Interactive Authoring Tool for Extensible MPEG-4 Textual Format (XMT)

Kyungae Cha[1] and **Sangwook Kim**[2]

Abstract. MPEG-4 is an ISO/IEC standard which defines a multimedia system for communicating interactive scenes containing various types of media objects. The Extensible MPEG-4 Textual format (XMT) framework provides interoperability between existing practices such as the Extensible 3D (X3D) and MPEG-4. This paper introduces an XMT authoring tool that supports a visual environment for building a spatio-temporal scenario of media objects comprising a multimedia scene. The authoring tool provides a comprehensive set of facilitative editing tools for composing multimedia scene, as well as tools for automatic generation of XMT documents and MPEG-4 contents. This paper also describes the functionality of the developed system and shows an example of its use.

1 INTRODUCTION

MPEG-4, one of the leading streaming media formats, is an ISO/IEC standard which defines a multimedia system for communicating interactive scenes with various types of media objects. In MPEG-4, a scene is accompanied with the description specifying how the objects should be combined in time and space in order to form the scene intended by the author.

The scene description is coded in a binary format called Binary Format for Scenes or BIFS[1,4,7,8,10,11], which is built on several concepts from the Virtual Reality Modeling Language(VRML)[5]. This binary form is suitable for low-overhead transmission so that BIFS basically provides an efficient application for the sender and the receiver[1,7].

On the other hand, the Extensible MPEG-4 Textual format (XMT) is a framework for representing MPEG-4 scene description using a textual syntax.

This paper presents an XMT document authoring tool that enables visual composition of an MPEG-4 scene and generates the corresponding XMT document and MEPG-4 contents. XMT is designed to provide a high-level abstraction for MPEG-4 functionalities and an easy interoperability between existing practices of content authors, such as the Extensible 3D (X3D) being developed by the Web3D Consortium and the Synchronized Multimedia Integration Language (SMIL) from the W3C consortium[7,11]. Thus authors can get multimedia contents, which are exchangeable and interoperable with X3D and SMIL, using the XMT authoring tool.

In the authoring system, authors can visually make a spatial arrangement of media objects and compose a temporal behavior of objects with timeline approach. Authors can also modify the material characteristics of each object using interactive and visual tools. Moreover, the visual scene is automatically transformed into an XMT-α and XMT-Ω format document.

In section 2, XMT formats are briefly discussed. In section 3, the various functions of the XMT authoring tool are described. The implementation of the proposed system is then presented in section 4. Finally section 5 gives conclusion and presents our future plans.

2 XMT-A AND XMT-Ω FORMATS

The XMT framework consists of two levels of textual syntax and semantics: XMT-α and XMT-Ω formats[7,10].

XMT-α is an XML-based version of MPEG-4 content which provides a straightforward, one-to-one mapping between the textual and the binary formats of an MPEG-4 scene description. XMT-α also provides interoperability with X3D[5], which improves upon VRML with new features such as flexible XML encoding and a modularization approach[6]. It contains a subset of the X3D as well as the X3D-like representations of MPEG-4 features such as Object Descriptors(OD), BIFS update commands and 2D composition[7].

XMT-**Ω** is a high-level abstraction of MPEG-4 features based on the SMIL[9]. It specifies objects and their relationships in terms of the author’s intention rather than coded nodes and route mechanism in BIFS. In the respect of reusing SMIL, XMT-**Ω** defines a subset of modules used in SMIL whose semantics are compatible. Moreover XMT-**Ω** format can be parsed and played directly by a W3C SMIL player, preprocessed to the corresponding X3D nodes and played by a VRML player. It may also be compiled to an MPEG-4 representation such as mp4 which can then be played by an MPEG-4 player. Figure 1 shows the interoperability of XMT between SMIL player, VRML player and MPEG-4 player.

