An Adaptive Multimedia-Presentation System*

Yitzhak Birk♣ and Frank Ghenassia
Technion - Israel Institute of Technology
Haifa 32000, Israel

Abstract

The inability to smoothly and dynamically adapt the speed of a multimedia presentation to the viewer's ability or desire substantially reduces its effectiveness for self-study and training. This paper describes an approach whereby a file of presentation directives is created off-line, and is then used along with the viewer-requested speed to control the presentation. The directives include a speed function for each segment of each track, as well as the specification of inter-track coordination events at selected segment endings. The simple model is shown to be quite powerful, and has been embodied in a prototype system. "Smooth" adaptivity complements existing "navigational" features.

1. Introduction

Multimedia presentation systems integrate many sources of information, both visual and auditory. Moreover, a viewer can typically select the material for viewing, "navigate" through reference material, etc. [HBR93].

Most of the work on multimedia systems has been carried out in two largely-unrelated prongs and by different communities of researchers: the "systems" community has looked at issues such the construction of large-scale servers capable of playing a large number of concurrent video streams, the effects of traveling through a communication network, etc.; the "applications" community has been busy trying to understand how to use the new technology for education and other applications [Sch94]. The latter community has focused, perhaps rightfully, on the content, the way in which it should be presented, and ways of encouraging the active participation of the user in the learning process. The main issue addressed in this paper, speed adaptation, appears to have fallen in the crack between these two major directions.

Their numerous impressive features notwithstanding, multimedia presentation systems presently have a critical shortcoming, namely their inability to fully adapt the coordinated presentation of the various media to the viewer's ability or desire. For example, the presentation speed of prerecorded "multimedia" material is fixed. (Some adaptation, e.g. of video only in variable-speed VCRs, is possible, but this is inadequate.) In contrast, even text on paper presents itself at the rate desired by the viewer! The problem is most pronounced in self-study and training situations, and can manifest itself directly in the loss of a significant fraction of user time, as well as indirectly through frustration and dissatisfaction. This shortcoming thus reduces the user's efficiency, thereby at least partly offsetting the benefits offered by multimedia.

This paper presents a model that can be used to specify the missing "soft" adaptivity along with the more traditional navigational instructions, and describes an operational prototype that implements this model on a PC platform. Some of the underlying systems issues are also discussed briefly. Although the model permits an unlimited number of properties of a presentation to be dynamically adapted to viewer requests, we will focus on speed as an important and challenging example. The new features are intended to complement those of existing systems, not to replace them.

1.1 Adapting the pace of a presentation

It is well known that a variable presentation speed of certain types of media, such as audio, cannot be achieved by simply varying the rate at which waveform samples are presented to an output device. To correctly vary the speed of a multimedia presentation, one must also address the question of the appropriate handling of the different tracks relative to one another so as to best imitate an original presentation made at the modified speed.

The sophistication of correct speed variation for multimedia presentations can be illustrated by the case of a "work-along" training movie and a trainee that listens to oral instructions and works along with the demonstration. Consider a "clumsy" trainee who cannot keep up with the pace of the demonstration, and therefore wishes to watch it at a substantially slower pace, e.g. 60 percent of the nominal one. Simply slowing the entire presentation down by 40 percent would make it sound awkward, and is definitely different from a human instructor's way of responding to a request for a

* This research was supported in part by the Fund for the Promotion of Research at the Technion and by equipment grants from In-Motion, Inc., Intel and Microsoft. Dr. Birk holds the Milton & Lilian Edwards Academic Lectureship.

♣ birk@ee.technion.ac.il

slower demonstration. A more natural solution might be to slow down the video but keep the audio speed unchanged or nearly so. Since the material is prerecorded, however, a specification of the handling of the time gap that opens between tracks is also required. Thus, there is much more to speed-change than a collection of individual signal-processing tasks.

1.2 The components of an adaptive multimedia-presentation system

The necessary components of an adaptive multimedia-presentation system must include techniques for adapting the presentation of individual components, as well as a model that permits one to specify their coordination when played at different speeds and a way of implementing this specification. The architecture and system prototype described in this paper are aimed at providing these components and demonstrating them along with conventional features.

