**INTERNATIONAL ORGANISATION FOR STANDARDISATION**

**ORGANISATION INTERNATIONALE DE NORMALISATION**

**ISO/IEC JTC 1/SC 29/WG 4**

**MPEG VIDEO CODING**

**ISO/IEC JTC 1/SC 29/WG 4 m56972**

**April 2021, Online**

**Title: On view IDs vs view indices in pruning graph signalling**

**Source: Bertrand Chupeau (InterDigital)**

**Abstract**

This document proposes to clarify the signalling of parent views in the pruning graph as specified in ISO/IEC DIS 23090-12 MPEG Immersive Video (MIV) by consistently only using view IDs instead of mixing with view indices.

# Introduction

Although correct, the syntax and semantics of the pruning graph parameters in the MIV specification are mixing view IDs and view indices, which may be confusing. This has been pointed out and discussed in GitLab (issue #244 of 23090-12 project).

For clarification and simplification purpose, this contribution proposes to signal parent view IDs instead of indices in the pruning\_parents( viewID ) syntax structure, and update the decoding process and Annex H accordingly. The modifications to the specification are indicated hereunder.

# Pruning parents syntax (clause 8.2.1.6.8)

Rename syntax element pp\_parent\_idx into pp\_parent\_id.

|  |  |
| --- | --- |
| pruning\_parents( viewID ) { | **Descriptor** |
| **pp\_is\_root\_flag**[ viewID ] | u(1) |
| if( !pp\_is\_root\_flag[ viewID ] ) { |  |
| **pp\_num\_parents\_minus1**[ viewID ] | u(v) |
| for( i = 0; i <= pp\_num\_parents\_minus1[ viewID ]; i++ ) |  |
| **pp\_parent\_id~~x~~**[ viewID ][ i ] | u(v) |
| } |  |
| } |  |

# Pruning parents semantics (clause 8.3.1.6.8)

Update the semantics accordingly.

**pp\_parent\_id~~x~~**[ viewID ][ i ] specifies the ~~index~~ ID of the i-th parent view for the view with view ID equal to viewID in the pruning graph at encoder stage. pp\_parent\_id~~x~~[ viewID ][ i ] ~~shall be in the range of 0 to mvp\_num\_views\_minus1 inclusive, but~~ shall not be equal to viewID~~ToIndex[ viewID ]~~.

# Common atlas frame MIV extension decoding process (clause 9.7.5)

## General decoding process for view parameters (clause 9.7.5.1)

Replace the ViewParentIndex output array with ViewParentID output array.

ViewParent~~Index~~ID[ viewID ] [ i ] specifies ~~index~~ID of the i-th parent view for the view with view ID equal to viewID in the pruning graph at encoder stage. ViewParent~~Index~~ID [ viewID ] [ i ] ~~shall be in the range of 0 to NumViews – 1, inclusive, and~~ shall not be equal to ViewID~~toIndex[viewID]~~.

(…)

At the start of each CVS, these arrays are initially set as follows:

NumViews = 0  
 for( v = 0; v <= 65 535 ; v++ ) {  
(…)  
 for( i = 0; i < pp\_num\_parents\_minus1[ v ]; i++) {   
 ViewParent~~Index~~ID[ v ][ i ] = 0  
 }  
 }

## Initializing view information process (clause 9.7.5.2)

Update the initialization process accordingly.

If mvp\_pruning\_graph\_params\_present\_flag equal to 1, the pruning related arrays are set as follows:

for( v = 0; v < NumViews; v++ ){  
 viewID = ViewIndexToID[ v ]  
 ViewRoot[ viewID ] = pp\_is\_root\_flag[ viewID ]  
 ViewNumParents[ viewID ] = pp\_num\_parents\_minus1[ viewID ] + 1  
 for( i = 0; i < ViewNumParents[ viewID ] ; i++ ){  
 ViewParent~~Index~~ID[ viewID ][ i ] = pp\_parent\_id~~x~~[ viewID ][ i ]  
 }  
 }

# Sample weighting recovery process in informative Annex H (clause H.7)

Update the entire process using view IDs instead of indices.

This process specifies how to use a pruning graph to recover a view blending weight for a sample within a pruned view. It is common for multiview renderers to have a weight per view that depends on the viewport position. This process recovers the sample weight when views are pruned.

