1. **INTERNATIONAL ORGANISATION FOR STANDARDISATION**

**ORGANISATION INTERNATIONALE DE NORMALISATION**

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

**CODING OF MOVING PICTURES AND AUDIO**

**ISO/IEC JTC1/SC29/WG11, N18339**

**March 2019 – Geneva, CH**

|  |  |
| --- | --- |
| **Source** | **Requirements** |
| **Status** | **Output** |
| **Title** | **Requirements on Integration of Scene Description in MPEG-I** |
| **Author** | **Imed Bouazizi** |

# Introduction

ISO/IEC 14496-11 includes hybrid natural/synthetic scene technologies. After close to 20 years MPEG is interested in exploring newer and more functionality-rich solutions that may be available in the industry.

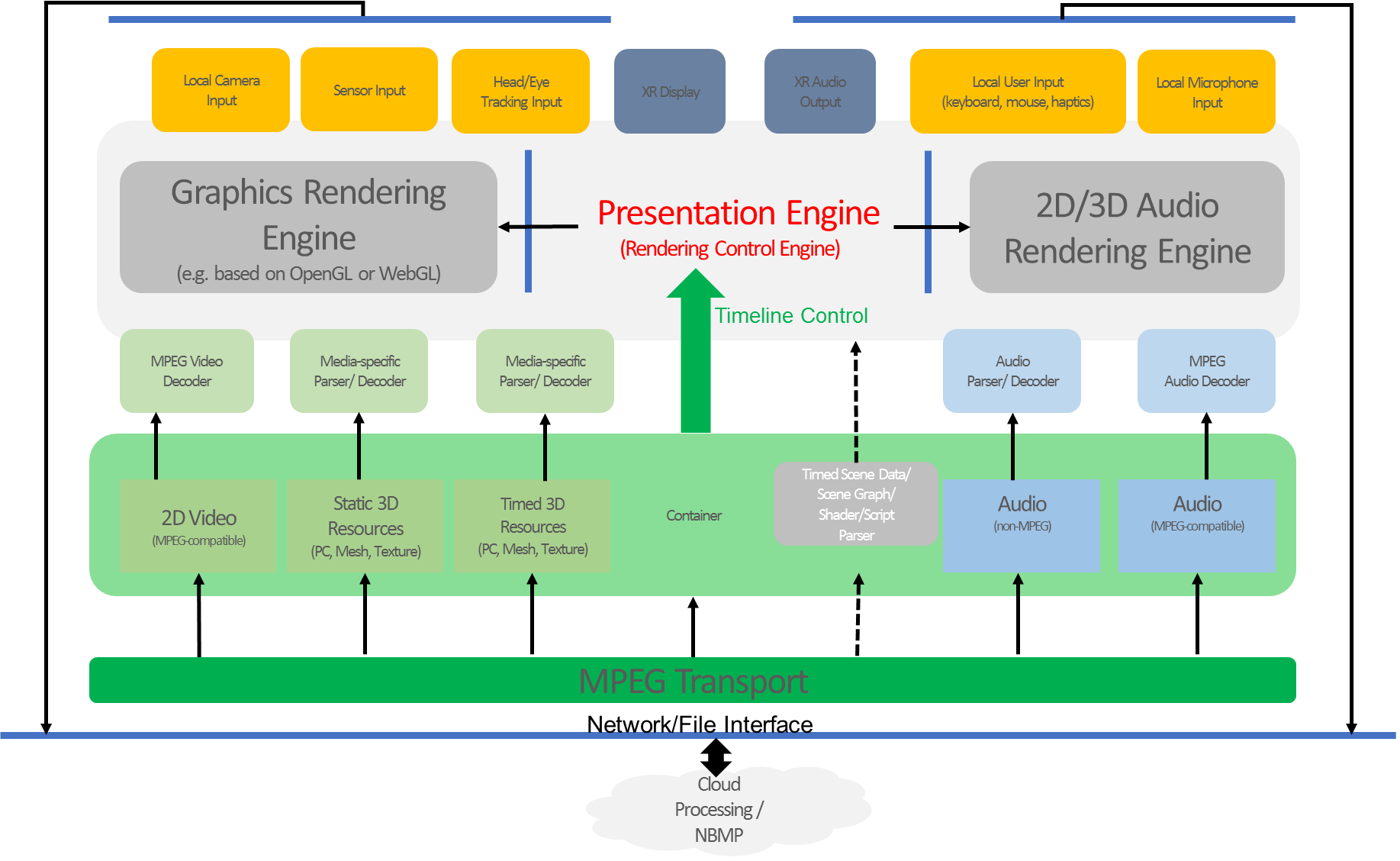
The scope of such a new work is the definition of a Format characterised by its ability to

