GRiNS: A GRaphical INterface for Creating and Playing SMIL Documents

Abstract

The W3C working group on synchronized multimedia has developed a language for Web-based Multimedia presentations called SMIL: the Synchronized Multimedia Integration Language (pronounced 'smile'). This paper presents GRiNS, an authoring and presentation environment that can be used to create SMIL-compliant documents and to play SMIL documents created with GRiNS or by hand.

1 Introduction

While the World-Wide Web is generally seen as *the* embodiment of the infra-structure of today's information age, it currently cannot handle documents containing continuous media such as audio and video very elegantly. HTML documents cannot express the synchronization primitives required to coordinate independent pieces of time-based data, and the HTTP protocol cannot provide the streamed delivery of time-based media objects required for continuous media data. The development of Java extensions to HTML, known as *Dynamic HTML* [5], provide one approach to introducing synchronization support into Web documents. This approach has the advantage that the author is given all of the control offered by a programming language in defining interactions within a document; this is similar to the use of the scripting language Lingo in CD-ROM authoring packages like Director [6]. It has the disadvantage that defining even simple synchronization relationships becomes a relatively difficult task for the vast majority of Web users who have little or no programming skills.

In early 1997, SYMM—a W3C working group on SYnchronized MultiMedia—was established to study the definition of a declarative multimedia format for the Web [17]. In such a format, the control interactions required for multimedia applications are encoded in a text file as a structured set of object relations. A declarative specification is often easier to edit and maintain than a program-based specification, and it can potentially provide a greater degree of accessibility to the network infrastructure by reducing the amount of programming required for creating any particular presentation. The first system to propose such a format was CMIF [2],[3],[4]. Other more recent examples are MADEUS [7] and RTSL [13]. The SYMM working group ultimately developed SMIL, the Synchronized Multimedia Integration Language [18] (which is pronounced *smile*). (A brief summary of SMIL is presented in the following section.) In developing SMIL, the W3C SYMM group has restricted its attention to the development of the base language, without specifying any particular playback or authoring environment functionality.

This paper presents a new authoring and runtime environment called GRiNS: a GRaphical INterface for SMIL. GRiNS consists of an authoring interface and a runtime player which can be used together (or separately) to create/play SMIL-compliant hypermedia documents. The authoring part of GRiNS can be used to encode SMIL presentations for any SMIL compliant browser or stand-alone player; the GRiNS player can take any SMIL-compliant document and render it using a stand-alone player. GRiNS is based largely on earlier experience with the CMIFed authoring system and the CMIF encoding format, both of which strongly influenced the development of SMIL.

In section 3, we give an overview of the GRiNS authoring and playback interfaces. Section 4 then discusses the availability of the GRiNS environment as part of the CHAMELEON suite of authoring and presentation tools. We begin the paper with a review of the SMIL language and a brief discussion of typical applications and runtime environments for which SMIL was developed.

2 Declarative Web-Based Hypermedia

This section briefly reviews the nature of Web-based multi-/hypermedia documents and the basic philosophy of the SMIL language. We also briefly sketch the type of runtime protocol support that is expected for processing SMIL documents. Interested readers are encouraged to read the referenced documents for a complete description of each topic.

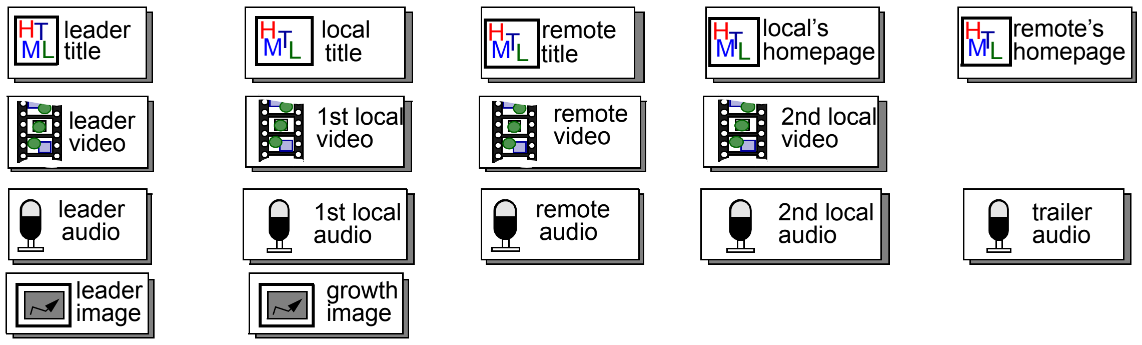


Figure 1: Media objects for use in (part of) a Web-based newscast.

Before we start, a short note on terminology: the scientific (and marketting) literature has not always been consistent in its uses of the terms *multimedia*, *hypermedia* or *continuous media*. In this document we will use the single term multimedia to mean a document or application that contains a mix of time-based and non-time-based media (such as audio/video and text/images, respectively) that need to be scheduled relative to one another. We include a fairly rich hypermedia model in this definition, in which links can be followed within one object, across a subset of the objects active at any one time, or across all the objects active at any time in the document.

2.1 A Typical Example SMIL Document

In order to focus our attention on the class of applications that this paper concerns itself with, we present an example of a generic Web application: that of a network newscast.[1] The basic premise of this newscast is a story on the explosive growth of the WWW. Several media objects have been defined that together make-up such a newscast. (Note that the selection of objects is arbitrary; we have selected a moderate level of complexity to illustrate the features of GRiNS.)

Let us assume that the objects shown in Figure 1 have been selected to make up the presentation. The newscast consists of an opening segment, a segment in which the local host gives background information, a segment in which a remote reporter gives an update, a segment in which the local host gives a wrap up, which leads into the trailing theme music. Finally, each correspondent also has a Web homepage that can be accessed from the story.

