Structured Multimedia Authoring

DICK C.A. BULTERMAN
CWI: Centrum voor Wiskunde en Informatica, Amsterdam

LYNDA HARDMAN
CWI and Technical University of Eindhoven

Authoring context sensitive, interactive multimedia presentations is much more complex than authoring either purely audiovisual applications or text. Interactions among media objects need to be described as a set of spatio-temporal relationships that account for synchronous and asynchronous interactions, as well as on-demand linking behavior. This paper considers the issues that need to be addressed by an authoring environment. We begin with a partitioning of concerns based on seven classes of authoring problems. We then describe a selection of multimedia authoring environments within four different authoring paradigms: structured, timeline, graph and scripting. We next provide observations and insights into the authoring process and argue that the structured paradigm provides the most useful framework for presentation authoring. We close with an example application of the structured multimedia authoring paradigm in the context of our own structure-based system GR*i*NS.

Categories and Subject Descriptors: H.5.1 [**Information Interfaces and Presentation**]: Multimedia Information Systems — *Evaluation/Methodology*; H.5.4 [**Information Interfaces and Presentation**]: Hypertext/Hypermedia — *Architectures*; D.2.6 [**Software Engineering**]: Programming Environments — *Graphical*.
General Terms: Design, Languages, Verification
Additional Keywords and Phrases: Multimedia authoring, Hypermedia, Synchronization, SMIL.

1. INTRODUCTION

The promise of multimedia computing is that it will redefine the way that people communicate with each other via computers. While the inclusion of audiovisual content in computer-based presentations has been technically possible since the beginning of the 1990's, it has still not become incorporated in the standard set of desktop tools, such as word processors, spreadsheets and slide-based presentations. The underlying problem is that authoring interactive multimedia presentations is a complex process that requires that an author specify different types of information at different levels. This includes the selection of media items to be included in the presentation, their spatial layout, any linking/interaction relationships, and, not the least, the temporal relationships among them. The delivery of multimedia presentations via a network infrastructure introduces additional authoring complexities, including the need to specify additional content (or document mark-up) to facilitate adaptation to the delivery environment (based on available bandwidth or device capabilities) and the needs of a diverse user community.

This research was supported under the ESPRIT-IV project CHAMELEON. More recent support came from Oratrix Development BV.
Authors' addresses: CWI, PO Box 94079, 1090 GB Amsterdam, The Netherlands; {Dick.Bulterman, Lynda.Hardman}@cwi.nl

Many systems have tackled the authoring complexity problem in different ways. While most commercial systems artificially reduce authoring complexity by limiting the presentation features available, several research systems have attempted to provide more comprehensive support for a wide range of presentation authoring needs. This article surveys selected commercial and research approaches in the context of four different paradigms for authoring multimedia documents: *structure-based*, *timeline-based*, *graph-based* and *script-based*.

The following section describes the general issues confronted when creating networked multimedia presentations. We then describe each of the four authoring paradigms in detail and discuss the advantages of using a structured approach. We next consider the process of multimedia authoring in the context of a representative example in which several paradigms are merged within a primarily structure-based framework. We conclude with a discussion on unsolved problems and future directions for research on authoring interactive multimedia presentations.

2. ISSUES IN MULTIMEDIA AUTHORING

An author of an interactive multimedia presentation has the goal of communicating a message. In order to achieve this goal, the author is required to specify the individual parts of a multimedia document (the *media assets*) and the relationships that exist among these assets. The result of the authoring process is either a final-format encoding of a presentation (for example, an MPEG-4 [ISO 1999] file containing visual, audio, navigation and descriptive data) or a more flexible intermediate specification that can be tailored to the context of the presentation environment at run-time (such as a SMIL 2.0 [W3C 2001, Bulterman and Rutledge 2004] specification containing a logical presentation structure and a set of alternatives for various presentation contexts). This paper will not contrast the benefits of final-form versus flexible-form encoding of multimedia; instead, we will concentrate on the process of creating the presentation itself. This presentation can be encoded — at least theoretically — in either format as part of a publishing step that is largely de-coupled from authoring the base presentation.

An *authoring system* allows the presentation creator to develop a narrative structure based on a collection of media assets and a creative intent that manages the presentation's visual and temporal flow. We refer to the way the system presents the underlying document structure to the author as an *authoring paradigm*.

In some respects, the task of authoring multimedia is comparable to creating a text document (such as this article) using a word processing system. Both activities require the collection/generation of source material and the placement of these sources within a presentation environment. Depending on the features supported by the formatter, authors may be able to vary the styling of the text, they may be able to vary the spatial layout, and they may be able to incorporate higher-level structures, such as chapters and sections. In the same way, multimedia authoring tools allow an author to integrate several types of information into a composite presentation and — depending on the system — allow spatial and temporal styling. Unlike text, however, the temporal dimension dominates the multimedia authoring process. Image, video and audio assets do not lend themselves to dynamic transformation or styling and a typical media presentation uses fixed and not flow-based layout.

In many respects, then, multimedia authoring is more akin to the process of making a movie. Here a (human) editor is concerned that the individual shots that have been created are assembled into sequences which are in turn grouped into scenes containing a single coherent thread of the story [Rubin and Davenport 1989]. Often separate audio tracks (containing background music or alternative languages) need to be synchronized with the base visual material. Increasingly, supplemental information — such as captions and metadata — also need to be integrated as part of the authoring process. While these concerns are also present in (networked) multimedia, a fundamental difference exists between movie content and multimedia presentations: interactivity. Historically, movies have been created as a strict linear production, with little or no user navigation available. Even DVD-style partitioning of content into chapters and sections ultimately resolves to a range of strictly ordered sequences in which content linking and user input play no role. Finally, movies are produced for a standard display environment — either wide-screen or TV — while modern network multimedia content can be displayed on devices ranging from high resolution desktop displays to low resolution mobile telephones.

As a result, the construction of context-sensitive, interactive, networked multimedia presentations is a non-trivial task that can only partially draw on experiences taken from authoring tools for each of the constituent media types. A multimedia presentation must describe both an abstract collection of candidate content and the mapping to a (set of) instances that ultimately get presented to the user.

In creating a multimedia presentation, a set of concerns can be identified that need to be supported by full-featured systems:

1. *Media assets*. The media objects used in a presentation need to be referenced. This seemingly trivial task is, in reality, no *so* trivial: aspects of a media object may not be known at authoring time (such as duration). Consider a typical evening newscast as an example. Here, there may be identifiable abstract assets, such as a set of national and local stories, the financial news, a sports round-up and the weather, but the exact nature of the daily instances of these will vary.
2. *Synchronization composition*. Across the collection of (abstract or physical) assets, a specification needs to be constructed that allows groups of assets to be presented in a coordinated manner. The synchronization can be based on hard timing relationships, a relative structural ordering, a set of constraints (such as having asset *A* repeat for the duration of asset *B*) or based on *a-temporal* composition (in which groups of assets are identified as belonging together without specifying predetermined timing relationships among the groups).
3. *Spatial layout*. In presentations containing more than one concurrently active media asset, a mechanism needs to exist to manage the rendering space. While such management can be implicit (as with video), it can also be explicit, dynamic and time-varying.
4. *Asynchronous events*. Fine-grained synchronization can be influenced by conditions generated asynchronously during the presentation. These can take the form of content-based events (such as markers embedded in the media content) or some form of user interaction (such as clicking a *next* or *start* button). Direct user interac-

tion can be applied as a timing constraint (such as specifying the end of an asset) or as navigation within or across presentations. (For a discussion on the detailed relationships of events and navigational linking, see [Hardman et al. 2000].)

5. *Adjunct/replacement content*. For presentations not distributed in a final form encoding, alternative content can be defined that can be used to adapt a presentation to specific user or device constraints. These alternatives can include *adjunct* content (that is, content presented along with a set of base media assets, such as text or audio captions) or *replacement* content (one or more assets that replace all or part of the base content).
6. *Performance analysis*. A critical element of a networked multimedia presentation is that shared resources are required to move (relatively) massive amounts of media from one or more servers to one or more clients. While the intra-stream synchronization problems are largely understood and supported by time-sensitive transport protocols (such as RTP [IETF 1995]/RTSP [IETF 1996]), an authoring system should provide an author with feedback on at least the performance implications of selecting a particular mix of media assets. Better authoring tools also provide performance optimization for various delivery scenarios.
7. *Publishing formats*. The world of networked multimedia has yet to converge on a single presentation encoding format. As a result, an authoring system should support multiple external formats, ranging from fixed final form encodings to flexible intermediate formats. The particular formats supported (such as MPEG-4 or SMIL) are less important in this paper than the inherent tranformability of the underlying authoring paradigm.

