | The dimension (in pixels) of the textures referenced in the specularImages array. |

Table 11: MPEG\_lights\_texture\_based properties

*Note: another Khronos extension (KHR\_lights\_environment* ***Error! Reference source not found.****) is proposed that also addresses image-based lights. It is based on the EXT\_lights\_image\_based extension. The major difference is that KTX2 is used as a container format for cubemap textures and that the cubemaps does not contain prefiltered mip-levels, this shall be done by implementations. Since the KTX2 format only addresses static images, it would be difficult to base our new extension on this one.*

**6 Processing Model**

The use cases described above require lights extension to describe light sources parameters (*KHR\_lights\_punctual* and *EXT\_lights\_image\_based*). *KHR\_animation\_pointe*r extension and our new *MPEG\_lights\_texture\_based* extension are used to implement time evolving lights properties.

When the *KHR\_animation\_pointer* extension is present in the scene description document and is associated to a light element, the Presentation Engine should adjust the glTF light parameters based on the data provided by the animation data.

As noted in the glTF 2.0 specifications, only the storage of animation keyframe is defined and no other runtime behaviors are defined (such as play/stop/loop…). The *MPEG\_animation\_timing* may be used to address these behaviors.

When the MPEG\_lights\_texture\_based extension is present in the scene description document, the presentation engine uses the referenced textures for the faces of the cubemap light.

* + 1. **m60868 [SD] [EE 8] – Timed lighting extensions**
    2. **MPEG\_light\_nature extension**
       1. **General**

MPEG light nature extension, identified by MPEG\_light\_nature, provides the ability to differentiate between real and virtual light sources present as glTF elements. Any light source information externally defined for glTF can be extended with this extension.

* + - 1. **Semantics**

The definition of all objects within MPEG\_light\_nature extension is provided in Table XXX.

**Table XXX – Definition of top-level objects of MPEG\_light\_nature extension**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Type** | **Default** | **Usage** | **Description** |
| nature | string |  | M | Indicates whether the lighting information corresponds to a physical light, a virtual light or unknown.  The allowed values are: “physical”, “virtual” or “unknown”. |

* + - 1. **Processing model**

When, in a scene description document, the scene references light elements, the renderer should determine whether the lighting information is a virtual or a physical lighting information based on the value of the “nature” field. If the scene contains both physical and virtual lighting information, then the renderer should relight the virtual elements of the scene (e.g. virtual objects) present in the scene with both lighting information. Regarding real elements of the scene, the renderer should only relight those with the virtual lighting information since by definition the real elements of the scene are already illuminated by the physical light sources in the scene.

* + 1. **m61176 [SD] On Lighting**

Image-Based Lighting

Image-based lighting (IBL) is the process of illuminating a scene using light images that are captured from the real-world environment. When performed accurately, IBL is able to produce realistic rendering of an augmented reality (AR) scene.

IBL consists of a set of basic steps:

* Capturing an HDR image of the real-world environment. This image is usually an omnidirectional or cube map image.
* Mapping the illumination, extracted from image, onto a representation of the environment.
* Shading the scene based on the mapped illumination and the scene objects.

The following image (found on the web) depicts an example of this process. The ball is a synthetic object, where as the rest is a real-world environment.

A picture containing tree, ground, outdoor, projectile

Description automatically generated

The first step is performed by obtaining a light probe image, which is an HDR omnidirectional image that reflects linearly the light intensity in every direction. When rendering the scene, the shader will do ray tracing to determine the incident illumination on the surface of the scene objects from the IBL environment image. The following figure depicts the shading procedure:

A picture containing text, watch

Description automatically generated

Where rv is the reflected view vector and n is the surface normal.

glTF 2.0 has an extension EXT\_lights\_environment, which provides the ability to define image-based lights in a glTF scene. It supports providing specular radiance and irradiance information in an environment map.

The following is an example of the extension:

|  |
| --- |
| "extensions": {  "EXT\_lights\_image\_based" : {  "lights": [  {  "intensity": 1.0,  "rotation": [0, 0, 0, 1],  "irradianceCoefficients": [...3 x 9 array of floats...],  "specularImageSize": 256,  "specularImages": [  [... 6 cube faces for mip 0 ...],  [... 6 cube faces for mip 1 ...],  ...  [... 6 cube faces for mip n ...]  ],  }  ]  }  } |

The HDR cubemap images can be provided in different mipmap levels.