Figure 2 shows two views of the newscast example, taken at different times in the presentation. On the left side, we see a portion of the introduction of a story on the growth of the World-Wide Web. In this portion, the local host is describing how sales of authoring software are expected to rise sharply in the next six months. Figure 2(b) shows a point later in the presentation, when the local host is chatting with a remote correspondent in Los Angeles, who describes that many Hollywood stars are already planning their own audio/video homepages on the Web.

The ability to link to various homepages makes the semantic content of the document dynamic. As shown in Figure 3, the information content can be augmented during the story depending on the

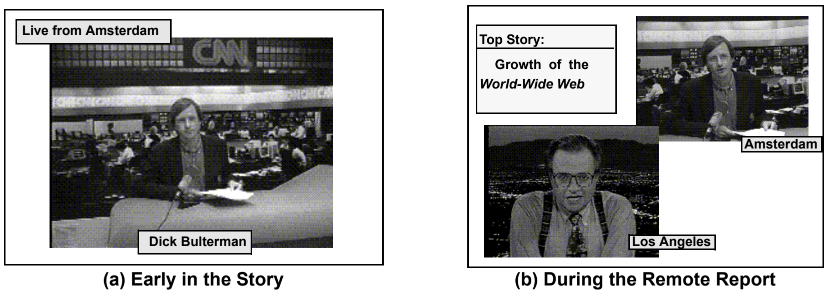


Figure 2: Two views of the Web newscast.

1.In 1991, as part of the first public CMIF paper, a network newscast was also used as a prototypical example. Sometimes the electronic world moves as slowly as the real one!

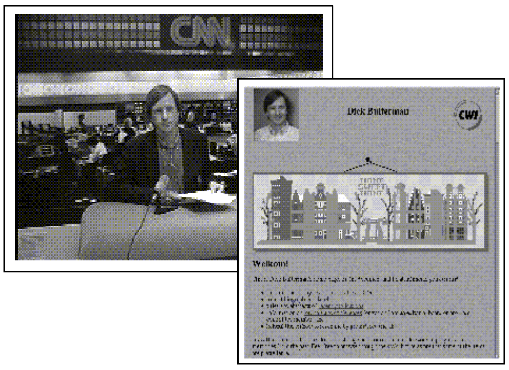


Figure 3: Augmenting information during the news.

viewer's interests. Links are not restricted to homepages, of course: any addressable object or document deemed relevant by the author should be available. Such behavior comes at a cost, however: we must be able to specify what happens to the base presentation when the link to the homepage is followed: should it pause, should it continue, or should it be replaced by the linked pages. (We discuss this in more detail below.)

A total description of the implementation details of the Web newscast is beyond the scope of this paper. We will, however, refer to it as a running example in the sections below.

2.2 A Brief SMIL Overview

SMIL is a declarative language for describing Web-based multimedia documents that can be played on a wide range of SMIL browsers. Such browsers may be stand-alone presentation systems that are tailored to a particular user community or they could be integrated into general purpose browsers. In SMIL-V1.0 (the version that we will consider in this paper), language primitives have been defined that allow for early experimentation and (relatively) easy implementation; this has been done to gain experience with the language while the protocol infrastructure required to optimally support SMIL-type documents is being developed and deployed.

The basic structure of SMIL is presented in Figure 4. Architecturally, SMIL is an integrating format. That is, it does not describe the contents of any part of a hypermedia presentation, but rather describes how various components are to be combined temporally and spatially to create a presentation. (It also defines how presentation resources can be managed and it allows anchors and links to be defined within the presentation.) Note that SMIL is not a replacement for individual formats (such as HTML for text, AIFF for audio or MPEG for video); it takes information objects encoded using these formats and combines them into a presentation.

Figure 5 contains a sample SMIL description of the newscast example sketched in Section 2.1. SMIL describes four fundamental aspects of a multimedia presentation:

- *temporal specifications*: primitives to encode the temporal structure of the application and the refinement of the (relative) start and end times of events;
- *spatial specifications*: primitives provided to support simple document layout;
- *alternative behavior specification*: primitives to express the various optional encodings within a