In the discussions below, we will rate each paradigm based on the range of authoring capabilities it generally exposes. Table I provides the metrics used. Note that our analysis of the authoring task concentrates primarily on designing an authoring system that supports assembly and interaction specification. We are less concerned with the creation of individual media items, since this requires the use of media-specific editors for the different media types used.

Table I. Properties of authoring paradigms.

1		2				3		4			5		6	7	
Media Assets		**Synchronization composition**				**Spatial Layout**		**Async. Interaction**			**Adaptive content**		**Performance**	**Publishing**	
Abstract	**Physical**	**Temporal**	**Abstract**	**Constraint-based**	**A-temporal**	**Implicit**	**Explicit**	**Events**	**User interaction**	**Navigation**	**Adjunct**	**Replacement**	**Environment Modelling**	**Final Format**	**Flexible Format**

3. AUTHORING PARADIGMS

In this section we provide an overview and analysis of the four dominant authoring approaches used in existing multimedia authoring systems. These paradigms are: *structure-based*, *timeline-based*, *graph-based* and *script-based*. While we use these to classify the authoring systems discussed in the next section, more than one paradigm may be present in any one system. Further details on the systems and the discussion can be found in [Hardman and Bulterman 1995] and chapter 4 of [Hardman 1998].

3.1 Structure-Based Paradigm

A structure-based paradigm uses an abstract representation of a presentation to define which media assets are included and when they get activated. The most common representation used is that of a document hierarchy or a document tree, with intermediate nodes containing composition elements and root nodes containing media asset pointers.

3.1.1 General characteristics of the structure-based paradigm. Structure-based authoring systems support the explicit representation and manipulation of the structure of a presentation. The structure collects media assets in the presentation into composite "sub-presentations" which can be grouped and manipulated as one entity. Multiple instances of the same object can belong to one or more groups, but each instance is treated as a logically separate entity. Nodes in the document tree can be used to define several types of composition: either a strict temporal ordering, a coarse general ordering (such as sequential or parallel composition) or a-temporal ordering (in which the actual temporal activation order is determined at runtime via links, events or user interaction). The structure can represent both a strict linear ordering or a set of selection (or *choice*) points that allow non-linear narratives to be specified. The structuring may group the media items indirectly, where, for example, higher-level concepts are associated with each other and each concept is associated with one or more (groups of) media items.

Figure 1 illustrates the structured paradigm for a multimedia presentation containing two text objects, two images and a video object. In the composition, a text object and image are displayed, followed by either a video or a second text/image set. Note that tim-

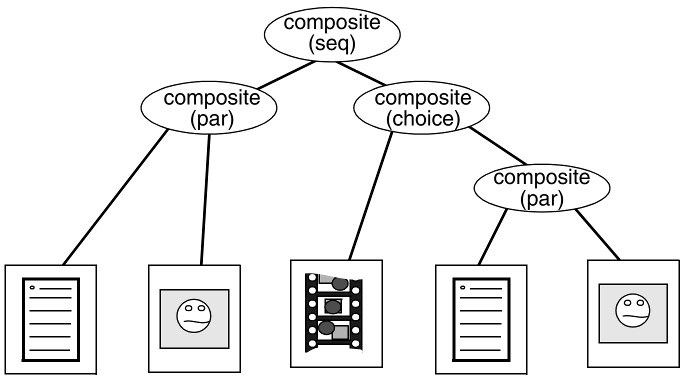

Fig. 1: Structure-based paradigm.

ing constraints can be supplied directly or indirectly with the media or the structure, but that often any binding to detailed timing can be delayed to publication time rather than (initial) authoring time.

3.1.2 Examples of the structure-based paradigm. While not the dominant commercial paradigm for describing presentations, the structured paradigm has enjoyed wide support within the research and advanced tool community. This section provides a brief overview of several important structure-based systems.

CMIFed [Rossum et al. 1993] is an authoring tool based on the CMIF format [Bulterman et al. 1991]. Both abstract and physical media selection is supported. CMIFed manipulates temporal composition using CMIF's parallel (*par*) and sequential (*seq*) time containers, and supports a-temporal composition using *bag* choice nodes. Constraint-base synchronization is not supported. Explicit layout is associated with temporal streams of media via the concept of a *layout channel*. CMIFed implements the navigation facilities of the Amsterdam Hypermedia Model (AHM) [Hardman et al. 1994]; event-based user interaction is not supported. Both adjunct and replacement alternative content are supported via the channel architecture. CMIFed did not integrate performance analysis but did support a flexible encoding of the final presentation. Many of the specification concepts introduced by CMIF/CMIFed/AHM were integrated into the W3C SMIL language. CMIFed evolved into the GR*i*NS [Bulterman et al. 1997] authoring environment, discussed in more detail in section 4 of this paper.

MAD (Movie Authoring and Design) [Baecker et al. 1996] represents a multimedia document as a nested hierarchy of acts, scenes and shots. Unlike CMIFed, the hierarchy is used to represent a logical decomposition of content rather than a logical composition of independent assets. It is representative of a simple structured approach that gets mapped to a fixed video-like temporal structure. MAD supports the specification of physical media only, and supports direct temporal composition in which the start time of each asset either is calculated from the start times and durations of preceding items and sub-items in the hierarchy or is specified by the author. The layout model is implicit. There is no support for asynchronous interaction or adaptive content. MAD uses a fixed implicit layout model and has no support for events or links.

MET++ [Hodges et al. 1989] uses a hierarchy of serial and parallel compositions of media items that is similar to CMIF. The temporal composition of the constituent items is derived from this composition hierarchy automatically. When the start time or duration of an event is altered all time positions are recalculated. Any object can be stretched or reduced in time. In the case of a composite object, the transformation is applied throughout the descendant hierarchy. The spatial layout is contained as a set of attributes on each media item (rather than CMIFed's abstract layout channels); the MET++ interface shows the values of attributes that vary over time, e.g. the horizontal and vertical positions in the figure. This representation could also be used for other object parameters, e.g. the volume of an audio object, or the fade-in rate of an image or video. MET++ does not provide support for asynchronous interaction, adaptive content or performance modelling.

Mbuild [Hamakawa and Rekimoto 1994] also uses a hierarchical structure of composites and multimedia assets for editing and reusing composite multimedia data. The timing of the presentation is determined when the highest ranking composite object is determined and is calculated using temporal glue. The main contribution of Mbuild is the explicit use of abstract media specification: authors are able to create empty hierarchical structures, reflecting the narrative of the presentation, and only later fill them with the desired media items.

Madeus [Tien and Roisin 2002] is one of the first systems to provide a uniform constraint-based specification of both temporal and spatial constraints between media elements or element compositions. The constraints are provided by the author and used to calculate the actual timing and spatial layout. The Madeus temporal view is similar to a timeline view (discussed below), but calculates the timing of elements (and composites) from the constraints specified among them rather than as absolute positions on the time axis. Madeus does not support abstract asset specification, a-temporal composition, any form of hyper-navigation or adaptive content. Madeus has evolved into *LimSee2*, discussed under the timeline paradigm.

HyperProp [Soares et al. 2000] supports the editing of hypermedia documents through three related views: structural view, temporal view and spatial view. The structural view shows nested boxes of the compositional structure of the document using a fish-eye representation. This allows users to choose between viewing global context and local details of the part of the presentation they are currently editing. The temporal view shows the temporal constraints among the elements and composites in the document. The temporal view is similar to the Madeus temporal view.

Encore [Adobe 2002] is a commercial DVD editing system that provides a quasi-structured view of a presentation. While not exposing a document hierarchy or tree directly, it does provide support for defining a structured composition of a DVD as a series of independent objects that are a-temporally scheduled. Within each object, chapters can be defined to support limited content-based navigation. (Time-based navigation is supported as a property of the video encoding of each asset.) Encore restricts the exposure of the structure of a disk to one or more (possibly hierarchical) menu pages. No support is provided for linking to content outside the scope of the presentation or for introducing content-based links within the presentation. There is limited support for adjunct content (in that captions for multiple languages can be added to a presentation), but no support for replacement content (such as displaying a slideshow instead of a video element on low-resource devices).