1. Store 3D data that are
   1. *Synthetic*, i.e. created by a computer, and
   2. *Natural*, i.e. captured from the real world, and
2. Expose interfaces between components of the Format
3. To enable 3D interactive experiences.

In particular, this document assumes that MPEG will integrate existing and well-established scene description formats that fulfil MPEG-I’s requirements into an MPEG-I solution, starting from the architecture and definition of the interfaces. This document defines requirements for selection of the relevant scene description formats and the interfaces to the rest of the components of MPEG-I.

# Overview

MPEG-I is studying the definition of a new scene graph/description format as part of an effort to enable immersive applications and services based on 6DoF and AR/VR/MR technologies. The MPEG-I Architecture group has proposed the following architecture to be used as a reference for all MPEG-I technologies:



This document sets requirements on the interfaces and APIs that need to be developed to integrate existing scene description solutions as part of the overall MPEG-I architecture. For this purpose, a subset of the architecture is used as a reference.



# Definitions

***TBD***

# Requirements

1. General
   1. If possible, the solution shall define interfaces to integrate existing scene description formats rather than define a new scene description format
   2. The solution shall reuse existing interfaces/API definitions (also from other SDOs) whenever possible and appropriate
2. Reference Scene Description Selection
   1. The scene description shall support audio, video and other media formats standardised by MPEG.
   2. The scene description shall enable the support of other visual or audio media formats.
   3. The scene description shall support definitions to indicate how sub-graphs and objects are related in terms of their temporal, spatial and logical relationships
   4. The scene description shall support composition of digital representations of natural and synthetic objects.
   5. The scene description shall support synchronisation between objects and attributes in the scene.
   6. The scene description shall support spatial and temporal random access.
   7. The scene description should support information to enable a renderer to perform path tracing.
   8. The scene description shall support sub-graph representation that allows modular rendering e.g. leafs in the scene description tree can also be packaged and referenced individually from a parent scene description and container.
   9. The scene description should support nodes and attributes in order to implement natural laws of light, energy propagation and physical kinematic operations.
   10. The scene description should support nodes and attributes in order to implement natural laws of acoustic energy propagation and physical kinematic operations.
   11. The scene description should support description of ray-traced camera parameters for rendering
   12. The scene description shall support parametric models for use in rendering environmental acoustic behaviour (e.g. reverberation, occlusion and directivity).
   13. The scene description shall support references (e.g. URLs) to external media resources in place of embedded file references
   14. The scene description shall support a mechanism to safely customize behaviour for nodes like camera, texture, geometry, audio, and object placement nodes through sandboxed, validated domain specific shaders or scripts for these nodes without affecting the functionality or forcing changes to the root node graph or other node types; i.e. provide a mechanism to safely extend the scene description.
3. I-s Interface
   1. *It shall be possible to update the whole scene-graph, a sub-graph, or a node in the scene description*
   2. *It shall be possible to correctly render a 6DoF Presentation after a random access in time*
   3. *It shall be possible to perform timed scene description updates*
   4. *It shall be possible to associate a scene description update with the corresponding scene description*
   5. *It shall be possible to use a scene description as the entry point to a 6DoF presentation*
4. I-m Interface
   1. *It shall be possible to access timed and non-timed, 2D and 3D media (meshes, point clouds, audio elements, …), stored locally or over the network*
   2. *It shall be possible to pre-fetch media that the presentation engine expects to be used in the presentation*
   3. *It shall be possible to retrieve media depending on the desired level of detail*
   4. *It shall be possible to retrieve and access referenced media partially in time and space*
   5. *It shall be possible to describe position, orientation, and visual/acoustic characteristics when rendering referenced media*
   6. *It shall be possible to synchronize media objects/resources and media components of a single object*
   7. *Audio elements shall be rendered consistently with their corresponding visual elements, if such visual elements exist.*
   8. *The specification shall enable synchronization of audio and video of users and the scene.*
5. I-l Interface
   1. *It shall be possible to discover and configure local capture modalities*
   2. *It shall be possible to adjust the presentation based on local capture modality availability*
   3. *It shall be possible to reference media objects that are captured locally using different capture modalities*
   4. *It shall be possible to provide feedback through available actuators*
6. I-i Interface
   1. It shall be possible to discover user interactivity modules
   2. it shall be possible to define custom interactivity procedures based on input from the user or from the user’s devices and sensors
7. Export
   1. *The scene description shall support information to enable a renderer to output raster data (image, and video), volumetric data (point clouds, meshes, arrays of voxels, and reflectance fields) and audio.*
   2. The scene description shall support a scriptable export output node for asynchronously exporting (as a file stream or buffer) any or all parts of the scene description connected to a node into a simpler or flattened representation.

