**INTERNATIONAL ORGANISATION FOR STANDARDISATION**

**ORGANISATION INTERNATIONALE DE NORMALISATION**

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

**CODING OF MOVING PICTURES AND AUDIO**

**ISO/IEC JTC1/SC29/WG11 MPEG/M50846**

**October 2019, Geneva, CH**

|  |  |
| --- | --- |
| **Source** | **Interdigital** |
| **Status** | **Input Document** |
| **Title** | **Generalized viewing space in Metadata for Immersive Video** |
| **Authors** | Renaud Doré, Bertrand Chupeau, Gérard Briand |

# Context

There has been a contribution [1] at last MPEG127 Goeteborg highlighting the lack in the WD [2] of the definition for a viewing space. This space would inform the client application where the rendering can occur in good conditions. The way how the application would be dealing with such an invention is application dependent, but one could list for example the following behavior:

* Turn the viewport gracefully to grey and fade out when approaching the limit of the viewing space
* Trigger inpainting, as mentioned by [1]
* Indicates to the viewer the direction where the viewing space extends further, in the case of the elongated form
* Etc.

The client has as its disposal the information of the camera extrinsic parameters. But it may not be obvious how to derive from that list the viewing space when the shape is not trivial. As a matter of fact, a new content PoznanFencing has been introduced in the CTC [3] at MPEG127 where the rig shape is no more compact but constitutes an arc of a more than 5 meters length, with cameras targeting at the arc center.

Also, it may be the decision of the content creator to define where the content is supposed to be viewed.

As a result, we agree with the authors of [1] that the definition of a viewing space is necessary in the MIV specification.

# Proposed solution

Our proposition starts from [1] but modifies the way how to generate an arbitrary shape.

The contribution [1] starts from the definition of elementary (or primitive) viewing box as defined by OMAF version 2 (sphere, cylinder, ellipsoid, ..) with size and positions. It then proposes logical operations on a list of these primitives by using CSG (Constructive Solid Geometry).

The limits of this CSG approach are twofold:

* It does not allow to define the viewing direction in all the viewing space, as mentioned in the Ad Hoc Meeting minutes
* It is not guaranteed to construct a smooth shape, and to elaborate a smooth degree of inclusion within the global viewing space: in a non trivial content like PoznanFencing already mentioned, when the client device will move inside the viewing space, it may suddenly pass from a one geometrical part to another. There will no way to define a smooth fade out for example. Or the number of primitives will have to be very high.

We propose instead to **create a path by using an ordered set of the viewing box primitives** along which an index is computed for all points of the viewing space. In short, the global viewing space is generated by displacement of the primitives along this space. In practice, it would be natural to place these primitives at the location of the cameras, but there can be more of them, or less.

# Rationale

Let the position P of the rendering device be known and equal to (xx,yy,zz).

Let the metadata be this ordered set { VBi, i=0 to n-1}. Each VBi does not only defines size and position, but also viewing direction.

We have to show that it is possible to derive from this list a smooth index. One possible implementation (which does not have to be normative of course) is the following;

1. At each VBi, a bisectrix plan is computed splitting the angle <VBi-1, VBi, VBi+1>, and simply orthogonal plan at each end of the list. It creates a partition of the 3D space.
2. The point looks for its closest neighbor VBj
3. The point looks for the relevant segment of attachment VBj-1-VBj or VBj-VBj+. Let’s call it VBk-VBk+1
4. The point P computes a distance index sk and sk+1 to VBk and VBk+1 respectively and elaborate its index s= sk/(sk + sk+1). By construction, this index goes from 0 to 1 when the point moves through its partition. When it crosses a partition frontier, it switches from 1 to 0 or vice-versa.
5. The point uses this index s for
   1. Interpolating the viewing box direction from the one of VBk and VBk+1
   2. Interpolating the projection plan to the VBk-VBk+1 segment leading a point P’
   3. Obtaining a shifted version of the elementary viewing box from P’ center

This implementation is illustrated in the following figure with a case similar to PoznanFencing.

