Editor: Peiya Liu
Siemens Corporate Research

SMIL 2.0

Part 1: Overview, Concepts, and Structure

Dick C.A. Bulterman
Oratrix Development, The Netherlands

The World Wide Web Consortium's Synchronized Multimedia Integration Language format for encoding multimedia presentations for delivery over the Web is a little-known but widely used standard. First released in mid-1998, SMIL has been installed on approximately 200,000,000 desktops worldwide, primarily because of its adoption in RealPlayer G2, Quicktime 4.1, and Internet Explorer 5.5. In August 2001, the W3C released a significant update with SMIL 2.0.[1]

In a two-part report on SMIL 2.0, I will discuss the basics of SMIL 2.0 and compare its features with other formats. This article will focus on SMIL's basic concepts and structure. Part two, in the January–March 2002 issue, will look at detailed examples of SMIL 2.0, covering both simple and complex examples. It'll also contrast the facilities in SMIL 2.0 and MPEG-4.

Design goals

We can partition the goals for SMIL 2.0's design across three broad categories:

- *Extend SMIL 1.0's functionality*. The designers of SMIL's first version purposefully kept the language simple and relatively frills-free. SMIL 2.0 provides some desirable additions, including support for increased interaction, enhanced timing semantics, extended content control and layout facilities, and new animation and transitions primitives.

- *Maintain a declarative, XML format*. Although integrating multimedia content is its main function, SMIL 2.0 was developed to remain a fully declarative rather than a procedural format. Therefore, a SMIL description doesn't define how to implement a presentation but leaves the specification's implementation up to the SMIL player. Also, in keeping with SMIL's first version, SMIL 2.0 is fully XML-compliant.

- *Introduce a module-based structure*. SMIL 1.0 was a simple, 29-page specification, but SMIL 2.0 has more than 50 modules (in more than 500 pages) that users can partition into 13 functional groups. Using a module structure, we can integrate key aspects of SMIL 2.0 into other XML-based languages without requiring support for the entire SMIL 2.0 specification. Even before the SMIL 2.0 language specification's release, parts of SMIL have been integrated into several other XML languages (such as Scalable Vector Graphics), and we can expect more examples of module reuse.

As of this summer, several commercial versions of SMIL 2.0 were available. Oratrix released the first, its Grins SMIL 2.0 player (http://www.oratrix.com/GRiNS), in September 2000.[2] Major mass-market media companies such as RealNetworks and Microsoft have also announced support for all or some of SMIL 2.0's features. The Third Generation Partnership Project (3GPP) consortium, the standardization body coordinating the deployment of next-generation wireless devices, recommended that industry use SMIL 2.0 as the basis for wireless multimedia devices.

What it isn't (and is)

Often, multimedia presentations are characterized by their content rather than their composition. Because of this, there has been some confusion about what a SMIL presentation is and isn't. It's easy to summarize what SMIL 2.0 isn't:

- *SMIL 2.0 isn't a Flash substitute*. Flash (http://www.macromedia.com/flash) is a proprietary content media type that is primarily used for small animations. SMIL 2.0 isn't a content media type because it doesn't define any particular type of media (such as vector or raster images, videos, text, or audio data). Instead of media content, SMIL describes media compo-