The remainder of this paper is organized as follows. In section 2, we present a model that permits the specification of the desired behavior. Section 3 describes the prototype and briefly discusses some underlying systems issues, and section 4 offers concluding remarks.

2. A Model for Adaptive Presentation of Multimedia Material

Our goals in designing the model are to permit a sufficiently flexible specification of the desired behavior of the presentation in response to viewer requests, while facilitating the specification of such behavior by an expert editor or editing system, avoiding viewer overload and facilitating extensibility. Also, we try to keep implementation complexity in check and to avoid the need for storing a modified copy of the material. In the remainder of this section, we present the model and point out how the above goals are achieved. Once again, the discussion will be in the context of adaptive speed, without implied restrictions.

2.1 A motivating example

Consider a recorded work-along demonstration. During certain parts of it, the instructor's face is shown and he is explaining issues without demonstrating anything. During the remainder of the time, the demonstration is presented on the screen and the instructor speaks in the background. An editor (human, off-line) is charged with preparing the presentation for adaptive viewing, assuming that the adaptivity is required for variable motor skills (rather than audio comprehension).

The editor decides, based on his expertise and on listening to the speaker, that the audio speed should be varied by no more than plus 30 or minus 20 percent. (The former may reflect comprehensibility limitations, and the latter - the limit of awkwardness.) Also, the editor chooses to impose lip sync. when the speaker's face is present; otherwise, he decides what to do on a case by case basis.

The editor begins by partitioning the movie into segments based on the presence of the speaker's face. When the face is present, he specifies that the speed be equal to the one requested by the viewer, subject to the bounds. When the face is not present, the editor notices that there are cases in which certain oral instructions must precede the video demonstration, while in other cases oral comments only make sense after something has been demonstrated. There are also silent periods in the audio, meaningless portions of video, and some excellent shots. The editor wishes to exploit the apparent flexibility in order to make the presentation as effective as possible at all speeds, and even to permit variability by more than the audio-variability bounds (perhaps by skipping silent periods in some cases). One important complication is that it is unreasonable for the editor to provide more than a single specification for the entire range of user-requested speed; yet, he must somehow make sure that his specification does not lead to unexpected results. Motivated by this example, we next present the actual model.

2.2 The model

A multimedia session can be viewed as a voyage among a set of multimedia presentations. The voyage is controlled by the viewer's requests while viewing a presentation or at the end of a presentation, in conjunction with prespecified behavior which may include a default voyage. For example, the default viewing of a title in an encyclopedia may consist of a high-level description, but hyperlinks may permit the viewer to request digressions in order to receive in-depth explanations about specific items [HBR93]. We refer to the navigational support as an interactive script, and to a sequential viewing session (with soft adaptivity but no navigation within it) - as a presentation. Although a presentation does not permit navigation within itself, it does provide hooks to support the script. For example, it may display navigation buttons that allow the user to request the termination of one of its components, or the termination of the ongoing presentation in favor of a specific different one. At a yet higher level, one can invoke multiple unrelated multimedia sessions on the same machine. Our focus in this paper is on the presentation and the levels below it.

A presentation consists of a set of tracks that are displayed concurrently. Examples of tracks might be the audio, video and foreign-language subtitles of a movie. Each track is partitioned (by the editor) into contiguous

segments. Each segment refers to a contiguous portion of source material of some type, and also includes a specification of an adaptivity filter. (For example, a segment might refer to a sequence of video frames and the filter might specify that the segment should be