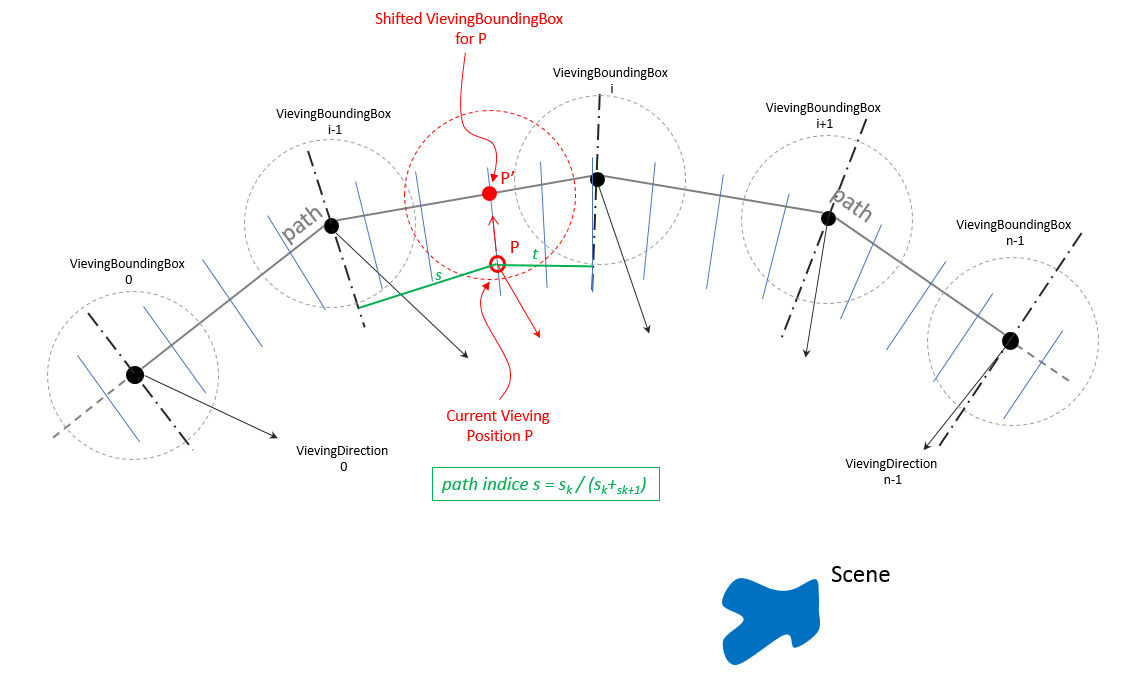


Figure : possible implementation using proposed metadata

The proposed simple algorithm has been computed for sake of demonstration on the PoznanFencing Case. The viewing box primitives have been (naturally) chosen to be at the location of camera v0, v2, v4, v6 and v8, taken in that order.

The Viewing space computed for that rig is illustrated below, where the viewing space is made visible by showing the contour in blue where the distance is equal to 1. As expected, it shows a tuboid form.

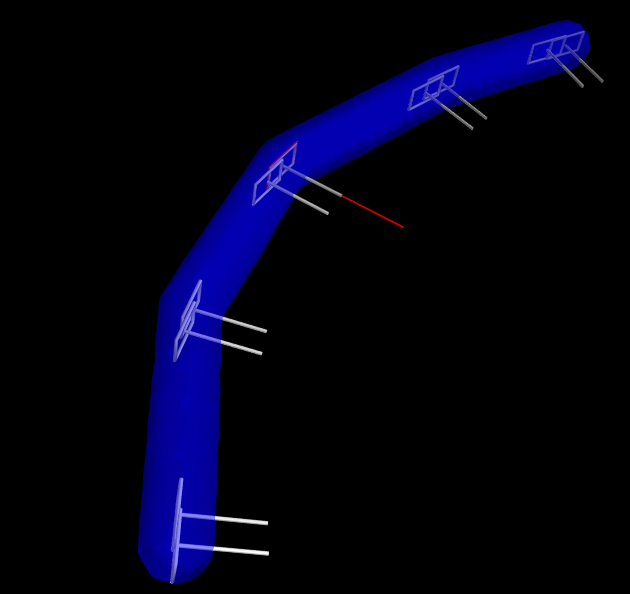


Figure : representation of viewing space in blue for PoznanFencing case when applying proposed algorithms

# Syntax update proposal

## Viewing space syntax

|  |  |
| --- | --- |
| viewing\_space( ) { | **Descriptor** |
| viewing\_box() |  |
| **num\_paths\_minus1** | u(8) |
| for( n = 0; n <= num\_paths\_minus1; n++ ) { |  |
| **num\_points\_minus1**[ n ] | u(32) |
| for( i = 0; i <= num\_points\_minus1[ n ]; i++ ) { |  |
| **viewing\_pos\_x**[ n ][ i ] | i(32) |
| **viewing\_pos\_y**[ n ][ i ] | i(32) |
| **viewing\_pos\_z**[ n ][ i ] | i(32) |
| **viewing\_dir\_azimuth\_centre**[ n ][ i ] | i(32) |
| **viewing\_dir\_elevation\_centre**[ n ][ i ] | i(32) |
| **viewing\_dir\_azimuth\_range**[ n ][ i ] | u(32) |
| **viewing\_dir\_elevation\_range**[ n ][ i ] | u(32) |
| } |  |
| } |  |
| } |  |