3.1.3 Advantages and disadvantages of the structure-based paradigm. Structure-based systems allow the explicit specification and manipulation of a presentation's structure. The advantage of this approach to authoring is that authors are able to use the structure as a storyboard, i.e., a representation of the narrative, for the presentation. The author is thus able to manipulate the narrative directly at an abstract level. Since the presentation consists of different levels of structure, it can be viewed at different levels of detail, providing easy navigation of the narrative structure. Another advantage is that since the structure is able to indicate an ordering it can be used to derive the timing for the presentation, as demonstrated in CMIFed, MAD, MET++ and Mbuild. The timing can thus be visualized and edited, at least to some extent, in the structure-based view. It may even be possible to have the structure displayed along a timeline, as illustrated by CMIFed and MET++.

The structured paradigm allows synchronization constraints to be defined between media assets, between media assets and a composition node, or between two compositions. Such relations can be implicitly defined in attributes or explicitly defined in external structures, such as CMIF's *sync arcs*.

Spatial layout is defined by assigning a position on the screen to the media item. This can be done on a per-asset basis (via attributes), using common abstract layout collections (such as the CMIF channel), or by specifying the *x* and *y* positions over time (as in MET++). These methods are not specific to the structure-based paradigm, and in all cases a separate layout view is required if explicit layout is supported.

Asynchronous interaction within a presentation can be specified directly within the structured paradigm, either as a set of predicates on composition objects/assets, or via additional structural element. Links can be created among structures, allowing, at least in principle, source and destination contexts to be specified along with the source anchor (i.e. the value from which the hotspot is derived).

The specification of adaptive content, either in terms of replacement media assets and compositions, or conditional adjunct content, can be specified via composition relationships. Adaptive content can also be specified as part of the layout dimension, as with CMIFed's channel structure, in which the set of active channels determines object visibility (and thus allows conditional or replacement content to be displayed based on channel visibility).

A drawback of the structure-based paradigm is that extra authoring effort must be expended to create the structure components in the hierarchy, rather than simply scheduling objects directly. Our hypothesis is that the benefits of understanding and manipulating the presentation structure will outweigh the initial effort.

3.2 Timeline-Based Paradigm

Timeline-based paradigms use an explicit temporal scale as a common activation reference for all media objects. Rather than manipulating logical relationships among media assets, timelines provide a virtual video-tape model for authoring in which only the relative ordering is emphasized.

3.2.1 General characteristics of the timeline-based paradigm. The timeline paradigm is a simple and popular model for ordering media assets in a presentation. Timelines show the constituent media assets placed along a time axis, typically on different tracks. A single track is usually assigned per media asset. This visualization gives a direct representation of the active period of each object during the presentation. Timeline based authoring systems allow the specification of the beginning and end times of display of a media item in relation to a time axis. Individual objects are manipulated rather than groups of objects, so that if the start time or duration of a media item is changed then this change is made independently of other objects on the timeline. All navigation is given in terms of a new position on the timeline.

Figure 2 illustrates the timeline paradigm for a presentation containing three text objects, four images, two videos and two separate music clips.

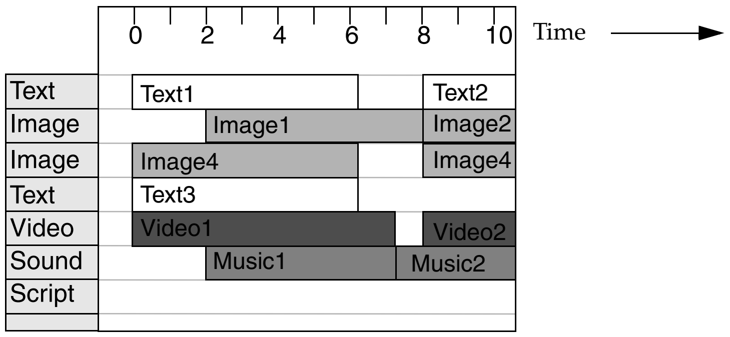

Fig. 2: Timeline paradigm.

3.2.2 Examples of the timeline-based paradigm. The timeline paradigm is widely used in commercial and restricted-functionality authoring systems. In this section, we consider three representative examples.

Director [Macromedia 1997] is a commercial system designed for creating animation-based presentations. The initial authoring interface is that of a presentation timeline, in which media assets are directly scheduled. (Note that for actual animation and interaction, the timeline paradigm is not used; users are expected to use a proprietary scripting language instead.) Graphics, text, audio and video media items can be placed on a timeline, or score as it is termed in the system. As is common in timeline systems, only physical assets can be specified, since the temporal characteristics of the individual asset dominates the authoring paradigm. Director allows only direct temporal composition via a timeline that is divided into discrete time intervals, called frames. The rate of play of the frames is variable and is determined by the current rate of play which can be changed at any frame. Constraint-based synchronization or a-temporal composition, however, is not supported. The layout model is explicit via a a spatial position in each frame, and the author can describe a path for the media item to follow through a series of frames. The timeline has a number of associated tracks, where, apart from a number of effects tracks, any media item can be placed on any track. Asynchronous interaction is supported via jumps to other parts of the timeline using a *goto frame* command in the scripting language, but not on the timeline directly. Other forms of linking are also not supported directly although each frame, media asset or anchor within a media asset can have an associated script that is executed when the end-user interacts with its associated object. Adaptive content is not supported via the timeline. Performance modelling is integrated into a publishing step. Recent versions of director provide support for CD-ROMs and for network-based delivery via *Shockwave*.

Integrator [Siochi et al. 1991] also applies a timeline model that represents the multimedia presentation as a set of tracks. While the main authoring metaphor is the timeline, composite objects can also be created, e.g. an image with a graphic overlay, or a slide show. A composite object can be placed on the control track of the timeline, and can be opened to view the layout of objects on its own timeline. The composite object appears as one object on the timeline, which makes it difficult to get an overview of all the objects making up the presentation. Also, time dependencies between objects at different levels of the hierarchy are impossible to specify. While the timeline is used to represent the parallel nature of multimedia applications, several flow operations can be added to the control track of the timeline, including iteration and conditionally branching constructs. These operations add to the power of the specification language, but make the visualization of the presentation on the timeline difficult to interpret. Transitions can be specified in the system. These are associated with media item events rather than being separate events themselves, and can occur at the beginning of an event (e.g. fade up from black), at the end of an event, or can join two events (e.g. a video dissolves to another video). In general, asynchronous composition is not supported, nor is adaptive content. Publishing is restricted to an internal format.

LimSee2 [WAM 2003] is an example of a multi-view approach to authoring, but in which the timeline paradigm dominates the synchronization control process. LimSee2 contains many of the constraint-related features of Madeus (discussed under the structured paradigm), but adds additional support for direct layout and constrained timing. LimSee2 uses physical media object selection, and supports both direct and constraint-based temporal scheduling. It provides some support for grouping media assets into abstract synchronization containers, but it does not provide support for a-temporal composition. The layout model is explicit and is based on a direct manipulation of objects in screen space. There is no visual relationship of layout constraints on the timeline. LimeSee2 does not support asynchronous interaction or adaptive content. It also does not support a performance model. Publication is to a subset of the SMIL 2.0 language.

3.2.3 Advantages and disadvantages of the timeline-based paradigm. The timeline authoring paradigm models the presentation as a collection of assets ordered over time. It uses a time axis as the main method of organizing the (temporal) positioning of media items in the presentation. It is visualized as a line with marked-off time intervals. The advantage of this approach, which is often used within continuous media asset editors (audio and video) is that the start times and durations of the media items in the presentation are displayed explicitly, and in principle can also be manipulated directly. A further advantage of the timeline is that temporal synchronization constraints, where they exist, can be shown and in principle manipulated directly. These constraints can be between media assets or between assets and the timeline.

In spite of its popularity, the timeline paradigm has a number of serious limitations. Since the temporal dimension dominates, the paradigm only provides useful feedback when the temporal characteristics of continuous media assets are known or can be explicitly specified. Conditional activation or asynchronous behavior is nearly impossible to model. Synchronization conditions cannot be expressed directly between (parts of) the media items themselves; as a result, if other durations are changed only the author (and not the authoring system) can resolve the specified constraints. Such flexibility is required when, for example, a video sequence is shortened or lengthened and the corresponding subtitles must stay synchronized with the correct parts of the video.

