USING MPEG-4 SCENE DESCRIPTION FOR OFFERING CUSTOMIZABLE AND INTERACTIVE MULTIMEDIA PRESENTATIONS

Wesley De Neve, Koen De Wolf, Davy De Schrijver, and Rik Van de Walle

Ghent University - IBBT, Department of Electronics and Information Systems, Multimedia Lab,
Sint-Pietersnieuwstraat 41, B-9000, Ghent, Belgium.
Email: {wesley.deneve, koen.dewolf, davy.deschrijver, rik.vandewalle}@ugent.be

ABSTRACT

MPEG-4 (Coding of Audiovisual Objects) is a collection of standards that enables the creation of rich media. The latter are usually defined as multimedia content with local and client-server based interactivity, natural audio and video mixed with their synthetic variants, 2D and 3D graphics, scripting, … In this paper, we will discuss how technology, as available in the MPEG-4 toolkit, can be used for the creation of advanced and interactive multimedia presentations. We hereby pay attention to their deployment in the current ecosystem of heterogeneous networks and devices. We also present some ideas on how to customize some of the scenes to the preferences of the end-user.

1. INTRODUCTION

Until recently, a multimedia presentation could be considered as a temporal and spatial composition of audiovisual elements. Video, audio, still images, and text are typical examples of such objects. Nowadays, multimedia presentations are no longer limited to the pure presentation of those different media elements. Streaming and interactivity have made their way in the world of content authoring. Interactivity was certainly a point of interest in the standardization of several multimedia description languages such as the Virtual Reality Modeling Language (VRML) and the Scalable Vector Graphics specification (SVG) [1].

The outline of this paper is as follows: Section 2 presents a short introduction to MPEG-4 Systems, followed by a discussion of the internals of some advanced interactive multimedia presentations in Section 3. Finally, Section 4 provides us with a conclusion.

2. MPEG-4 SYSTEMS

The term Systems in MPEG-1 and MPEG-2 refers in essence to the multiplexing and synchronization of video and audio streams. Among other tools, MPEG-4 Systems extends this with the definition of a full-fledged file format and advanced scene description. The latter actually consists of a three-tier architecture: the actual scene description framework (defining the temporal and spatial layout of a scene), the object descriptor framework (describing the characteristics of the audiovisual objects that are referenced in the scene description), and the elementary stream descriptor framework (describing the properties of the elementary streams that are part of an audiovisual object, such as a scalable video object).

Scene, object, and elementary stream description can be realized by making use of the eXtensible MPEG-4 Textual format (XMT). XMT is in fact a collective term for two multimedia description languages: XMT-A and XMT-Ω. XMT-A is a low-level scene description language while XMT-Ω allows a more high-level description of a scene. Descriptions in both languages are typically compiled to the BInary Format for Scenes (BIFS). The latter is then wrapped in an MP4 file container, often together with the audiovisual objects that are used in the scene in question.

Because it is impossible to discuss all aspects of MPEG-4 Systems in a few paragraphs, we would like to refer the reader to [2] and [3] for a more in-depth overview.

3. MPEG-4 SCENE DESCRIPTION IN ACTION

This section explains the internals of three advanced and interactive MPEG-4 scenes. For each demo, we provide a use case scenario and a discussion of some relevant implementation aspects. All multimedia presentations are described by making use of BIFSText, an alternative for XMT-A using the VRML syntax. BIFSText is fully supported by the tools as available in the GPAC suite (GPAC Project on Advanced Content) [4]. This package also contains a sophisticated MPEG-4 terminal, in particular the Osmo media player. Streaming was done by relying on Apple Computer's Darwin Streaming Server.