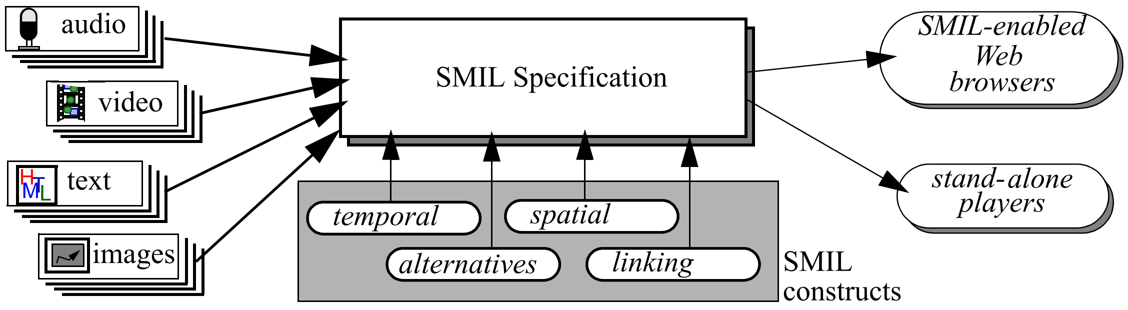


Figure 4: SMIL architecture.

```
 1 <smil sync="soft">
 2  <head>
 3   <layout type="text/smil-basic">
 4    <channel id="matise"/>
 5    <channel id="m_title" left="4%" top="4%" width="47%" height="22%"/>
 6    <channel id="v-main" left="52%" top="5%" width="45%" height="42%"/>
 7    <channel id="a_uk_main"/>
 8    <channel id="music"/>
 9    <channel id="pix" left="5%" top="28%" width="46%" height="34%"/>
10    <channel id="v-remote" left="3%" top="44%" width="46%" height="40%"/>
11    <channel id="r_uk"/>
12    <channel id="r_title" left="52%" top="58%" width="42%" height="22%"/>
13   </layout>
14  </head>
15  <body>
16   <seq id="WebGrowth">
17    <par id="opening_segment">
18     <text id="leader_title" channel="m_title" href="the.news/html/title.html"/>
19     <switch id="news_leader">
20      <video channel="v-main" href="mpeg/logo1.mpv"/>
21      <img channel="v-main" href="images/logo.gif"/>
22     </switch>
23     <audio id="leader_music" channel="music" href="audio/logo1.aiff"/>
24    </par>
25    <seq id="story_1_web_growth">
26     <par id="node_22">
27      <text id="dcab_intro" channel="m_title" href="html/dcab_intro.html" dur="8.000s"/>
28      <a href="archives-dcab.smi#1" show="new">
29       <video channel="v-main" href="mpeg/dcab1.mpv"/>
30      </a>
31      <seq id="sequences">
32       <par id="background_info">
33        <audio id="a_1a" channel="a_uk_main" href="audio/uk/dcab1.aiff" begin="0.900s"/>
34        <img id="web_growth" channel="pix" href="images/webgrowth.gif" dur="16.000s"
36           begin="id(a_1a)(begin)+8.600s"/>
35       </par>
36       <par id="live_link-up">
37        <a href="archives-larry.smi#1" show="pause">
38         <video id=r_larry" channel="v-remote" href="mpeg/larry.mpv"/>
39        </a>
40        <text id="remote_title" channel="r_title" href="html/r_title.html" dur="6.000s"
             begin="id(r_larry)(begin)+1.800s"/>
41        <audio id="remote_voiceover" channel="r_uk" href="audio/uk/larry1.aiff"
             begin="id(r_larry)(begin)+1.700s"/>
42       </par>
43      </seq>
44     </par>
45     <par id="node_56">
46      <video channel="v-main" href="dcab.zout.mpv"/>
47      <audio id="trailer_voiceover" channel="a_uk_main" href="audio/uk/dcab3.aiff"/>
48      <audio id="trailer" channel="music" href="audio/logo2.aiff"
           begin="id(wrap-up)(begin)+6.500s"/>
49     </par>
50    </seq>
51   </seq>
52  </body>
53 </smil>
```

Figure 5: A SMIL description of the Web Growth story. (Line numbers have been added for clarity.)

document based on systems or user requirements; and
- *hypermedia support*: mechanisms for linking parts of a presentation.

We describe how SMIL encodes these specifications in the following paragraphs.

2.2.1 SMIL *temporal specifications*

SMIL provides coarse-grain and fine-grain declarative temporal structuring of an application. These are placed between the `<body>` ... `<body/>` tags of a document (lines 15 thru 52 of Figure 5). SMIL also provides the general attribute `SYNC` (currently with values `HARD` and `SOFT`) that can be used over a whole document or a document part to indicate how strictly synchronization must be supported by the player. Line 1 of the example says, in effect: do your best to meet the specification, but keep going even if not all relationships can be met. Coarse grain temporal information is given in terms of two structuring elements, taken from CMIF:

- `<seq> ... </seq>`: A set of objects that occur in sequence (e.g., lines 16-51 and 31-43).
- `<par> ... </par>`: A collection of objects that occur in parallel (lines 17-24 or 36-44).

Elements defined within a `<seq>` group have the semantics that a successor is guaranteed to start after the completion of a predecessor element. Elements within a `<par>` group have the semantics that, by default, they all start at the same time. Once started, all elements are active for the time determined by their encoding or for an explicitly defined duration. Elements within a `<par>` group can also be defined to end at the same time, either based on the length of the longest or shortest component or on the end time of an explicit master element. Note that if objects within a `<par>` group are of unequal length, they will either start or end at different times, depending on the attributes of the group.

Fine grain synchronization control is specified in each of the object references through a number of timing control relationships:

- *explicit durations*: a `DUR="`*length*`"` attribute can be used to state the presentation time of the object (line 34);
- *absolute offsets*: the start time of an object can be given as an absolute offset from the start time of the enclosing structural element by using a `BEGIN="`*time*`"` attribute (line 33);
- *relative offsets*: the start time of an object can be given in terms of the start time of another sibling object using a `BEGIN="`*object_id* `+` *time*`"` attribute (line 41).

(Unless otherwise specified, all objects are displayed for their implicit durations—defined by the object encoding or the length of the enclosing `<par>` group.) The specification of a relative start time is a restricted version of CMIF's *sync_arcs* [4] to define fine-grain timing within a document. (The timing constructs with SMIL are also called sync_arcs.) At present, only explicit time offsets into objects are supported, but a natural extension is to allow *content markers*, which provide content-based tags into a media object.

2.2.2 Layout specifications

In order to guarantee interoperability among various players, SMIL-V1.0 supports basic primitives for layout control that must be supported on all SMIL players. This structure uses an indirect format, in which each media object reference contains the name of an associated *output channel* that describes how objects associated with that channel are to be presented. A separate layout resolution section in the SMIL `<head>` section maps these channels to output resources (screen space or audio). A example of a visible object reference is shown in line 29, which is resolved by the definition in line 6. Non-visible objects (such as audio) can also be assigned to output channels (see lines 33 and 7). Each player/ browser is responsible for mapping the logical output channels to physical devices. A priority attribute is being considered to aid in resolving conflicts during resource allocation.

