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/M8931
October 2002, Shanghai, China

Source: ENST
Status: For consideration at the 62nd MPEG meeting
Title: Analysis of the streaming text requirements
Author: Cyril Concolato, Jean-Claude Dufourd, Jean Le Feuvre

Introduction

The goal of this document is to provide an analysis of the requirements of the Core Experiment on Streaming Text. It focuses on those coming from the 3GPP specification because they include those coming from the ITU-T J.123 specification. We will expose among all these requirements, those that are already fulfilled by BIFS and those that are not. In the latter case, we will explain what needs to be done to fulfill the remaining requirements using BIFS structures. Along with this document, some BIFS translations of the 3GPP test sequences are provided. Finally, we will explain how the Streaming Text activity could benefit from the definition of new nodes and/or new stream in terms of authoring and reusability of the content.

1. Analysis of the requirements

This contribution does not deal with the requirements concerning Unicode encoding, character support, font support and metrics. It does not either deal with language issues.

a. Text rendering position and composition

In the solution for Timed Text proposed by the 3GPP consortium, the text rendering, position and composition properties are not specified with the scene description (SMIL). Instead, there are set at the file format level. Theses properties are:

- definition of a text region: width, height, translation vector from the video region
- definition of the rendering order (layer) for the text region with respect to the video region
- definition of a text box within the text region: width, height, translation vector from the text region

This positioning mechanism can be easily emulated in BIFS using an **OrderedGroup** node to enable the layer property and the region can be emulated using **Layer2D** nodes. This is summarized in the BIFS text below:

```
OrderedGroup {
 order [VideoLayer TextLayer]
 children [
  DEF VideoRegion Shape {
   geometry Rectangle { size VideoWidth VideoHeight }
   appearance Appearance { texture MovieTexture { url "20" } }
  }
  Transform2D {
   translation -VideoWidth/2+TextTX+TextWidth/2 VideoHeight/2-TextTY-TextHeight/2
   children [
    DEF TextRegion Layer2D {
     size TextWidth TextHeight
     children [
      DEF TextBoxTransform Transform2D {
       translation -TextWidth/2+TextBoxTX+TextBoxWidth/2 TextHeight/2-TextBoxTY-
       TextBoxHeight/2
       children [
        DEF TextBoxLayer2D Layer2D {
         size TextBoxWidth TextBoxHeight
         children [
          …
         ]
        }
       ]
      }
     ]
    }
   ]
  }
 ]
}
```

b. Sample and Sample Description Format

The Timed Text proposal of the 3GPP consortium extensively uses the file format notion of *Samples* and *Sample Descriptions*. A sample contains the text to be displayed plus the timing information and its display properties. A sample description contains the static information needed to display the text contained in each sample. It is similar to an ESD in the MPEG-4 Framework. But in the ISO Media File Format, the sample description may change over time. Therefore, some display properties that one can consider as static information can actually change over time (e.g. background-color). Let's look into further details.

The *TextSampleEntry* sample description sets the following general properties of the text, which can be overridden by a *TextSample* sample:

- Scrolling (in, out, in/out)
- Scrolling directions (right-left, left-right, top-bottom, bottom-top)
- Karaoke continuous
- Horizontal and Vertical text justification
- Text direction (vertical or horizontal)
- Text box and background color
- Default text style (font-face, font-size, font-style, color)
- Font table

The *TextSample* sample contains the text to be displayed. The text is displayed using the default properties of the sample description corresponding to this sample. Some rendering properties can be overridden by some text modifiers. These modifiers have the possibility

to alter the properties on a character-per-character basis. For instance, a *TextStyle* modifier can change the color of the characters 10 to 20 of the sample text string.

When implementing the *TextSample*/*TextSampleEntry* concepts in BIFS, the main problems are the fact that: the sample description associated to a sample can change over time or be overridden, and the *TextModifier* can split the text string in several chunks of text that have different rendering properties. For these reason one way to implement it in BIFS is to merge the information of the *TextSampleEntry* and the one contained in the sample. Therefore all the text properties are included explicitly in the sample.

The algorithm to convert a couple *TextSample*/*TextSampleEntry* into BIFS could be the following:

- Determine the default properties for the *TextSample* text string from the associated *TextSampleEntry*
- Apply the *TextModifiers* and determine the largest chunks of characters that have the same rendering and animation properties (color, highlighting, karaoke properties …)
- Create a **Text** node for each chunk of characters and create the BIFS structure needed for the karaoke, blinking or scrolling animations (**TimeSensors**, **Interpolators**, **ROUTEs**), or for the linking property (**Anchor**).
- Use a **Form** node to handle the positioning of all the **Text** nodes.

