



# SMIL 2.0

## XML for Web Multimedia

Lloyd Rutledge • CWI, Amsterdam

On 7 August, the World Wide Web Consortium (W3C) released version 2.0 of Synchronized Multimedia Integration Language, or SMIL (pronounced “smile”). Three years ago, SMIL 1.0 introduced a basic foundation for Web multimedia and, as the sidebar “SMIL Support” (on page 83) describes, it quickly gained widespread use. With a specification document about 15 times as large as version 1.0, SMIL 2.0 builds on this foundation and marks an enormous step forward in multimedia functionality.

Although Web multimedia has long been obtainable with proprietary formats or Java programs, it’s been largely inaccessible to most Web authors and isolated from the Web’s technical framework. SMIL’s HTML-like syntax aims to do for multimedia what HTML did for hypertext: bring it into every living room, with an easy-to-author descriptive format that works with readily available cross-platform players. SMIL lets authors create simple multimedia simply and add more complex behavior

incrementally. But SMIL isn’t just HTML-like, it’s XML, which makes it part of the W3C’s family of XML-related standards including scalable vector graphics (SVG), cascading style sheets (CSS), XPointer, XSLT, namespaces, and XHTML.

SMIL’s features fall into five categories: media content, layout, timing, linking, and adaptivity. The latter brings altogether new features to the Web, letting authors adapt content to different market groups, user abilities, system configurations, and runtime system delays. In this tutorial, I’ll cover each feature category and its basic constructs using a simple SMIL presentation built with the SMIL 2.0 Language Profile, which is the flagship SMIL-defined language for multimedia browsers.

### SMIL 2.0 Features

Figures 1a and 1b show an example SMIL presentation, Fiets Amsterdam Tour. Fiets (pronounced “feets,” which is Dutch for “bicycle”) is an interactive multimedia tour of Amsterdam.

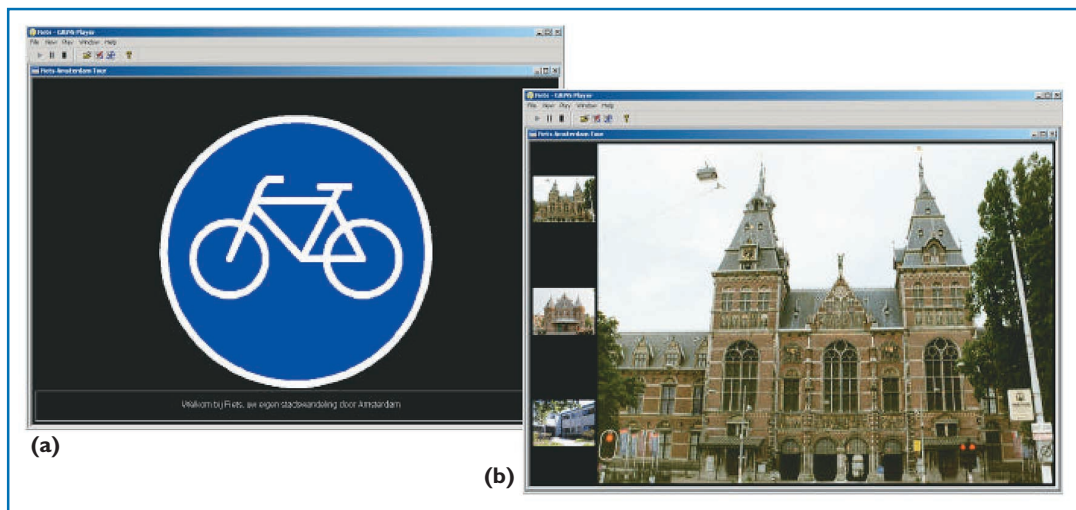


Figure 1. The Fiets Amsterdam Tour SMIL presentation. (a) Display of Fiets greeting section, set for Dutch captions, on GRiNS Player for SMIL 2.0. (b) Display of Fiets thumbnail section, on RealPlayer I.