[1] Department of Computer Science, Kyungpook National University, Daegu, Korea, email : chaka@woorisol.knu.ac.kr

[2] Department of Computer Science, Kyungpook National University, Daegu, Korea, email : swkim@cs.knu.ac.kr

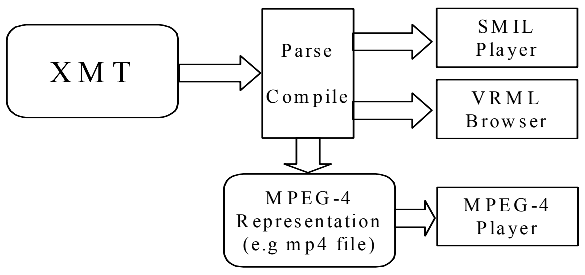

Figure 1. The interoperability of XMT

3 XMT AUTHORING SYSTEM

This section shows the XMT document authoring environment of our system and the authoring process in creating an MPEG-4 scene and an XMT document. The main functionalities of the system are also described.

3.1 System Structure

The following figure shows the system structure and every component of the XMT authoring tool.

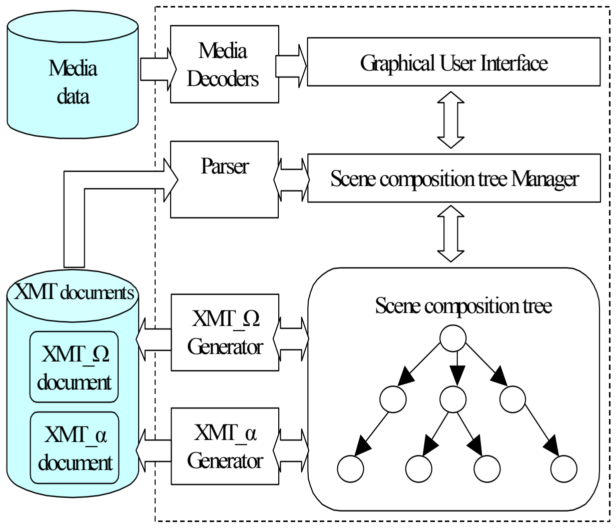

Figure 2. System Structure

Authors compose an MPEG-4 scene with various editing tools provided in the graphical user interface. Following the authoring process, the scene composition tree, which represents the visual scene as internal data structure, is built and modified.

Using the scene composition tree, the XMT-α or XMT-**Ω** generator makes a corresponding XMT format document. At this time the author can choose the output format that he/she wants. The XMT format files can be parsed and then displayed in the user interface as a visual scene. The author can also modify the visual scene and recreate the XMT file.

3.2 Graphical User Interface

The graphical user interface provides a set of drawing tools and editing tools for various media types such as JPEG image, MPEG-1 video, G.723 audio and graphical objects (Rectangle, Circle, and others). These tools enable authors to compose audio-visual scenes with direct manipulation technique and see them immediately. Figure 3 presents an overview of the graphical user interface and a simple example of a scene.

Authors first select from the toolbar one of the tools they want to add in the scene and then draw the selected object. For image objects, the object is drawn in the interface window. For video objects, the first frame of the video is drawn in the interface window.

Whenever a new media object is added in the scene, the system automatically assigns the object ID, start time and end time of the object with default value. The bottom portion of figure 3 shows the timeline window where the timelines of objects are arranged. The layer of timeline represents the drawing order of corresponding objects, which is determined following the object addition sequence. Here the timeline window shows the initial state, i.e. no modification is occurred.