IBL lighting and the EXT\_lights\_image\_based extension seem to be the right approach to address the needs and requirements of the EE on lighting. We suggest that based on this, support for the EXT\_lights\_image\_based extension should be recommended for AR scenarios that require PBR.

* + 1. **m62011 [SD][EE8] Lighting signalling for MPEG-I SD**

### MPEG\_lights\_punctual extension

#### General

MPEG light punctual extension, identified by MPEG\_light\_punctual, provides the ability to differentiate between real and virtual light sources present as KHR\_lights\_punctual elements as well as providing accessor to retrieve timed dependent information it.

#### Semantics

The definition of all objects within MPEG\_light\_punctual extension is provided in Table XXX.

**Table XXX – Definition of top-level objects of MPEG\_light\_punctual extension**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Type** | **Default** | **Usage** | **Description** |
| light | integer |  | M | Index of the item in the array of the lights parameter of the KHR\_lights\_punctual extension |
| nature | string |  | M | Indicates whether the lighting information corresponds to a physical light, a virtual light or unknown. |
| color\_accessor | integer |  | O | Provides a reference to the accessor giving a sequence of color value as described in color of in KHR\_lights\_punctual.  In the absence of the accessor, the value in the referenced light is considered as the constant value for the rendering duration. |
| intensity\_accessor | integer |  | O | Provides a reference to the accessor giving a sequence of intensity value as described in intensity of KHR\_lights\_punctual.  In the absence of the accessor, the value in the referenced light is considered as the constant value for the rendering duration. |
| range\_accessor | integer |  | O | Provides a reference to the accessor giving a sequence of range value as described in range of in KHR\_lights\_punctual.  In the absence of the accessor, the value in the referenced light is considered as the constant value for the rendering duration. |

#### Processing model

When, in a scene description document, the scene references light elements, the renderer determines whether the lighting information is a virtual or a physical lighting information based on the value of the “nature” field. If the scene contains both physical and virtual lighting information, then the renderer should relight the virtual elements of the scene (e.g. virtual objects) present in the scene with both lighting information. Regarding real elements of the scene, the renderer should only relight those with the virtual lighting information since by definition the real elements of the scene are already illuminated by the physical light sources in the scene.

* + 1. **m63127 [SD][AMD2][EE8] Dynamic lighting in scene description**

The definition of top-level objects within MPEG\_lights\_dynamic extension is provided in **Table *2***.

**Table 2 – Definition of top-level objects of MPEG\_lights\_dynamic extension**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Type** | **Default** | **Usage** | **Description** |
| accessor | integer |  | M | Provides a reference to a timed accessor giving a sequence of timed samples containing light parameters. |

The timed light samples may be expressed in JSON, as illustrated in ***Table 3***. A sample only contains parameters that evolve.

**Table 3 - Example of a JSON file to control a MPEG\_lights\_texture\_based light source.**

{

“duration”: 15,

“samples”: [

{

“cts”: 0.0,

{

“intensity”:0,

"irradianceCoefficients": [...3 x 9 array of floats...]

}

},

{

“cts”: 1.0,

{

“intensity”:2,

"irradianceCoefficients": [...3 x 9 array of floats...],

“rotation”: […]

}

},

{

“cts”: 2.0,

{

“rotation”: […]

}

},

},

…..

]

}

The following table gives an example of glTF description file showing the use of the *MPEG\_lights\_dynamic* extension with an *MPEG\_lights\_texture\_based* light source.

**Table 4 - MPEG\_lights\_dynamic extension in a glTF file**

"extensions": {

"**MPEG\_lights\_texture\_based**": {

"lights": [

{

"intensity": 1.0,

"rotation": [0, 0, 0, 1],

"irradianceCoefficients": [...3 x 9 array of floats...],

"specularImages": [

[... 1 or 6 textures ...]