2.2.3 Alternate Behavior Specifications

SMIL provides a means for defining alternate behavior within a document via the `<switch>` construct. The `<switch>` allows an author to specify a number of semantically equivalent encodings, one of which can be selected at runtime by the player/browser. This selection could take place based on profiles, user preference, or environmental characteristics. In lines 19-22 of Figure 5, a specification is given that says: play either the video or still image defined, depending on system or presentation constraints active at runtime. The switch statement is similar to that supported in RTSL. It is a simplified version of the *shadow channel* concept in CMIF.

2.2.4 Hypermedia and SMIL

HTML supports hypertext linking functionality via a straight-forward process: each document has a single focus (the browser window or frame) and anchors and links can be easily placed within the document text. When the link is followed, the source text is replaced by the destination. In an SMIL presentation, the situation is much more difficult. First, the location of a given anchor may move over time as the entity associated with the anchor moves (such as following a bouncing ball in a video)—and even if it does not move, it still may be visible for only part of the object's duration. Second, since SMIL is an integrating format, conflicts may arise on ownership of anchors and the semantics of following any given link.

In SMIL-V1.0, a pragmatic approach to linking is followed. Anchors and links within media objects are followed within the context of that media object. An anchor can also be defined at the SMIL level, which affects the presentation of the whole SMIL document. This effect depends on the value of a `SHOW` attribute, which may have values:

- `REPLACE:` the presentation of the destination resource replaces the complete, current presentation (this is the default);
- `NEW:` the presentation of the destination resource starts in a new context (perhaps a new window) not affecting the source presentation; or
- `PAUSE`: the link is followed and a new context is created, but the source context is suspended.

In our example, two links are defined: one on lines 28-30 and the other on lines 37-39. If the link associated with line 29 is followed, the current presentation's audio and video streams keep going, and a new SMIL presentation containing an HTML homepage is added. If the link at line 38 is followed, the current video object pauses while the homepage is show. (A question: what happens to the other active objects?)

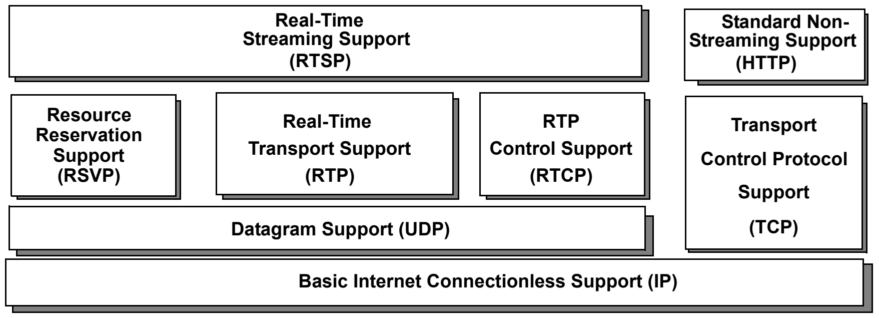
General support for hypermedia is a complex task. Interested readers are invited to study the approaches defined for the Amsterdam Hypermedia Model [3], which serves as the basis for the current SMIL proposal.

2.3 The SMIL Runtime Environment

Figure 5 shows the general relationship of the protocols that will be used to support SMIL presentations. IP, TCP and UDP are standard protocols from the Internet suite. HTTP and TCP are the standard protocols used for the current fetch-and-store policy on the Web.

In order to provide more control over how information is sent, new protocols are being devised to augment or replace the existing protocol stacks. RSVP [8],[9] is a resource reservation protocol that was designed for uni-directional multicast applications, but which also could be used for simplex unicast applications. While it seems unlikely that RSVP will play a major short-term role in public networks, it may be a useful tool within Intranets for guaranteed end-to-end bandwidth allocation. RTP [11],[12] and RTCP [10] are the *Real-time Transport Protocol* and the RTP *Control Protocol*, respectively. Note that RTP/RTCP do not guarantee real-time (or on-time) packet delivery. Instead, they provide a framework with which a given application can implement on-time delivery of data packets. RTP/RTCP can be used by the sender/receiver to adjust the manner in which data is buffered at either end of the transfer, or it can be used to dynamically select appropriate data encodings—but only if this is supported by all parties in the transfer pipeline. RTSP [14] is a streaming protocol that can be integrated with the protocols discussed above. (A streaming protocol is one that does not wait for an entire object to be delivered before rendering can begin.) As with RTP, RTSP does not provide a complete streaming solution; rather, it provides a framework in which applications-level streaming support can be implemented.

The nautre of the runtime environment is essential in understanding the functionality available


Real-Time Streaming Support (RTSP)
Standard Non-Streaming Support (HTTP)
Resource Reservation Support (RSVP)
Real-Time Transport Support (RTP)
RTP Control Support (RTCP)
Transport Control Protocol Support (TCP)
Datagram Support (UDP)
Basic Internet Connectionless Support (IP)
Figure 5: Types and relationships among network protocols for multimedia.

with SMIL. Most of the new network protocols are building blocks that must be tailored by the player and the server. Different policies will need to evolve to handle different types of data, or different QoS levels within one type. It will probably take years to settle these issues for the general case, which means that some incompatibility is inherent in using SMIL in a heterogeneous environment. While this may limit the 'user experience' initially, the very existence of SMIL could serve as a catalyst within the research and development community for addressing data control in terms of a common document specification.