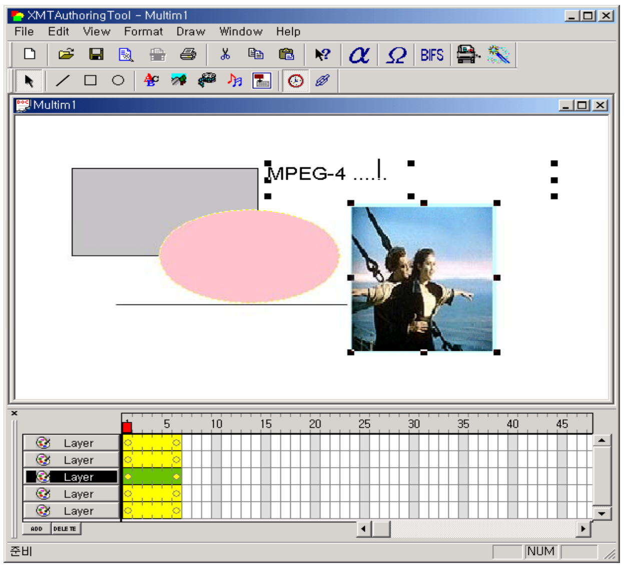

Figure 3. Graphical user interface

3.2.1 Spatial composition

In the user interface, each object participated in a scene is contained in a rectangular tracker so that they are treated as individual objects. Thus the author can move, resize or remove the objects directly for composing a spatial arrangement of the scene.

The spatial attributes of an object can be specified in terms of the spatial position of the object's bounding rectangle, which is represented as a rectangular tracker containing the object in the user interface. The spatial position of bounding rectangle of an object (i.e. the spatial attribute of the object) is specified as the form of (x,y,h,w), where w denotes the width of the bounding rectangle; h denotes the height, while x and y denote the coordinates of the center of the rectangle with referring to the center of whole rectangle of the presentation as origin of coordinate system.

The author can also apply material characteristics such as color, transparency, and border type using editing tools. These material properties of an object are specified as object property node in the internal form of our authoring system. The spatial and material attributes of each object are automatically specified by the system from the visual scene.

3.2.2 Interactive scenario composition

In the presentation of an MPEG-4 scene, user interaction is possible within the set in the scene description. Assume that the author designs the following scenario for the scene in figure 3.

Example 1. If an end user clicks the circle object, the fill color of the rectangle object will be changed through the gradient from red to green.

Here, the circle object and the rectangle object refer to the source object and the destination object respectively. To make an interactive scenario, the event type(e.g. user's click), the source and destination object and the responding action type(e.g. change fill color), etc., should be specified. We denote the interactive information as event object which is represented as a quadruple (*destination object ID, event type, action type, key values*). The key values mean an array of values to be used to change the parameters of the action type field. The event object for the above example is specified as (*3000, click, fill color, ((1.00 0.00 0.00),(0.00 0.50 0.00)*), if the rectangle object as the destination has the number 3000 for its object ID.

We provide a dialog based interface in order to facilitate the interactive scenario authoring process. The event object specification is done by selecting an event type and attributes of the destination object that the author wants the event type to change, without the need for an extra description.

3.2.3 Temporal scenario composition

For composing temporal scenario of objects, the author can modify the timeline of each object, i.e., the author directly modifies the length and position of timelines in the timeline window. Moreover the author can declare the temporal relationships among objects, which are maintained through the authoring process. Consider the following scenario for the scene in figure 3.

Example 2. The text object is rendered at end of the image object.

The scenario can be specified if the author modifies the timelines of the two objects like figure 4 and he/she declares the two objects as a sequence group which maintains the objects play sequentially.

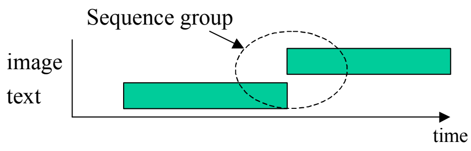

Figure 4. An example of timeline modification and temporal relationship declaration

The timeline of the image object is automatically updated to maintain the relationship each time the duration of the text object is modified.

3.3 Scene Composition Tree

The resulting graphical user interface is represented as a scene composition tree designed to organize the composed scene into a hierarchical structural form. Whenever a new object is created in the user interface, the corresponding object node is also created.