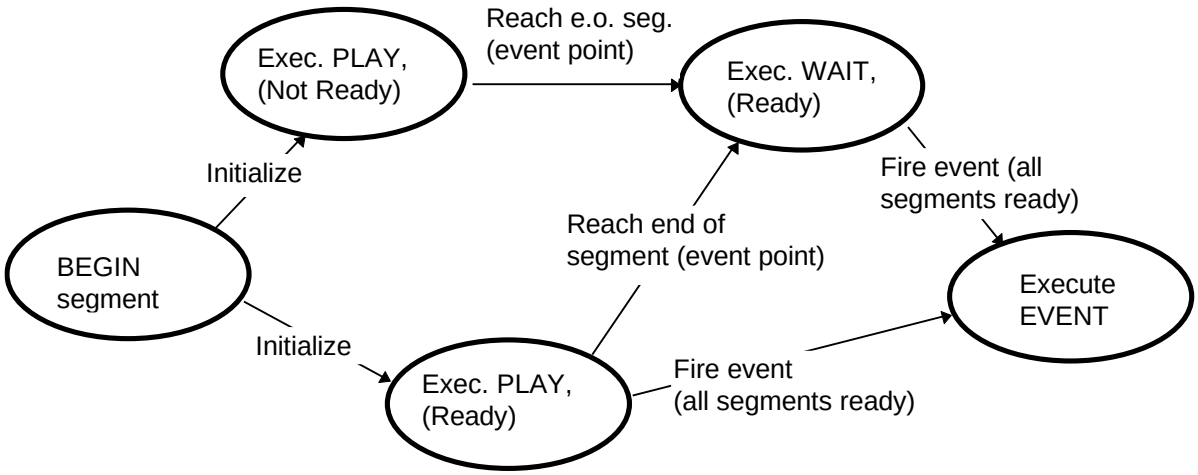

Figure 1: The finite state machine representing the execution of a segment which is also a member of an event set. The top two states correspond to the case in which the segment's completion is a necessary condition for the firing of the event; the bottom state corresponds to the case in which the segment needn't be completed prior to the event. The directives for a segment include instructions for its execution during the PLAY, WAIT (it has been completed but the event has yet to fire) and EVENT (the event has just fired) states. In the PLAY phase, the segment is played according to its speed function. In the WAIT state, it may "freeze", repeat (PLAY, Ready) or complete and cause the next segment of its track to be executed. Once the event fires, a segment that has yet to be completed may continue, possibly at a different speed, or may be discarded in favor of its track's next segment. A completed segment will typically be discarded at this point.

displayed at the nominal rate times the factor requested by the viewer but must remain within certain bounds.)

To control the degree of "desynchronization" among tracks, i.e., the relative playing positions, sets of inter-segment boundaries, at most one from each track (per set), are defined as "events". The specification of an event is used to determine the relative behavior of the different tracks. For example, an event specification may state that all tracks reaching their respective event points must wait until all other members of the event set reach their respective points, at which time the processing of the next segment of every member track commences. Similarly, one could specify that when any two tracks reach the event point, the remaining ones must skip to their respective points, at which time processing continues in the next segments of all tracks. Finally, a member track may be specified as being affected by the event but not participating in triggering it, or vice versa.

Figure 1 depicts a finite state machine corresponding to the execution of a single segment which is a member of an event set. For example, a segment whose completion is not a condition for the firing of the event, and which must be terminated once the event fires (this corresponds to skipping to the end of the segment), goes through the following sequence of states: BEGIN; PLAY (Ready); [WAIT (Ready)]; EVENT. It visits the WAIT state if and only if its playback is completed prior to the firing the event.

The final element of the model is the Multimedia Object, MMobject for short. It implements the methods necessary to process a track. An MMobject is associated with at least one output. Each output is either a destination device (audio card, screen, disk, etc.) or a redirector (a pipe), whose output will in turn be considered by another MMobject as an input.

In a presentation, each object receives its input from one or several I/O devices (abstracted as objects) and similarly sends its output to I/O objects. The representation is based on the well-known model of the source/filter/sink [Gib91] whereby each I/O object either represents a source or a sink and each MMobject represents a filter. Consequently, an MMobject only contains the methods required to process and format the data. As may already be evident to some of the readers, this model is very well suited for implementation as an object-oriented architecture. (Other multimedia systems have used such architectures as well [AHR93], [Din93], [MHB90].)

A multimedia presentation employs one or several MMobjects that act concurrently. This composition is what is visible to the viewer. For example, a presentation that displays a movie and occasionally shows slides would employ one MMobject that handles the slides, one that processes the video stream, and one in charge of the audio track (even though in some cases the audio and video tracks may be processed within the same MMobject). Each track is assigned to an

MMobject of the appropriate type. The MMobject itself contains no data.

The intuitive presentation of the model suggests that all tracks are partitioned at points that would be played simultaneously at the nominal rate (without adaptivity), but this need not be the case. Also, filters can be quite complex, including an ability to react to "discrete" user requests such as pressing buttons.