1070-986X/01/$10.00 © 2001 IEEE

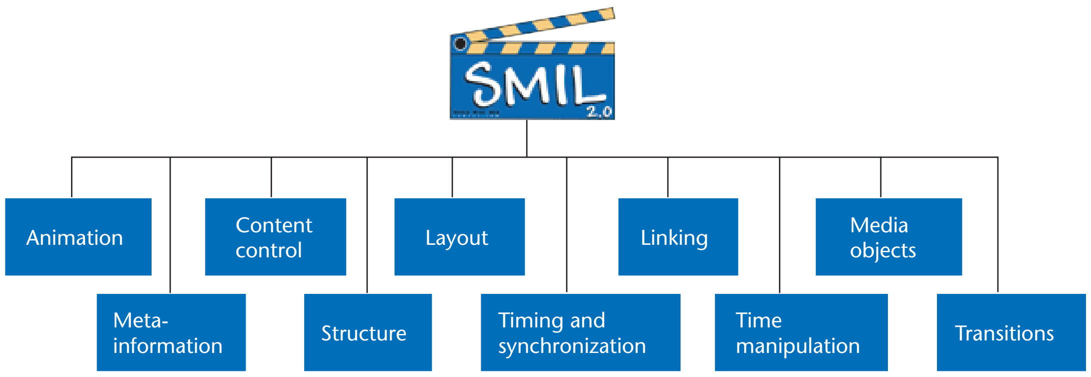


Figure 1. SMIL 2.0's functional grouping of module sets.

sition. A SMIL presentation can include Flash objects.

- *SMIL 2.0 isn't an MPEG-4 substitute.* MPEG-4 is a highly touted (but lightly implemented and deployed) format for describing a media object's content and interaction.[3] More precisely, MPEG-4 is a family of protocols that covers a wide range of media-related concerns but not a specific solution to any one class of media presentation. As we'll see, researchers have done considerable work to coordinate the development of SMIL 2.0 and MPEG-4 through the Extensible MPEG-4 Textual (XMT)[4] format specification.

- *SMIL 2.0 isn't a Dynamic HTML[5] substitute.* D-HTML was introduced as a way to introduce local time and animation effects into static HTML Web pages. Although some of the animation primitives in SMIL 2.0 resemble the functionality of some D-HTML uses, SMIL's scope is much broader than the local nature of D-HTML.

What is SMIL 2.0 then? It's a collection of XML elements and attributes that we can use to describe the temporal and spatial coordination of one or more media objects. SMIL 2.0 lets users define how independent media objects should be integrated during a presentation's lifetime. For example, that presentation may be delivered via a streaming server or played locally.

SMIL 2.0 defines 10 major functional groupings of elements and attributes (see Figure 1). Of these, the timing and synchronization grouping is the core of the SMIL specification. The functional groupings represent collections of individual SMIL 2.0 modules, each of which defines elements and attributes intended to address specific multimedia issues in a reusable manner. The number of modules per functional grouping varies from two to about 20. Generally, the more modules per grouping the finer each module's granularity.

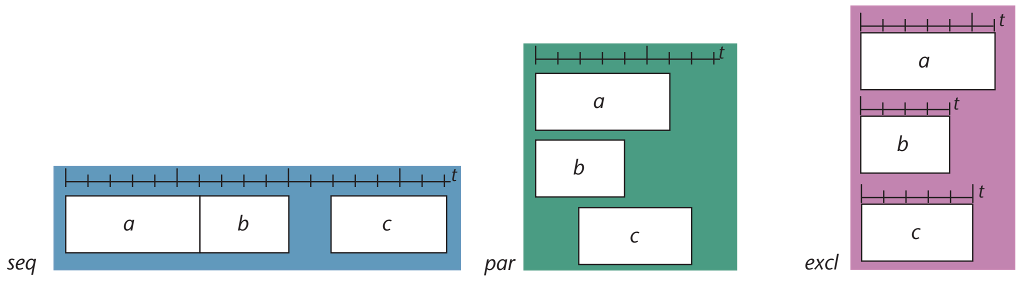
SMIL 2.0's developers intended for its modularization to facilitate the reuse of SMIL functionality in other XML languages and help define a number of SMIL profiles. Each profile provides a collection of modules that users can customize to its primary goal. The initial partitioning of SMIL 2.0 profiles includes SMIL 2.0 language, SMIL basic, and XHTML+SMIL. Of these, the SMIL 2.0 language and basic profiles completed the required review and implementation requirements of the W3C for the summer 2001 release of SMIL 2.0. The XHTML+SMIL profile,[6] which integrates SMIL timing, animation, and transitions modules (among others) into XHTML, wasn't completed and remains under development by the W3C. In addition to the three SMIL profiles, W3C also released a version of SMIL Animation in July 2001 for final review by its members.