3 GRiNS: Authoring and Presentation for SMIL

GRiNS is an authoring and presentation system for SMIL documents. It is a part of the CHAMELEON multimedia document processing suite, shown in Figure 6. Where the CHAMELEON suite provides tools for the authoring of adaptive documents—that is, documents that can be converted to a variety of presentation formats based on the (dynamic) needs of the document authors—GRiNS provides dedicated tools for creating and presenting SMIL documents. Both GRiNS and CHAMELEON are based on technology developed for CWI's CMIF environment.

The GRiNS authoring tool is based on a structured approach to document creation [2]. The GRiNS presentation tool provides a flexible presentation interface that supports all of the semantics of the SMIL V1.0 specification. In the following sections, we give examples of how GRiNS can be used to create the WebNews example. We then discuss some of the issues involved in providing support for the GRiNS presenter.

3.1 The GRiNS Authoring Environment

The GRiNS authoring environment supports creation of a SMIL document in terms of three views:

- the *logical structure view*, where coarse-grain definition of the temporal structure takes place;
- the *virtual timeline view*, where fine-grain temporal interactions are defined; and
- the *playout view*, where layout projections are defined.

Hyperlink definition and specification of alternative data objects can occur in either the logical structure view or the virtual timeline view.

3.1.1 The Logical Structure View

If we were to define the Web Growth story in terms of its over-all structure, we would wind up with a representation similar to that in Figure 7. Here we see that the story starts with a opening sequence (containing a logo and a title), and ends with a closing jingle. In between is the "meat" of the story. It contains an introduction by the local host, followed by a report by the remote correspondent, and then concluded by a wrap-up by the local host. This 'table of contents' view defines the basic structure of the story. It may be reusable.

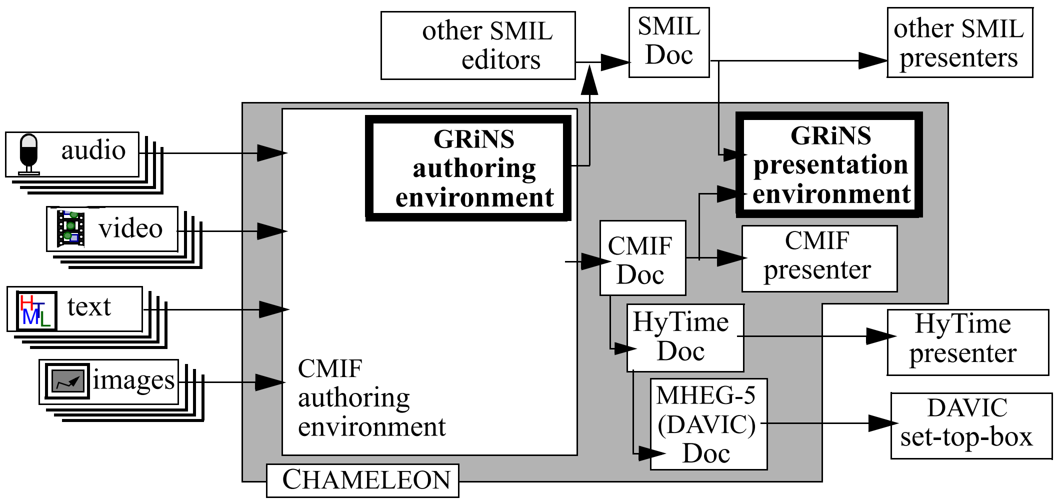


Figure 6: The CHAMELEON/GRiNS architecture.

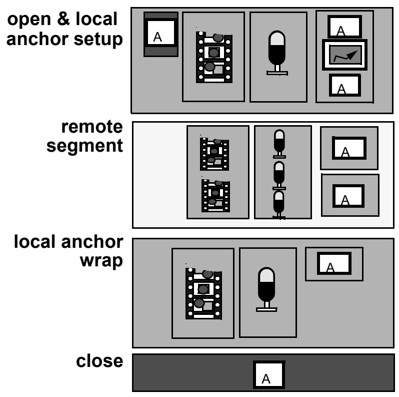
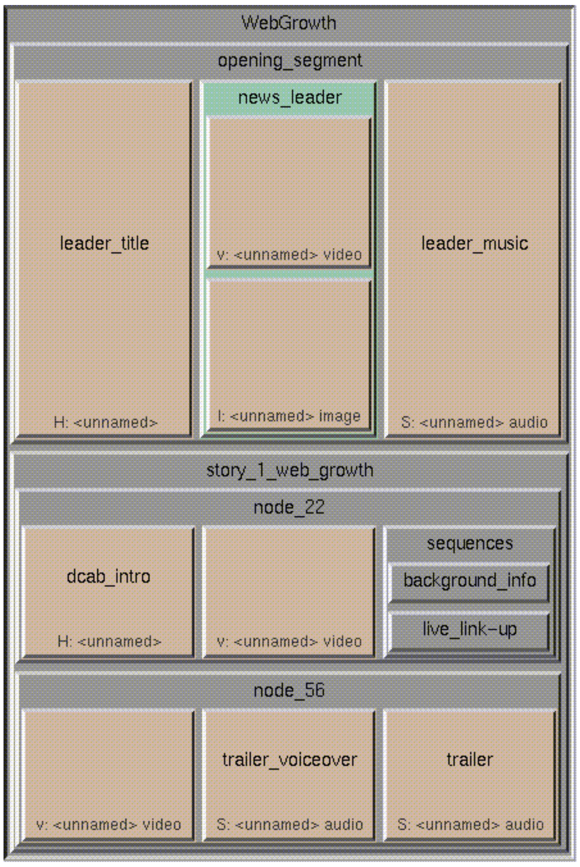


Figure 7: The structure of the Web Growth story. (Time flows from top to bottom.)