# Potential MPEG Activities

# Overview

Currently, an effort is ongoing to investigate the integration of MPEG media with several scene Graph formats: OSG, USD, glTF2, a-frame, and MPEG-4 BIFS. In addition, the Audio group has defined a format for audio scenes that will be used for their spatial audio renderer.   
It was agreed that MPEG would focus its efforts on a shortlisted set of formats, starting from gltf2. Extensions to glTF2 are possible and easy to make. It was mentioned that there is already an effort to integrate V-PCC encoded point clouds into glTF2 by decoding into the glTF2 PointSet format. Audio extensions and support for other visual objects are thought to be possible as well. Further study of this integration will be performed and document on the MPEG Github repository. It is expected that this effort will help identify the gaps and clarify the added value that MPEG can bring.

# Scene Graph – MPEG Media Interfaces

The media content can be referenced as 3D objects, 2D textures (images and videos) to 3D objects, or audio objects.

The architecture depicts the media interface that connects the presentation engine to the media through the scene description. The media may arrive in a compressed or encoded format (including content encryption) and is passed to the appropriate decoding process to produce a raw representation that is access by the presentation engine.

The presentation engine parses the Scene Graph to identify the referenced media ahead of their usage time. It passes that information to the media retrieval engine with certain constraints. The presentation engine must have a way to associate referenced media with the decoded media data to enable appropriate consumption. Some objects might be decoded into multiple raw data buffers, e.g. a mesh object with its texture. Tight synchronization between the object’s components is essential.



The following components are identified to be standardized:

* Decoded Data Buffer: a buffer that holds decoded media data such as 3D object geometry (vertices) or textures
* Interface to media retrieval engine: an asynchronous request/response interface to request partial media and to receive a reference to the decoded data buffer that holds the media
* Media formats and protocols to perform partial media access and retrieval

## 5.2 Decoded Data Buffer

The decoded data buffer is a buffer that holds raw media data in different formats, e.g. RGB video frames, YUV, raw geometry, Bitmaps, etc… Decoded data buffers may hold a single raw sample or a set of raw samples. In the latter case, the decoded raw samples are presented in the same order as their presentation order. Access to the the decoded data buffer is currently very simple and limited to fetching the next raw sample in presentation order.

Each decoded raw sample shall be associated with a presentation timestamp. It shall also enable partial spatial access to the decoded content. It shall be possible to link a set of Decoded Data Buffers together, so that access to them is done in a synchronous way, allowing simultaneous fetching of content of a logical sample (spread over these buffers) from all linked buffers together. Operations like seeking to a specific position, advancing or gowing backward by one frame are also performed in a synchronized way across the linked Decoded Data Buffers.

The format of the Decoded Data Buffer is to be standardized by MPEG.

## 5.3 Media Retrieval Interface

The media retrieval interface is used by the presentation engine to request media data for consumption. The presentation engine uses the information from the Scene Description as well as the current user’s position, viewport, and device information to determine which media resources (or parts thereof) will be needed at what time. The Media Retrieval Engine will notify the Presentation Engine upon availability of the content. It will indicate in which Decoded Data Buffers the decoded content will be available and in what format.

This interface is to be standardized by MPEG.