Structure, timelines, and SMIL

A SMIL presentation is a structured composition of autonomous media objects. As Figure 2 (next page) shows, we can use three basic timing containers in a SMIL presentation:

- *seq*, or sequential time container. A seq container's children are rendered so that a successor child can't begin before its predecessor child completes. A successor child might have an additional start delay, but this delay can't resolve to be negative in relation to the end time of its predecessor. (See Figure 2a).

- *par*, or parallel time container. A par container's children are all rendered in parallel. In terms of a SMIL's timing model, this doesn't



```
<seq>
  <img id="a" dur="6s"
    begin="0s" src="..." />
  <img id="b" dur="4s"
    begin="0s" src="..." />
  <img id="c" dur="5s"
    begin="2s" src="..." />
</seq>
```

(a)

```
<par>
  <img id="a" dur="6s"
    begin="0s" src="..."/>
  <img id="b" dur="4s"
    begin="0s" src="..."/>
  <img id="c" dur="5s"
    begin="2s" src="..."/>
</par>
```

(b)

```
<excl>
  <img id="a" dur="6s" src="..."
    begin="0s;x.activateEvent" />
  <img id="b" dur="4s" src="..."
    begin="y.activateEvent" />
  <img id="c" dur="5s" src="..."
    begin="z.activateEvent" />
</excl>
```

(c)

Figure 2. SMIL 2.0 time containers. (a) A SMIL seq container plus a SMIL 2.0 code fragment. Note that begin times are relative to the predecessor's end. (b) A par container, with the same timing offsets. Note that begin times are relative to the containing par. (c) An excl container. Object a starts at time 0 of the excl and whenever object x (not shown) is activated. Objects b and c start only when objects y and z (not shown) are activated. Because the actual activation times depends on event activity, we can't use a common timeline to model the relationships among a, b, and c.

mean that they get rendered at the same time but that they share a common timebase defined by the par container, and any or all of the children can be active at any time that the parent par is active. The par is the most general SMIL time container. (See Figure 2b).

- *excl*, or exclusive time container. Only one of the children of an excl can be active at a time. The children's activation order depends on the begin attribute of the excl's children. Typically, each will have an event-based start condition (such as **`begin="button1.activateEvent"`**) that lets one of the children start on demand. The excl container is new in SMIL 2.0. (See Figure 2c).

We can nest the three basic time container types in a presentation hierarchy. That is, any child of a par, seq, or excl can be a simple media object or an embedded par, seq, or excl container. As in SMIL 1.0, the hierarchy can represent relatively static presentation timing. Introducing event-based activation or termination in SMIL 2.0 also lets users define a dynamic activation path. Note that most other multimedia formats only support a par-like semantic, not the seq or excl semantics.

A fundamental property of multimedia presentations is that they contain media objects requiring some notion of time for correct presentation. To model presentations, we often use a timeline metaphor. To understand SMIL 2.0, we can apply the timeline metaphor to containers that hold only static objects for which we know all timing information (including activation time, termination time, and object duration) at the time of authoring. For containers like the excl where the activation or termination time is unknown until the object actually starts, or for par's containing objects with unresolved begin or end times, the timeline metaphor is essentially useless. While this presents a challenge to authoring software designers, it provides a SMIL 2.0 document with unprecedented temporal flexibility and adaptability because the effective presentation timeline is determined at runtime, based on the individual media objects activated and their activation order.

Unlike media formats that force users to explicitly define when each object begins and ends relative to a single timeline, SMIL provides a logical timing framework in which we can use the structured relationship of objects to define most timing relationships among objects.[7] When writing a SMIL file, authors don't have to worry about the exact starting or ending times because these can be gleaned from the presentation's par, seq, and excl structure. In a SMIL specification, the objects' structured composition determines the timeline, rather than the timeline being the basic compositional unit.