|  |  |
| --- | --- |
| viewing\_box( ) { | **Descriptor** |
| **shape\_type** | u(3) |
| if( shape\_type == 0 ) { |  |
| sphere\_box() |  |
| } |  |
| if( shape type == 1 ) { |  |
| cuboid\_box() |  |
| } |  |
| } |  |

|  |  |
| --- | --- |
| sphere\_box( ) { | **Descriptor** |
| **scale** | u(8) |
| **radius** | u(32) |
| } |  |

|  |  |
| --- | --- |
| cuboid\_box( ) { | **Descriptor** |
| **scale** | u(8) |
| **x\_dim** | u(32) |
| **y\_dim** | u(32) |
| **z\_dim** | u(32) |
| } |  |

## Viewing space semantics

**num\_paths\_minus1** plus 1 specifies the number of navigation paths which are used to define the viewing space.

**num\_points\_minus1**[ n ] plus 1 specifies the number of sampled 3D points along the n-th navigation path.

**viewing\_pos\_x**[ n ][ i ], **viewing\_pos\_y**[ n ][ i ] and **viewing\_pos\_z**[ n ][ i ] are fixed-point 16.16 values defining the 3D coordinates of the i-th sampled position along the n-th path in the global reference coordinate system. The points are ordered according to the curvilinear abscissa along the path; the n-th path is therefore defined in-between the first and last endpoints with index 0 and index num\_points\_minus1[ n ], respectively.

**viewing\_dir\_azimuth\_centre**[ n ][ i ] and **viewing\_dir\_elevation\_centre**[ n ][ i ] specify the azimuth and elevation values, respectively, which define the central viewing direction at the i-th position along the n-th navigation path in units of 2−16 degrees, according to the global reference coordinate axes. Azimuth values shall be in the range of −180 \* 216 to 180 \* 216 − 1, inclusive. Elevation values shall be in the range of −90 \* 216 to 90 \* 216, inclusive.

**viewing\_dir\_azimuth\_range**[ n ][ i ] and **viewing\_dir\_elevation\_range**[ n ][ i ] specifiy the azimuth and elevation ranges, respectively, through the central viewing direction, at the i-th position along the n-th path in units of 2−16 degrees. Azimuth range shall be in the range 0 to 360\*216. Elevation range shall be in the range 0 to 180\*216.

**shape\_type** specifies the geometric shape of the elementary viewing bounding box. shape\_type equal to 0 indicates that the viewing bounding box is a sphere. shape\_type equal to 1 indicates that the viewing bounding box is a cuboid. The values 2 to 7 are reserved.

**scale** is an integer value between -128 and +127 which indicates the exponent base 2 of the scale value used in measurement. The base unit is one meter.

**radius** specifies the radius of a sphere as 16.16 fixed-point value in distance scale.

**x\_dim, y\_dim, z\_dim** specify the dimensions of a cuboid along the X, Y and Z directions of the global reference coordinate system, respectively, as 16.16 fixed-point value in distance scale. When associated with the i-th sample along the n-th viewing path, the cuboid viewing box is centered at that position and oriented along the associated central viewing direction.

# Conclusion

We propose to extend the proposition of [1] by metadata under the form of an ordered set of viewing box primitives. It is compact, costs about nothing in term of bitrate, and fulfills the extended requirements mentioned in this document. We have computed a possible behavior of a client application and have shown that it generates a smooth behavior compatible with user experience expectation.

We recommend to push these very small metadata in the WD3.

# References

1. Generalized viewing space signaling in Metadata for Immersive Video, Vinod Kumar Malamal Vadakital, Kimmo Roimela, Lauri Ilola, Jaallo Keranen, m49226 July 2019
2. “Working Draft 2 of Immersive Video”, ISO/IEC JTC1/SC29 WG11 (MPEG) output document N18576, Gothenburg, SW, July 2019.
3. “Common Test Conditions for Immersive Video”, ISO/IEC JTC1/SC29 WG11 (MPEG) output document N18563, Gothenburg, SW, July 2019.
4. S Deshpande, Y.K. Wang, “Technologies under consideration for ISO/IEC 23090-2 (OMAF)”, ISO/IEC JTC1/SC29/WG11 MPEG/w18403, April. 2019, Geneva, Switzerland.

[**5]** AhG report on MPEG-I Visual Technologies, Gothenburg, SW, m49711, July 2019