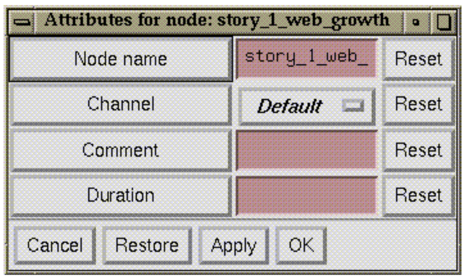
The GRiNS logical structure view allows an author to construct a SMIL document in terms of a similar nested hierarchy of media objects. The interface provides a scalable view of the document being created. Note that only structure is being edited here: much of this creation takes place before (or while) actual media objects have been created.

Figure 8(a) shows a part of the WebNews hierarchy in terms of the logical structure view. (The green box in the middle shows two alternates in the presentation: a video or a still image, one of which will be selected at runtime. Figure 8(b) shows a typical dialog box that is used to provide details of how a particular element is to be included in the presentation. The logical structure view has facilities cutting and pasting parts of the presentation, and it allows individual objects or sub-structured to be previewed without having to play the entire application.

During design and specification, the values of object attributes—including location, default or express duration or synchronization on composite objects—can be entered by the author. In practice,



(a)


Attributes for node: story_1_web_growth

Node name	story_1_web_	Reset
Channel	Default	Reset
Comment		Reset
Duration		Reset

(b)

Figure 8: The (a) logical structure of the WebNews and (b) the attributes associated with one object.

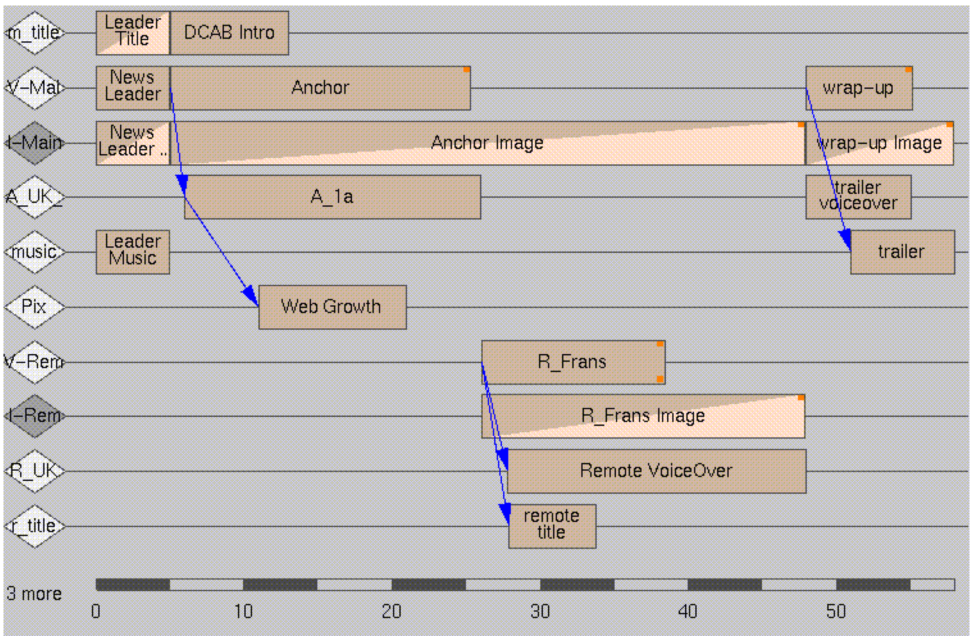


Figure 9: The virtual timeline view of the WebNews. (Time flows from left to right.)

duration of an object or a group will be based on the enclosing structure, will which be calculated automatically. Note that while the basic paradigm of the logical structure view is a hierarchical structure, the author can also specify loop constructs which give (sub-)parts of the presentation a cyclic character.

3.1.2 The Virtual Timeline View

The logical structure view is intended to represent the coarse timing relationships reflected in the `<par>` and `<seq>` constructs. While the attributes associated with an element (either an object or a composite structure definition) can be used to define more fine-grained relationships—such as DURATION or the desire to REPEAT an object during its activation period—these relationships are often difficult to visualize with only a logical structure view. For this reason, GRiNS also supports a timeline projection of an application. Unlike other timeline systems, which use a timeline as the initial view of the application, the GRiNS timeline is virtual: it displays the logical timing relationships as calculated from the logical structure view. This means that the user is not tied to a particular clock or frame rate, or to a particular architecture. (The actual timing relationships will only be known at execution time.) Note that the virtual timeline view is a generated projection, rather than a direct authoring interface. It is used as a visualization aid, since it reflects the view of the GRiNS scheduler on the behavior of the document.

Figure 9 shows a virtual timeline of the Web Growth story. Rather than illustrating structure, this view shows each of the components and their relative start and end times. The timeline view provides an insight into the actual temporal relationships among the elements in the presentation. Not only does it show the declared length and start times of objects whose duration or start offset is pre-defined, it also shows the *derived length* of objects who duration depends on the structure of the document. The timing view shows pre-defined durations as solid blocks and derived durations as blocks with a diagonal line.

When working with the virtual timeline view, the user can define exact start and end offsets within <par> groups using declarative mechanisms (the *sync_arc*), shown as an arrow in the figure. Sync arc are meant to provide declarative specifications of timing relationships which can be evaluated at runtime or by a scheduling pre-processor.