Fiets starts with a splash screen logo (Figure 1a) and an audio introduction. The splash screen fades in at the beginning and fades out when the welcoming audio track is complete. Users can select from either English or Dutch. For hearing-impaired users, Fiets shows closed captions of the voice-over at the bottom of the display. Figure 1a fades to Figure 1b, which shows three thumbnail images of Amsterdam buildings. Users can click on these images, in any order, to see a corresponding full-sized image in the main display area. Figure 2 (next page) shows the SMIL code for the presentations in Figure 1a and 1b.

### Media Content

The ‘I’ in SMIL stands for “Integration.” This is because SMIL does not create media, but rather integrates existing multiple media into a single presentation. To specify media elements, SMIL presentations refer to files in other formats. Once the browser locates the media items, SMIL constructs control how the media will be presented.

The primary SMIL construct for retrieving media content is the `<ref>` element, and its synonyms `<img>`, `<video>`, `<audio>`, `<text>`, `<animation>`, and `<textstream>`. Each instance of these elements locates one media object to integrate into the presentation. The media object element `src` attribute, taken directly from HTML, states the Web location of the file holding the media object. All media in a SMIL presentation are brought in using the `src` attribute. In Figure 2, these constructs are coded in blue.

SMIL 2.0 also specifies transition effects that accompany media as they start and stop playing. In Fiets, these make the logo in Figure 1a fade in and out. In Figure 2, line 22 shows the `transIn` and `transOut` attributes, which indicate the transition to apply when the media object appears and disappears, respectively. The attributes’ value refers to a `<transition>` element defined for the SMIL presentation in the document head (see line 16). This code selects transition type “fade” from the more than 100 choices that SMIL offers. The fade gives the Fiets logo a one-second fade-in and a one-second fade-out.

### Layout

Once you’ve selected multiple media items as content, you must coordinate their display in the multimedia presentation. SMIL layout lets you control how each media object is arranged on the screen and integrated into the overall presentation. In paper-based and Web publishing, designers pay

## SMIL Resources

For more information on SMIL and SMIL tools, see the following resources.

### Online

- Resources and examples used in this article • <http://www.cwi.nl/~lloyd/Papers/Spotlight/>
- SMIL 2.0 Specification • <http://www.w3.org/TR/SMIL20/>
- W3C SMIL Links Page • <http://www.w3.org/AudioVideo/>
- W3C Web Content Accessibility Guidelines 1.0 • <http://www.w3.org/TR/WAI-WEBCONTENT/>
- RealNetworks RealPlayer • <http://www.real.com/products/player/>
- Oratrix GRiNS Player and Editor • <http://www.oratrix.com/Products/G2P?zone=G2P>

### Books

- H. Williamson, *SMIL for Dummies*, Hungry Minds, New York, 2001.
- M. Slowinski and T. Kennedy, *SMIL: Adding Multimedia to the Web*, Sams Publishing, Indianapolis, Ind., forthcoming in 2001.
- L. Rutledge and D.C.A. Bulterman, *SMIL: Interactive Multimedia on the Web*, Pearson Education, New York, forthcoming in 2002.

particular attention to text and image layout, as it contributes much to the readers’ appreciation and understanding of the material.

In Figure 2, most of the SMIL layout code (shown in red) appears within the `<layout>` element in the document head. The `<layout>` element specifies one layout for the whole presentation. It contains `<topLayout>` elements, each of which defines a presentation window. Although Fiets has only one window, SMIL 2.0 presentation lets you include as many window-defining `<topLayout>` elements as you want. Each window has multiple rectangular regions. In Figure 2, red layout code is also scattered throughout the code main body, assigning region attributes to media object elements and indicating the region in which they’ll play.