For similar reasons, a-temporal composition or adaptive content cannot be specified directly via the timeline. Where links are specified, these are via scripts associated with an object being displayed on the screen. The script defines the destination of the link and may also contain some sort of transition information, such as dissolve to next scene in 2 seconds.

Since no high-level ordering of media assets is supported, spatial layout is specified by assigning positional attributes to the media objects. This can be done by positioning the item where it should appear or by specifying the *x* and *y* positions over time. Neither method is specific to the timeline-based paradigm. It is difficult to get an overview of the position of the object compared with other objects and over time. Although no overview is available, the timeline does provide the author with easy access to a screen view from any point along the timeline.

The main advantage of the timeline model for multimedia is that, for non-interactive, non-adaptive presentations, the timeline metaphor is easily understood. For restricted presentations (with only a few objects) it provides a compelling initial view of the temporal relationships, but for more complex presentations the lack of logical structuring of components provides a lack of semantic focus. Also scene breaks, or any overview of the narrative, need to be recognized implicitly, for example, by an abrupt change in the objects on the timeline. Because scenes are not represented explicitly it is also not possible to create synchronization constraints in relation to a scene. Control flow can be added to a presentation as an object on the timeline or as an external script, but its effects cannot be visualized using the timeline.

3.3 Graph-Based Paradigm

The graph-based paradigm includes systems that use some form of (possibly) restricted directed graph model to characterize the activation and synchronization of multimedia assets. There are two dominant sub-paradigms: the flowchart model and the directed graph model. Of the latter type, various forms of timed Petri net have been a dominant paradigm.

3.3.1 General characteristics of the graph-based paradigm. A graph gives the author a visual representation of the components in a presentation and the logical sequence of ordering among those components. Authoring with a graph is similar to programming the presentation in a procedural way, but with an interface improved by icons for visualizing the actions that take place. The narrative of the presentation can be reflected in the nodes (and node substructures) used. The high-level order of displaying or removing objects and other events is shown, but time is not represented explicitly. The destinations of choice points are given in terms of jumping to a new node set in the graph.

Figure 3a illustrates a typical activation flowchart; Figure 3b illustrates a typical directed graph model as used by Firefly [Buchanan and Zellweger 2005]. Note that various types of control primitives are available as nodes and that extensive timing annotation can be added to the graph.

3.3.2 Examples of the graph-based paradigm. Many graphical approaches have been developed to model multimedia presentations, but relatively few graph-based interfaces. This section considers four representative graph approaches. Note that since nearly all approaches use the graph to model control flow, only this aspect is discussed in the overviews.

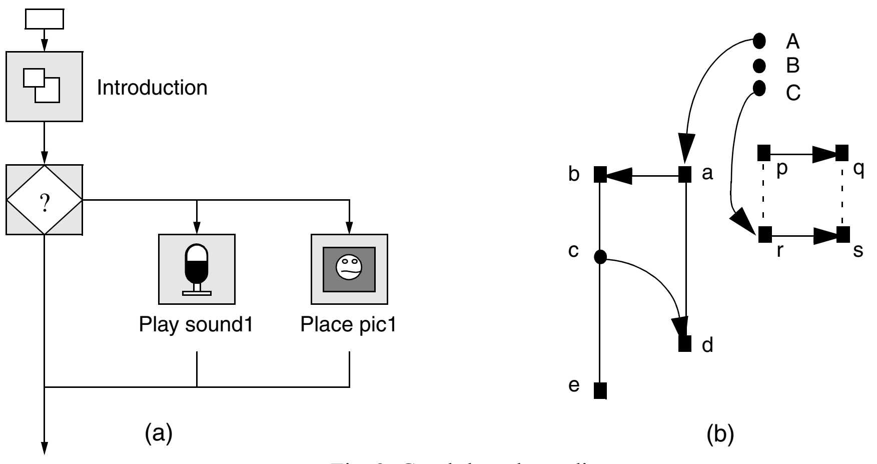

Fig. 3: Graph-based paradigm.

Authorware ([Kogel (Buford) 1994, Kogel (Buford) and Heines 1993, Macromedia 1997]) is a commercial system for creating interactive multimedia presentations for computer based training and kiosk applications. Icons representing actions are selected and incorporated into a flowchart defining the sequence of events in a presentation. Flowcharts can be grouped into subroutines and nested to arbitrary levels. This grouping is often necessary, since there is a limit to the display area for any one flowchart. The hierarchy of subroutines can be used by the author as an outline, or storyboard, for working on the presentation top down first by stating the sections in the presentation and then filling them in. The flowcharts remain procedural however, and there is no way of getting an overview (via a timeline) of which media items will be played on the screen when. Interactions, on the other hand, can be fairly complex and go far beyond links, which are implemented as "jump to there" commands.

Firefly [Buchanan and Zellweger 2005] is a temporal behavior specification of a document. (Firefly is more extensively described elsewhere in this issue.) Firefly combines a constraint language with a general timeline notion, both mapped onto a graph language. Unlike strict timeline systems, the graph notation allows unpredictable presentation timing to be described (these are the dotted lines in Figure 3b). Firefly also de-couples specification-time behavior from actual runtime behavior — the latter is computed by the system formatter based on the resolution of presentation time constraints. The advantages of Firefly-like approaches is that they provide a very rich framework for expressing constraint-based relationships. The disadvantage with these mechanisms is that there is only limited support at during authoring to manage the inherent complexity of non-trivial presentations: even the simplest document explodes in nodes and states.

Eventor [Eun et al. 1994] is an event editor that uses a divide and conquer approach to multimedia design, supported by three different views of the presentation: a temporal synchronizer, spatial synchronizer and user interaction builder. Eventor distinguishes timeline-based and flowchart paradigms (which they refer to as event-based), and aim to incorporate the advantages of both in an authoring system. Eventor is based on CCS, Calculus of Communicating Systems, a formal specification mechanism. This allows a formal specification of the behavior of the presentation, which can be used for checking, for example, syntactic correctness. In addition, they provide automatic aids for validating, for example, temporal constraints. Basic units of programming in Eventor are media items (called basic objects). Temporal synchronization can be specified among media items and composite objects. The temporal synchronizer visualizes the composite objects in a flowchart style structure. The spatial synchronizer allows the author to specify paths and scaling transformations by demonstration which are tightly coupled to temporal synchronizations.

Timed Petri Nets [Little and Ghafoor 1990] represent a class of formal solutions to specifying multimedia presentations based on a network of nodes that contain explicit synchronization predicates. Little's original work was not geared to modelling a particular media presentation, but to specify a set of temporal constraints that could be resolved by selecting appropriate media from a database of alternatives. The temporal specification was, in essence, a formal query mechanism for accessing stored content based on temporal and (presumably) semantic media descriptions. The general notion of timed Petri nets have been explored extensively in the research literature, but they have not resulted in any large scale authoring systems for general use. Part of the problem with Petri net approaches is that there is considerable overhead in specifying all of the constraints associated with the activation of a particular media object. The model is very flexible, but its flexibility often results in a loss of focus for users who are less interested in formal analysis than simply specifying a narrative path.

3.3.3 Advantages and disadvantages of the graph-based paradigm. The graph model paradigm supports logical and physical media asset specification and a clear control view: nodes are executed in turn, determined by the surrounding control structure. The advantage of the graph paradigm is that it provides an abstraction model that can incorporate arbitrarily powerful a-temporal interaction commands. For example, standard multiple choice question formats are easily supported. (These structures are very difficult to model on a timeline.) The paradigm also supports abstract grouping of nodes in the form of nested graphs, allowing the narrative structure of the presentation to be reflected by the different levels of graphs. Although this is a useful view of the general control flow, it is much less useful for specifying detailed synchronization of the relative activation, termination and persistence of objects, e.g., for background images that may have been displayed before a sub-graph was executed. This means that a sub-scene cannot be played independently since the state of the presentation is known only by playing the entire presentation.

The specification of spatial layout is orthogonal to the specification of control flow, and is usually defined in terms of individual asset attributes. This makes visualizing the coordinated control of logically-related media objects (such as adjunct content) difficult. Also, specifying user interaction based on areas within objects is difficult.