The object node has its corresponding object type, object ID and values specifying spatio-temporal attributes. The scene composition tree is modified through the attachment of the new object node. At the same time, the property node of the object is attached as a child node of the new object node. The property node as well as the tree structure can be changed throughout the authoring process. The tree structure can be changed while objects are added, replaced, or removed. If the author creates event information, an event object which contains destination object ID, event type and values of transition status is created and attached to the source object node as its child node. Thus an event object does not specify its source object ID.

3.4 Generation of XMT Document

The resulting graphical user interface is represented as the scene composition tree. From the scene composition tree, both of the XMT-α and XMT-Ω document corresponding to the visual scene are directly generated.

3.4.1 XMT-α generation

In XMT-α format, each object is represented as an element similar to the object node described in BIFS. Thus, the XMT-α format document can be generated following the BIFS generation rules.

The XMT-α generator searches the scene composition tree until it meets the audio and visual object node. It then creates the corresponding object element of the XMT-α document using spatio-temporal attributes of the object node. With the value specified in the object's property node in the scene composition tree, the XMT-α generator can describe geometric attributes such as position, size and shape of the object or material attributes such as fill color and border style.

Figure 5 and Figure 6 show a portion of XMT-α and BIFS text for the scene of example 1 respectively. In this case, when the XMT-α generator finds the circle object node in the scene composition tree, it also meets the circle object's property node as well as its event node at the object node's child. Using the information written in the event node, the route and sensor nodes can be described.

3.4.2 XMT-Ω generation

XMT-Ω syntax and semantics have been designed using extensible media (xMedia) objects as basic building blocks[7].

The elements within XMT-Ω abstract the geometry and the behavior of the corresponding object in the visual scene. Thus, if an object is associated with an event object node, its behavior should be defined by a set of animation and timing element.

Figure 7 shows the XMT-Ω format document corresponding the XMT-α format in figure 5. The rectangle object is defined with the elements describing the object's spatial and material attributes as well as the animate elements describing a change of fill color which responds to a click on the circle object.

Likewise figure 8 shows a portion of XMT-Ω document specifying the scenario of example 2. It represents a temporal relationship and synchronization module expression using SMIL timing constraints. A 'seq' container defines a sequence of elements in which elements play one after the other. The text object starts one second after the presentation begins and 19 seconds later disappears. When the text object disappears, the image object whose temporal duration is 23 seconds starts. Figure 9 represents the BIFS text corresponding XMT-Ω in figure 8.

```
<Transform2D DEF="Transform2D3000"
      translation="-163.00 17.00" scale="1.00 1.00"
      rotationAngle="0.00" >
    <children>
      <Shape>
      <Appearance>
        <Material2D DEF="Material2D3000"
              emissiveColor="0.75 0.75 0.75" ...>
        <LineProperties DEF="LineProperties3000"
              ...
        </Material2D>
      </Appearance>
      <Rectangle DEF="Rectangle3000" USE="Group0"
                  size="162.00 110.00">
    </Rectangle>
    </Shape>
    <TimeSensor DEF="TimeSI3000I0" cycleInterval="3.00"
            enabled="FALSE"  loop="TRUE"
            startTime="0.00" stopTime="-1.00" >
    </TimeSensor>
    <ColorInterpolator DEF="ColorInter3000I0"
                key="0.00 1.00"
          keyValue="0.00 0.50 0.00 1.00 0.00 0.00 " >
    </ColorInterpolator>
    ...
    <Circle DEF="Circle3001"  USE="Group0"
          radius="57.00">
        ...
    <TouchSensor DEF="TouchS3001" enabled="TRUE" >
    </TouchSensor>
                  ...
<Route fromNode="TouchS3001" fromField="isActive"
            toNode="TimeSI3000I0" toField="enabled" />
<Route fromNode="TimeSI3000I0"
         fromField="fraction_changed"
         toNode="ColorInter3000I0" toField="set_fraction" />
<Route fromNode="ColorInter3000I0"
            fromField="value_changed"
            toNode="Material2D3000"
            toField="emissiveColor" />
```

