INTERNATIONAL ORGANISATION FOR STANDARDISATION
ORGANISATION INTERNATIONALE DE NORMALISATION
ISO/IEC JTC1/SC29/WG11
CODING OF MOVING PICTURES AND AUDIO

ISO/IEC JTC1/SC29/WG11
MPEG2002/M8929
October 2002, Shanghai, China

Source: ENST
Status: For consideration at the 62nd MPEG meeting
Title: Improvements for WD 1.0 on ATG
Author: Concolato Cyril, Dufourd Jean-Claude, Le Feuvre Jean

I Introduction

This document lists all the problems encountered while we were trying to implement or create content with the new features proposed in the WD 1.0 of ATG. We also raise some problems with the current BIFS specification that we faced when answering the Streaming Text Core Experiment. Since these problems are related to 2D graphics and Text rendering, we think they could be solved by the ATG activity. The problems or improvements are listed here with no particular order.

II CompositeTexture2D

II.1 Problem Statement

In order to get a well defined mapping of the texture's scene to the object a player has to know the size in pixels of the texture to use. Viewport provides this functionality, however Viewport is not mandatory in the node. Moreover, the node has 2 exposed fields, pixelWidth and pixelHeight, considered as optional. These fields "specify the ideal size in pixels of this map. The default values result in an undefined size being used. This is a hint for the content creator to define the quality of the texture mapping". This is however not correct. Modifying the pixelWidth and pixelHeight in the scene results in the same functionality as using an appropriate TextureTransform scaling.

Moreover when drawing a background in a composite texture, the size of the background (ie the size of the overall texture as created by the author) is needed, and pixelWidth/pixelHeight cannot be used for that purpose since they can be dynamically modified. From the implementation point of view, the allocation of an off-screen surface of a given size for the lifetime of the texture is also needed, otherwise a player may have to reallocate this surface on the fly or worse, ignore some part of the composite texture.

Lastly, a well-defined pixel size is absolutely needed to compute the texture world coordinates when using meter metrics; otherwise different player may draw the texture in different ways. Using the texture children's bounding rectangle is not a solution since these children may be animated, resulting in a variable bounding rect.

In order to clarify this and since MPEG is considering updating CompositeTexture2D to add the repeatS and repeatT fields, we suggest redefining the CompositeTexture2D node, or at worst changing its semantics to reflect these problems.

II.2 Node Rewrite

We suggest redefining the node given the lack of bit streams using this node and the lack of players for the two profiles which include the node.

CompositeTexture2D {			
eventIn	MFNode	**addChildren**	
eventIn	MFNode	**removeChildren**	
exposedField	MFNode	**children**	[]
field	SFInt32	**pixelWidth**	-1
field	SFInt32	**pixelHeight**	-1
field	SFBool	**repeatS**	TRUE
field	SFBool	**repeatT**	TRUE
}			

- pixelWidth, pixelHeight: indicates the size in pixel of the texture object. Dimensions shall be greater than zero.
- CompositeTexture2D holds a stack to bindable (background2D and Viewport) potentially used in its children field.

Note:

viewport and background fields are no longer present since they are not useful - the same functionalities can be achieved through regular mechanisms of bindable nodes. We even argue that the presence of the viewport and background fields in this node contradicts the original design of the VRML bindable nodes.

II.3 Proposed modification

In case WG11 doesn't feel appropriate to change the encoding of the node we suggest redefining the semantics as follows:

-PixelWidth, PixelHeight: indicates the size in pixel of the texture object. Value (-1, -1) is forbidden. Although these are exposedFields they shall not be dynamically modified since they represent the physical size of the off-screen surface used for compositing.

-The field repeatSandT of type SFInt32 shall be added to the CompositeTexture2D node. The meaning of this field will be that of the combined repeatS and repeatT of the ImageTexture node. The value 0 is equivalent to false, false. The value 1 is equivalent to repeatS = true, repeatT = false. . The value 2 is equivalent to repeatS = false, repeatT = true. The value 3 is equivalent to repeatS = true, repeatT = true.

III Layer2D and Background2D

III.1 Problematic

When using a background2D with picture inside a layer2D, a problem arises when changing the size of the layer2D: the background2D is currently forced to fill the parent surface. The current specification says:

"The top-left corner of the image is mapped to the top-left corner of the **Layer2D** and the right-bottom corner of the image is stretched to the right-bottom corner of the **Layer2D**, regardless of the current transformation. Scaling and/or rotation do not have any effect on this node. The background image will always exactly fill the entire **Layer2D**, regardless of **Layer2D** size, without tiling or cropping."