The decoupling of timing resolution from a presentation specification lets, for example, a presentation's entire timing be changed based on the content's dynamic associations. Consider the following SMIL 2.0 code fragment:

```
<par endsync="select">
 <img id="btn_a" src="..." dur="10s" />
 <img id="btn_b" src="..." dur="5s" />
 <excl id="select">
   <text src=".../todays_txt.html"
       begin="btn_a.activeEvent"
       dur="25s"/>
   <video src=".../todays_video.mpg"
       begin="btn_b.activeEvent" />
 </excl>
 <audio src=".../todays_tune.mp3"
     repeat="indefinite"/>
</par>
```

The outer par contains two button images (`btn_a` and `btn_b`), an audio object reference, and an excl container (named `select`). The excl contains text and video object references. Note that each of the media object references is indirect: they point to external data that can change on a hourly, daily, or weekly basis.

The button images display for 10 and 5 seconds, respectively, when the par starts. The background music also begins and repeats indefinitely—that is, it repeats until the containing par ends. Because the par contains the directive `endsync="select"`, it will end when the `select` object ends. The `select` object ends when either the text or the video object ends, depending on which one the presentation viewer activates. (If the viewer selects `btn_a`, the text object will show for 25 seconds; if the viewer selects `btn_b`, the video object will play for that object's intrinsic duration, whatever that is.) All the while, the background music keeps repeating.

SMIL 2.0 timing and synchronization

The timing and synchronization functional group represents the core of SMIL 2.0 functionality. The group consists of 19 modules, each of which defines a collection of XML elements and attributes that control some timing aspect.

We've already discussed the three basic SMIL 2.0 timing elements seq, par, and excl. Each of these elements form parent timing containers in which we can place media objects or other timing containers. For any content within a container (whether a media object or structure container), these are the primary issues:

- When does the element begin?
- How long is it active?
- What happens to it when it's no longer active?
- Other than strict temporal, are there other conditions that cause an element to end?

We answer these questions by specifying a set of attribute values on either the parent time container or any of its children. SMIL 2.0 has an extensive set of attributes to control timing, most of which carry sane defaults so that we can easily accomplish basic timing and synchronization operations.

Figure 3 gives the basic collection of timing and synchronization attributes. Note that not all the SMIL 2.0 profiles support all the attributes, and the syntax of defining and setting the attribute values can vary, but the core functionality of timing, extended activation control, object persistence, and repeating element control are reasonably universal. A design objective of SMIL 2.0 was that each attribute should have a well-defined semantic that remains constant across profiles.

Begin, end, and dur

The begin and end attributes are similar in terms of syntax and semantics. The primary differences between these attributes in SMIL 1.0 and SMIL 2.0 is that SMIL 2.0 lets us specify multiple values for begin and end. (Figure 2 shows an example of this.) The first satisfied begin/end value will cause a corresponding element to start or end. It's possible to mix both scheduled and event-based activation/termination in one attribute. For example, `begin="3s;button.activateEvent"` will cause the associated element to start either at 3 seconds after the default time at which the element would otherwise be able to start, or when the `activateEvent` event associated with another element with an ID of button occurs (typically via a mouse click). (The default begin time varies with