3.1. Video-on-demand: a trailer mosaic

3.1.1. Use case scenario

This application, inspired by the PCTV project of the Belgian internet provider Telenet and by the trailer page of Apple Computer, provides a consumer with the possibility to choose between different movie trailers. Before starting the actual playback of the trailer, the user is supposed to make an additional choice between a low and high quality version of the movie selected. The two different versions of each trailer are stored as hinted MP4 files in the movie repository of the Darwin Streaming Server (server-based simulstore). On top of that, the consumer also has the possibility to navigate to a website containing more information about the movie in question. This can be done by clicking on a hyperlink that is integrated in the MPEG-4 scene. The end-user can also return to the main screen in case he/she wants to watch another movie trailer.

3.1.2. Implementation aspects

In the main scene, the user has to pick out a movie trailer by clicking on one of the preview images. The images in question are in fact textured buttons, defined by a prototype. A prototype allows the definition of a new type of node, having its own fields and corresponding behavior. The collection of fields in the `ImageButton` prototype definition looks as follows:

```
PROTO ImageButton[
 exposedField SFVec2F size 140 106 # button size
 exposedField MFString textureURL [] # texture
 eventOut SFBool isOver # mouse over
 eventOut SFBool isActive # click
]
```

A similar template was used for the definition of the hyperlink that enables an end-user to navigate to a relevant website.

With respect to the layout of the different objects, one could realize this by making use of the proper translations. However, this is cumbersome because it requires knowledge about all dimensions of the objects in the scene. Therefore, we have made use of the functionality of the `Form` node in order to take care of the layout of our scene. This node makes it possible to position its children in an automatic way, hereby considering some given constraints such as the alignment of its children.

Browsing through the different screens (trailer selection, version selection, and playback) is implemented by relying on a `Switch` node. Embedding an MPEG-4 trailer in the scene is realized by making use of an `Inline` node. The latter is referencing an `Object Descriptor` (`OD`) containing an RTSP URL to the appropriate movie trailer, as depicted in Figure 1.

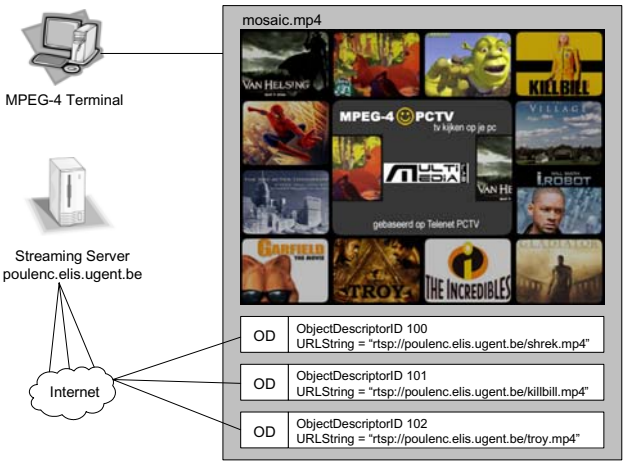

Figure 1: A trailer mosaic

In this implementation, we have also made use of the `Script` node. This node not only provides a convenient way to execute complex computations, it also offers direct access to the fields of the nodes used in the scene description. Furthermore, the `Script` node makes it possible to keep track of the current state of the presentation. For those reasons, it simplifies many tasks and it keeps the scene description well organized (for instance, by avoiding the declaration of too many `Routes`). With respect to our demo, the `Script` node manages tables containing URLs for the different images, websites, and trailer versions used. The node in question is also aware of the current active movie trailer.

3.2. A media player offering DVD functionality

3.2.1. Use case scenario

In the demo as just mentioned, the user is supposed to choose between two different versions of the same trailer. In this demo, we not only implement the characteristics of a basic media player (play, pause, stop, and repeat), but we also extend a part of the functionality of the first demo. To be more specific, the consumer now has the possibility to choose between four versions of the same video content: the original version of the trailer is available, as well as versions having a lower frame rate (temporal scalability), a lower resolution (spatial scalability), and a lower visual quality (SNR scalability). Furthermore, it is also possible to change the quality of the audio: one can choose for high quality, low quality, or no audio at all. Similar to the language and chapter selection features of a DVD, it is also possible to switch between subtitles in different languages and to jump to a particular scene. Note that all media data are now locally stored in an MP4 container (client-based simulstore). This could be useful for situations in which bandwidth and storage are not an issue, but battery life is.