The main advantages of using flowchart-based graphs is their ability to support a-temporal composition. The main advantage of using directed graphs is that they can provide a framework for formal timing analysis of an presentation. The main disadvantage of both approaches is that complex temporal and synchronization relations are cumbersome to author and that the explosion of intermediate nodes means that modelling any presentation containing a significant number of media assets is time consuming and severely limited.

3.4 Script-Based Paradigm

In the structure-, timeline- and graph-based paradigms, a visual interface is presented that allows various aspects of a presentation to be manipulated in terms of declarative requirements. A script-based approach uses a procedural representation to specify the behavior of a presentation. This approach provides nearly unlimited control over synchronous and asynchronous behavior.

3.4.1 General characteristics of the script-based paradigm. A script-based system provides the author with a programming language where positions and timings of individual media items, and other events, can be specified. Authoring the presentation is programming. The destinations of choice points are given in terms of jumping to a new procedure. Figure 4 shows a typical activation script.

3.4.2 Examples of the script-based paradigm. There are as many examples of script-based control as there are programming languages. Several systems have been developed for explicit control of multimedia presentations and are reviewed in this section.

Videobook [Ogawa et al. 1990] incorporates a time-based, composite media data sequence with the hypertext node and link concept, allowing the construction of composite multimedia nodes. The system presents media items and hotspots according to a script specifying their presentation parameters, timing, and layout. The script is visualized as a three-dimensional display showing the layout of each object along a timeline. The system thus provides a low-level scripting language for the author to specify a presentation, which is then given a higher-level visualization along a timeline. The author is provided with some amount of structuring support, since each scene is defined in a separate script and scripts for scenes can contain nested sub-scenes. Synchronization of events is specified by giving the start time of an event with respect to the scene. Although the multimedia document has an underlying structure-based paradigm, the structure is interpreted from the author-defined script, rather than being used as the basis of editing and for generating the script.

```
set win=main_win
set cursor=wait
clear win
put background “pastel.pic”
put text “heading1.txt” at 10,0
put picture “gables.pic” at 20,0
put picture “logo.pic” at 40, 10
put text ”contents.txt” at 20,10
set cursor=active
```

Fig. 4: Script-based paradigm.

Harmony [Fujikawa et al. 1991], like Videobook, attempts to integrate dynamic media items into a hypertext system. Each object is considered as a node and there are links between nodes. Links are used to specify the timing relations between nodes. The notion of a composite object, or object group, is introduced, where, if a composite is the destination of a link, a message is broadcast to all members of the composite when the link is traversed. A scenario viewer displays the (derived) structure of a scenario.

Nsync [Bailey et al. 1998] combines the use of a scripting language with a run-time constraint manager to support the construction of interactive presentations in which user input determines the ultimate scheduling of media objects. The specification language uses a persistent form of the *if-then-else* construct (replacing *if* by the notion of *when*) to describe a series of constraint conditions that influence presentation navigation. The constraints are usually not temporal expressions, but variables that are used to determine presentation state; state changes allow a sequence of presentation components to be activated. Using Nsync, both synchronous mutliple choice interaction point and asynchronous interaction (that is, interaction that is enable across a specific temporal interval) can be supported. Unlike the AHM, Nsync does not rely on a link-based model to navigate across independent story fragements but uses an event model to trigger various substructures within the presentation. This has an advantage when performing run-time scheduling, although it is not clear if the lack of a structured story segmentation provides new maintenance problems similar to have a go-to based programming structure.

3.4.3 Advantages and disadvantages of the script-based paradigm. Script systems are in essence similar to the graph paradigm in terms of their flexibility and power of expression, and through the direct use of the scripting language they are likely to be more flexible. In terms of authoring support, however, they lack tools for viewing the procedure calls in any structured way. This limitation, in turn, leads to more likely program structure errors. Even if the narrative structure of the presentation has been reflected in the script structure, it remains difficult to manipulate at a high level. As with the graph paradigm, timing information for the presentation is embedded in the lines of code (with the exception of *command streams* where the lines of code have explicit times). Spatial layout information is also given via the lines of code. Since the structure, timing and spatial layout information is present in the code it is possible to derive a structure or time-based visualization. For example, in Videobook the space and time coordinates of items are shown in a 3D time-based representation and in Harmony the structure can be viewed. With no further support, this is a tedious, low-level method for specifying a multimedia presentation. It can, however, be the most flexible, since with a more general language the author is not restricted to the actions supplied by an authoring system for manipulating a document model.

While the script paradigm provides a compelling implementation model for detailed interaction, it has a substantial limitation in that it cannot be easily interpreted or transformed outside of the context of the scripting language. This means that every presentation is, essentially, provided only in a special-purpose final format encoding. If a programming language is not supported on a particular platform, or if the presentation needs to be dynamically altered to support adaptive content, this is usually impossible with a script-based approach.

Table II. Comparison of authoring paradigms.

	1		2				3		4			5		6	7	
	Media Assets		**Synchronization composition**				**Spatial Layout**		**Async. Interaction**			**Adaptive content**		**Perfor-mance**	**Publishing**	
	Abstract	**Physical**	**Temporal**	**Abstract**	**Constraint-based**	**A-temporal**	**Implicit**	**Explicit**	**Events**	**User interaction**	**Navigation**	**Adjunct**	**Replacement**	**Environment Modelling**	**Final Format**	**Flexible Format**
Structure	++	++	0	++	+	+	0	+	0	+	+	++	++	0	-	++
Timeline	--	++	++	-	+	--	+	-	--	--	-	--	--	+	+	0
Graph	++	++	-	+	0	++	-	-	+	++	+	+	+	0	--	-
Script	--	++	+	--	+	+	-	+	++	++	++	-	-	--	-	--

Rating Scale: ++ very good, + good, 0 neutral, - bad, -- very bad/not possible.

3.5 Summary

Table II provides a summary ranking of general authoring paradigm properties based on the issues developed in Section 2. For each paradigm, a ranking is provided that identifies the ability of that paradigm to support basic authoring issues. Note that individual systems within each paradigm may deviate from this ranking.

Structure based systems are good for viewing, editing and navigating the narrative structure of a presentation, allowing different levels of detail to be shown as appropriate. While not ideal for viewing the timing of the presentation directly, the structure can be used for editing the timing by allowing (some) timing relations to be derived from the structure. The structure itself gives an ordering of the display of media items. Layout information is specified per event, and an overview of the layout at a particular time is possible only by playing the presentation. Interaction, other than playing the presentation, is restricted to specifying and following links. Links, however, can in principle be defined using source and destination anchors and contexts.

Timeline based systems have no direct means of editing the narrative structure of the presentation directly, although it can be perceived and navigated as discontinuities of groups of objects along the timeline. The timeline is, however, the best way of showing when objects are displayed on the screen and visualizing synchronization relationships among events. It is not necessarily the best way of editing the timing, since although timing of individual objects can be changed, every object has to be manipulated individually, unless some form of structuring is present. Layout is specified per object per time unit, so an overview of all objects at a certain time is possible. The layout, and other properties of an event, can also be shown as a function of time. Interaction specification is even more restricted than in structure based systems, since links are often only jumps to some other point on the timeline.

The graph and script paradigms are comparable in power of expression, where editing and viewing the result tend to be more cumbersome with scripts. Reflecting the narrative structure in the structure of the graph, or script procedure calls, is possible but not compulsory. Similarly, navigation of the narrative is only as easy as the procedural correspondence maintained during editing. Timing information, on the other hand, cannot be shown (although as mentioned previously, this may be derived for viewing in a timeline). Layout information is specified per object, generally as part of the command to display the object. An overview of the layout at a particular time is possible only by playing the presentation. The flexibility of the interaction that can be specified is high, and the graph tools can help with its specification. Viewing the interaction remains a problem, with the only options being to run the presentation to check all the paths within the graph/script specification.

