MPEG-Pro, an Authoring System for MPEG-4 with Temporal Constraints and Template Guided Editing

Souhila Boughoufalah, Jean-Claude Dufourd, Frederic Bouilhaguet,
ENST Paris Dept ComElec, 46, rue Barrault, 75013 Paris
E-mail: { souhila, dufourd, bouilhaguet }@ enst.fr

Abstract

From multimedia history, we can see that however powerful the technologies underlying multimedia systems, only relevant authoring applications will bring success to these systems. From this assumption MPEG-Pro, a first MPEG-4 authoring system prototype has been designed to integrate all main authoring capabilities from Audio/Visual encoding to the final delivery of a complete MPEG-4 application. Even if commercial tools are very popular in the authoring world, they all provide a specific proprietary solution. In this sense, the essential benefit of MPEG-Pro is to allow the production of standard contents.
MPEG-Pro architecture follows very closely the MPEG-4 structure. Therefore it inherits the benefits and the drawbacks of this format. Among these limitations, MPEG-4 temporal model is based on time-stamped events and gives no means to express temporal dependencies between Audio/Visual objects. We introduce a solution to this limitation by adding a new layer to the authoring system to manage temporal constraints.
In this paper, we present first the current authoring prototype. We describe its graphical interface and its modular architecture which allow a large scope of extensibility to be able to integrate new high-level authoring features.
Then we present our introduction of temporal constraints to the editing system and our template system which aims at hiding much of the scene description complexity of MPEG-4.

1 Introduction

MPEG-4 Authoring is quite a challenge. Far from the past "simplicity" of MPEG-2 one-video-plus-2-audio-streams, MPEG-4 allows the content creator to compose together spatially and temporally large numbers of objects of many different types: rectangular video, arbitrarily shaped video, still image, speech synthesis, voice, music, text, 2D graphics, 3D, and more…

In order to specify this composition, MPEG-4 contains the BInary Format for Scenes (BIFS) [1] which is very close to a machine language. It is thus very important to have a means to present to the author a much higher level interface. Next to the composition step, the MPEG-4 content creation chain will be the result of the integration of a number of very different macro modules: audio and visual encoders, scene description encoder, MPEG-4 File Format manager —this includes the specialization of the content towards a particular delivery technology—, event handling, scripting support which all are unique features that represent an innovative effort in the development of the MPEG-4 authoring tool.

In this paper, we present MPEG-PRO: our solution to help authors creating MPEG-4 contents from the end-user interface specification phase to the cross-platform MP4 file. We show our choice of an open and modular architecture of an MPEG-4 Authoring System able to integrate new modules. At last, we discuss the raised problems, we present our introduction of temporal constraints to MPEG-Pro and we present the future directions.

2 Toward an MPEG-4 Authoring System

2.1 User interface

To ease the authoring process, powerful graphical tools must be provided to the author [4]. It is well known that due to the temporal aspect of multimedia applications, giving instantly to the author a real perception of what he is editing is very difficult. Offering him an environment with multiple, synchronized views can help to reach this goal [3].

In a first step, we have chosen a timeline based edition and we justify this choice by two major reasons:
- MPEG-4 scene description format is based on time-stamped updates (events).
- Timeline editors are well-adapted to multimedia presentations because of the intuitive placement of time events provided by this graphical representation [2] [6].

The interface is composed of three main views and a timeline directing the whole edition/presentation. as shown in figure 1.

♦ **Edit/Preview** : By integrating the presentation and editing phases in the same view, we enable the author to see a result, even partial, of the created object. This views integration tries to approach the WYSIWYG paradigm[5]. If the author inserts a JPEG in the scene, he can immediately see on the presentation window his image located exactly where he has decided to put it. But if he assigns a particular behavior to the JPEG —for example an animation behavior or an interactive one—, he cannot see the result until he plays the scene.
We can say that integration of the two views is very useful to compose spatially the scene. Toggling between the editing mode and playing mode is performed by playing/pausing the presentation. In editing mode, the system interprets the user events as meant for the authoring part of the application . The system inserts each event in the "BIFS update" list with the appropriate parameters. In playing mode, the system transmits the events to the appropriate BIFS sensor nodes. The "Time Engine" triggers the actions and calls the "Scene Tree Manager" to execute the appropriate routes/updates/ conditionals.