Figure 3 (page 81) shows how you place regions and visual displays in their windows using the attributes `top`, `bottom`, `left`, `right`, `height`, and `width`. Each of these has a numeric value in either pixels or percentages. Using these values, you can position a region relative to the boundaries of its containing block, which is typically defined by the region’s parent element – either another region or a `<topLayout>` window. You need only assign two of the three attributes along each dimension. If you assign fewer than two attributes for a dimension, SMIL defaults the missing attributes to the containing block’s boundary.

### Timing

SMIL’s foremost contribution to Web formats is its sense of timing. Without SMIL, XML-defined Web

```

1 <!DOCTYPE smil PUBLIC "-//W3C//DTD SMIL 2.0//EN" "http://www.w3.org/TR/REC-smil/SMIL20.dtd">
2 <smil xmlns="http://www.w3.org/2001/SMIL20/Language">
3 <head>
4 <layout>
5 <topLayout title="Fiets Amsterdam Tour" backgroundColor="black" width="1010" height="665">
6 <region regionName="splashScreen" top="5" left="5" bottom="5" right="5"/>
7 <region regionName="buildingImage" top="5" right="5" width="875" height="655"/>
8 <region regionName="closedCaptioning" bottom="5" left="5" right="5" height="60"/>
9 <region title="Thumbnail Bar" top="5" left="5" bottom="5" width="120">
10 <region regionName="stationThumb" fit="meet" height="90" top="65" />
11 <region regionName="churchThumb" fit="meet" height="90" top="280"/>
12 <region regionName="CWI-INSThumb" fit="meet" height="90" top="495"/>
13 </region>
14 </topLayout>
15 </layout>
16 <transition id="fadels" type="fade" dur="1s"/>
17 </head>
18 <body>
19 <seq>
20 <par title="Greeting Section" end="greet.end+1s">
21 
23 <par id="greet" begin="1s">
24 <switch>
25 <par systemLanguage="en">
26 <audio src="welcome.wav" region="buildingImage"
27 alt="Welcome to Fiets, your self-guided tour of Amsterdam (spoken)" />
28 <text src="welcome.html" region="closedCaptioning" systemCaptions="on"
29 alt="Welcome to Fiets, your self-guided tour of Amsterdam (captions)"/>
30 </par>
31 <par systemLanguage="nl">
32 <audio src="welkom.wav" region="buildingImage"
33 alt="Welkom bij Fiets, uw eigen stadswandeling door Amsterdam (gesproken)" />
34 <text src="welkom.html" region="closedCaptioning" systemCaptions="on"
35 alt="Welkom bij Fiets, uw eigen stadswandeling door Amsterdam (ondertiteling)"/>
36 </par>
37 </switch>
38 </par>
39 </par>
40 <par title="Thumbnail Section" dur="indefinite">
41 <par>
42 <a href="#station" alt="Show Centraal Station" >
43 
44 </a>
45 <a href="#church" alt="Show the Oude Kerk (Old Church)">
46 
47 </a>
48 <a href="#CWI-INS" alt="Show the CWI-INS building" >
49 
50 </a>
51 </par>
52 <excl dur="indefinite">
53 
54 
55 
56 </excl>
57 </par>
58 </seq>
59 </body>
60 </smil>

```

Color key:      Media content      Layout      Timing      Linking      Adaptivity

Figure 2. SMIL code for *Fiets Amsterdam Tour*. The **src** attributes (blue code) indicate the media object's Web location. SMIL layout instructions (red code) appear mainly within the **layout** element in the document head. Timing elements (purple code) dominate the document body's hierarchical composition.