4. APPLYING AUTHORING PARADIGMS

The paradigms developed in Section 3 are not mutually exclusive, but reflect a difference in emphasis. There is no single approach that provides the ideal solution to an author's task and more often a combination is appropriate. In this section, we review the development of a presentation using the GR*i*NS authoring system. GR*i*NS, which grew out of our work on CMIFed and the facilities of the SMIL language, is an example of a structure-based authoring environment that provides multiple document views. The authoring paradigms incorporated in the system are structure and timeline editing. A graph model is used internally to determine presentation scheduling and performance analysis, but only the result is exposed to the author. Neither CMIFed or GR*i*NS support a scripting model (because of limitations in transforming the resulting encoding), but a SMIL presentation authored by GR*i*NS can be manipulated via a SMIL player (such as Ambulant [Bulterman et al. 2004]) that supports a SMIL DOM interface.

4.1 Presentation Model

The definition of a multimedia presentation will be illustrated by authoring an interactive tour of New York City, the site of the ACM's 2004 Multimedia Conference. The presentation is designed to represent a wide range of authoring issues.

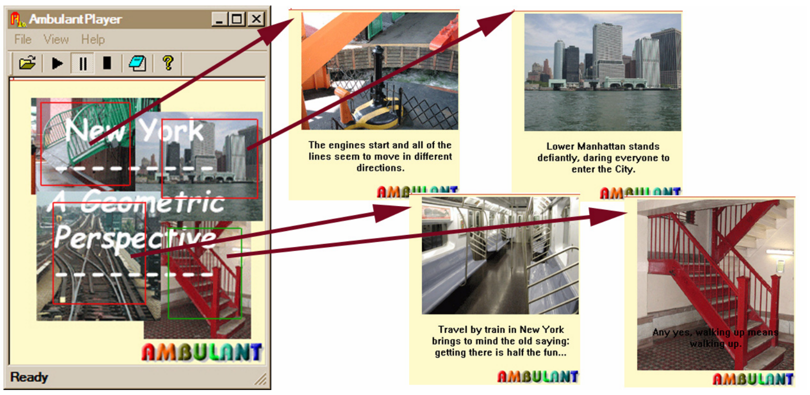
Ambulant Player
File View Help
New York
A Geometric Perspective
Ready
AMBULANT
The engines start and all of the lines seem to move in different directions.
Lower Manhattan stands defiantly, daring everyone to enter the City.
Travel by train in New York brings to mind the old saying: getting there is half the fun...
Any yes, walking up means walking up.

Fig. 5: Overview of the New York presentation.

The basic architecture of the presentation is shown in Figure 5. At left is a menu screen that contains a number of hotspot areas. (These are shown by the four thick black boxes, which are not visible to the viewer of the presentation.) When any of the hotspots are selected, a corresponding presentation fragment is played. The presentation has four such fragments: 1) a ride on the Staten Island Ferry; 2) an overview of the former Dutch colony of lower Manhattan; 3) a rider's guide to the New York subway; and 4) a visit to a quaint neighborhood in Queens. This architecture, which is similar to a DVD-style menu with independent chapters, is a prime example of a-temporal media composition: all four presentation fragments are logically active in parallel, but only one (or none) will be physically active at any one time.

Each of the four fragments contains a visual media component (image or video), an audio soundtrack and a set of captions. Some of the images contain anchors to related Web sites; selecting these will pause the presentation.

4.2 Authoring Views

Figure 6 provides a hierarchical tree representation (at a high level of abstraction) of the New York presentation. Using the composition primitives supported by SMIL 2.0, we see a menu node accompanied by the four presentation fragments described above, all of which are grouped within an a-temporal *exclusive* node. (The activation arcs are not shown.) Note that only the *Menu* and *Ferry* nodes are decomposed into the media components. This image illustrates a drawback of the hierarchical model: the visual complexity of the presentation can be significant, and you often cannot see a particular leaf node because of the forest of branches.

4.2.1 Viewing the basic presentation. Figure 7 shows an overview of the New York presentation as represented by the GR*i*NS editor. GR*i*NS is a multi-view editor that is based on a structured composition model. In the figure, we see a nested hierarchy, represented as a collection of embedded structure containers. The container labelled *New York Tour* contains the menu image shown in Figure 5 (note the interaction anchor symbol at top left of this node) and a complex node labelled *City Tour*. The *City Tour* node contains

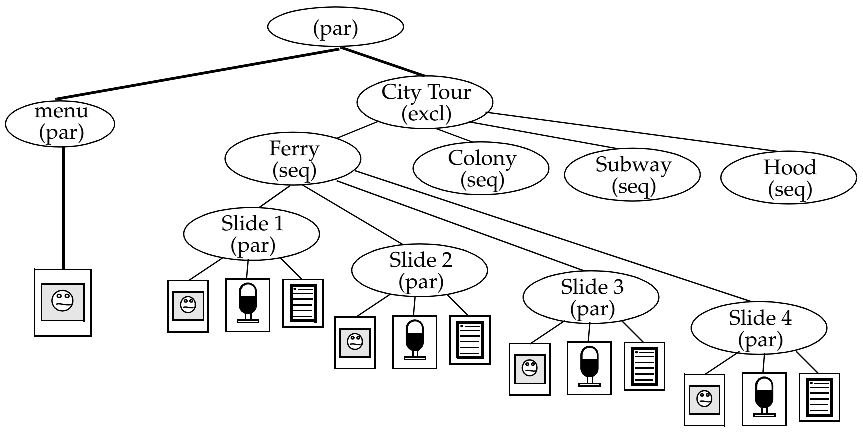

Fig. 6: Hierarchical decomposition of *New York*.

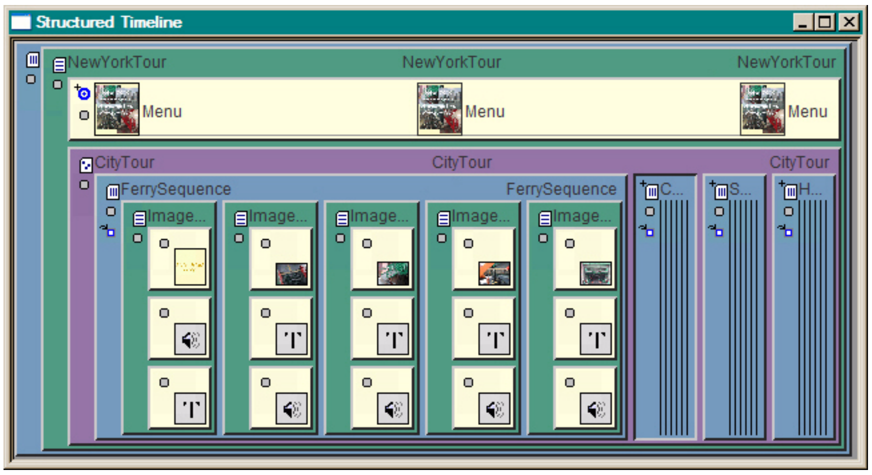

Fig. 7: GR*i*NS structured representation of the presentation in Figure 6.

four composite children, each of which hold one of the *ferry*, *colony*, *subway* and (*neighbor)hood* fragments. (Of these, only the *Ferry Sequence* is expanded in this image; the others are collapsed to save screen space in the editor.)

By default, GR*i*NS illustrates the basic structure of the presentation. Although an ordering can be determined (shown in the sequence of slides within the *Ferry Sequence*), there is no direct representation of the presentation timeline. Within any structure component, however, GR*i*NS provides a timeline view that illustrates the temporal composition of objects once that structure container is activated. This view is illustrated in Figure 8. Here we see a zoomed in view of the *Ferry Sequence* only, with a system-generated time axis shown at the top of the composition. The time axis is conventional, except that the room needed to draw the various sub-structure containers is not counted in the timescale. These *temporal discontinuities* are represented as dotted lines on the timescale.

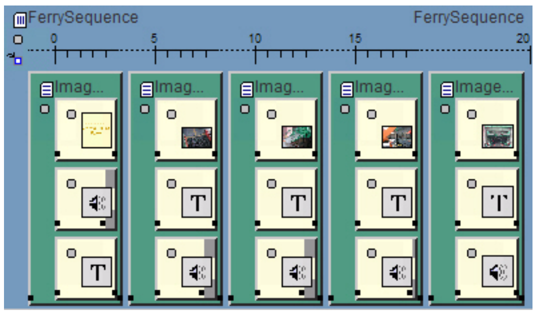

Fig. 8: Merging structural and temporal representations.