In summary, then, the specification of a multimedia presentation includes the set of tracks, each specified as a sequence of segments with their associated filters and mappings to MMobjects, along with a sequence of events, each specified as a list of inter-segment boundaries and the requested actions. The actual implementation can be carried out in a variety of ways without altering the model.

2.3 Meeting the goals

An important decision was to confine the explicit coordination among tracks to segment boundaries using the event mechanism, while permitting independent adaptivity of each track within a segment. This permits the use of a single adaptivity specification for the entire range of user-requested speeds, without worrying about runaway situations beyond the limited range of a segment. Yet, the model permits a great degree of flexibility in shaping the presentation specification. Viewer overload is avoided by permitting a single user-control to govern a set of related tracks. (E.g., a single speed control over multiple tracks.) Finally, the use of an auxiliary file to hold the presentation directives leaves the original material in tact, permitting it to be stored on inexpensive media such as CD-ROM.

3. Implementation

3.1 System overview

A prototype multimedia presentation system that implements the above model has been built in our lab, with two goals in mind: (i) demonstrating the feasibility of implementation on a widely available general-purpose platform, and (ii) evaluating the usefulness of adaptive presentations. The chosen platform is a PC with a Pentium processor running Microsoft Windows NT [Cus93], assisted by compression/decompression hardware as well as an audio processing board capable of altering the speed of voice without distortion. (We are presently working on a software-only implementation.) The system presently supports still pictures, compressed video in Motion JPEG format, and audio. Other formats, including MPEG, will also be supported. The system is highly modular and extensible due to the clean model and the object-oriented architecture. The code was written in C++. (The initial prototype of the playback system was ported to Windows 3.11 due to the lack of NT device drivers for the audio and video cards.) The system comprises two main components: an editing system and a playback system.

The MMeditor permits an "expert" to view the material, mark segments, and specify both the per-segment speed functions and the synchronization event on segment boundaries. Figure 2 depicts a snapshot of the screen of an initial version of the editor. The MMeditor's output is a file containing presentation (speed) directives for use by the playback system. An example of presentation directives is depicted in Figure **3**, wherein "*" represents the possibility of additional similar items. The files containing the original material are untouched, permitting the use of read-only media such as CD-ROM for storing them.

The playback system receives as input the original material and the presentation directives, as well as (dynamically) the user's speed- and navigation-requests, and generates the requested presentation.

3.2 Scheduling

Since the presentation is often composed of several tracks, and tight adherence to the presentation directives must be maintained, proper scheduling is very important. This importance is further accentuated when attempting to use a non real-time operating system like Windows NT.

Proper scheduling comprises two elements: sufficiently accurate timing of the invocation of MMobjects (even when there is a single periodic object and no load), and the ability to allocate time slices of the correct lengths to different objects in the right order.

The requirement for accurate timing in our case is no different than for any media player, and is a modest one due to the fact that the final precise streaming of data is always controlled by (simple) dedicated hardware, which is capable of buffering significant amounts of data at its input or to pull the data directly from memory. Moreover, the inter-track coordination, both within segments and on segment boundaries, is

based on examining data structures containing the current position of the various tracks, and this data is in the "native" units of the respective tracks, so no time measurements are involved. For example, we compute the video frame number that should go along with the current audio sample number, taking into account the relative speeds, and check whether it matches the current frame number. If it does not, the speed of the audio is changed slightly.

upon other user-level tasks to provide services, and this time is not associated with the original user-level task. There are various workarounds, and more comprehensive solutions are appearing on the horizon [GA91], [FL93], [MST93], [HK94], [FM95]. Fortunately, measurement errors have no cumulative effect since the next invocation of a periodic MMobject is independent of its completion time. Our user-level scheduler runs at a very high priority level, and the remaining threads of the playback system run at a