## 5.4 Media Formats and Protocols

The Media Retrieval Engine accesses the media data and makes it available in timely manner to the Presentation Engine based on its request. The Media Retrieval Engine is responsible for invoking the necessary media decoding procedures. The decoded media data is stored in one ore more Decoded Data Buffers.

In order to perform these operations, support from the underlying media formats, storage formats, and network protocols is required. These will be leveraged to optimize the media fetching procedures. Examples are formats to support viewpoint/viewport dependent streaming of 3DoF to 6DoF content.

Formats, storage, and streaming of this media content is expected to be standardized.

# References

[1] “ITU-T/ITU-R/ISO/IEC Common Patent Policy”, March 2007#

# Annex A - Reference Scene Graph Formats

# A.1 Introduction

A set of scene graph formats has been identified to be used as reference for the design of the i-m interface. These formats are listed here:

* OpenSceneGraph (OSG)
* Universal Scene Description (USD)
* glTF 2.0
* MPEG BIFS

The next step of this evaluation is to answer the following questions for each of these reference scene graph formats:

* Does your scene description allow to add external media such as still images textures, videos, audio sources, etc.
  + If so, how is the media referenced?
* Does the Graph require that the media be included in the package or can it be external?
  + If external, what type of data reference do you permit?
* How does it handle the API to real-time media that is external, for example can it stream media into the scene (similar to W3C MSE)?
* How does it handle the synchronization of different external media objects/streams?
* Does it support late binding of media in the scene description?
* If the media is adaptive, does the scene graph handle the adaptation logic?
* Does the Scene Graph have mechanisms to update the scene, e.g. programmatically or through external means?

# A.2 Analysis of OSG

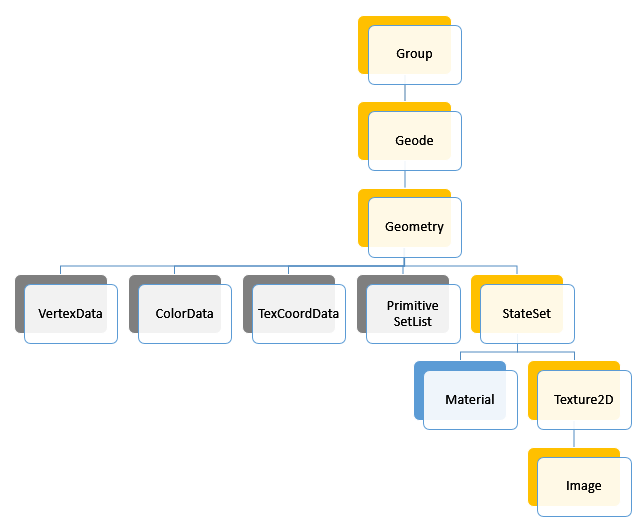
**Does your scene description allow to add external media such as still images textures, videos, audio sources, etc?**

**If so, how is the media referenced?**

→ Yes, OSG allows to add external media.

1) Example of still image textures

|  |
| --- |
| **osg::Group** {  UniqueID 1  Name "cow.osg"  DataVariance STATIC  **Children 1 {**  **osg::Geode {**  UniqueID 2  Name "cow.osg"  DataVariance DYNAMIC  **Drawables 1 {**  **osg::Geometry {**  UniqueID 3  DataVariance DYNAMIC  **StateSet** TRUE {  **osg::StateSet** {  UniqueID 4  DataVariance STATIC  **ModeList** 2 {  GL\_CULL\_FACE OFF  GL\_LIGHTING ON  }  **AttributeList** 1 {  osg::Material {  UniqueID 5  DataVariance STATIC  Ambient TRUE Front 0.5 0.5 0.5 1 Back 0.5 0.5 0.5 1  Diffuse TRUE Front 1 1 1 1 Back 1 1 1 1  Specular TRUE Front 1 1 1 1 Back 1 1 1 1  Emission TRUE Front 0 0 0 1 Back 0 0 0 1  Shininess TRUE Front 1 Back 1  }  Value OFF  }  **TextureModeList** 1 {  Data 3 {  GL\_TEXTURE\_GEN\_S ON  GL\_TEXTURE\_GEN\_T ON  GL\_TEXTURE\_2D ON  }  }  **TextureAttributeList** 1 {  Data 2 {  **osg::Texture2D** {  UniqueID 6  DataVariance STATIC  WRAP\_S REPEAT  WRAP\_T REPEAT  WRAP\_R REPEAT  MIN\_FILTER NEAREST\_MIPMAP\_LINEAR  MAG\_FILTER LINEAR  UnRefImageDataAfterApply TRUE  ResizeNonPowerOfTwoHint TRUE  **Image** TRUE {  UniqueID 7  **FileName "Images/reflect.rgb"**  WriteHint 0 2  DataVariance STATIC  }  }  Value OFF  osg::TexGen {  UniqueID 8  DataVariance STATIC  Mode SPHERE\_MAP  PlaneS 1 0 0 0  PlaneT 0 1 0 0  PlaneR 0 0 1 0  PlaneQ 0 0 0 1  }  Value OFF  }… |