Figure 5. A portion of XMT-α for the example 1

```
DEF Transform2D3000 Transform2D {
      translation -163.00 17.00
      scale 1.00 1.00
      rotationAngle 0.00
      children [
        Shape {
         appearance Appearance {
        material DEF Material2D3000 Material2D {
                emissiveColor 0.75 0.75 0.75
                filled TRUE
```

```
                    transparency -1.00
                        ...
                    geometry Rectangle {
                            size 162.00 110.00   }
            DEF TimeSI3000I0 TimeSensor {
                            cycleInterval 3.00
                            enabled FALSE
                            loop TRUE
                            startTime 0.00
                            stopTime -1.00           }
            DEF ColorInter3000I0 ColorInterpolator {
                            key [
                               0.00
                                1.00  ]
                            keyValue [
                                0.00 0.50 0.00
                                1.00 0.00 0.00    ]
                             ...
            geometry Circle {  radius 57.00  }
            DEF TouchS3001 TouchSensor {
                            enabled TRUE         }
                                                ...
ROUTE TouchS3001.isActive TO TimeSI3000I0.enabled
ROUTE TimeSI3000I0.fraction_changed TO
                          ColorInter3000I0.set_fraction
ROUTE ColorInter3000I0.value_changed TO
                      Material2D3000.emissiveColor
```

Figure 6. A portion of BIFS text corresponding the XMT-α in figure 5

```
<rectangle ID="rectangle_3000" parent="group_0"
            size="162.00 110.00">
<transformation ID="transformation_3000" translation="-163
            17" scale="1.00 1.00"></transformation>
<material ID="material_3000" color="#c0c0c0"
            transparency="-1.00" filled="true" >
<animateColor attributeName="color"
        dur="1s" begin="circle_3001.click"
      values="#000000; #010000" keyTimes="0.00; 1.0" />
</material>
</rectangle>
<circle ID="circle_3001" parent="group_0" radius="57.00">
<transformation ID="transformation_3001"
     ...
<material ID="material_3001" color="#ffd700"
```

Figure 7. A portion of XMT-Ω corresponding XMT-α in figure 5

```
<seq begin="1s" >
  <string ID="string_3002" parent="group_0"
            textLines="MPEG-4 ......"  dur="19s">
   <fontStyle ID="fontStyle_3002" family="Arial"
                 horizontal="true" justify="BEGIN"
                 language="(null)" leftToRight="true"
                size="-21.00"   spacing="34.00"   style="PLAIN"
                topToBottom="true">
      </fontStyle>
    <transformation ID="transformation_3002"
                   translation="100 91" scale="1.00 1.00">
    </transformation>
    <material ID="material_3002" color="#ffff00"
                  transparency="-1.00" filled="true" >
    </material>
  </string>
```

```
   <image ID="image_1000" parent="group_0"
                         src="D:\sample_image.gif" dur="23s">
   <transformation ID="transformation_1000"
               translation="113 -25" scale="1.00 1.00">…
</seq>
```

Figure 8. A portion of XMT-Ω for the example 2

```
DEF Switch3002 Switch {
      whichChoice 1
           choice [
                 DEF Transform2D3002 Transform2D {
                          . . .
                     Shape {
                       appearance Appearance {
                          material DEF Material2D3002 Material2D
{
                                      emissiveColor 1.00 1.00 0.00
                                      filled TRUE
                                      transparency -1.00
                          . . .
             geometry Text { string [ "MPEG-4 ......"]
                          fontStyle DEF FontStyle3002
                                   FontStyle {
                                      family "Arial "
                                      horizontal TRUE
                                      justify "BEGIN"
                                      language "(null)"
                                      leftToRight TRUE
                                      size -21.00
                                      spacing 34.00
                                      style "PLAIN"
                                      topToBottom TRUE . . .
DEF Switch1000 Switch {
         whichChoice 1
             choice [
                  DEF Transform2D1000 Transform2D {
                          . . .
                          appearance Appearance {
                                       texture ImageTexture {
                                                 url 1
                                                 repeatS TRUE
                                                 repeatT TRUE
             }                                   geometry Bitmap {
                          . . .
AT 1000 { REPLACE Switch3002.whichChoice BY 0 }
AT 20000 { REPLACE Switch3002.whichChoice BY 1 }
AT 20000 { REPLACE Switch1000.whichChoice BY 0 }
AT 43000 { REPLACE Switch1000.whichChoice BY 1 }
```