Timing control
- `begin`
- `end`
- `dur`

Extended activation
- `endsync`
- `min`
- `max`

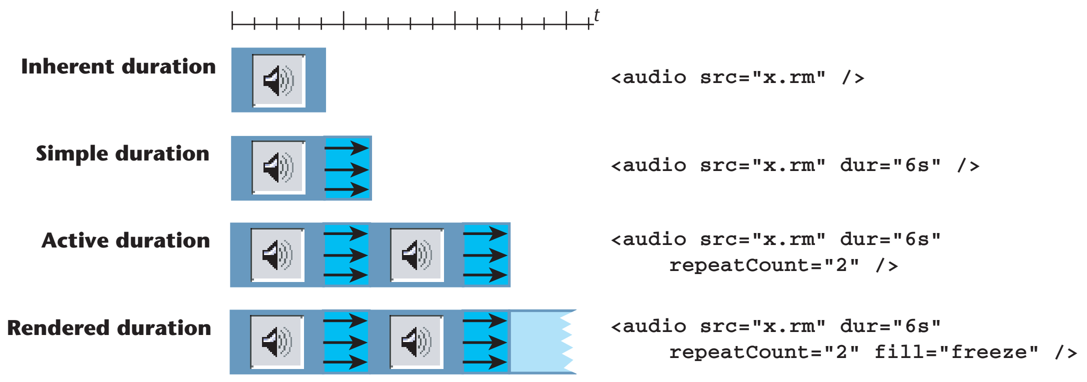
Object persistence
- `fill`

Repeating control
- `repeatCount`
- `repeatDur`

Synchronization
- `syncBehavior`
- `syncTolerance`
- `syncMaster`

XML timing integration
- `timeContainer`
- `timeAction`

Figure 3. Primary SMIL 2.0 timing and synchronization attributes.



```
<audio src="x.rm" />

<audio src="x.rm" dur="6s" />

<audio src="x.rm" dur="6s"
     repeatCount="2" />

<audio src="x.rm" dur="6s"
     repeatCount="2" fill="freeze" />
```

Figure 4. Various SMIL 2.0 duration concepts. (a) The inherent duration is the duration of the media object, if any. In this example, the inherent duration is 4 seconds. (b) The simple duration is the inherent duration modified by the* `dur` *attribute. (c) The active duration is the simple duration modified by the* `repeatCount` *(and* `repeatDur`*) attributes. (d) The perceived duration is an element's visual behavior after its active duration and before its parent time container ends.

the parent time container. For a par and excl, it's when the container starts. For a seq, it's when the previous element in the seq ends or the first child element starts.)

Once started, an element will have a certain duration, which we can determine in several ways (see Figure 4). Generally, so-called discrete media (media without an inherent notion of time, such as images or a page of text) have a default 0-second duration, and continuous media (media with an inherent notion of duration, such as an audio or video object) use that inherent duration as a default. (Some implementations of SMIL 1.0 used 5 seconds for the default duration of a discrete-media item. All SMIL 2.0 implementations should use the real default value of 0.)

An object with a 0-second duration wouldn't ordinarily be visible during a presentation. Such objects can be displayed if the `fill` attribute is set to a value of "freeze." A frozen object is displayed after the end of its active duration until its parent time container ends. For discrete media, this will be the image or text, for continuous media, it will be the last frame or sample. Consider this:

```
<par dur="10s">
  <img begin="3.5s" fill="freeze"
     src="..." />
</par>
```

It will display an image 3.5 seconds after the parent par begins. The object's active duration will be 0 seconds, but the object will remain visible for 6.5 seconds (until the parent par ends). Two `fill` values define a rendering duration that extends beyond the parent: transition and hold.

As in SMIL 1.0, if a par has multiple children active, it can specify that the entire par ends when the first child ends, the last child ends, or a named child ends. We do this with the **`endsync`** attribute. (The default is **`endsync="last"`**.)

SMIL 2.0 has two new attributes that provide extra duration control: **`min`** and **`max`**. We can use these attributes, which developed out of the SMIL 2.0 and MPEG-4 integration work, to define a lower or upper bound on the active duration, regardless of that element's timing characteristics.

What goes around, comes around

The default behavior of all elements is that they play once, for either an implicit or explicit duration. The SMIL 2.0 **`repeatCount`** attribute lets an iteration factor be defined that acts as a multiplier of the object's simple duration. We can use a special value **`indefinite`** to specify that an element repeat continually until the parent time container ends. (Although it's tempting to define this as forever, this isn't correct: the parent time container defines the context of **`indefinite`**.) The **`repeatDur`** attribute defines a duration for all the repeated iterations.

Synchronization behavior

In a perfect world, a SMIL 2.0 player would perfectly implement all the defined timing in a specification. Unfortunately, the world is imperfect and unpredictable. To provide some measure of control in the face of unpredictably, SMIL 2.0 provides three high-level synchronization control attributes:

- **`syncBehavior`** lets a presentation define whether there can be slippage in implementing the presentation's composite timeline,
- **`syncTolerance`** defines how much slip is allowed, and

- `syncMaster` lets a particular element become the master timebase against which all others are measured.