],

}

]

}

"**KHR\_lights\_punctual**": {

"lights": [

{

"color": [1,1,1],

"intensity": 0.5,

"type": "directional",

"name": "Sun"

},

{

"color": [0,0,1],

"intensity": 1000,

"spot": {

"innerConeAngle": 0.3337942263919169,

"outerConeAngle": 0.39269909262657166

},

"type": "spot",

"name": "Spot.001"

}

]

}

},

"scene": 0,

"scenes": [

{

"name": "Scene",

"nodes": [0, 2,…],

"extensions": {

"MPEG\_lights\_texture\_based": {

"lights”: [0]

}

"MPEG\_lights\_dynamic”: {

"accessor”: 0

}

}

}

],

"nodes": [

{

"mesh”: 0,

"name”: "Cube",

"scale”: [

5,

0.05000000074505806,

5

]

},

{

"extensions”: {

"KHR\_lights\_punctual”: {

"light”: 0

},

"MPEG\_lights\_dynamic”: {

"accessor”: 1

}

},

"name”: "Light\_Orientation",

"rotation”: [

-0.7071067690849304,

0,

0,

0.7071067690849304

]

},

…

# Processing Model

When the presentation engine encounters an *MPEG\_lights\_dynamic* extension beside a light description, it must parse the content of the JSON file referenced by the extension and make the light parameters evolve accordingly. The parameters that are not specified in a sample are kept unchanged.

# Alternative design

For the scenario of having dynamic punctual light (*KHR\_lights\_punctual*), it was proposed during the MPEG meeting that a possible solution aligned with the current design philosophy is to define a new punctual light extension as follows:

**Table XXX – Definition of top-level objects of MPEG\_light\_punctual extension**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Type** | **Default** | **Usage** | **Description** |
| light | integer |  | M | Index of the item in the array of the lights parameter of the KHR\_lights\_punctual extension |
| nature | string |  | M | Indicates whether the lighting information corresponds to a physical light, a virtual light or unknown. |
| color\_accessor | integer |  | O | Provides a reference to the accessor giving a sequence of color value as described in color of in KHR\_lights\_punctual.  In the absence of the accessor, the value in the referenced light is considered as the constant value for the rendering duration. |
| intensity\_accessor | integer |  | O | Provides a reference to the accessor giving a sequence of intensity value as described in intensity of KHR\_lights\_punctual.  In the absence of the accessor, the value in the referenced light is considered as the constant value for the rendering duration. |
| range\_accessor | integer |  | O | Provides a reference to the accessor giving a sequence of range value as described in range of in KHR\_lights\_punctual.  In the absence of the accessor, the value in the referenced light is considered as the constant value for the rendering duration. |

* + 1. **m63116 [SD] [AMD2] Lighting updates proposal**

The updates proposal is related to the semantics (section 8.4.2 of CDAM2) and the processing model (section 8.4.3 of CDAM2) in revision mode.

## Semantics

**Table 2 – Definitions of item in the lights array of MPEG\_lights\_texture\_based extension**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Type** | **Default** | **Usage** | **Description** |
| name | string | N/A | O | Name of the light |
| nature | string | physical | O | Indicates whether the lighting information corresponds to a physical light, a virtual light, both or unknown.  *Editor’s note: To be re-evaluated in light of the possible work on real environment integration in SD whether this attribute remains here or goes to a new node to be defined.* |
|  |  |  |  |  |
| projection | string | equirectangular | O | Provides the projection type (which may be equirectangular or cubemap) of the specular images. In the latter case, the 6 faces of the radiance map should be packed according to the order described by the OMAF specification in section 5.2.3.  *Editor’s note: To be studied if the projection could reference a general specification and not an application standard that has no relation with ISO/IEC 23090-14.* |
| rotation\_accessor | integer | N/A | O | Provides a reference to the accessor giving a sequence of quaternions as described in rotation of in EXT\_lights\_image\_based.  When an anchor refers to a light source, the rotation of the anchor takes precedence over this rotation attribute.  *Editor’s note: Study if position and rotation attribute should be consistent, i.e. either both static or both dynamic.*  *Editor’s note: Study if improvements can be done in case the information is static, i.e. not using an accessor just for one value.* |
| intensity\_accessor | integer | N/A | O | Provides a reference to the accessor giving a sequence of intensity values as described in intensity of in EXT\_lights\_image\_based.  *Editor’s note: The absence of the parameter needs to be specified.*  *Editor’s note: Study if possible improvement can be done in case the information is static, i.e. not using an accessor just for one value.* |
| irradiance\_accessor | integer | N/A | M | Provides a reference to the accessor giving a sequence of irradiance coefficient samples. Irradiance coefficients are defined as in irradianceCoefficients of EXT\_lights\_image\_based.  See Khronos EXT\_lights\_image\_based for the definition of the irradiance coefficient sample.  *Editor’s note: study if sample format needs to be specified and provide it if desired.*  *Editor’s note: she absence of the parameter needs to be specified.*  *Editor’s note: study if improvements can be done in case the information is static, i.e. not using an accessor just for one value.* |
| specular\_images | array | N/A | M | Provides a list of references to textures providing specular images (environment map).  The length of the array shall be equal to “1” or “6”.  When the length of the array is equal to “1”, the specular images are provided in a packed format as defined by the projection attribute.  When the length of the array is equal to “6”, the projection attribute shall be of type cubemap and the order of faces are TBD.  *Editor’s note: Study if Khronos definition could be used, i.e. cube faces are defined in the following order and adhere to the standard orientations as shown in the Khronos EXT\_lights\_image\_based extension.*  *Editor’s note: The absence of the parameter needs to be specified.* |