A BIFS access unit corresponding to a text sample may be described by the following BIFS text:

```
AT CTS {
 REPLACE TextRegion.children[0] BY Transform2D {
  translation 0.0 0.0 #TextBox translation
  children [
   Shape { #TextBox Background
    geometry Rectangle {
     size 176 30 #TextBox size
    }
    appearance Appearance {
     material Material2D {
     emissiveColor 1.0 1.0 1.0 #TextBox background-color
      transparency 0.0 #TextBox background-transparency
      filled TRUE
     }
    }
   }
   DEF ScrollTransform Transform2D { #Scroll
    children [
     Form { #TextBox
      size 176 30 #TextBox size
      groups [ 1 -1 ]
      constraints [ "AT" "AL" ]
      groupsIndex [ 0 1 -1 0 1 -1 ]
      children [
       Shape {
        geometry Text {
         string [ "This is demonstration No1" ]
         fontStyle FontStyle {
          family "Serif"
          size 12
          style "PLAIN "
         }
        }
        appearance Appearance {
```

```
            material Material2D {
              emissiveColor 0.0 0.0 1.0
              transparency 0.0
              filled TRUE
            }
          }
        }
      ]
    }
  ]
 }
}
```

c. Text Modifiers

In this section we will analyze the text modifiers one-by-one and see how they can be mapped onto some BIFS structures. We will not go into much details but rather focus on the points which are not easily translatable.

- **Style modifier**

The 3GPP consortium's proposal contains a TextStyle modifier that can alter the style of the text in the following manner:

- Change the font
- Change the text color including transparency
- Change the font-size
- Change the font-face to one of the following value: plain, bold, italic, underlined

All these properties can be easily mapped onto the fields of a **FontStyle** node except for the underlined value of the font-face which does not exist neither in BIFS, nor in VRML.

- **Highlight and highlight color modifiers**

Two 3GPP text style modifiers consist in highlighting pieces of text by changing the text color and the background color surrounding the text. This is not yet feasible in BIFS for two reasons. First, the functionality per se is not available. Second, to simulate it in BIFS, one would have to put a rectangle, behind the text, filled with the highlight color. But, as mentioned many times during the discussions on Advanced Text and Graphics, it is currently impossible in BIFS to determine the exact position and size of a rendered text.

- **Karaoke modifier**

The concept of karaoke can be mapped onto BIFS structure using discrete interpolators to determine at each instant in time the color of each chunk of text and therefore of each **Text** node. Both continuous and discontinuous karaoke mechanism can be implemented in BIFS. But, since this modifier uses the functionality of text highlighting as described above, it suffers the same limitation.

- **Scrolling and Scroll delay modifier**

The scrolling property as per 3GPP differs a little bit from the scrolling property as per the **Layout** node. Indeed, the 3GPP spec envisages the scrolling in three modes: scroll in, where the text is at first not visible and scrolls to its default position; scroll out, where the

text is at first positioned at its default position and then scrolls until completely out of sight; and scroll in-out, where the latter modes are merged. The **Layout** node as currently defined does not allow this kind of behavior. If the children of the node are scrolling, they can scroll only following an in-out mode.

To map the scrolling mechanism of the 3GPP spec, one needs to implement in BIFS the whole mechanism using ROUTEs, TimeSensor and Interpolator. Therefore the scroll delay introduced by a *ScrollDelayModifier* could be easily implemented by setting the startTime of the TimeSensor controlling the animation to this delay. But, this is not that simple, one would indeed face the problem of determining when to stop the scrolling animation. This instant can be computed from the scrolling speed, the text position and the text length. As previously stated, the latter element is unknown with the current BIFS specification. So, the 3GPP scrolling mechanism cannot be efficiently done in BIFS.

- **HyperText modifier**

The hypertext modifier is quite simple to implement in BIFS. Once, a chunk of characters with the same linking property has been built, one has to generate an **Anchor** node that will contain the **Text** node. The *URLString* from the 3GGP spec exactly matches the **url** field of the **Anchor** node and the *altString* matches the **description** field.

- **TextBox modifier**

The 3GPP *TextBox* modifier allows to change the positioning of the text box within the text region. In BIFS, the result of a *TextBox* modifier would consist in a change of the **translation** field of the `TextBoxTransform` node as defined above.

- **Blinking text**

Blinking text is not a built-in feature of the current BIFS specification. However, it is possible to implement this feature using ROUTEs, ColorInterpolator and TimeSensors.