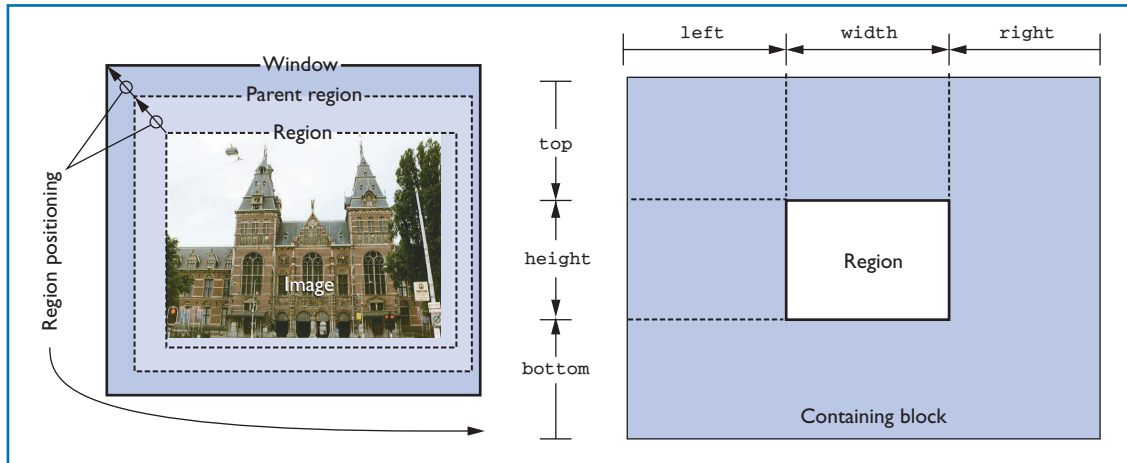


Figure 3. SMIL layout model. The layout attributes let you position regions within containing blocks.

presentations are static: Users can move displays using the scroll bar and switch between them using hyperlinks, but each presentation is itself unchanging. With SMIL, XML presentations change over time, with or without user interaction. This applies to more than just SMIL presentations; developers have added SMIL timing constructs to other XML-based formats as well.

**Temporal composites.** In Fiets, as in most SMIL presentations, timing elements (the purple code in Figure 2) dominate the hierarchical composition of the document body. SMIL's timing elements, `<seq>` and `<par>`, appear often and are always parents of other elements – typically other timing composites or media object elements. The `<seq>` and `<par>` elements specify different temporal relationships between their children.

- The `<seq>` element specifies that its children play in sequence, one after the other, in the order in which they appear in the code. That is, the second child starts when the first one ends, the third child starts when the second ends, and so on. The `<seq>` element itself ends, by default, when the last of its children ends. In Figure 2, lines 19 and 58 show the start and end tags of Fiets' single `<seq>` element. Its two children specify Fiets' two main parts: the introductory speech and logo, followed by the building thumbnails.
- The `<par>` element specifies that its children play in parallel, starting at the same time. The `<par>` element itself ends, by default, when all its children stop playing. In Figure 2, lines 25 through 30 show one of several `<par>` elements; this one simultaneously plays the welcoming English audio and corresponding closed-caption text.

SMIL's temporal composites let you specify timing information without using time stamps or numbers. Instead, you build the timing structure around the media's implicit duration. For example, a sequence of videos plays each video until it ends, then starts the next video, and thus you don't have to set begin times. Videos in a `<par>` play at the same time. Events triggered by the `<par>`'s ending happen, by default, when the longest-playing video ends. If the `<par>` itself is a child of a `<seq>`, then the next element in the sequence starts when the last video ends. Such temporal composition makes creating presentations easier, because you don't need to calculate time markers. It also makes presentations more adaptive to runtime delays: Timing is based on the end of media presentations, even if their ending is delayed.

**Timing attributes.** To fine-tune the general time line defined by SMIL's temporal composites, you assign timing attributes – `begin`, `end`, and `dur` (duration) – to media object elements and timing composites. When you set the `begin` attribute, an element will start a set length of time after its default start. Thus, a child of a `<par>` would start at a set time after the `<par>` begins and a `<seq>` child would start at a set time after its previous sibling ends. Similarly, an `end` attribute stops an element a set length of time after its default start time. Finally, the `dur` attribute sets an explicit duration for the element to play, overriding its implicit duration.