**7.1 Pruning graph parsing process**

This process converts the global variables output of the decoding process of the pruning graph information into variables for input to the sample weighting recovery process as specified in subclause H.7.2.

Outputs of this process are:

* for each of the views, with index v in the range of 0 to NumViews - 1, inclusive, and view ID equal to viewID:
  + the variables ViewLeaf[ ~~v~~viewID ] and ViewNumChildren[ ~~v~~viewID ],
  + the 1D array ViewChild~~Index~~ID[ ~~v~~viewID ] of size ViewNumChildren[ ~~v~~viewID ].

The process operates as follows:

for ( v = 0; v < NumViews; v++ ) {  
 viewID = ViewIndexToID[ v ]  
 ViewLeaf[ ~~v~~viewID ] = 1  
 ViewNumChildren[ ~~v~~viewID ] = 0  
 }   
   
 for ( v = 0; v < NumViews; v++ ) {  
 viewID = ViewIndexToID[ v ]  
 if ( ViewRoot[ viewID ] == 0 ) {  
 for ( i = 0; i < ViewNumParents[ viewID ]; i++) {  
 j = ViewParent~~Index~~ID[ viewID ][ i ]  
 ViewLeaf[ j ] = 0  
 ViewChild~~Index~~ID[ j ][ ViewNumChildren[ j ]++ ] = ~~v~~viewID  
 }  
 }  
 }

**7.2 Sample weighting recovery process**

This process computes the weight of the contribution to the target view of a sample from a reconstructed pruned view with ~~index v~~ view ID equal to viewID.

Inputs to this process are:

* the position p = ( x, y ) of a sample with valid depth in a reconstructed view with ~~index v~~ view ID equal to viewID,
* for each of the views with index ~~k~~v, in the range of 0 to NumViews - 1, inclusive, and view ID k equal to ViewIndexToID[ v ]:
  + the view weight with respect to target view, ViewWeight[ k ] determined by external means,
  + if mvp\_pruning\_graph\_params\_present\_flag is equal to 1, the pruning graph information composed of the variables ViewLeaf[ k ] and ViewNumChildren[ k ] and the 1D array ViewChild~~Index~~ID[ k ] of size ViewNumChildren[ k ].

Output to this process is:

* the weight of the contribution of the sample to the target view, sampleWeight.

The pixel weighting recovery process operates as follows for pixel p with coordinates (x, y):

if ( mvp\_pruning\_graph\_params\_present\_flag == 0 )  
 sampleWeight = ViewWeight[ ~~v~~viewID ]  
 else  
 sampleWeight = ViewWeight[ ~~v~~viewID ] +  
 ComputeChildrenWeight (~~v~~viewID, Unproject(~~v~~viewID, p))

The invoked function ComputeChildrenWeight(~~v~~viewID, P) recursively operates as follows:

ComputeChildrenWeight(~~v~~viewID, P) {  
 w = 0  
 if ( ViewLeaf[ ~~v~~viewID ] == 0 ) {  
 for( i = 0; i < ViewNumChildren[ ~~v~~viewID ] ; i++) {  
 vChild = ViewChild~~Index~~ID[ ~~v~~viewID ][ i ]  
 pOnChild = Project( vChild, P )  
   
 if( IsInViewport( vChild, pOnChild ) == 1) {  
 if ( IsOccupied( vChild, pOnChild ) == 0 ) {  
 w += ViewWeight[ vChild ] + ComputeChildrenWeight( vChild, P )   
 }  
 }  
 else {  
 w += ComputeChildrenWeight (vChild, P)  
 }  
 }  
 }  
 return w  
 }

where:

* the function UnProject(~~v~~viewID, p) returns the 3D point P which projects onto pixel p in the ~~v-th~~ view with ID equal to viewID as specified in subclause H.2.4,
* the function Project(~~v~~viewID, P) returns the pixel coordinates of the projection of 3D point P in the ~~v-th~~ view with ID equal to viewID,
* the function IsInViewport(~~v~~viewID , p) returns true if pixel p is inside the viewport of the ~~v-th~~ view with ID equal to viewID,
* the functionIsOccupied(~~v~~viewID, p) returns false if the depth of pixel p in the ~~v-th~~ reconstructed view with ID equal to viewID is invalid.

# Recommendations

Adopt into the Study of DIS document.