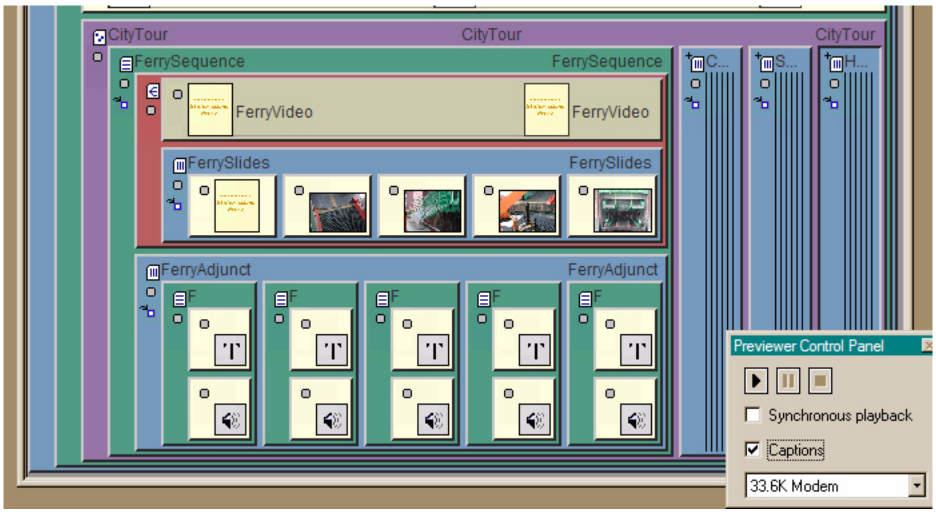

Fig. 9: Supporting both replacement and adjunct alternative content.

4.2.2 Supporting advanced presentation features. The presentation illustrated up to this point is essentially a simple slideshow. A more interesting case arises when we consider a more complex sub-structure for each of the content containers in the presentation. Assume, for example, that our presentation consists of either a video for each of the fragments or — if there is not enough presentation bandwidth — a collection of image extracts. Assume that we also want to make the text captions optional: only viewers who explicitly request captions will see them. This structure is represented within GR*i*NS as shown in Figure 9.

Figure 9 shows that a new structure container has been added within the *Ferry Sequence*. It contains a node labelled *FerryVideo* and a structure container labelled *FerrySlides*. Depending on the bitrate measured by the presentation player, either the video or the audio will be shown. In this image, the bitrate has been artificially set to a low modem value, as shown in the editor's preview control panel. As a result, the video object is darkened in the structure view and each of the slides are highlighted.

In Figure 9, the captions setting in the player control panel is turned on, meaning that the captions in each text block will be shown. (As illustrated in Figure 5, these captions are shown in a separate text whether the video or the slides are selected.) Figure 10 shows the authoring view when the captions are turned off. Note that they are not removed from the display, but are darkened to indicate that they are not currently active.

The timeline mapping of any structured presentation is computed dynamically by the GR*i*NS editor. Only the nodes that are active in the context of current display parameter settings are included in the timeline calculations. This illustrates that a timeline can represent one instance of a document's abstract structure, but not the structure itself.

4.2.3 Other editor features. The purpose of this example was to indicate how a structured paradigm can be integrated with other views to effectively represent the complex syn-

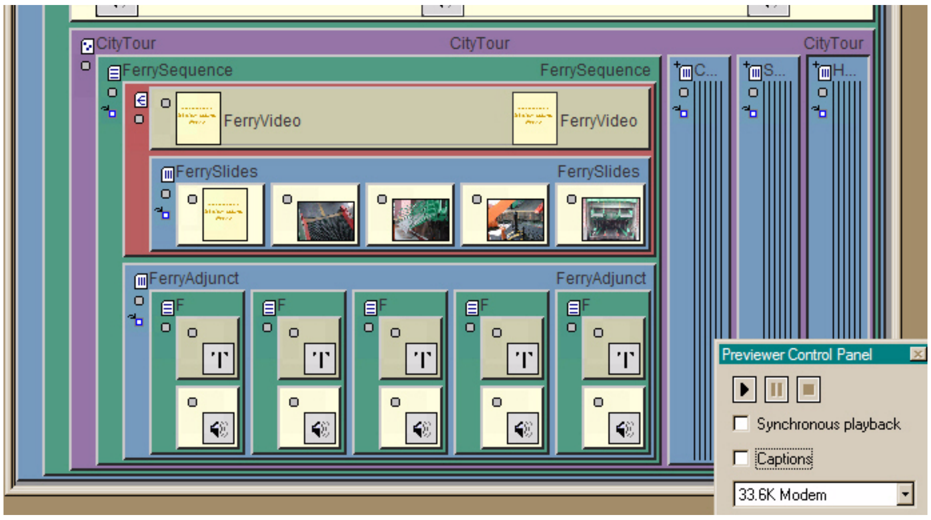

Fig. 10: Showing that adjunct alternative content is disabled.

chronization, activation and selection mechanisms that can be supported by modern multimedia encoding format. It was not intended to illustrate all features supported by GR*i*NS. For completeness, GR*i*NS also supports extensive layout and animation views, full hyperlinking and asynchronous composition facilities and extensive performance modelling for a wide range of device classes. It also support publication of documents to several desktop and mobile encodings.

4.3 Summary

The task of creating a multimedia presentation is multi-levelled and time-consuming. In order to support the author in this task we developed GR*i*NS for creating and editing interactive multimedia presentations. This environment supports a structured approach to authoring multimedia, allowing the author to view and manipulate the presentation in the way best suited to the current authoring task (e.g. specifying layout, or altering synchronization relations). The structured timeline view allows the editing and viewing of the presentation's hierarchical structure, from which high-level timing constraints are derived. The layout view shows abstract resource usage by the presentation and supports specification of precise timing constraints. The player allows the user to play parts, or all, of the multimedia presentation, and to activate and deactivate channels.

5. CONCLUSION

When the early work on CMIFed was carried out at the beginning of the 1990's the Web itself was in its infancy. Then, it was already a "cool thing" that computers were able to support multimedia directly, rather than using separate video disc devices to avoid the processing power required to display digital video. Now, at the beginning of the 2000's, transporting and manipulating large numbers of bits is much less of a problem and SMIL has become an integral part of the Web infrastructure for desktop and mobile devices. Unfortunately, we have not seen the hoped-for uptake of authoring systems for SMIL or for any other format. Instead, SMIL is used as a simple wrapper for single media assets, and little of its extensive support for a-temporal composition, layout, linking and adaptive content are being exploited. The reasons for this are not completely clear, but one is certainly the high complexity of authoring interactive multimedia in a more abstract, transformable manner. While there has been fairly wide acceptance for authoring individual media objects (such as audio, video, images and animations — often using very complex tools), there has been little interest in moving general multimedia presentations to a higher level of abstraction. Perhaps the most compelling reason for the limited creation of complex interactive hypermedia presentations is that there is often no short-term payback model to compensate for the increased effort of producing compelling presentations: multimedia remains a 'cost center' instead of a 'profit center' for nearly all applications.

As we see it there are two, not necessarily exclusive, potential futures for authoring interactive multimedia presentations. The first is to refine the human authoring interface to make the presentation creation process less cumbersome; this can include the integration of more explicit formal knowledge about the process [Falkovych et al. 2004] or systems in which authoring itself become an active but transparent activity that may be constrained for various application domains [Bulterman 2003]. The second is to remove the need for authoring per se and tie content aggregation to an end-user's request for information, where the creation of the presentation is carried out automatically; here, the goal is to satisfy the information need of the user without aspiring to create exceptional presentations.

Whatever the approach, it is clear that significant new work needs to be done to move multimedia beyond the creation of atomic objects (however rich the internal structure) so that, as with the text-based Web, the user can play a more important role in accessing, relating and navigating through the vast sources of multimedia media content available.

6. ACKNOWLEDGEMENTS

This paper is an update of our earlier work on structured multimedia authoring [Hardman et al. 1993]. The original paper was co-authored by Guido van Rossum, who provided much of the initial implementation support for CMIFed. Guido, who used CMIFed to develop and debug his Python language [Rossum and Drake 2003], has since moved across the ocean and concentrated his efforts on Python's developments. Other members of the multimedia group at CWI who contributed to the work on the CMIF editor are Jack Jansen, Robert van Liere, Sjoerd Mullender and Dik Winter. Many of the interface improvements and feature extensions were further developed for GR*i*NS by Kleanthis Kelanthous and Alain Uginet for Oratrix. We wish to thank Prof. Larry Rowe for his comments and suggestions. Finally, we thank Arnold Smeulders of the University of Amsterdam, who was Lynda Hardman's thesis advisor.