Fiets uses these three attributes to refine its synchronization. In Figure 2, line 23 has the assignment `begin="1s"`, which starts the `<par>` element grouping that contains the greeting audio and captions one second after the logo appears,

leaving time for its one-second fade-in. Line 52 assigns a duration of "indefinite" to Fiets' thumbnail section. This gives the thumbnail section no scheduled stop time, which means an unscheduled event must end it, such as a user closing the presentation. Finally, line 20 has a more complex assignment: `end="greet.end+1s"` synchronizes the greeting section's ending with the ending of the "greet" element (the audio and closed-caption welcome), adding one second for the logo to fade out.

### Linking

Although SMIL timing lets presentations progress by themselves, users need control to access the information they need – and to have a more engaging experience. SMIL's hyperlinking constructs provide the key navigation paths for users to traverse through Web multimedia. SMIL uses the same Web hyperlinking constructs as HTML, while also accounting for the impact of timing on user interaction.

**SMIL lets you tailor content according to characteristics such as language and computing environment.**

As in HTML, SMIL's primary linking constructs are the `<a>` element and its `href` attribute. The `<a>` element contains media that users can "click" (or otherwise activate) to trigger the link. The `href` value is a URI to the content triggered by the link, such as another SMIL presentation.

SMIL linking uses the `#` character for linking to document portions, just as HTML does. In both languages, a `#` in an `href` value links to the document portion named after the `#`. The document containing the portion is specified before the `#`; if nothing appears before it, the link is to elsewhere in the same document. In HTML, browsers typically scroll the display to where the named portion is shown. In SMIL, the linked-to SMIL document starts playing at the named point, effectively fast-forwarding the whole presentation up to that point.

SMIL 2.0 introduces the `<exc1>` (exclusive) element, which lets users see Web linking in context: some portions of the document continue playing while other parts change to show the link's destination. This is possible because the `<exc1>` element's children play one at a time, in any order, and have no implicit synchronization with the rest of the presentation. Whenever one child is trig-

gered to begin, other playing siblings stop and the rest of the timing is unaffected (unless you set explicit synchronization relations). The most common use case for this is "jukebox buttons," which let users trigger links in any order to play subpresentations that cancel each other out.

Fiets uses such jukebox buttons for its thumbnail section. In Figure 2, the `<par>` spanning lines 41 through 51 shows thumbnail images for the three buildings. Each image is packaged in an `<a>` element linking to the display of the corresponding larger image. The larger images are contained in the `<exc1>` spanning lines 52 through 56. At the beginning, none of the larger images is shown; each time a user clicks a smaller image, the corresponding larger image appears, canceling the display of any other larger image showing at the time.

### Adaptivity

Web documents are available to people all over the world. In creating such documents, you often must account for a wide variability in how users process information. SMIL helps you achieve this through adaptivity, letting you tailor content according to characteristics such as language, perceptual abilities, and computing environment.

For adaptivity, the most important SMIL element is `<switch>`. The browser examines the children of each `<switch>` element in the order in which they are encoded. Once it finds a child that passes the tests for playability, it selects that child for play and ignores all the others. If it finds no children appropriate for play, none are played.

To determine playability, SMIL primarily relies on test attributes. These attributes typically state the conditions that must be true for an element to be played. In Fiets, for example, the developers used the `systemLanguage` attribute to indicate that an element is only appropriate for users fluent in the stated language. In Figure 2, the attribute assignments on lines 25 and 31 state that the language of each version of the verbal greeting is, respectively, English and Dutch. The `<par>` elements with these assignments are in a `<switch>`, so only one (or possibly none) is played for each user.