This is however not consistent: "fill the entire **Layer2D"** could be interpreted in 2 different ways: "fill the rectangle defined by layer2D.size since this is the only size info available for layer2D", or "fill the entire display area and clip to the rectangle defined by the layer2D.size" field, since it seems more logical and avoids changing the background aspect when the layer size changes. But unfortunately the spec forbids that when saying "The top-left corner of the image is mapped to the top-left corner of the **Layer2D** and the right-bottom corner of the image is stretched to the right-bottom corner of the **Layer2D"** points which can only be computed through the layer2D.size field.

III.2 Proposed corrections

We suggest to remove the problematic sentence and clarify as follows:

"Background2D is stretched to fit exactly the entire parent surface and is not subject to scaling and rotations. The parent surface is defined as follows:

- In case of CompositeTexture2D, it is the rectangle defined by the texture size in pixel.
- Otherwise it is the full display area.

Background2D is subject to clipping when inside Layer2D, the clipper being defined by the layer2D size field."

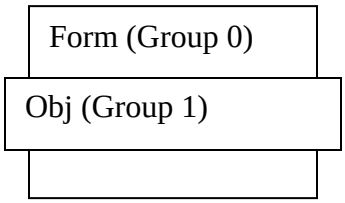
IV Form problems

IV.1 Problematic

Note 1: The Form specification uses an undefined word, "component". This word shall be replaced by "group".

Note 2: The children of the Form node shall all be centered before applying the constraints; this is was removed from the specification by error.

Assume we have the following case before constraints are applied:



Now let's apply the following constraint: "AR 0 1". According to the spec, the right-most point of all groups in the constraint shall be used as the right alignment origin of all groups in this constraint. However in this case one of the groups is the form node itself, which is not affected by alignment constraints. An author would however believe that writing "AR 0 1" would result in having Group 1 right-aligned with the form node. This is not the case and the object will actually not move, nor the form, thus this constraint will have no effect. We believe this is a serious flaw in the Form node and propose to change it.

IV.2 Proposed Changes

- Replace "component" by "group" in the Form specification
- Update the example with recent names (Shape instead of Shape2D… ☺)
- Add the following text before the alignment description:

"In case the form itself is specified in alignment constraint (group index 0), the form rectangle shall be used as the base of the alignment computation and other groups in the constraint list shall be aligned as specified by the constraint"

V XLineProperties

V.1 Problem Statement

Version 1.0 of the WD on ATG proposes the creation of a new node to perform better line rendering. One of the new features is the transparency field that allows to draw outlines of shapes with some level of transparency. The current WD says:

"The **transparency** field specifies the transparency of the outline of a Shape when drawn. It supercedes the value of the transparency of a material node."

We believe that it is under-specified because the outline of a shape is drawn partly on top of that shape and partly outside of it. Indeed, if the shape and the outline have two different transparency values the result would be a shape drawn with an outline that will have two colors as illustrated below:

```
Shape {
  geometry Rectangle { size 100 50 }
  appearance Appearance {
    material Material2D {
      emissiveColor 1 0 0
      filled TRUE
      transparency 0.5
      lineProps XLineProperties {
        lineColor 0 1 0
        width 10
        transparency 0.5
```

```
      }
    }
  }
}
```

V.2 Proposed Changes

We suggest to add a Boolean field to the XLineProperties node to allow the drawing of the outline within the shape and not centered on the real shape outline.
The new field would be:
Syntax: field SFBool **isCenterAligned** TRUE
Semantics: Specifies that the outline of the shape is aligned on the center of the line that is drawn. If set to false, the outline is aligned on the inside of the edge of the shape.

VI Holes in shapes

VI.2 Problem Statement

BIFS define a lot of 2D and 3D graphics primitives. Among the 3D graphics primitive, there is the Extrusion node which allows to create holes in 3D shapes. Unfortunately there is no such equivalent in 2D, though it is used in lots of proprietary formats (namely Flash) and standard ones (SVG). With the current specification, there are several ways to do simple holes. For example, one could use IndexedFaceSet node with self intersecting faces or one could transform a Shape that has a whole in it into 3 shapes: the shape without outline and 2 shapes for the inner outline and the outer outline.

VI.2 Proposed Changes

We suggest to define a new Node called Hole which is an SF2DNode as follows:
Syntax:

```
Hole {
        exposedField SFNode geometry NULL
}
```

Semantics:
The **Hole** node defines an geometry to perform a hole in the next child of the parent grouping node. The hole applies only to the next sibling, whether that sibling is a single Shape or a group of nodes.