Playing Temporal HTML Documents

Franck Rousseau*

f.rousseau@opengroup.org
The Open Group Research Institute
2, avenue de Vignate
38610 Gières, France

Andrzej Duda

Andrzej.Duda@imag.fr
LSR-IMAG
BP 72
38402 St Martin d’Hères Cedex, France

Abstract

The WWW has become a major publishing medium. To integrate synchronized multimedia in this environment in a seamless way we propose temporal extensions to HTML and an execution architecture to support these extensions. We have created a prototype using Java and adapted it to make it run in a standard browser.

1 Introduction

We have defined temporal extensions to HTML [1] to allow specification and playback of synchronized multimedia presentations in the WWW context. Components of such presentations may include continuous multimedia data (audio and video clips) either stored on WWW servers or coming from live-data sources. The extensions are based on three concepts: hypertime links for temporal composition, common time bases for close lip-sync synchronization between media objects, and dynamic layout for positioning and movement.

We have also defined an execution architecture [2] to play back documents specified using the temporal extensions to HTML. The architecture is based on three concepts that directly support these high level concepts: synchronization events, synchronization managers, and synchronizable media objects. We have prototyped our architecture using Java, SableCC, and DOM (Document Object Model). Since no Web browser implements DOM yet, in fact we use Dynamic HTML for our prototype. This short paper presents how we have used these technologies to integrate our architecture within a standard WWW browser environment.

2 Temporal Extensions to HTML

We propose to use a simple functional paradigm derived from temporal point nets to specify temporal composition: a temporal link between an origin and a target. We call it a hypertime link by analogy to its WWW companion. A hypertime link has an explicit temporal semantics: it relates two media samples (e.g. video frames) and assigns the origin’s time instant to the target. It can start or skip an object at the specified target position or stop it if the target is the end of the object. Following a hypertime link is automatic in the sense that it does not require any user interaction and consists of skipping in time from the origin media sample to the target. The functional relation has a nice analogy with hypertext links which expresses a relation between an origin place and a target one, and whose activation allows the user to jump (instantaneously, in theory) from one place to another in the space of documents.

To specify which objects should be kept synchronized, how often, and what is the nature of this close synchronization (in other words, who is the master of the time), we define the notion of a time base. A time base is a virtual time space in which media objects “live”. A time base defines a common time coordinate space for all objects that are related by some relations, for example master-slave dependency. A time base can be seen as a perfect time space in which real world phenomena such as jitter or drift do not exist and media objects behave in a perfect manner. We define the nature of synchronization between media segments using the notions of master and slave. A master-slave relationship defines a master controlling a slave according to its needs. We extend this notion to multiple masters and slaves through the common time base: a master can accelerate time or slow it down, hence slaves and other masters must adjust to time.

We suppose that a synchronized multimedia document may include a variety of media objects having temporal behavior. A media object defines time evolution of media samples of one type. Media samples must be presented at precise time instants defined by the rate of presentation. The rate may be imposed by the author, adapted to match the duration of another object, or adjusted to synchronize with other objects. A media object schedules the presentation of samples within a given time base. In this way, objects in the same time base are synchronized.

We define a dynamic layout as a special case of a media object. It defines the temporal behavior of the physical layout. The only difference is that layouts are neither masters nor slaves since they do not contain any media sample to be synchronized. It encapsulates frames, a means for defining regions of screen in which media objects are presented. Frames can be mixed with static elements such as traditional HTML text paragraphs or images. Frames can include other layouts to specify nested layouts thus providing nested coordinate spaces. Layouts and frames being considered as objects they can fully participate in the temporal composi-

*Franck Rousseau has been supported by Bull SA.

tion using hypertime links. This approach allows seamless integration of spatial and temporal dimensions into a multimedia document.

3 Execution Architecture for Synchronized Multimedia Documents

To support the model presented above, we have defined an execution architecture to play back documents specified using the proposed temporal extension to HTML. The extensions are fairly low-level, so all the proposed concepts have their counterparts at the system level. The architecture is based on three components: synchronizable objects, synchronization events, and synchronization managers.

A synchronizable object integrates media and synchronization: it encapsulates the services needed for a media to be played back and for controlling its execution. A synchronizable object is a media object wrapped using a synchronizable interface that defines methods needed by the underlying multimedia architecture to handle intramedia and intermedia synchronization and scheduling. Synchronizable objects generate synchronization events and are controlled by synchronization managers. By clearly separating media processing functions from synchronization, the architecture becomes modular and dynamically extensible: objects that deal with new media or compression formats, can be integrated in a seamless way.

Synchronization events convey time information between various entities. A synchronization event defines an action aimed at a target that will happen in the future. When its deadline is reached, the event is sent to the target to perform the required action. A hypertime link can be easily implemented using such a synchronization event. The deadline for a hypertime link can be obtained from the relative temporal information associated with the link.

Synchronization managers implement time bases. They manage a pool of synchronizable objects belonging to the same time base. They handle synchronization events on behalf of objects and enforce the synchronization policy defined in the time base by means of roles (master or slave) and synchronization points. Managers do not handle time themselves, but use the internal scheduler for scheduling events.

The three concepts on which our architecture is based, make the notion of time virtual. We can say that time in our architecture is elastic, which means that it provides flexible adaptive synchronization in a computing environment that does not have strong real-time support.

4 Prototype

We have prototyped the concepts of our multimedia architecture using Java, SableCC, and Dynamic HTML. We would rather use DOM which is meant to be a standard, but it is not yet implemented in available browsers. Dynamic HTML provides similar though less generic functionalities that are sufficient to create a first prototype.

Our multimedia documents integrate two aspects: the first related to the traditional definition of an HTML document that describes the structure and the presentation style and the second one concerned with the temporal behavior defined using our extensions. To play back such a document, we decompose it into two parts corresponding to these aspects. The structure and the presentation style are defined using Dynamic HTML and CSS (Cascading Style Sheets).

The prototype is aimed at a target that will happen in the future. When its moment, the prototype can handle MPEG-1 and MPEG-2 video, audio in .au, .wav, and .aiff format and text used for closed-captioned video. Several audio tracks in various formats and sampling rates can be mixed dynamically. A MPEG-2 decoder has been written in Java: it is a port of mpeg2dec (version 1.2 of July 19, 1996) available at the MPEG Software Simulation Group that has been encapsulated and slightly modified to allow synchronization control.

The Java components generated from temporal extensions are linked with the synchronization classes of our execution architecture. The synchronization classes control the Dynamic HTML. In this way, we are able to play back our multimedia documents in a standard browser environment with minor Java extensions. In the future, DOM will allow the use of standard HTML in place of Dynamic HTML and should provide more flexibility in manipulating objects through its API.

5 Conclusion and perspectives

Our approach has the advantage of sticking closely to a standard browser environment so that existing media components can be easily reused. We work on adding support for external interactions to allow the user to control a presentation. We also work on making our architecture distributed by integrating streaming functionalities. In this way, we will be able to incorporate media streams coming over network connections and synchronize them with other media objects. A suitable transport negotiation protocol will take into account different bandwidth and delay requirements of a stream. As our architecture is already based on the notion of elastic time, which means that it adapts to available resources, it will be especially suitable in a computing environment that does not have strong real-time support such as the Internet. In this case, it provides flexible adaptive synchronization and the best possible quality of presentation.

References