♦ **Scene Tree** : This view provides a structural view of the scene as a tree (a BIFS scene is a *graph*, but for ease of presentation, the *graph* is reduced to a *tree* for display). Since the edit view cannot help to see the objects behavior, the scene tree provides more detailed information concerning the objects. This view gives a snapshot representing graphically the current scene graph at the time selected on the timeline. Even if it is a VRML-like representation presenting a hierarchy of nodes and the routes connecting them, in coordination with the edit view and with the tool selection

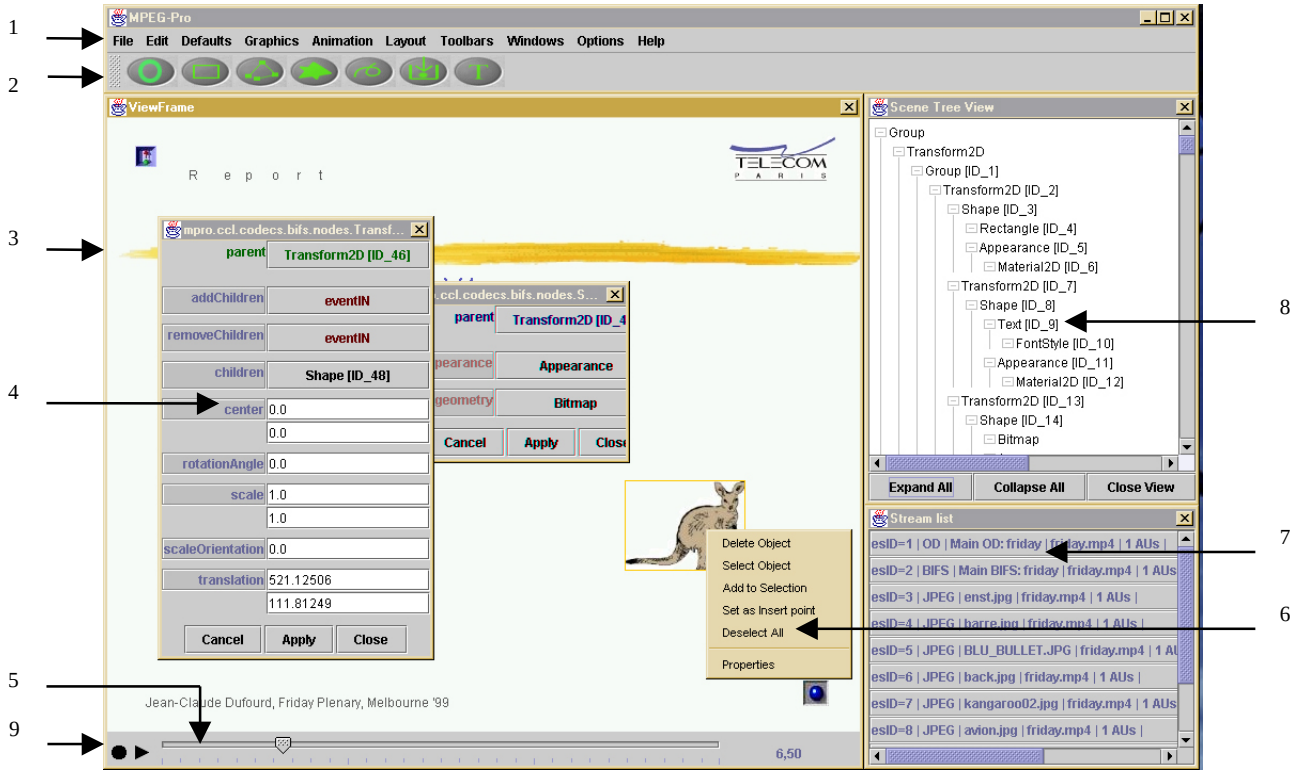

Figure 1 MPEG-Pro User Interface

properties we can easily follow the behavior of a particular object.

On figure 1, once the author launches the authoring tool, he can load an existing MP4 file, through the menu bar (1). The scene is displayed in the Main frame (3) at time 0. The scene can be played/stopped (9). The timeline (5) provides a temporal axis representing a physical representation of time. The author can set the current time by sliding the cursor forward or backward and edit the objects at the appropriate time.

The Scene Tree structure (8) displays the scene graph corresponding to the current time. From the editing frame, contextual menus (6) are associated with objects to perform different delete/selection operations or to display/modify the object fields in an appropriate property window (4). If the author wants to create new objects, he can do it from the menu bar (1) or from a Toolbar (2). The streams list (7) enables the insertion of audio/visual streams imported from other MP4 files and the edition of existing media nodes.

2.2 Introducing Temporal constraints

In the first version of our prototype, we have designed an authoring system that matches closely to the MPEG-4 system.

In MPEG-4, spatial composition is achieved by structuring the scene as a classical hierarchy of nodes on which can be applied spatial constraints by using specific nodes : Transform and Form nodes [7]. Temporal composition in MPEG-4 is achieved via time-stamped events. Hence, we have designed an editing directed by a timeline with absolute time values.

However intuitive this approach is for an author, it presents some limitations. For example, in some situations where we want to maintain a temporal relationship between two scene elements, modifying one element will break the relationship and the other element must be edited by hand to restore the relationship. As for spatial composition, MPEG-4 provides means to have spatial constraints between, expressing temporal dependencies between objects is not possible with MPEG-4.