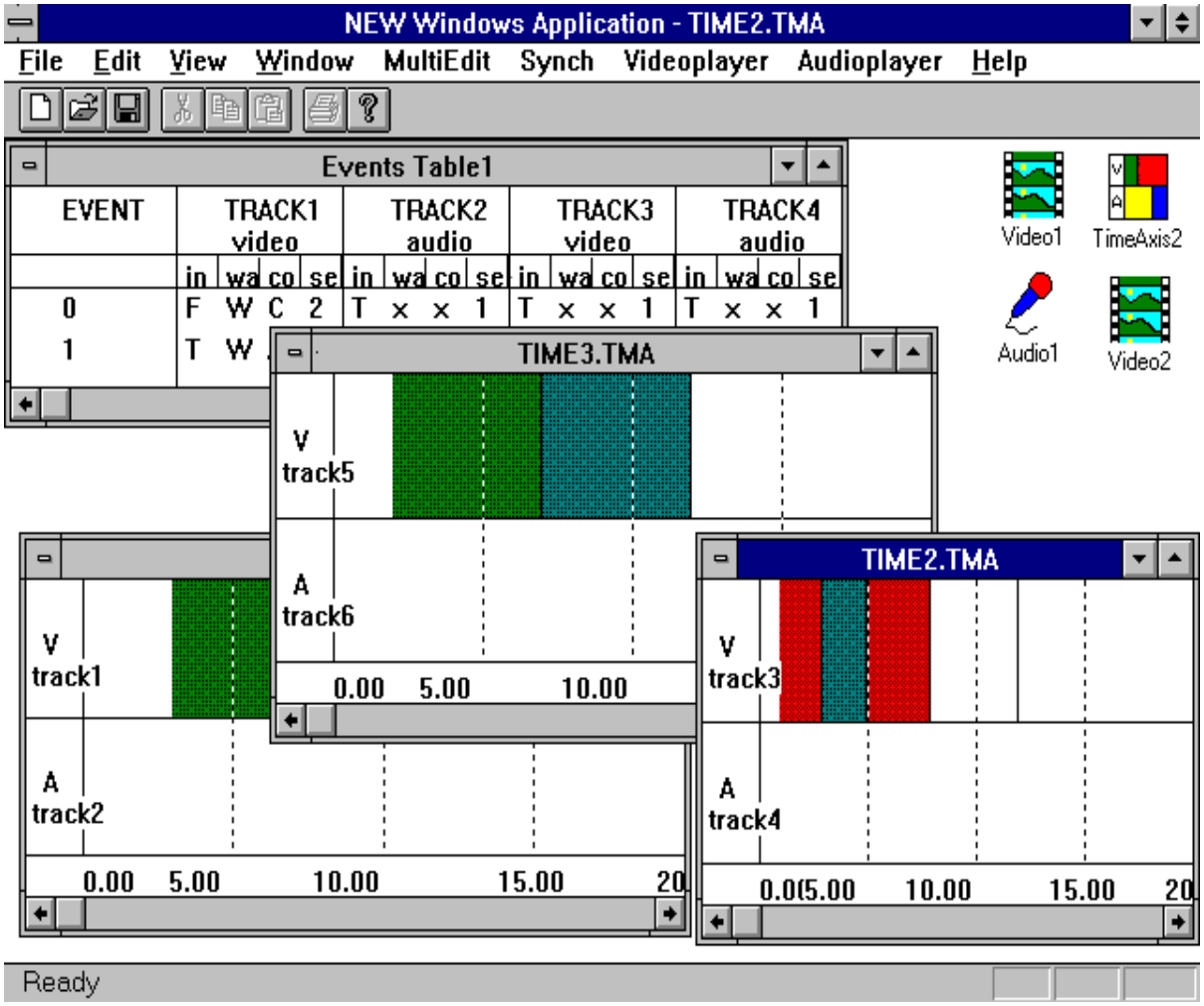

Figure 2: Representative MMeditor screen. Time lines and event lists are shown.

For the correct allocation of time slices, one needs to know the amount of CPU time required for each MMobject (period and duty cycle for a periodic object), and must also be able to measure the CPU time it has consumed. (Similarly for other critical resources.) For the required amount of time, we use the worst case (computed off line), although one could be more daring at the risk of an occasional missed deadline.