**Figure 1.** OSG Scene graph example

2) Example of video

Video can be included as still image texture.

|  |
| --- |
| osg::ref\_ptr<osg::Geode> geode = new osg::Geode;  osg::ref\_ptr<osg::Drawable> quad = osg::createTexturedQuadGeometry( osg::Vec3(), osg::Vec3(1.0f, 0.0f, 0.0f), osg::Vec3( 0.0f, 0.0f, 1.0f) );  osg::ref\_ptr<osg::Texture2D> texture = new osg::Texture2D;  osg::ref\_ptr<osg::Image> image;  image = osgDB::readImageFile( "/dev/video0.ffmpeg" );  osg::ImageStream\* imageStream = dynamic\_cast<osg::ImageStream\*>( image.get() );  if ( imageStream ) imageStream->play();  texture->setImage( image.get() );  quad->getOrCreateStateSet()->setTextureAttributeAndModes( 0, texture.get() );  geode->addDrawable( quad.get() );  //Start the viewer.  osgViewer::Viewer viewer;  viewer.setSceneData( geode.get() ); |

osg:ImageStream class supports an image stream, which manages sub-images in the data buffer. It can be derived to read data from video files or the internet. OSG has already had a few built-in plugins that support the loading and playing of AVI, MPG, MOV, and other file format.

**Does the Graph require that the media be included in the package or can it be external?**

→ The media can be external.

However, the shader source can be in the package as well as external.

|  |
| --- |
| osg::Geode {  UniqueID 1  Name "glsl\_julia.osg"  DataVariance DYNAMIC  StateSet TRUE {  osg::StateSet {  UniqueID 2  DataVariance STATIC  AttributeList 1 {  osg::Program {  UniqueID 3  DataVariance STATIC  Shaders 2 {  osg::Shader {  UniqueID 4  DataVariance DYNAMIC  Type VERTEX  ShaderSource 35 {  "//"  "// Vertex shader for drawing … sets"  "//"  "// Authors: … "  …} |

**If external, what type of data reference do you permit?**

→ There are 45 plugins in the OSG[2]. These plugins provide the support for reading and writing both native and 3rd party file formats.

1) List of available movie & image

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **name** | **Description** | **Extensions** | **Read** | **Write** |
| bmp | Window bitmap | .bmp | yes | yes |
| dds | DirectX image format | .dds | yes | yes |
| gdal | GDAL image loaders | All extensions supported by GDAL | yes | no |
| gif | GIF Image | .gif | yes | no |
| hdr | High Dynamic Range image | .hdr | yes | yes |
| jp2 |  | .jps .jpc | yes | yes |
| jpeg | Jpeg image | .jpeg .jpg | yes | yes |
| Openflight | Image attribute files | .attr | yes | yes |
| pic | PIC image | .pic | yes | no |
| png | Portable Network Graphics | .png | yes | yes |
| pnm |  | .pnm. ppm .pgm .pbm | yes | yes |
| rgb |  | .rgb .sgi .rgba .int .inta .bw | yes | yes |
| tga | Targa image | .tga | yes | no |
| tiff | Tiff image | .tiff .tif | yes | yes |
| quicktime | Quicktime movie loader | .mov .mpg .mpv .mp4 .m4v .dc .avi .flv .swf | yes | yes |
|  | Quicktime image loader | rgb. rgba .jpg .jpeg .tif .tiff .gif .png .pict .pct .tga .psd | yes | yes |
| xine | Xine image stream | .mpg .mpv .db .flv .mov .avi .wmv. xine | yes | no |