An element does not have to be in a `<switch>` to be determined inappropriate for inclusion – elements anywhere in the document are subject to the same test. This enhances adaptivity by letting you play optional elements for some presentations and not for others. Fiets applies this feature with closed captioning. The captioning elements on code lines

28 and 34 have the `systemCaptions="on"` attribute assignment, meaning they are shown only if captions are turned on in a user's browser. Audio descriptions make multimedia accessible to sight-impaired users by offering a spoken account of visual presentation elements. The elements in SMIL's `systemAudioDesc="on"` test attribute assignments play only if the user requests audio descriptions.

Descriptive constructs enhance adaptivity by alerting disabled users to presentation content that they can't directly perceive. The HTML and SMIL attributes `title`, `alt`, and `longdesc` give textual accounts of subdocument and media item content. The W3C strongly suggests using the `alt` attribute in particular for all linking and media object elements (for more on this, see the W3C's accessibility guidelines in the "SMIL Resources" sidebar on page 79). Users who cannot perceive a document's presented forms can then read (or hear, through speech synthesizers) their descriptions instead. You can also use these attributes as descriptive comments in the SMIL code. Editing tools like GRiNS help incorporate the descriptions into interface displays.

## Modularization

Like XML, SMIL is a metalanguage that lets you create other languages. By placing constructs into modules, SMIL 2.0 lets you combine these modules into your own *profile* – a tailored final-form language for multimedia presentation. W3C provides several examples of SMIL profiles: the SMIL 2.0 Language Profile, SMIL Basic, XHTML+SMIL, and animated SVG.

SMIL Basic is a subset that contains only features that suit the limitations of a mobile environment. Developers can thus create SMIL presentations that won't overwhelm mobile phones and palmtop devices. SMIL Basic presentations can also play on SMIL 2.0 Language Profile browsers. SMIL Basic is defined along with the SMIL 2.0 Language Profile in the SMIL 2.0 Recommendation specification.

XHTML+SMIL combines SMIL and HTML constructs into one new language. This enables new use cases, such as XML-based PowerPoint-like presentations. Such presentations mix timing and text in a layout that is based on text flow as defined by HTML and CSS, rather than SMIL's two-dimensional layout. XHTML+SMIL also inherits HTML's text-flow-based hierarchical composition, rather than SMIL's temporal composition, which affects how authors structure and

## SMIL Support

Like most W3C recommendations, SMIL 2.0 was developed and released with strong industrial involvement and support. RealNetworks, Microsoft, and Oratrix have already announced SMIL 2.0 support in their upcoming browser releases.

RealNetworks' RealPlayer began playing SMIL presentations soon after SMIL 1.0 was released. Subsequent distribution of RealPlayer has put SMIL on the desktops of more than 15 million users. SMIL 1.0 frequently synchronizes media on RealNetworks' Web site and documents it links to, for example. RealPlayer users are often unaware they are browsing SMIL presentations, and the same will likely be true as RealPlayer upgrades to SMIL 2.0.

Oratrix's GRiNS has also been playing SMIL presentations from the start. For the past year, GRiNS has kept up with every public release of the SMIL 2.0 draft specification and is now playing the Proposed Recommendation version of the SMIL 2.0 Language Profile. GRiNS's main function, however, is as an editor for SMIL 2.0, SMIL Basic, and XHTML+SMIL. It is particularly useful for managing large-scale SMIL-defined multimedia. Figure A shows a structural view of the Fiets presentation in Oratrix's GRiNS editor for SMIL 2.0, scheduled for public release in September.

Microsoft's Internet Explorer 5.5 (the currently released version) plays HTML+TIME, a prototype hybrid of SMIL constructs and HTML. IE 6.0 will play XHTML+SMIL and further the extensive, albeit quiet, spread of SMIL through the Web.

Other browsers are providing single interfaces to multiple languages as well. For example, Helsinki University of Technology has developed X-Smiles (<http://www.xsmiles.org>), a tool that presents XHTML, XML, SMIL, and other XML-based languages in one browser, making the existence of multiple formats irrelevant to users.