The measurement of consumed time is presently imperfect under Windows NT, which is a microkernel-based operating system. The reason is that while time consumed by a user-level task and by the kernel on its behalf can be measured, the kernel may in turn also call

slightly lower priority. Consequently, the time slices allocated by our scheduler to our threads are usually consumed entirely on behalf of the desired MMobject, and the measurements are reasonably accurate. This, combined with refraining from excessively loading the computer and the fact that data is buffered, has resulted in smooth operation. Nevertheless, we are planning to take a closer look at this issue in the near future.

```
[Directives Header]
Title                                   =   Example
Author                                  =   Frank Ghenassia
Date                                    =   10.12.94
[Event Table]
*       [Event]
                Number                  =   6
*               [Track]
                        Notif           =   TRUE
                        Wait            =   FREEZE
                        Cont            =   CONT
*       [Track]
                [Track Header]
                        type            =   Video
                        Input           =   Disk
                        Output          =   Net,Screen
*               [Segment]
                        Source          =   c:\video\example.jpg
                        StartOffset     =   1000
                        EndOffset       =   15000
                        [Filter]
                                Speed   =   1.20
                        Event           =   2;4
[End]
```

Figure 3: Presentation directives format. For each segment participating in an event, we specify whether its reaching the event point is a necessary condition for the event to trigger, what it should do from the time it reaches the event point until the event triggers, and what it should do once the event triggers. For each segment, information includes the speed function as well as the source of the data.

4. Conclusion

The need for continuous, "soft", dynamic adaptivity of multimedia presentations has previously received little attention. In this paper, we defined the problem and suggested a model for describing the desired adaptivity. Our operational system shows that a sufficiently-high-quality system can be implemented on a PC with a standard operating system. The present implementation uses special-purpose boards, but a "software only" implementation will be possible in the near future on PCs, since many of the functions required for "multimedia" will be included in the basic configurations and even in the general-purpose CPUs.

The presentation directives could be interleaved with the prerecorded material, but storing them separately permits adaptivity to be added to any prerecorded material, and multiple sets of directives may be created for the same material.

Our prototype demonstrates the usefulness of this kind of adaptivity as a complementary feature to the existing "navigational" ones, and can be used to develop expertise in correctly specifying the adaptivity for different kinds of material, situations and purposes. We are presently working on a second-generation of the implementation, focusing on increasing the friendliness of the editor and the generality of the event specifications and speed functions.

5. References

[HBR93] L.Hardman, D.C.A Bulterman, G.van Rossum, "The Amsterdam hypermedia model: extending hypertext to support *real* multimedia", TR CS-R9306 1993, Univ. of Amsterdam.

[Sch94] R.C. Schank, "Active learning through multimedia", *IEEE Multimedia,* vol. 1, no. 1, pp. 69-78, Spr. '95.

[Gib91] S. Gibbs, "Composite multimedia and active objects", OOPSLA'91, pp97-112.

[AHR93] F.Arbab, I.Herman, G.J. Reynolds "An object model for multimedia programming", Comp. Graphics Forum (Eurographics'93 Conf. issue).

[Din93] D. Dingeldein, "Modeling multimedia-objects with MME", ZGDV, Darmstadt, Germany, TR, 1993.

[MHB90] F. Manola, M.F. Hornick, A.P. Buchmann, “Object data model facilities for multimedia data types”, TR TM-0332-11-90-165, GTE Labs, Dec. ‘90.

[Cus93] H. Custer, “Inside Windows NT”, Microsoft Press, 1st ed., 1993.

[GA91] R. Govindan and D.P. Anderson, “Scheduling and IPC Mechanisms for Continuous Media”, Proc. 13th ACM Symp.on Operating Sys. Principles, pp. 68-80, Oct. ‘91.

[FL93] B. Ford, J. Lepreau, “Microkernels should support passive objects”, 3rd Intnl. Workshop on Object Orientation in Operating Sys., Dec. 9-10, ‘93, Ashville NC, U.S.A.

[MST93] C.W. Mercer, S. Savage, H. Tokuda, “Processor capacity reserves for multimedia O.S.”, TR CMU-CS-93-157, Carnegie Mellon Univ., May ‘93.

[HK94] G. Hamilton, P. Kougiouris, “The Spring nucleus: a microkernel for objects”, Proc. 1993 Summer USENIX Conf., June ‘93.

[FM95] R. Fishtein and A. Mendelson, “Thread Set - enhanced kernel support for scheduling and resource management in Mach 3.0 operating system”, in preparation.