XML integration

When used in a native SMIL 2.0 document (one in which the outer XML tag is `<smil>`), the nature and meaning of various timing elements is clear. When integrating SMIL timing into other XML languages, we require a mechanism to identify timing containers. The SMIL 2.0 specification does this using the `timeContainer` and `timeAction` attributes.

Events and hyperlinking

In the normal course of processing, a SMIL 2.0 document's activation hierarchy determines the rendering of document elements. The user can influence the elements selected by using SMIL 2.0 events. The event architecture lets document components that are waiting be activated or terminated to actually start or stop. SMIL 2.0 allows several uses of events, but the most important new semantic in the language is the combination of events and the `begin` and `end` attributes. In further combination with the excl element, events provide a powerful mechanism for conditional content activation.

SMIL 2.0 also supports a rich hyperlinking architecture. Unlike links in HTML or XHTML, the fundamental concept of the SMIL link is that it models a temporal seek in a presentation. Rather than simply activating a target element, the target play state that is activated is identical to the state that the presentation would have been in if the target point had been arrived at naturally. (One exception is that all event-based activation is ignored.) This means that all nodes temporally between the link's source and destination must be evaluated to see if they would have contributed to the final target play state. The temporal seeking and activation facility allows polished presentation construction, but its implementation in the player isn't for the faint of heart.

Content control elements and attributes

One of the major innovations of SMIL is support for conditional content via the switch element:

```
<switch>
  <video src="..."
   systemBitrate="115200"/>
  <seq systemBitrate="57344">
    <img src="img1.png" dur="5s"/>
    ...
    <img src="img12.png" dur="9s"/>
  </seq>
  <text src="desc.html" dur="30s"/>
</switch>
```

In this fragment, a video object is rendered if the system bit rate (actual or configured) is set at 112 Kbytes or above. If the bit rate is 56 Kbytes or above, but below 112 Kbytes, a player shows a sequence of images instead. If no other element had been selected in the switch, a player shows a text object.

One of the SMIL 1.0 switch element's limitations was that it only allowed selection based on a predefined list of system attributes. SMIL 2.0 extends this notion with a user-defined set of test attributes: `custom test` attributes. The author can define the `custom test` attributes, and the user can select them directly or indirectly via the player. SMIL 2.0 also lets test attributes be used inline—that is, outside the switch element. Any media object reference containing a `system` or `custom test` attribute will be evaluated as if it were wrapped into a single-element switch.

To a first approximation, both the event mechanism and the content control facilities provide a means for dynamically selecting objects in a presentation. The basic difference between these facilities is that the event mechanism works on objects that the SMIL 2.0 scheduler recognizes, while the content control facility determines which object the scheduler gets to evaluate. The actual selection process associated with the conditional control primitives can be static or dynamic. That is, the selection can be done at parse time (when the document is loaded) or at each iteration through the document. The specific selection policy is a property of the SMIL player.

Transitions and animation

SMIL 2.0 significantly extends the facilities available for performing local operations on media objects in a document. Two of the most visible of these are support for transitions and animation. SMIL's transition support allows a set of basic transitions to be defined as part of the SMIL head element and then to instance one of the available transition types as an input or output transition on a media object. For example, consider the SMIL 2.0 fragment in Figure 5 (next page). We can apply transitions to all visual media or collections of media. Each transition can have certain timing properties (duration) and other transition-specific properties (direction).