2) List of available 3D database

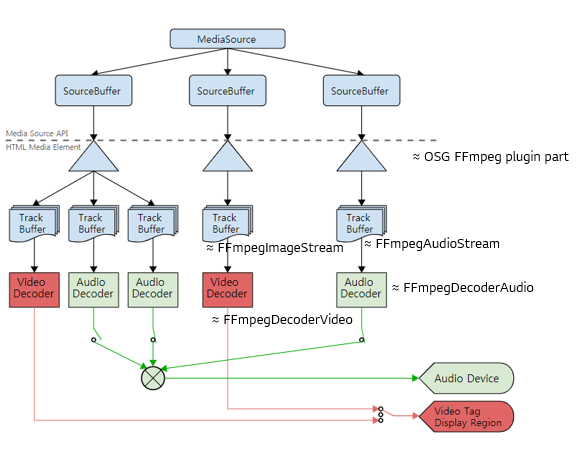
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Plugin name** | **Description** | **Extensions** | **Read** | **Write** |
| 3dc | 3DC point cloud reader | .3dc .asc | yes | no |
| 3ds | 3D Studio | .3ds | yes | yes |
| ac3d | AC3D modeler | .ac | yes | yes |
| bsp | Quake3 BSP | .bsp | yes | no |
| dae | COLLADA 1.4.x | .dae | yes | yes |
| dw | Design Workshop Database | .dw | yes | no |
| dxf | Autodesk DXF Reader | .dxf | yes | no |
| fbx | Autodesk FBX | .fbx | yes | yes |
| geo | Geo | .gem .geo | yes | no |
| Inventor | Open Inventor format | .iv .wrl | yes | yes |
| ive | **Native osg binary** | .ive | yes | yes |
| logo | Logo database | .logo | yes | no |
| lwo | Lightwave Object | .lwo .lw .geo | yes | no |
| lws | Lightwave Scene | .lws | yes | no |
| md2 | Quake MD2 | .md2 | yes | no |
| obj | Alias Wavefront | .obj | yes | yes |
| ogr |  | .ogr | yes | no |
| Openflight | Multigen Openflight | .flt | yes | yes |
| osg | **Native osg ascii** | .osg | yes | yes |
| pfb | Performer loader | All extensions supported by Performer loaders | yes | no |
| shp | ESRI Shapefile | .shp | yes | no |
| stl | Stereolithography file | .stl .sta | yes | no |
| vrml | VRML2 | .wrl | yes | no |
| x | DirectX 3D model | .x | yes | no |

## *How does it handle the API to real-time media that is external, for example can it stream media into the scene (similar to W3C MSE)?*

→ Yes, OSG can handle real-time media.

OSG includes ffmpeg built-in plugin, which supports audio and video decoding and frame control. osg::ImageSteam and osg::AudioStream class are extended to FFmpegImageStream and FFmpegAudioStream in ffmpeg plugin. FFmpegImageStream and FFmpegAudioStream with the ffmpeg decoder can request the next packet, fetch the decoded frame/sample data and store the data in the buffer.

Figure 2 shows how W3C MSE MediaSource connected to HTMLMediaElement[3] with the corresponding objects of OSG ffmpeg plugin.



**Figure 2.** W3C MSE: MediaSource object and HTMLMediaElement

## *How does it handle the synchronization of different external media objects/streams?*

→ OSG can handle time-based synchronization between different objects in scene.

OSG has one osg::Timer instance, which is referred from the objects. osg::ImageStream and osg::AudioStream will be played based on time. osg::Sequence is group node which allows automatic time-based switching between children. osg::AnimationPath applies a time varying transformation pathway.