If changes are made to the application or to any of the attributes of the media objects, these are immediately reflected by in the virtual timeline view. As with the logical structure view, the user can select any group of objects and preview that part of the document. When the presentation view is active (see next section), the virtual timeline view also displays how the playout scheduler activates, arms and

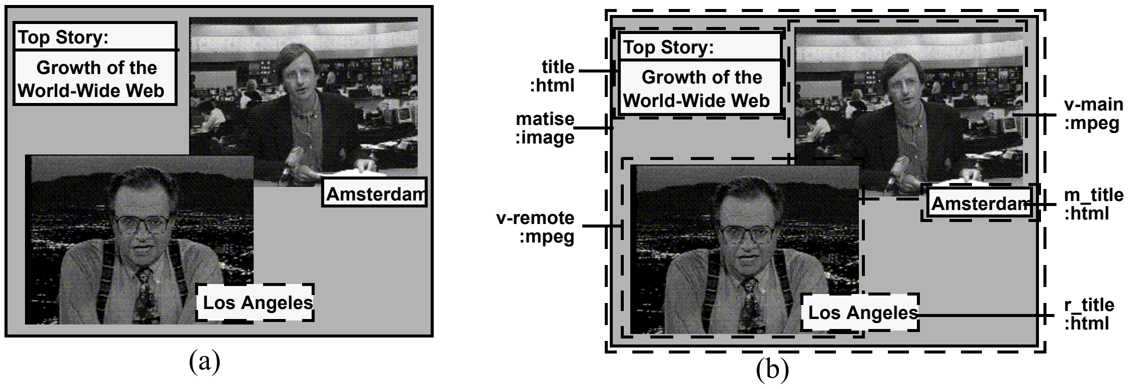


Figure 10: The (a) presentation view and (b) associated channel map.

pre-fetches data during the presentation.

Note that both the logical structure and the virtual timeline views are authoring views; they are not available when a document is viewed via the playout engine only. Both views isolate the user from the syntax of the SMIL language.

3.1.3 The Presentation View

The presentation view has two purposes: first, during authoring, it provides a WYSIWYG view of the document under development, and second, it provides a mechanism to interactively layout the output channels associated with the document. Figure 10(a) shows part of the presentation being developed, while Figure 10(b) illustrates the channel map associated with the part of the document. During playout, the values of the layout attributes can be dynamically changed if required.

The presentation view also activates the presentation controls of the playout engine. These are discussed in the next section.

3.2 The GRiNS Playout Engine

The GRiNS authoring interface is actually an augmented GRiNS playout engine. It is a playout engine that also provides a structure and timeline editor. GRiNS also provides a 'stand-alone' playout engine which can play any compliant SMIL-V1.0 document.

The architecture of the playout engine is show in Figure 11. It consists of a document interpreter that, guided by user interaction, steps thru a SMIL document. The document interacts with a link processor (which implements the `NEW, PAUSE, REPLACE` semantics of the link), and a channel scheduler which builds a Petri-net based representation of the document. Actual media display is handled by

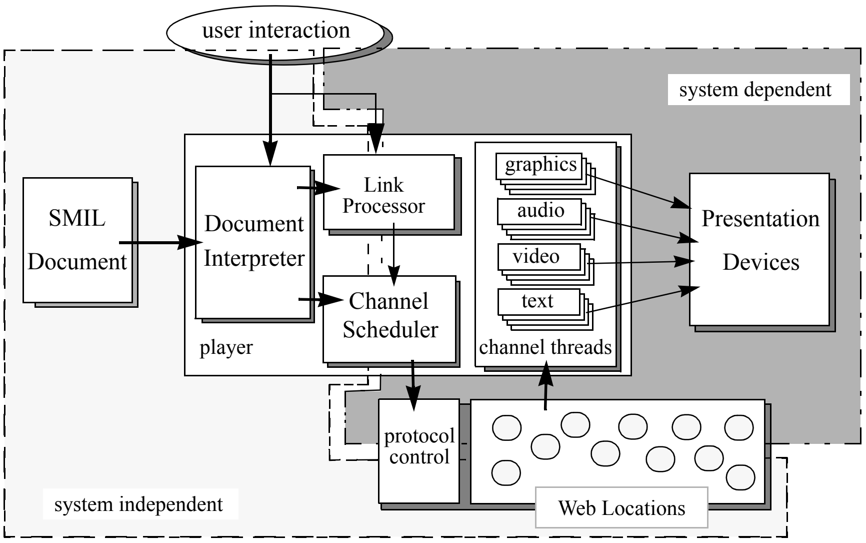


Figure 11: The GRiNS playout engine architecture.

a set of channel threads (or their architectural equivalent on various platforms), which interact with presentation devices. The channel scheduler and the various channel threads implement the application and media dependent parts of the RTP/RTCP protocol stacks.

As part of the general SMIL language, an author can specify alternative objects that can be scheduled based on resources available at the player, the server or the network. The GRiNS player also allows users to explicitly state which channels they wish to have active. This is useful for supporting multi-lingual applications or applications in which users are given an active choice on the formats displayed.

The GRiNS playout engine is implemented as a stand-alone application, and is not integrated directly into a full-function Web browser. This makes it suitable for separate distribution (such as on a CD-ROM), or for occasions when only limited ability to alter the presentation state of a document is required. (For example, if you are making a document to display a certain type of information, the stand-alone player can restrict the user's ability to enter random URLs and to browse other documents during presentation.) The use of a separate playout engine also allows us to experiment with new schedulers, protocols or data types much more easily than in a full-feature browser.