**Table 3 – MPEG\_lights\_texture\_based object instantiation at the scene level**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Type** | **Default** | **Usage** | **Description** |
| light | integer | N/A | M | Reference to an item in the lights array of the MPEG\_lights\_texture\_based extension. |

**Table 4 – MPEG\_lights\_texture\_based object instantiation at the node level**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Type** | **Default** | **Usage** | **Description** |
| light | integer | N/A | M | Reference to an item in the lights array of the MPEG\_lights\_texture\_based extension. |

## Processing model

When a scene description file becomes available, the Presentation Engine parses the related glTF file and creates the lights defined in the MPEG\_lights\_texture\_based array provided at the glTF file top level.

These lights are then instantiated based on the location of the MPEG\_lights\_texture\_based extension within the scene graph as follows:

* At the scene level: the light is global (i.e. for the entire scene)
* At the node level: the pose of the light is determined through the node TRS hierarchy
* At the anchor level: the pose of the light corresponds to the pose of the anchor
  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. **Test case mix-1: Mix of virtual and real light sources**

|  |  |
| --- | --- |
| **Test case identifier** | mix-1 |
| **Description** | A real scene containing an object on a table is illuminated by an ambient light as well as a main directional light from the top. Next to the real object, a virtual object is located on the table. In addition, a virtual spot illuminates the objects and the surroundings of the objects with a red colour. |
| **Test assets** | Scene with the following 3D objects:   * + Virtual spot light of red colour   + Virtual object   AR Scene recorded with ARCore containing:   * + Ambient lighting information   + Main directional light and intensity   + Video of the scene with the table and the object |
| **Current support** | The following features are supported:   * + Support for shadows   + Support for virtual spot light |
| **Criteria** | A renderer needs to be able to:   * + Render the shadow associated from both the real and virtual lights of the virtual objects   + Render the red colour of the virtual light onto the real scene (object and surrounding) |
| **Location** | Test asset was provided and agreed at MPEG #141. It can be found in m62010. But an FTP failure prevents the upload from happening right now. |

* 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 12 is expected.

Table 12 - 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:  A picture containing night sky  Description automatically generated  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. **Technical solution development**
     1. **Design principles**

**Principle #1**

Reuse light source descriptions defined in Khronos and common to AR frameworks.

**Principle #2**

Signalling agnostic to virtual or real scenes (addressing both VR and AR applications).

**Principle #3**

Signalling common to virtual and real sources of light.

**Principle #4**

Signalling the nature of the source of light for appropriate rendering.

**Principle #5**

Signalling the nature of the source of light for appropriate rendering.

**Principle #6**

Reusing as much as possible glTF extensions related to the lighting topics.

* + 1. On existing **glTF lighting** extension**s**

The EXT\_lights\_image\_based extension should be recommended for AR scenarios that require PBR.

* + 1. **On dynamicity of the lighting information**

For changing properties of the lighting information on the presentation time basis, The KHR\_animation\_pointer is used.

The KHR\_animation\_pointer is used at the “animations/channel” level to specify a JSON pointer that indicates a glTF element to animate. Any element in the glTF file (including elements from an extension) may be specified, provided that this element is “*Animatable*”. The keyframes data and their timing are provided in a separate binary file or in the gltf file as a base64 coded buffer. They are accessed via accessors, bufferview and buffer as for any other glTF animations. The *input* value of the animation sampler points to the list of times and the *output* value points to a list of values for each keyframe.

[Editor’s note: As documented in the related GitLab [issue](http://mpegx.int-evry.fr/software/MPEG/Systems/SceneDescription/MPEG-Contributions/-/issues/397), the use of the KHR\_animation\_pointer extension is conditioned to the feedback from Khronos.]

* 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>