To express and compute temporal dependencies between objects, we introduce temporal constraints management to our system. Basically, our constraints are expressed by binary constraints [2] and can be resumed by the following inequality :

$Time(o) - Time(o_i) <= delta$

Time(o) : start or end time of the object o.

delta : a time difference value.

A set of higher level expressions are given to the author. For example, to express: Object1 Co-begin Object2 is interpreted by the two inequalities :

$start(o1) - start(o2) <= 0$

$start(o1) - start(o2) <= 0$

The temporal constraint solver implementing a constraints resolution algorithm observes all time fields it manages and dynamically arranges the values if one of these is changed i.e. if a constraint object time is changed. As MPEG-4 does not provide today any temporal constraints expression, temporal constraints information are only available for the authoring system to help the author modifying his content. Once saved,

they will be stored as a hint track in the MP4 file and can be read again by the authoring tool for further processing.

2.3 Templates

Templates are customizable reusable BIFS/Java authoring elements. Templates are intended to help non-professional authors to easily customize an already created content and add/use it in their content. Templates are our means of choice to propose to the user objects of a higher level than BIFS nodes.

Customization includes changing properties and insertion of objects by means of Customization points. Objects may be BIFS objects or other templates. Customization points are points in the BIFS that can be operated on. The user detects the customization points through editing hot spots. Alternatively, the user may request a list of all the customization points of a template. We have defined mainly two major template types :

- Complete template : a customizable (entire) scene composed of BIFS/Java objects and other templates. Such a template may be composed of *building block* templates.
- Building block (Atomic) template : elementary BIFS/Java element to be customized and intended to be added to a complete template. Such a template cannot be stand-alone example : a button, a layout.

Complex templates include a Java "checker" class which implements the advanced checking of the customization variable values and/or the generation of ad-hoc pieces of BIFS. These checker classes access a specific API of the authoring tool. Templates can theoretically be designed by any user. However, we prefer to think of these as highly tuned objects which will embody a large part of the MPEG-4 expertise contained in the authoring tool.

3 MPEG-Pro Architecture

3.1 The Core/Extensions model

Our architecture is based on two main module types: Core and Extension. A Core module has lots of communication with other Core modules and Extensions. An Extension only communicates heavily with Core modules; if communication with another extension is needed, it is done through an event mechanism in the Core (Actions). Core modules are needed in all configurations of the authoring. Possible configurations of the authoring tool include:

- non-graphical tool for off-line encoding or scripted manipulations of scenes,
- beginner's version without some features
- full expert version.

The Core/Extensions architecture has two benefits: an extension developer only needs to know the Core, and the presence or absence of any extension is transparent to the other extensions. As shown in figure 2, the Core contains Authoring management, MP4 and ObjectDescriptor management, media decoding and inter-extension communication (Actions). Authoring management consists in encoding and decoding BIFS, maintaining an internal form for BIFS and OD for the scene and updates, exposing a scene modification API to the rest of the tool, managing time, maintaining an authoring history for undo/redo and initiating communication between MP4 Manager, Media Handler and the extensions.

The MP4 Manager reads MP4 files into memory, exposes an API to access the meta-information and the streams, imports raw streams by creating the needed meta-information, and writes the edited scenes into MP4 files. The MP4 manager is also responsible for Object Descriptor encoding and decoding. The Media Handler provides asynchronous decoding of the media streams. It receives samples from the MP4 Manager and makes available decoded buffers to the Renderer.

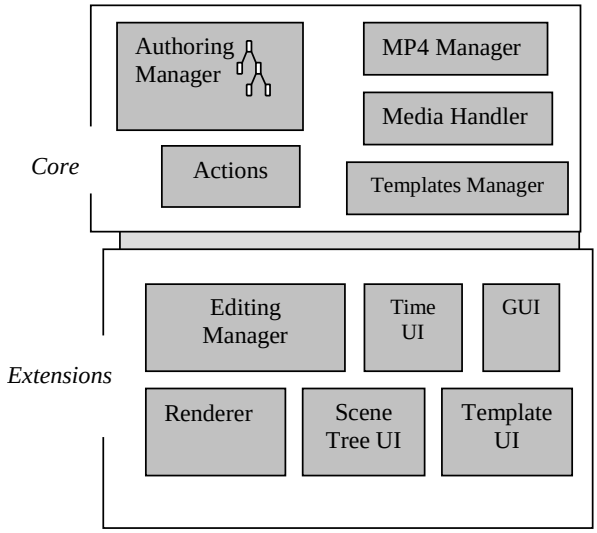

Figure 2 The Core / Extensions Architecture

The Templates Manager includes a set of GUI elements for each customization type (insertion point, primitive datatype, …), the management of the saving of the Java checker class, the symbol table, the customization points and the template instance information as a part of the hint track,

Extensions manage features such as a scene tree display, a preview window, a timeline or a properties display. The extension's code interacts and communicates with the Authoring Manager.