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<smil xmlns="http://www.w3.org/SMIL20/Language">
  <head>
    <layout>
    ...
    </layout>
    <transition id="fade" type="fade" dur=î1sî/>
    <transition id="push" type="pushWipe" dur=î0.5sî/>
  </head>
  <body>
    ...
    <img src="..." transIn="fade"/>
    ...
    <video src="..." transOut="push"/>
    ...
  </body>
</smil>
```

Figure 5. Transitions architecture in SMIL 2.0. We can define the transitions in the head section and reference them in the body section as either input or output transitions.

Animation in SMIL 2.0 comes in two flavors. First, some animations apply to attributes and elements in the SMIL presentation. This includes animating the position of a rendering region or a background color. A second type of animation support is the SMIL 2.0 animation specification, which gives generalized temporal animation support for integration into XML languages.

Layout

Unlike HTML, which uses an indirect layout model via cascading style sheets, SMIL also supports a direct layout model for managing a presentation's visual and audio rendering space. In SMIL 2.0, the SMIL basic layout mechanism is supported with elements and attributes that lets users specify layout as a hierarchy of rendering regions and support multiple top-level presentation windows. Supporting hierarchical layout is especially important when integrating animation into a presentation. We can move content that is logically grouped together in concert by animating the common parent region's position. Multiple top-level windows let a single presentation segment control and render operations in different parts of the screen in a coordinated manner.

Another enhancement to SMIL 2.0 layout is the support for subregion positioning. This facility lets users place an object at a particular (X, Y) offset within a region or aligned to a registration point. (In SMIL 1.0, we had to place all objects at a region's top left corner.) The placement occurs in-line, as part of the media object reference. SMIL 2.0 also supports an alignment mechanism for content in regions. This lets us center a set of images of varying size at a specific point in a region. SMIL 2.0 also lets multiple objects be active in a region simultaneously, relaxing a restriction from SMIL 1.0.

Conclusion

There are several places to obtain SMIL 2.0 players. The W3C maintains a Web site with links to available players, editors, and other tools (see http://www.w3.org/AudioVideo/.) The facilities available in SMIL, combined with the increasing availability of connection bandwidth, provide authors with a new toolbox of presentation control primitives. The diversity of expected future delivery environments (such as broadband, conventional modem, and wireless) demand such a toolbox so that we can build reusable and tailored presentations.

In part 2 of this article, we'll look at how we can use SMIL 2.0's features to build various presentations for multiple target platforms, including Microsoft's IE-6, Real's new RealONE player, and Oratrix's Grins environment. **MM**

References

1. J. Ayers et al., *Synchronized Multimedia Integration Language (SMIL) 2.0*, World Wide Web Consortium Recommendation, Aug. 2001, http://www.w3.org/TR/2001/REC-smil20-20010807/.
2. D.C.A. Bulterman et al., "GRiNS: A GRaphical INterface for Creating and Playing SMIL Documents," *Proc. 7th Int'l World Wide Web Conf.* (WWW7), Elsevier Science Publishers, Amsterdam, 1998.
3. R. Koenen, ed., *ISO/IEC JTC1/SC29/WG11*, *Overview of the MPEG-4 Standard*, Int'l Organization for Standardization, Geneva, Mar. 2001.
4. M. Kim et al., "eXtensible MPEG-4 Textual Format," contribution to ISO-IEC JTC1/SC29/WG11 MPEG00/6110, Int'l Organization for Standardization, Geneva, May 2000.
5. *Dynamic HTML in Netscape Communicator*, Netscape Corp., http://developer.netscape.com/docs/manuals/communicator/dynhtml, 1997.
6. D. Newman, P. Schmitz, and A. Patterson, "XHTML+SMIL Profile," W3C Working Draft, World Wide Web Consortium, Aug. 2001, http://www.w3.org/TR/2001/WD-XHTMLplusSMIL-20010807/.
7. L. Hardman, D.C.A. Bulterman, and G. van Rossum, "Structured Multimedia Authoring," *Proc. ACM Multimedia 93*, ACM Press, New York, 1993, pp. 283-290.

Readers may contact Bulterman at Oratrix Development, Amsterdam, The Netherlands, email Dick.Bulterman@oratrix.com.

Contact Standards editor Peiya Liu, Siemens Corporate Research, 755 College Rd. East, Princeton, NJ 08540, email pliu@scr.siemens.com.