2. Synthesis on the existing BIFS structures

In the previous section we have seen that apart from the problems on text positioning, already raised in the Advanced Text and Graphics activity, the complete 3GPP specification on Timed Text could be implemented in a pure BIFS manner. Nevertheless, the current specification need to be improved in order to efficiently answer the Streaming Text requirements. Therefore, we propose to add the following items to the working draft on Advance Text and Graphics.

a. Add features to the FontStyle node

There are three features that could improve the efficiency of the proposed BIFS solution for streaming text. We will expose each of these features and propose some modification to the current BIFS specification.

- Highlighting

The requirements for the highlighting features could be restated in the following way:

- Being able to highlight a text, i.e. fill the bounding box of the text with a different color than the text color
- Support two modes of highlighting:
  - inverse video mode, i.e. the text color becomes the bounding box color and the background color becomes the text color
  - using a specified highlight color to fill the bounding box

To fulfill this requirement, we suggest to create a new node, called **Highlight**, as follows:

```
Highlight {
        exposedField SFBool useInversedVideoMode
        exposedField SFColor highlightColor
}
```

This node would be of type **HighlightNodeType** and be put in a new field of the FontStyle node, called **highlight**.

- Blinking and Underlining

The proposed requirement is the following:
It should be possible to have underlined text and blinking text.

We suggest to add two more values for the **style** attributes of the **FontStyle** node: "BLINK" and "UNDERLINE". We do not propose to set a blinking rate.

b. Selecting the scrolling mode

As previously stated, BIFS misses the ability to perform scrolling in or out of **Text** elements. The current specification, quoted below, shows what is currently possible with the **Layout** node:

" The **scrollRate** field specifies the scroll rate in meters per second. When **scrollRate** is zero, then there is no scrolling and the remaining scroll-related fields are ignored.
The **smoothScroll** field selects between smooth and line-by-line/character-by-character scrolling of children. When TRUE, smooth scroll is applied."

We suggest to add a new SFInt32 field to the **Layout** node, called **scrollMode**, which could take the value –1 for scroll-in mode, +1 for scroll-out and 0 for the scroll in-out and the default behaviour would be scroll in-out as it is currently the case. Finally, we suggest to add the **scrollDelay** field even if the delay can be done using routes to set the **scrollRate** field after some time.

3. Notes on complexity

In this document, we have shown that provided some small changes, all the rendering features of the 3GG specification for Timed Text can be mapped to BIFS structures. The proposed solution does not claim to be the only BIFS solution to fulfill the requirements. However, some remarks can be made on the complexity of any BIFS solution by analyzing the proposed one.

a. Compression

The sizes of the text streams which are dealt with currently in 3GPP Timed Text are around 1kb. Therefore compression seems to be irrelevant.

b. Decoding and rendering complexity

Whatever BIFS solution is proposed, the positioning and the rendering of the text is rather complicated task. It needs to be done using the **Layout** or the **Form** node. Indeed, since there is no way to determine the size and the positioning of the characters irrespectively of the terminal, a BIFS solution has to rely on implicit positioning of text elements. But, the only two nodes to do that in BIFS are **Form** and **Layout**. These nodes are quite complex to implement and require high profiles (*Complete2D*) and levels. Moreover, such simple content is targeted for simple devices like mobile devices and/or PDAs which have limited capabilities.

c. Content creation and reusability

The drawback of a 100% BIFS solution is in term of content creation and/or reusability. Indeed, contrary to the 3GPP solution, the proposed solution merges the general scene description (video, audio, graphics …) with the text rendering description. Therefore, to add, replace or remove the text content from some existing scenes, one would have to analyze the BIFS content, determine if possible the part of the scene related to an existing text content and remove or replace it. This is quite painful and may sometimes be impossible because of the complexity of the scene. On the opposite, the file format solution proposed by the 3GPP consortium offers an easiness of authoring and reusability. Indeed, adding, removing or replacing the text content is just an operation on file tracks.

d. Recommendations

Based on the previous analysis of a 100% BIFS solution and on the above notes on complexity, we recommend that a solution that uses a separate stream/track be preferred. We suggest that this stream be interfaced with the scene via a new node, which would do all the positioning and rendering of text. This latter text should be in the stream along with its display properties. In particular, it should be possible to specify, as per the 3GPP spec, some text rendering properties valid for runs of characters.

One way to explore could be to have an AnimationStream node pointing to a new stream type where the DecoderConfigDescriptor would do the static positioning of the text as well as the handling of default properties (color, font …) and the stream itself would contain dynamic text and text properties. These latter information could be encoded in BIFS or in some other format. The advantage of using the AnimationStream node would be the ability to seek in the text stream from the scene using MediaControl.