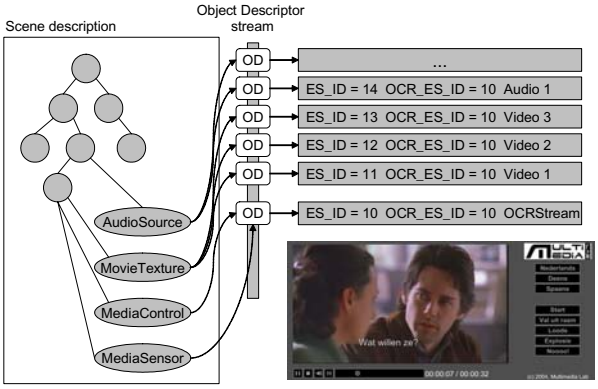

Figure 2: A media player offering DVD functionality

3.2.2. Implementation aspects

Figure 2 shows the interface of the media player. The video is positioned at the left. At the bottom, we can find the playback controls, the buttons that make it possible to switch between the different versions of the same video and audio content, and a status bar. The language and scene selection controls can be found at the right (from top to bottom respectively). Similar to the first demo, the positioning of the different audiovisual objects is done by making use of the `Form` node.

The most important challenge in this demo is the synchronization of all media streams. It is easy to see that the video, audio and subtitle streams have to be in sync all the time. This requirement especially holds true when the user selects an alternate video, audio, or subtitle version.

Enforcing synchronization between a video and an audio stream is typically done by making the audio stream dependent on the video stream. In that way, both elementary streams share the same clock, i.e. the clock of the video stream. As such, we could enforce synchronization by making all streams dependent on one elementary stream. However, it is important to know that this particular elementary stream has to be active during the entire presentation. This is not necessarily the case since the user may switch between different versions of the video, audio, and subtitle streams all the time.

In order to deal with this problem, it is possible to make use of an `Object Clock Reference Stream` (`OCR`). This elementary stream actually contains clock values instead of media samples. An `OCRStream` is not linked to an audiovisual object but it is possible for the other elementary streams to be dependent on this stream. As such, they all share the same clock. The latter is illustrated in Figure 2. It is only required that the duration of the `OCRStream` has to be the same as the duration of all other streams in order to achieve an appropriate synchronization, irrespective of the video, audio, or subtitle stream selected.

Figure 3: A shot browser

For controlling the playback of the presentation, a `MediaControl` and a `MediaSensor` node were used. The `URL` field in these nodes refers to the just mentioned `OCRStream`.

Subtitling is done by making use of an animation stream. Such a stream is created for every supported language. The different subtitle streams are also linked to the `OCRStream` in order to achieve synchronization with the active video and audio stream. An animation stream for subtitling actually consists of a series of timed commands. For example, the BIFSText syntax of an instruction, stored as a random access sample in the stream with ID 4 and used for updating a subtitle after 5003 clock ticks, looks as follows:

```
RAP AT 5003 IN 4 {
  REPLACE TXT.string BY ["What do they want?"]
}
```

The `Conditional` node is used for the conditional execution of timed commands. The latter commands have to be placed in the `buffer` field of this node. Changing the value of the `activate` field to `true` or changing the value of the `reverseActivate` field to `false`, results in the execution of the buffered commands. By relying on this behavior, it is possible to realize the actual switching between the different streams. The following example illustrates the usage. When a user clicks on the button for changing the current version of the video stream, this event is detected and forwarded by a `Route` to the appropriate `Conditional` node. The latter then executes the instructions in its buffer. This boils down to updating the value of the `URL` field of the `MovieTexture` node such that it contains a reference to another `Object Descriptor`.

Switching between the different scenes or chapters is realized in a similar way. The major difference is the fact that the `mediaStartTime` field of the `MediaControl` node has to be updated with the start time of the scene in question.