4 Current Status and Availability

The GRiNS authoring environment and playout engine have been implemented on all major presentation platforms (Windows-95/NT, Macintosh and UNIX). They are intended to provide a reference architecture for playing out SMIL documents and to integrate structured authoring concepts in SMIL document creation. While a full function description of all of the features of the environment are beyond the scope of this paper, we do have documentation available for interested parties. General information is available at GRiNS Web page (http://www.cwi.nl/Chameleon/GRiNS).

The present version of the GRiNS environment is implemented as a combination of Python-based machine independent and dependent code, and a collection of media output drivers. The quality and availability of drivers for any particular media type vary slightly across platform, but we a making a concerted effort to provide a high degree of cross-platform compatibility. (This is an on-going process.)

We feel the SMIL can play a significant role in providing a common document representation that can be interpreted by many players. Each player can make its own contribution in terms of quality of service or implementation efficiency. In this respect, we feel that our main focus will be on the development of GRiNS authoring tools that will work with anyone's playout engines. Until these engines become prevalent, we will also provide our own environment to assist in SMIL experimentation.

The GRiNS environment was developed in part within the ESPRIT-4 CHAMELEON project. This project is considering commercialization of some or all of the parts of the CHAMELEON architecture, including the GRiNS environment. Plans exist for the free or low-cost distribution of the playout environment to not-for-profit organizations or development partners. Availability of recent versions of the authoring interface is also being considered. Interested parties should contact the authors for more details.[1]

Acknowledgments

The work of the W3C's SYMM working group was coordinated by Philipp Hoschka of W3C/INRIA. A complete list of contributors is available at [17]. CWI's activity in the GRiNS project is funded in part through the ESPRIT-IV project CHAMELEON [1] of the European Union. Additional sources of funding have been the ACTS SEMPER [15] and the Telematics STEM [16] projects.

1. *Note to the reviewers*: discussions on availability of the environment and its documentation were not completed at the time this paper was written. It is expected that a distribution policy will be defined before the final paper version is required.

References

[1] CHAMELEON: An Authoring Environment for Adaptive Multimedia Presentations. ESPRIT-IV Project 20597. See `http://www.cwi.nl/Chameleon/`. See also `http://www.cwi.nl/~dcab/`.

[2] D.C.A. Bulterman, R. van Liere and G. van Rossum: *A Structure for Transportable, Dynamic Multimedia Documents* (with R. van Liere and G. van Rossum), Proceedings of 1991 Usenix Spring Conference on Multimedia Systems, Nashville, TN, 1991 pp. 137-155. See also `http://www.cwi.nl/~dcab/`.

[3] L. Hardman, D.C.A. Bulterman and G. van Rossum: *The Amsterdam Hypermedia Model: Adding Time and Context to the Dexter Model* CACM 37(2), Feb. 1994, pp 50-62. See also `http://www.cwi.nl/~dcab`.

[4] D.C.A. Bulterman: *Synchronization of Multi-Sourced Multimedia Data for Heterogeneous Target Systems*, in Network and Operating Systems Support for Digital Audio and Video (P. Venkat Rangan, Ed.), LNCS-712, Springer-Verlag, 1993, pp. 119-129. See also `http://www.cwi.nl/~dcab/`.

[5] W3C: *HTML 4.0: W3C's Next Version of HTML*. `http://www.w3.org/MarkUp/Cougar/`

[6] Macromedia, Inc.: *Director 6.0*. `http://www.macromedia.com/software/director/`

[7] C. Roisin, M. Jourdan, N. Layaïda and L. Sabry-Ismail: *Authoring Environment for Interactive Multimedia Documents,* Proc. MMM'97, Singapore. (Elsewhere in these proceedings.) See also: `http://opera.inrialpes.fr/OPERA/multimedia-eng.html`

[8] Zhang, L., Braden, R., Estrin, D., Herzog, S., and S. Jamin, *Resource ReSerVation Protocol (RSVP) -- Version 1 Functional Specification*, `ftp://ftp.ietf.org/internet-drafts/draft-ietf-rsvp-spec-16.ps`.

[9] Zhang, L., Deering, S., Estrin, D., Shenker, S., and D. Zappala, *RSVP: A NewResource ReSerVation Protocol*, `ftp://parcftp.xerox.com/pub/net-research/rsvp.ps.Z`, IEEE Network, Vol 7, no 5, pp 8-18, September 1993.

[10]D. Bettati, et al: Connection Establishment for Multi-Party Real-Time Communication, Proc. NOSSDAV '95, Durham NH 1995. See `http://www.cs.berkeley.edu/~amit/research/Postscript/estab-nossdav.ps`.

[11]H. Schulzrinne, S. Casner, R. Frederick, V. Jacobson, rfc1889: *RTP: A Transport Protocol for Real-Time Applications*, 01/25/1996. `http://www.cis.ohio-state.edu/htbin/rfc/rfc1889.html`.

[12]H. Schulzrinne, *RTP*, `http://www.cs.columbia.edu/~hgs/rtp/`.

[13]RTSL: *Real-Time Streaming Language*. See `http://www.real.com/server/intranet/index.html`

[14]H. Schulzrinne: *A real-time stream control protocol (RTSP)*, `http://www.cs.columbia.edu/~hgs/rtsp/draft/draft-ietf-mmusic-stream-00.txt` (11/26/96).

[15]SEMPER: *Secure Electronic Marketplace for Europe*, EU ACTS project 0042, 1995-1998. See `http://www.semper.org/`.

[16]STEM: *Sustainable Telematics for the Environment*, EU Telematics Project EN-1014, 1995-1996.

[17]W3C Synchronized Multimedia Working Group. `http://www.w3.org/AudioVideo/Group/`.

[18]W3C SMIL Draft Specification. `http://www.w3.org/AudioVideo/Group/WD-smil-format.html`.