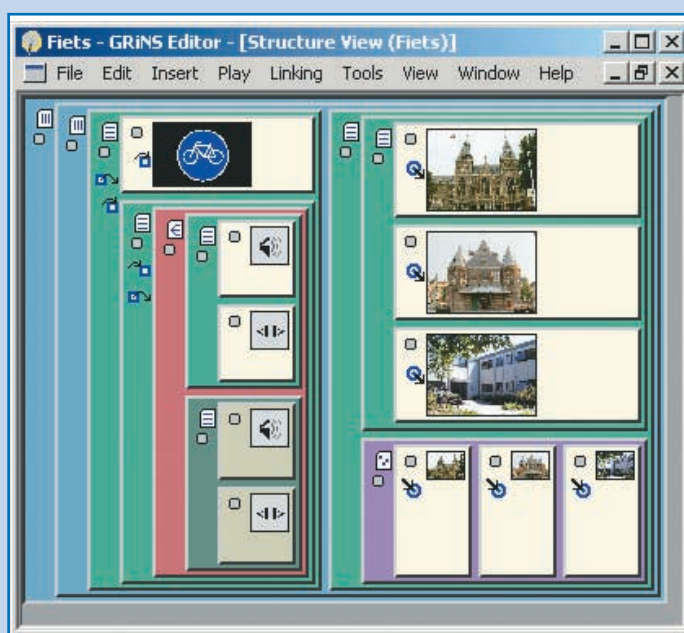


Figure A. GRiNS editor view of Fiets Amsterdam Tour. In addition to facilitating presentation editing, GRiNS helps developers manage large-scale SMIL-defined multimedia.

maintain their documents. SMIL 2.0 and XHTML+SMIL are separate W3C projects, and the latter is still a work in progress.

SMIL also provides SMIL Animation, a package of animation constructs that authors can apply to other XML formats. SVG, for example, applies SMIL timing constructs to create elaborate animated graphics. Like XHTML+SMIL, this mix of SMIL constructs and direct XML-based encoding of content offers an example of how flexibly SMIL 2.0's constructs blend with other XML syntax.

### Conclusion

SMIL 2.0's release makes full-fledged multimedia accessible to every user, author, and implementer, as well as to the Web infrastructure itself. SMIL is emerging not so much as a new class of presentations and tools but as a tool to make the existing Web and its various interfaces more animated, informative, and accessible.

SMIL 2.0 is coming out now not just as an XML syntax but also as a collection of readily available products. Upcoming releases of RealPlayer and Internet Explorer will quickly put SMIL 2.0 playback capabilities on millions of computers world-

wide. GRiNS, already released, plays 2.0 now and helps build large-scale interactive multimedia for playback on all these players. The next step is to see what SMIL presentations media artisans create, and how they use the new power of SMIL 2.0 to change the face of the Web as we know it. □

### Acknowledgments

My work on SMIL was funded in part by the MIA project in Amsterdam. This article benefited from the valuable feedback of Jacco van Ossenbruggen of CWI, Dick C.A. Bulterman and Sjoerd Mullender of Oratrix, and Geoff Freed of WGBH. Portions of the article were adapted from the forthcoming book, *SMIL: Interactive Multimedia on the Web*.

Lloyd Rutledge is a researcher in multimedia and human-computer interaction at CWI (Centrum voor Wiskunde en Informatica) in Amsterdam, the Netherlands. His research interests include semantic annotation of stored media, multimedia authoring, and semiautomatic generation of hypermedia presentations. He is a member of the W3C SYMM working group, which develops SMIL.

Readers can contact the author at [Lloyd.Rutledge@cwi.nl](mailto:Lloyd.Rutledge@cwi.nl).

# Get access

to individual IEEE Computer Society documents online.

More than 57,000 articles  
and conference papers available!

*US\$5 per article for members*

*US\$10 for nonmembers*

<http://computer.org/publications/dlib/>