3.2 Function Overview

As said above, MPEG-Pro is based on timeline editing. All the metaphors used in this tool match closely the MPEG-4 systems architecture. Every author action is time-stamped and interpreted as a BIFS-update.
In a scene, there is at least one update: the Replace-Scene command which builds the scene graph at time 0. Hence, playing a scene means executing all the update commands sorted by time.

A complete application is managed by the Authoring Manager. More than one scene may be included in an application. Each scene, identified by a scene ID, has its own context :

- a node graph,
- a ROUTE list,
- a list of BIFS updates,
- a list of OD updates.

For each scene, a scene controller maintains the updates lists. It maintains the coherence of each scene context: each author modification is registered as one or several update commands (BIFS or OD) and inserted after consistency checking in the appropriate updates lists as shown in figure 3.

The standard scenario for modifying the scene is to place the scene tree at a specific time, to check the feasibility of the next BIFS command by applying it to the (time-localized) scene graph and then to modify the updates defining the scene

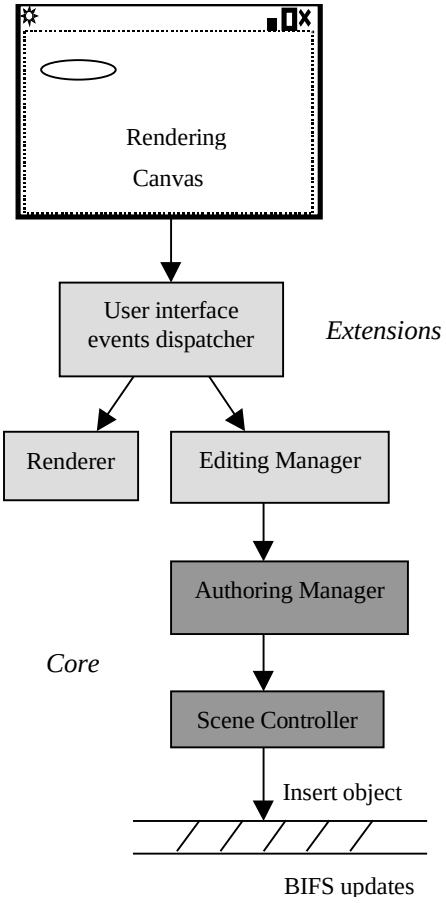

Figure 3 Authoring event path

3.3 Implementation specifics

MPEG-Pro is written in Java and interfaced with the MPEG-4 implementation group (IM1) decoders. It follows the MPEG-4 v1 specification, and builds scenes compliant with the Complete 2D profiles.

4 Conclusion and Future

MPEG-Pro is an authoring System for MPEG-4 contents. In this paper, we have presented our editing environment and the underlying architecture. We have focused on the importance user interfaces take in the multimedia editing area as well as the importance of specifying an architecture modular and flexible. We have explained our choice of an editing :

- directed by a timeline with addition to temporal constraints
- integrating the Edit/Play phase
- based on multiple synchronized views
- integrating a templates guided authoring.

We have deployed our efforts in the conception of an open and flexible architecture able to integrate new extensions without changing the Core of the system.

Our future work will address the following point:

- A navigational view: in order to help the author to create very large contents. In this case a navigational view presenting the hyperlink graph obtained from the interactive points. This view gives the author a global perception of the content.

References

1. N2501, FDIS of ISO/IEC 14496-1 (MPEG-4 Systems)
2. Nabil Layaïda ``Madeus : Système d'édition et de présentation de documents multimédia structurés, PhD Thesis from the University Joseph Fourier, Grenoble, France, June 1997.
3. M. Jourdan, C. Roisin, L. Tardif, ``Multiviews Interfaces for Multimedia Authoring Environments", Proceedings of the 54th Conference on Multimedia Modeling, , ed., pp., Lausanne, November 1998.
4. Blair MacIntyre, Steven Feiner “Future multimedia user interfaces” Multimedia Systems Volume 4 Issue 5 (1996) pp 250-268 © Springer Berlin Heidelberg 1996
5. HUDSON S. E. HSI C “The Walk-through Approach To Authoring Multimedia Documents”, Proceedings of the ACM 94 Multimedia Conference, pp 173-180, San Francisco, 1994.
6. MACROMEDIA, Director 7 online http://www.macromedia.com/, 1999.
7. Citta Baral, Graciela Gonzalez “Specifying Generic Multimedia 3D Visualizations and Temporal Presentations from Database Queries” Proceedings of the ICMCS’99 , Florence, June 1999.