3.3. An automated shot browser

3.3.1. Use case scenario

As just mentioned, the second demo makes it possible to jump to a certain scene during the playback of the presentation. Hereby, the scenes in question not only have to be detected in a manual way, but they also have to be manually included in the scene description. In this section, we will discuss an application that makes it possible to generate an MPEG-4 presentation in an automatic way, hereby allowing the user to browse through the different shots that make up the corresponding video stream.

3.3.2. Implementation aspects

For detecting the shots, two different algorithms were implemented: one based on color histograms and one based on Sobel edge detection. Both algorithms are part of a DirectShow transform filter that is able to log the temporal segmentation information in plain text, as well as in an XML document that is conforming to the MPEG-7 specification. With respect to the plain text description, a small Java application was developed that is able to transform the segmentation information into an MPEG-4 scene browser. The latter is depicted in Figure 3.

The scene browser in question consists of a selection menu that is activated when the user clicks on the video. This menu consists of several buttons for controlling the playback of the movie. In addition to these buttons, there is also the possibility to visualize the type of transition between two consecutive shots (cut, fade in, and fade out). We also have implemented the possibility to browse through the shots and to navigate to a selected shot.

For the implementation of the selection menu, the `Layout` node was used. This grouping node makes it possible to arrange its children automatically in the available space. Several parameters can influence the functioning of this node, for instance allowing to change the alignment of the child nodes and to change the scrolling behavior (direction and speed).

The children of the `Layout` node are in fact buttons: one for every shot. In addition to those, two other buttons are provided that make it possible for the user to navigate between the different shots. When the user positions its pointer device on top of such a button, the `scrollRate` field of the `Layout` node is given a value different from zero. Scrolling in the other direction is realized by reversing the sign of the `scrollRate` field. When the mouse is not on top of one of the two buttons, the `scrollRate` field is given a value of zero.

It is also possible to drag the navigation bar. This is implemented by relying on the functionality of a `PlaneSensor` node.

4. CONCLUSION AND FUTURE WORK

In this paper, we have discussed how several MPEG-4 tools can be used for the creation of highly customizable and interactive multimedia presentations. This was done from a practical point of view. The authors hereby hope to have contributed for a better understanding of the way in which MPEG-4 offers an integrated and interoperable solution that fulfills the requirements of many modern multimedia applications. For instance, think about the usage of video-on-demand in heterogeneous usage environments and the creation of DVD-alike applications.

With respect to the different implementations, several improvements are possible. For the trailer mosaic and the media player, it would be useful to realize the stream switching in an automatic way. As such, it would be possible to anticipate the load on the server, the network, or the terminal. The second and the third demo would also benefit from a streaming based approach. Concerning the shot browser, it would also be useful to work with scenes (a meaningful collection of shots), especially when those are annotated by using keywords.

5. ACKNOWLEDGMENTS

The authors would like to thank Bart Mollet for his valuable assistance in the implementation of the different demos during the completion of his Master's Thesis.

The research activities that have been described in this paper were funded by Ghent University, the Interdisciplinary Institute for Broadband Technology (IBBT), the Institute for the Promotion of Innovation by Science and Technology in Flanders (IWT), the Fund for Scientific Research-Flanders (FWO-Flanders), the Belgian Federal Office for Scientific, Technical and Cultural Affairs (OSTC), and the European Union.

6. REFERENCES

[1] C. Concolato and J.C. Dufourd, "Comparison of MPEG-4 BIFS and some other multimedia description languages", Workshop and Exhibition on MPEG-4, San Jose, June 2002.

[2] F. Pereira and T. Ebrahimi, Eds., *The MPEG-4 Book*, Prentice Hall, January 2002.

[3] Aaron E. Walsh and Mikael Bourges-Sevenier, *MPEG-4 Jump-Start*, Pearson Education, December 2001.

[4] Jean Le Feuvre and Cyril Concolato, *GPAC Project on Advanced Content*, available on http://gpac.sourceforge.net.