Figure 9. A portion of BIFS text corresponding XMT-Ω in figure 8

All the XMT and BIFS text which are shown the above, are generated automatically from the visual scene.

3.5 XMT Parsing

The XMT framework is based on XML, thus valid XMT element nesting can be defined in the Document Type Declaration (DTD) and parsed using XML parser. XML4C[3] is used as a validating XML parser written in a portable subset of C++ for parsing XMT documents.

A valid XMT document can be transformed as a form of scene composition tree using DOM API [2]. DOM API provides a tree-based API to compile an XML document into an internal tree structure and navigate the tree.

Media elements described within the parsed XMT document are represented as object nodes with their corresponding property nodes. Thus the scene described in the XMT document can be visualized by rendering the corresponding media object nodes using the scene composition tree. The visualized scene can also be modified and rewritten as XMT document.

4 IMPLEMENTATION

The proposed XMT authoring tool is developed using C++ under the Windows 95/98/NT platform. The system supports the Complete2D profile for MPEG-4 contents.

5 CONCLUSION

The XMT document authoring tool provides visual and direct manipulating authoring technique. In the system, common users can create an MPEG-4 scene and its XMT format document although they are not familiar with XMT syntax and semantics.

Moreover, the visual scene is automatically transformed into XMT-α or XMT-Ω document without syntax error. Likewise, a sophisticated scene, which may be very difficult to create using text description, can be generated. In the future, it is necessary to support more types of media data and scene nodes such as 3D objects and a more facilitative authoring interface.

REFERENCES

[1] A. Puri and A. Eleftheriadis, “MPEG-4: An Object-Based Multimedia Coding Standard Supporting Mobile Applications,” Mobile Networks and Applications, vol. 3, pp. 5–32, 1998.
[2] Document Object Model (DOM) Level 1 Specification, W3C Recommendation, October, 1998. http://www.w3.org/TR/REC-DOM-Level-1/
[3] http://www.alphaworks.ibm.com/tech/xml4c/
[4] ISO/IEC 14496-1:1999 Information technology - Coding of audio-visual objects - Part 1: Systems ISO/IEC JTC1/SC29/WG11 N2501, 1999.
[5] ISO/ICE FDIS 14772:200x, Information Technology-Computer graphics and image processing--The Virtual Reality Modeling Language (VRML)
[6] ISO/IEC xxxxx:200x, X3D, Information technology -- Computer graphics and image processing -- X3D.
[7] M. Kim, S. Wood, L.T. Cheok, “Extensible MPEG-4 textual format (XMT),” in Proc. on ACM multimedia 2000 workshops, Los Angeles, California, United States, 2000, pp. 71–74.
[8] S. Battista, F. Casalino and C. Lande, “MPEG-4: A Multimedia Standard for the Third Millennium, Part 1,” IEEE Multimedia, vol. 6, no. 4, pp.74–83, 1999.
[9] Synchronized Multimedia Integration Language (SMIL) 1.0 Specification, W3C Recommendation, June, 1998. http://www.w3.org/TR/1998/REC-smil-19980615
[10]WG11(MPEG), MPEG-4 Overview (V.16 La Baule Version) document, ISO/IEC JTC1/SC29/WG11 N3747, October 2000.
[11]WG11(MPEG), MPEG-4 Overview (V.18 Singapore Version) document, ISO/IEC JTC1/SC29/WG11 N4030, March 2001.