The following table shows the elements of the OSG sequence.

|  |  |  |
| --- | --- | --- |
| **Element** | **Type** | **Description** |
| FrameTime | List of floats | Specifies the display duration of each child |
| LastFrameTime | float | Special time to display last frame of the last loop |
| LoopMode | integer | 0= LOOP start over at the beginning  1= SWING pingpong animation back and forth |
| IntervalBegin | integer | First child to take part in animation |
| IntervalEnd | integer | Last child to take part in animation |
| DurationSpeed | float | Multiplier of real-time clock, set to N to go N times faster |
| DurationNReps | integer | How many times to repeat, -1 means repeat forever |
| SequenceMode | integer | 0= START, 1=STOP, 2=PAUSE, 3=RESUME |

The following shows the example of osg::AnimationPath.

|  |
| --- |
| osg::Group {  UniqueID 9  DataVariance DYNAMIC  Children 3 {  osg::MaxtrixTransform {  UniqueID 10  Name "hours"  DataVariance DYNAMIC  UpdateCallback TRUE {  osg::AnimationPathCallback {  UniqueID 11  DataVariance DYNAMIC  AnimationPath TRUE {  osg::AnimationPath {  UniqueID 12  DataVariance DYNAMIC  TimeControlPointMap 3 {  Time 0 {  Position 0 0 0.5  Rotation 0 -0.707107 -0.707107 0  Scale 1 1 1  }  Time 21600 {  Position 0 0 0.5  Rotation 0 -0.707107 0.707107 0  Scale 1 1 1  } …  …} |

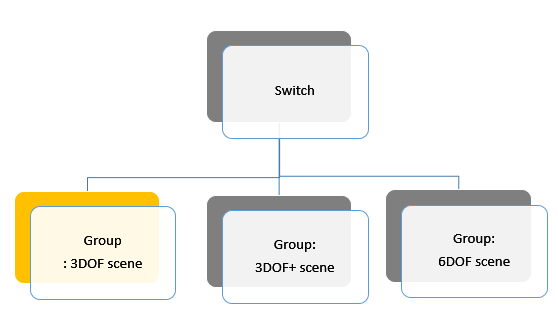
## *Does it support late binding of media in the scene description?*

→ Yes, late binding of media can be implemented in OSG plugin like ffmpeg plugin.

## *If the media is adaptive, does the scene graph handle the adaptation logic?*

→ No, OSG itself cannot handle the adaptation logic without modification.

Adaptive media contents can be represented with OSG. osg::Switch is a group node that allows switching between children. Typical uses would be for objects which might need to be rendered differently at different times. One or more of children can be selected to be rendered. Thus, we can create the scene like Figure 3. According to the capability of the player, an object can be activated. For instance, if the player cannot handle 3DOF+ and 6DOF, the first child node in Figure 3 can be selected.



**Figure 3.** An example with switch node:   
The 3DOF scene is selected to render, the others will not be rendered.

However, the selection logic is missing. Checking the capability of the player and selecting the proper child node from the switch node is required to be implemented inside of the player.

# Annex B Reference Usage Scenarios

# B.1 360 Degree Image and Video

A Scene Graph is used to describe a scene that consists of a simple 360 degree image. The image is referenced locally or through the network.

Different variations of this usage scenario are also considered:

* The referenced content is a 360 video formatted as an OMAF file
* The video is streamed over the network e.g. using DASH
* The geometry of the image or video is different than a sphere, e.g. a cube
* The video is in stereoscopic format
* The video is included in an OMAF file as one or multiple tracks, but the file also contains an audio track

# B.2 3D Audio and 360 Degree Video

A Scene Graph is used to describe a scene that consists of a simple 360 degree image and 3D audio, both contained in one local file. As an extension to the above

* The audio is stereo audio
* The audio is 3D audio
* The audio contains 3D audio and a non-diegetic content

# B.3 360 Degree Video and Subtitle

A Scene Graph is used to describe a scene that consists of a simple 360 degree image and a subtitle track. The rendering of the subtitle track is handled is controlled the scene description.

# B.4 Combination with local objects

In extensions to 2.1, 2.2 and 2.3 a local object is present with the scene description:

* An image is placed as an overlay to the scene
* A local audio source is added

# B.5 A 3D scene with a 2D Video

In this usage scenario, a 2D video is used to display content of a 3D flat screen hanging on a wall as shown in the following figure.



The Scene Graph is used to compose the 3D and 2D scene and to display the 2D video on the 3D area. The 2D video can be referenced locally or through the network.