7. REFERENCES

ADOBE SYSTEMS 2002. Encore 1.5 DVD Editing Software. See: http://www.adobe.com/

BAECKER, R., ROSENTHAL, A.J., FRIEDLANDER, N., SMITH, E. AND COHEN, A. 1996. A Multimedia System for Authoring Motion Pictures. In Proceedings *ACM Multimedia '96*, Boston MA, Nov., 31 - 42.

BAILEY, B., KONSTAN, J., COOLEY, R. AND DEJONG, M. 1998. Nsync - A Toolkit for Building Interactive Multimedia Presentations. In Proceedings ACM Multimedia '98, Bristol, UK, 257-266.

BUCHANAN, M.C. AND ZELLWEGER, P.T. 2005. Automatic Temporal Layout Mechanisms Revisited, ACM Transactions on Multimedia Computing, Communications and Applications, 1(1). (Elsewhere in this issue.)

BULTERMAN, D.C.A., ROSSUM, G. VAN AND LIERE, R. VAN 1991. A Structure for Transportable, Dynamic Multimedia Documents. In Proc. *Summer USENIX Conference*, Nashville, Tennessee, 137-155.

BULTERMAN, D.C.A., HARDMAN, L., JANSEN, J. MULLENDER, K.S. AND RUTLEDGE, L. 1997. GRiNS: A Graphical Interface for SMIL, Proc. *WWW-7*, Brisbane, Australia.

BULTERMAN, D.C.A. 2003. Using SMIL to Encode Interactive, Peer-Level Multimedia Annotations. Proc. *ACM DocumentEngineering 2003*, Grenoble, France, November, 32-41.

BULTERMAN, D.C.A. AND RUTLEDGE, L. 2004. SMIL 2.0: Interactive Multimedia for Web and Mobile Devices, Springer-Verlag, Heidelberg. ISBN 3-540-20234-X.

BULTERMAN, D.C.A., JANSEN, J., KLEANTHIS, K., BLOM, K. AND BENDEN, D. 2004. The Ambulant SMIL 2.0 Open Source SMIL Player, Proc. *ACM Multimedia 2004*, New York, October, 492-495.

EUN, S., NO, E.S., KIM, H.C., YOON, H. AND MAENG, S.R. 1994. Eventor: An Authoring System for Interactive Multimedia Applications. *Multimedia Systems* 2: 129 - 140.

FALKOVYCH, K., WERNER, J. AND NACK, F. 2004. Semantic-Based Support for the Semi-Automatic Construction of Multimedia Presentations. Position paper *First International Workshop on Interaction Design and the Semantic Web*, NYC, May.

FUJIKAWA, K., SHIMOJO, S., MATSUURA, T., NISHIO, S. AND MIYAHARA, H. 1991. Multimedia Presentation System 'Harmony' with Temporal and Active Media. In: *Proc. of the Summer 1991 USENIX Conference*, June, Nashville, TN, 75 - 93.

HAMAKAWA, R. AND REKIMOTO, J. 1994. Object composition and playback models for handling multimedia data. *Multimedia Systems* 2: 26 - 35.

HARDMAN, L., ROSSUM, G. VAN AND BULTERMAN, D.C.A. 1993. Structured Multimedia Authoring. In Proceedings: *ACM Multimedia '93*, Anaheim CA, Aug., 283 - 289.

HARDMAN, L., BULTERMAN, D.C.A. AND ROSSUM, G. VAN 1994. The Amsterdam Hypermedia Model: Adding Time and Context to the Dexter Model. *Communications of the ACM*, 37 (2), Feb., 50 - 62.

HARDMAN, L. AND BULTERMAN, D.C.A., 1995. Authoring Support for Durable Interactive Multimedia Presentations. In: *State of The Art Report in Eurographics '95*, Maastricht, The Netherlands, August 29 - September. Available at http://ftp.cwi.nl/mmpapers/eg95.ps.gz.

HARDMAN, L. 1998. *Modelling and Authoring Hypermedia Documents*, Ph.D. Thesis, Univ. Amsterdam. ISBN 90-74795-93-5. Available at http://www.cwi.nl/~lynda/thesis/.

HARDMAN, L., SCHMITZ, P., OSSENBRUGGEN, J. VAN, KATE, W.R.T. TEN AND RUTLEDGE, L. 2000. The Link vs. the Event: Activating and Deactivating Elements in Time-Based Hypermedia. In: *New Review of Hypermedia and Multimedia* 6, pp. 89-109.

HODGES, M.E., SASNETT, R.M. AND ACKERMAN, M.S. 1989. A construction set for multimedia applications. *IEEE Software*, 6(1), Jan., 37 - 43.

IETF (INTERNET ENGINEERING TASK FORCE) 1995. Realtime Transport Protocol, http://www.cs.columbia.edu/~hgs/rtp/

IETF (INTERNET ENGINEERING TASK FORCE) 1996. Real-Time Streaming Protocol, http://www.cs.columbia.edu/~hgs/rtsp/

ISO (INTERNATIONAL STANDARDS ORGANIZATION) 1999. ISO/IEC 14496:1999 (MPEG-4), Available at: http://wwwam.HHI.DE/mpeg-video/standards/mpeg-4.htm

KOEGEL (BUFORD), J.F. AND HEINES, J.M. 1993. Improving Visual Programming Languages for Multimedia Authoring, Proc. *ED-MEDIA '93, World Conference on Educational Multimedia and Hypermedia*, Charlottsville, Virginia, June, 286 - 293.

KOEGEL (BUFORD), J.F. (ed.) 1994. *Multimedia Systems*. Addison-Wesley, New York, New York. ISBN 0-201-53258-1.

LITTLE, T.D.C. AND GHAFOOR, A. 1990. Multimedia Object Models for Synchronization and Databases. Proc. *Sixth International Conference on Data Engineering*, Los Angeles, CA, USA, pages 20-27.

MACROMEDIA. 1997. Authorware version 4. Director version 6. See: http://www.macromedia.com/.

OGAWA, R., HARADA, H. AND KANEKO, A. 1990. Scenario-based hypermedia: A model and a system. In proceedings: *ECHT '90*, Nov., INRIA France, 38 - 51.

ROSSUM, G. VAN, JANSEN,J., MULLENDER, K S. AND BULTERMAN, D.C.A. 1993. CMIFed: a presentation environment for portable hypermedia documents. In Proceedings: *ACM Multimedia '93*, Anaheim CA, Aug., 183 - 188.

ROSSUM, G. van AND DRAKE, F. 2003. An Introduction to Python, Network Theory Ltd. See also: http://www.python.org/

RUBIN, B. AND DAVENPORT, G. 1989. Structured content modeling for cinematic information. *SIGCHI Bulletin*, Oct., 21(2), 78 - 79.

SIOCHI, A., FOX, E.A., HIX, D., SCHWARTZ, E.E., NARASIMHAN, A. AND WAKE, W. 1991. The Integrator: A Prototype for Flexible Development of Interactive Digital Multimedia Applications. *Interactive Multimedia* 2(3), 5 - 26.

SOARES, L.F.G., RODRIGUES, R. AND MUCHALUAT SAADE, D. 2000. Modeling, authoring and formatting hypermedia documents in the HyperProp system, Multimedia Systems 8(2). 118-134.

TIEN, T.T. AND ROISIN, C. 2002. A Multimedia Model Based on Structured Media and Sub-Elements for Complex Multimedia Authoring and Presentation, *International Journal of Software Engineering and Knowledge Engineering*, Vol. 12, No. 5, Pp. 473-500.

WAM RESEARCH GROUP 2003. The LimSee2 Authoring System for SMIL. See: http://wam.inrialpes.fr/software/limsee2/ .

W3C (WORLD WIDE WEB CONSORTIUM) 2001. Synchronized Multimedia Integration Language, Version 2.0.*W3C Recommendation*. Editors: J. Ayers, D. C. A. Bulterman, A. Cohen, et al., http://www.w3.org/TR/Rec-smil20/

Figure Captions:

Table Captions: