Chapter 13
SMIL: Synchronized Multimedia Integration Language

Dick C. A. Bulterman

Abstract The period from 1995 to 2010 can be considered to be networked multimedia's Golden Age: Many formats were defined that allowed content to be captured, stored, retrieved, and presented in a networked, distributed environment. The Golden Age happened because network infrastructures had enough bandwidth available to meet the presentation needs for intramedia synchronization, and content codecs were making even complex audio/video objects storable on network servers. This period marked the end of the CD-ROM era for multimedia content distribution. Unlike the relative simplicity of CD-ROM multimedia, where timing constraints were well-understood and pre-delivery content customization was relatively simple, the network multimedia era demanded new languages that would allow content to be defined as a collection of independent media components that needed to be located, fetched, synchronized, and presented on a large collection of user devices (under greatly varying network characteristics). One of the most ambitious projects to define an open and commonly available multimedia content integration language was W3C's SMIL. In a period of approximately ten years, SMIL grew from a simple synchronization language to a full content integration and scheduling facility for a wide range of Web documents. This chapter considers the timing and synchronization aspects of SMIL.

Keywords Structured timing · Hierarchical synchronization · Web-based multimedia

D. C. A. Bulterman (✉)
Vrije Universiteit Amsterdam and CWI, Amsterdam, The Netherlands
e-mail: dick.bulterman@gmail.com

M. Montagud et al. (eds.), *MediaSync*,
https://doi.org/10.1007/978-3-319-65840-7_13

13.1 Introduction

This chapter provides an overview of the W3C Synchronized Multimedia Integration Language (SMIL), which was developed and is maintained by the World Wide Web Consortium (W3C) [1]. SMIL has been defined as a series of *W3C Recommendations* [2–5], the term used by W3C to indicate a member-reviewed and member-approved standard for use within the W3C's suite of protocols and formats. SMIL is a comprehensive presentation format that can be used to structure the timing, layout, and user control of a set of content objects. While SMIL can be used to address a wide array of temporal control applications, this chapter will concentrate on SMIL as a multimedia presentation language.

The SMIL Recommendation is structured as a set of *modules*, each providing a collection of elements, attributes, and attribute values that can be selectively included when designing XML languages. The SMIL modules have also been grouped into a number of *profiles* that each serves the implementation needs of individual use domains. Since this book is particularly concerned with media synchronization, the discussion of SMIL in this chapter will largely be limited to the timing and synchronization aspects of the language. We will also consider how timing and synchronization are influenced by user interaction: mostly by activating temporal hyperlinks, but also where parts of a presentation are activated by event-based user interaction.

In the sections below, we start with a short history of SMIL, tracing the roots of the formats on which SMIL was based. We then provide a short primer on how SMIL is structured, with enough background to understand the examples given later in the chapter. We then consider the details of SMIL timing and interaction. We close with a short reflection on the success and limitations of the language.[1]

13.2 A Brief History of SMIL

In 1996, W3C started an activity to determine how audio and video could best be supported in the context of the a-temporal HTML text and image content that was then dominant on the World Wide Web. This activity, lead by Philipp Hoschka, resulted in the definition of the Synchronized Multimedia (SYMM) working group [7]. The working group consisted of several major providers of media players and embedded technology (most notably Apple, Intel, Microsoft, Philips, RealNetworks), a collection of academic institutions (CWI, GMD, and INRIA), and organizations interested in accessibility (WGBH and the Daisy consortium). After about nine months of work, the group published the first version of SMIL in June 1998. The architecture of SMIL was based largely on CWI's CMIF format [8], with modifications, restrictions, and extensions flowing from the design-by-committee

[1]This chapter draws on material published in [6].

process. Subsequent editions of SMIL, up to SMIL 3.0, were produced through December 2008, when the working group ceased active development.

From its earliest version, SMIL differed significantly from other media support activities available at that time. Unlike Apple's Quicktime and the Windows Media Player—the two dominant content delivery sources at the time—SMIL was not a *content object delivery platform* in which a single video or audio item could be played, but a *presentation delivery platform*, in which several independent media objects could be gathered (potentially from multiple servers), scheduled, and then presented via an open Web interface. Treating a media presentation as a collection of objects rather than a single item remains a unique approach even in current (W3C) multimedia recommendations.

The general model used by SMIL was that a media presentation consisted of an XML-formatted scheduling file[2] (the.*smil* presentation) and a series of external media objects. A SMIL-compliant player would interpret the presentation file and then implement a scheduling algorithm that would correspond to the needs of the presentation specification. Different players could employ different implementation strategies to meet the needs of the SMIL specification. Another innovation of SMIL was a *declarative* approach to defining the media object interactions in a presentation. SMIL is not a scripting or programming language that implements the mechanics of content delivery and playout, but a specification language that allows an author to define what they would like to have happen. It is up to the playout engine to resolve any constraints at playback time, such as lack of network bandwidth, limited screen size, or lack of interaction facilities.

The focus on media scheduling meant that SMIL was not a low-level content creation language, but a media aggregation and synchronization language. In early versions of SMIL, there was a rigid separation between content creation and content scheduling: All objects scheduled by SMIL were required to be stored in external files. Later versions relaxed this limitation, allowing plain and synchronized text to be defined directly within the SMIL file.

In 1996, the Internet was a heterogeneous environment. SMIL was defined to interact within this environment, with different user agents, different network speeds, different client screen sizes, different user language preferences, and users of differing natural abilities (in the sense that users could be deaf, blind, or otherwise differently-abled). Interoperability was a key concern of SMIL, which meant that limiting the specification to any one scripting language or any one delivery platform would not meet the language's needs. SMIL was largely successful in providing an interoperable scheduling language. Unfortunately, unlike text and images, there was—and is—no universally available set of video and audio formats that could ensure interoperability of presentation content. Early commercial and open implementations of SMIL-compliant players did not support interoperability in content codecs. As a result, a SMIL presentation written for use with the

[2]SMIL was not only an XML-compliant language, it was the *first* XML language released as a W3C recommendation.

RealNetworks G2 player could not be played with Apple's Quicktime player, even though both were early adopters of the SMIL language. This severely limited the impact of SMIL.

13.3 SMIL Presentation Basics

A SMIL presentation is an XML-formatted specification containing references to media content objects, a temporal scheduling, and synchronization model that determines when these objects are presented (and how persistent they are) and a layout model that can help a playback agent determine where the objects should be placed relative to one another. The specification also allows transitions, metadata, temporal hyperlinks, and a host of other secondary presentation features to be defined. Most of these are beyond the scope of this chapter (but are treated extensively in [6]).

In order to understand the basic structure of a simple SMIL presentation, consider the following SMIL definition[3]:

```
0)  <!DOCTYPE SMIL PUBLIC "-//W3C//DTD SMIL 3.0 Language//EN"
                          "http://www.w3.org/2008/SMIL30/SMIL30Language.dtd">
1)  <SMIL xmlns="http://www.w3.org/ns/SMIL" version="3.0" baseProfile="Language">
2)   <head>
3)      <meta name="title" content="SlideShow"/>
4)      <meta name="generator" content="GRiNS for SMIL 3.0"/>
5)      <meta name="author" content="Dick Bulterman"/>
6)      <layout>
7)          <root-layout xml:id="Winter" backgroundColor="#ffffcc" width="240" height="269"/>
8)          <region xml:id="Title" left="0" width="240" top="0" height="29"/>
9)          <region xml:id="Image" left="0" width="240" top="29" height="180" z-index="2"/>
10)         <region xml:id="Text" left="0" width="240" top="209" height="42" fit="meet"/>
11)         <region xml:id="Buttons" left="0" width="240" top="251" height="15"/>
12)         <region xml:id="Sound"/>
13)     </layout>
14)  </head>
15)  <body>
16)     <par>
17)         <img region="Title" src="M/title.gif"/>
18)         <seq>
19)             <video region="Image" src="M/v0.mp4"/>
20)             <video region="Image" src="M/v1.mp4"/>
21)             <video region="Image" src="M/v2.mp4"/>
22)             <video region="Image" src="M/v3.mp4"/>
23)         </seq>
24)     </par>
25)  <body>
26) </SMIL>
```

[3]Line numbers have been added to simplify references in the text. They are not part of the SMIL language.

(This structure is reused later in this chapter.) Line 0 defines some XML basics for this file: the DOCTYPE and the DTD. This defines the format of the SMIL file itself, not of the presentation defined in the file. Line 1 defines the dialect of SMIL used in this file and the SMIL profile (collection of modules) that is expected to be supported by the SMIL agent processing the presentation.

Lines 2–14 provide information that the agent can use to process the file. It may contain metadata defining the name, the author, and the system used to generate the presentation. It also includes a layout specification that determines where objects are placed and their relative layering. SMIL Layout was a contentious facility: Most members of W3C wanted SMIL to use CSS layout features [9]. These gave the user agent nearly total control over (relative) object placement. The SYMM group felt that multimedia presentations were different from text content and that the semantic meaning of object placement (such as having captions overlay or be close to content) justified having a separate layout model.

Lines 15–25 define the presentation body. In this presentation, an image containing a presentation title is presented in parallel with a sequence of video objects. Both the title image and the sequence start at the same time. The members of the sequence start when their predecessor is finished.

One design goal of SMIL was that simple things should be able to be done as simply as possible, but that complex scheduling operations should also be possible without having to resort to a scripting language. In that sense, the following minimalist SMIL presentation could be used to display a simple sequence of video objects:

```
0)  <SMIL>
1)  <body>
2)      <video src="M/v0.mp4"/>
3)      <video src="M/v1.mp4"/>
4)      <video src="M/v2.mp4"/>
5)      <video src="M/v3.mp4"/>
6)  <body>
7)  </SMIL>
```

In this presentation, a default DTD and SMIL profile is used, as determined by the user agent. The agent defines a default layout structure for the presentation. The <body> element, which in SMIL defaults to the behavior of a <seq> element, is used to structure the presentation of a series of videos. The implicit duration of the videos themselves determines the length of the presentation. Granted, there is not much control on where a user agent displays the content, but the authoring overload of creating the presentation is minimal.

Finally, as a basic XML refresher:

- Each line of text between "<" and ">" characters is an XML statement.
- Each statement begins with an element name, defined by the language (in this case, SMIL).

- Each element defines a number of attributes (such as "src"). Attributes may allow attribute values to be defined (constrained by the language).

Of course, the full XML specification is slightly more complex than this summary, but this should be enough to understand the examples below.

13.4 SMIL Timing and Synchronization

SMIL timing defines when elements in a presentation get scheduled and, once scheduled, how long they will be active. The SMIL timing facilities are the core contribution of the SMIL standard: using the elements and attributes defined in the SMIL Recommendation, time can be integrated into any XML language.

In a SMIL-based document, every media object and nearly every structural element has a specific timing scope. While media timing plays an important role in determining the overall duration of a presentation, the structure of the document is also used to simplify and optimize the rendering of media presentations. SMIL timing defines a collection of elements that determine the relative start and end times of document objects and a collection of attributes that control the duration, persistence, repetition, and accuracy of timing relations in a document. SMIL can be used directly as the host language for a document (as is done in the various SMIL profiles), but it can also serve as the basis for integrating time-based coordination of otherwise static elements (as is done in SVG animation and in XHTML + SMIL).

13.5 SMIL Timing Model Basics

The timing approach used by SMIL to specify the (relative) begin times of media objects and their durations is based on a *structured timing model*. This means that the nested presentation structure in a SMIL document—and not only hard-coded clock values—is used to define the high-level activation and synchronization of objects. For many simple SMIL documents, this timing is implied: The SMIL agent can figure them out at presentation time. If more precise control over a presentation is required (such as inserting delays or specifying interactive behavior), SMIL also provides more complex timing mechanisms.

13.5.1 A Simple Slideshow Presentation

We introduce the timing issues addressed by SMIL in terms of the slideshow presentation depicted in Fig. 13.1. This presentation contains a single background

image on top of which a sequence of slides is placed, each with an accompanying image containing a text label and an audio file containing spoken commentary. The presentation also contains a single background music object that is played throughout the presentation. The timing in this presentation is dominated by two object sets: a background music object, which determines the duration of the total presentation, and various spoken commentary objects, which determine the duration of each of the image slides. This means that an outer time base for the entire presentation is defined and a set of inner time bases for each slide in the presentation.

13.5.2 Media Object and Presentation Timing Definitions

A basic property of multimedia presentations is that they require some degree of temporal coordination among the objects being presented. The more complex a presentation—in terms of either number of simultaneous objects or number of synchronization control points—the greater the amount of control information required. SMIL uses *timing elements* and *timing attributes* to provide the activation and synchronization control information in a presentation. In general, timing attributes are used to control the timing behavior of media object, and timing elements are used to control the behavior of the presentation as a whole.

13.5.2.1 Media Timing

Multimedia presentations typically contain two types of media objects: *discrete* media and *continuous* media. Discrete media objects, such as the text labels, the background image, and each of the slide images in Fig. 13.1, have no implicit duration. If referenced in a SMIL file without any additional timing attributes, their duration will be 0 s—which is not very long. *Continuous* media, such as the background music object and each of the spoken commentaries in Fig. 13.1, have implicit durations that are defined within their media encodings. If referenced in a SMIL file without any additional timing attributes, they will be rendered for the full duration defined by the object.

In Fig. 13.1, each slide image and the associated text labels should be displayed for duration that is defined by the accompanying spoken commentaries. (Each slide/text/audio group will have different durations, since not all spoken commentary is equally long.) SMIL provides a range of attributes that allow the duration of objects to be explicitly defined and refined, and it provides a general inheritance model in which the durations of both discrete and continuous media can be obtained by the context in which a media object is presented in relation to other objects. This allows the durations of the images and text to match that of the spoken audio.

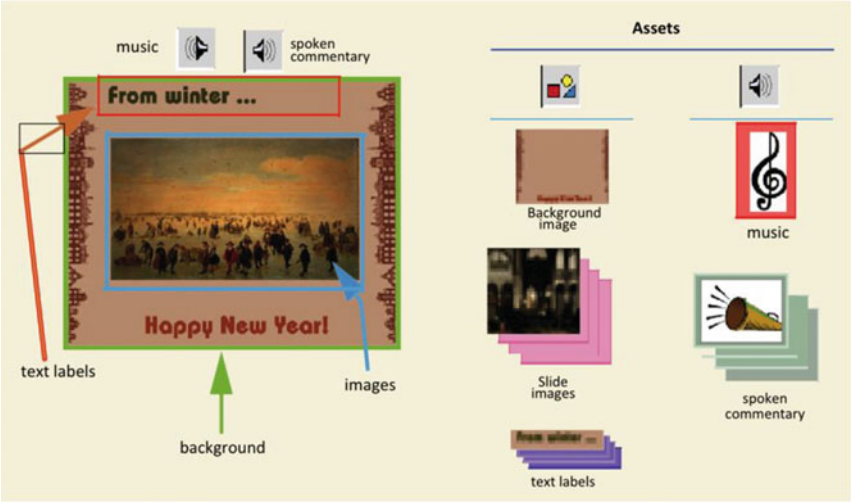


Fig. 13.1 Elements used in a slideshow presentation

13.5.2.2 Presentation Timing

A SMIL file contains references to one or more of media objects and a set of timing primitives that determine when these objects get started relative to one another. The total timing of each of the media objects, plus any additional timing control defined in the SMIL file, determines the duration of the composite presentation. Sometimes, this composite duration can be calculated in advance, but often it cannot.

The basic timing of the slideshow presentation described in Fig. 13.1 is deterministic: That is, we can determine the full timing in advance of the presentation's execution by evaluating the timing of each of the continuous media objects. It is important to understand, however, that the presentation is only deterministic if several potential presentation-time delays are ignored—these include any streaming delays associated with bringing the media object from a server to the presentation device, or any delays at the client associated with decoding and rendering individual media objects. For local presentations, such as CD-ROM multimedia, it is safe to assume that all the delays in the system can be predicted in advance and factored into the presentation timing. For Web-based multimedia, where the delays when obtaining media may be considerable (and unpredictable) and where there may be wide variability in the performance of end-user devices, assuming that presentations are fully deterministic which is a dangerous strategy.

A presentation with uncertain timing characteristics is *non-deterministic*. In addition to the network streaming delays discussed above, non-deterministic timing can also be the result of content substitution within the presentation or as a result of using interactive, event-based presentation timing. SMIL has elements and attributes to handle these cases as well.

13.5.3 SMIL and Timelines

A timeline metaphor is often used to model presentations. A timeline is a simple graph showing time on one axis and one or more media objects on the other axis. An example timeline, showing the elements and objects in Fig. 13.1, is shown in Fig. 13.2. This timeline shows the media sorted by layout: The left axis shows the various classes of media objects used, and the bottom axis shows the cumulative duration. It is also possible to define separate lines for each media object, but this is usually less space efficient.

A timeline exposes the exact temporal relationships among media items. These can translated to a text format by assigning explicit begin times for each media object and defining durations to discrete objects. One such encoding is the time-list structure of the following code fragment. The background audio and image objects (lines 0 and 1) are followed by the set of slide images (lines 2–8), the spoken commentary (lines 9–15), and the image-encoded text labels (lines 16–22). The begin times are determined by the duration of the spoken commentary objects. While this example could be used as the basis for SMIL timing, this would be unwise, since it is insensitive to delays and requires that all timing relationships be pre-calculated.

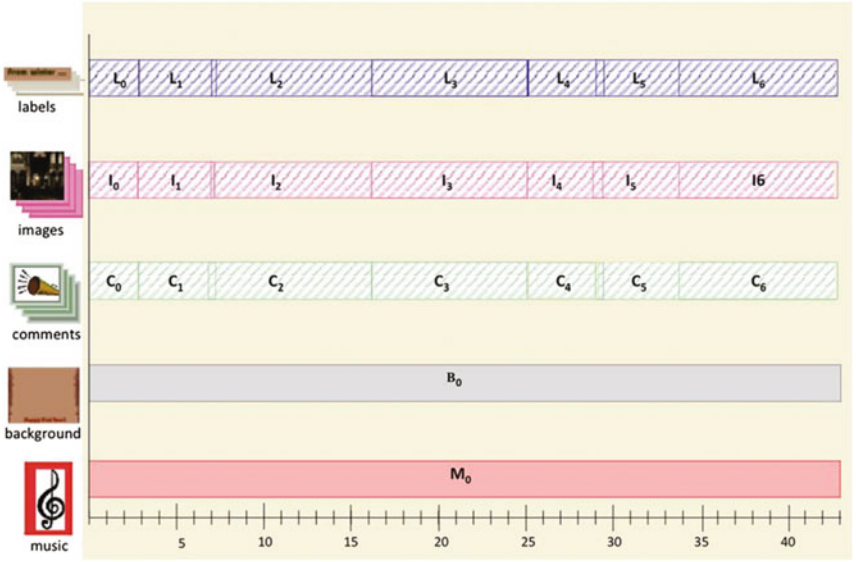


Fig. 13.2 Timeline representation of presentation in Fig. 13.1

```
0)   <audio xml:id="M0" begin="0s"    ... />
1)   <img xml:id="B0" begin="0s" dur="43s"... />
2)   <img xml:id="I0" begin="0s" dur="3s" ... />
3)   <img xml:id="I1" begin="3s" dur="4s" ... />
4)   <img xml:id="I2" begin="7s" dur="9s" ... />
5)   <img xml:id="I3" begin="16s" dur="9s" ... />
6)   <img xml:id="I4" begin="25s" dur="4s" ... />
7)   <img xml:id="I5" begin="29s" dur="5s" ... />
8)   <img xml:id="I6" begin="34s" dur="9s" ... />
9)   <audio xml:id="C0" begin="0s"    .../>
10) <audio xml:id="C1" begin="3s"   ... />
11)  <audio xml:id="C2" begin="7s"   ... />
12) <audio xml:id="C3" begin="16s"   ... />
13) <audio xml:id="C4" begin="25s"   ... />
14) <audio xml:id="C5" begin="29s"   ... />
15) <audio xml:id="C6" begin="34s"   ... />
16)  <img xml:id="L0" begin="0s" dur="3s" ... />
17) <img xml:id="L1" begin="3s" dur="4s" ... />
18)  <img xml:id="L2" begin="7s" dur="9s" ... />
19)  <img xml:id="L3" begin="16s" dur="9s" ... />
20) <img xml:id="L4" begin="25s" dur="4s" ... />
21) <img xml:id="L5" begin="29s" dur="5s" ... />
22) <img xml:id="L6" begin="34s" dur="9s" .../>
```

A better approach is to use structure-based timing primitives, since this allows timing to be deduced from the content rather than duplicating content and timing information.

13.5.4 SMIL and Structure-Based Timing

The main advantage of the timeline model is that it is an easy-to-understand representation of continuous media objects under deterministic timing conditions. As such, it is a representation used often in video and audio media editors. Unfortunately, while deterministic timing is good for modeling video tape, it does not scale well to most Web environments: If one of the image objects arrives later than planned, or if the presentation agent is slow in rendering the audio, the timeline does not really help in maintaining order among objects. Things become even more troublesome if we don't know the implicit duration of the audio items when constructing the timeline or if the duration of the object changes over the lifetime of the presentation. (Since the SMIL file does not contain the media—it contains a pointer to the media—the timing and the update histories of the media object are decoupled from its use.) Finally, if we add content substitution or interactive timing to the presentation (such as having the follow-on slide begin on a mouse click rather than at a fixed time), the timeline representation loses almost all of its utility.

In order to provide a more realistic framework for Web documents, SMIL is based on a structured timing framework in which the structured relationships among objects can be used to define most timing. SMIL encodes its timing

relationships by defining a logical timing hierarchy rather than an exact timeline. The hierarchy for Fig. 13.1 is shown in Fig. 13.3. Here, we see a set of yellow logical parallel nodes (P0–P7) and one blue logical sequential node (S0). The parallel components say activate the sub-components together as a unit, and the sequential component says activate the sub-components sequentially. The SMIL textual encoding of the presentation hierarchy is as follows:

```
0)  <par xml:id="P0" >
1)  <audio xml:id="M0" .../>
2)  <image xml:id="B0" .../>
3)  <seq xml:id="S0" >
4)      <par xml:id="P1" >
5)          <img xml:id="I0" .../>
6)          <audio xml:id="C0" ... />
7)          <img xml:id="L0" ... />
8)      </par>
9)      <par xml:id="P2" >
10)         <img xml:id="I1"  ... />
11)         <audio xml:id="C1" ... />
12)         <img xml:id="L1" ... />
13)     </par>
14)     ...
15)     <par xml:id="P7" >
16)         <img xml:id="I6"  ... />
17)         <audio xml:id="C6" ... />
18)         <img xml:id="L6" ... />
19)     </par>
20) </seq>
21)  </par>
```

While a timeline can state that objects I_2, C_2, and L_2 all start at 7 s into the presentation and that they each have duration of 9 s, the SMIL hierarchy can state what is really going on logically:

- That objects C_i, I_i, and L_i are to be treated as a logical group that get scheduled together (that is, they begin and end together);
- That the duration of I_i and L_i depends on the duration of C_i,
- That all three objects are to begin after object C_{i-1}—and, by extension, I_{i-1} and L_{i-1}—end.

Note that none of these relationships depends on the exact duration of any of the objects—you can construct a SMIL file before you know anything about the actual media being used.

A single timeline for one instance of a SMIL specification (that is, for one of the—potentially many—run-time uses of the presentation) can be constructed by combining the structured composition of objects with a model of the execution environment that contains information on the performance of the network

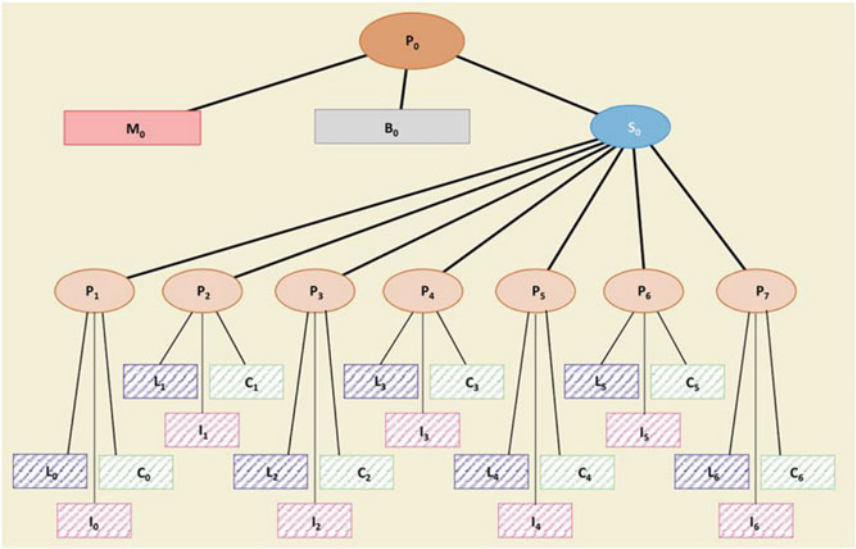


Fig. 13.3 The SMIL structured representation of the presentation in Fig. 13.1

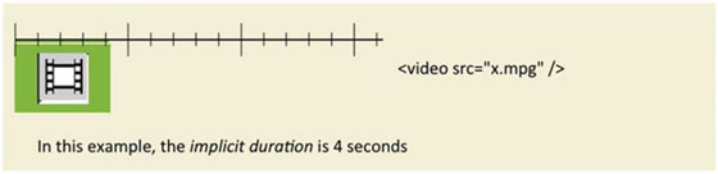
connection, the preferences of the user, etc. A timeline model based on the explicit media timings alone is not rich enough to model the various structured paths throughout a SMIL presentation.

13.5.5 Durations, Time, and Timebases

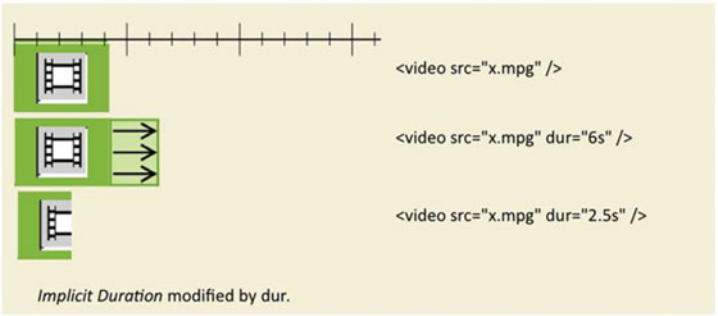
One of the most powerful features of SMIL is a flexible time model in which various aspects of an element's behavior can be determined by the context in which it is being presented. In order to use this model, it is important to understand a number of temporal distinctions and constraints applied by the SMIL model. These include defining the active period of an element and defining the way that delays and relative starting/ending times can be expressed in a document.

13.5.5.1 Defining the Active Period of an Element

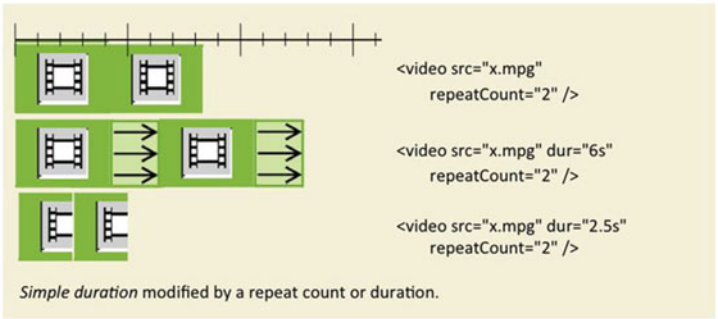
Most media formats require that the duration of all of the component media objects be explicitly defined. SMIL has several attributes that support direct duration definition, but it also provides attributes that allow you to specify or limit several layers of logical object durations. These layered durations, which are illustrated in Fig. 13.4, are as follows:



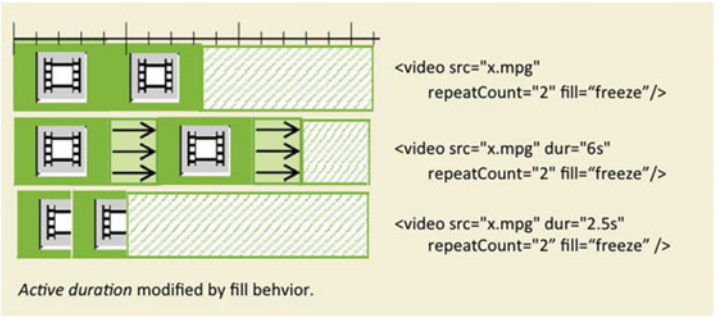
(a) Implicit Duration



(b) Simple Duration



(c) Active Duration



(d) Rendered Duration

Fig. 13.4 SMIL durations

- *Intrinsic duration*: This is the duration of a media object as encoded in the (external to SMIL) media file. Most discrete media items such as images or plain text have an intrinsic duration of 0 s; some quasi-discrete media—such as animated images—may have a longer intrinsic duration. Continuous media objects have duration that is equal to the temporal length of the object. Many media formats (but not all!) define the intrinsic duration explicitly in the media encoding.
- *Implicit duration*: This is the duration that SMIL uses as the basis for scheduling an object. It is usually equal to the intrinsic duration, if available. Discrete objects are modeled as having an implicit duration of 0 s. If the intrinsic duration for continuous objects is not available (such as often this case with MP3 objects), the SMIL agent typically will have to scan the entire object to determine its duration. (This can be a time-consuming process.) The implicit duration forms the starting point for the calculation of other logical timing durations have been defined within SMIL. Note that the implicit duration is a SMIL concept, separate from an object's intrinsic duration. See Fig. 13.4a.
- *Simple duration*: It is possible to modify an object's implicit duration with an explicit duration via SMIL's *dur* attribute. The result of applying an explicit duration (if any) to an object yields its simple duration. (If the implicit duration is not modified by a *dur* attribute, then the implicit and simple durations are the same.) The simple duration of an object may be longer or shorter than the implicit duration. Simple durations can also be defined to have special values that logically limit or stretch the duration of objects; these are the media and indefinite values, as discussed below. See Fig. 13.4b.
- *Active duration*: SMIL defines a number of attributes that allow an element to be repeated. These attributes modify the element's simple duration, and the resulting repeated duration is called the object's active duration. If an element's simple duration is shorter than its implicit duration, only the first part of the element will be repeated. If the element's simple duration is longer than its implicit duration, the entire element plus the temporal difference between the implicit and simple durations will be repeated. (During this "extra" time, either nothing will be rendered or the final frame/sample of the media object will be rendered: The behavior depends on the media *type*.) The *end* attribute can be used to define when the active duration ends. If an element does not repeat and is not shortened by end, its simple and active durations are the same. See Fig. 13.4c.
- *Rendered duration*: The active duration of an element ends after its *dur*/*end* and *repeat* attributes have been applied. This does not mean that an object disappears at the end of its active duration. SMIL provides an attribute to control the persistence of an object after its active duration has ended: the *fill* attribute. If *fill* is set to "freeze," the element will remain rendered until the end of its parent time container. If *fill* is set to "remove," the object is removed from the screen as soon as its active duration ends. For discrete media with a *fill* = "freeze" attribute, the object will simply be rendered as if its active duration was extended; for visual continuous media, the last frame or sample of the object will be rendered.

It is important to understand that each of these durations applies to every temporal element in SMIL. Every element has an implicit, simple, active, and rendered duration. Luckily, most of these durations will be managed by the SMIL agent, but understanding each of these durations, and their impact on presentation timing, can help clarify why a SMIL agent behaves the way it does.

13.5.5.2 Clock Values

Many timing attributes are based on clock values. These values can take several different forms, and they may serve as all or part of an attribute's time value. All clock values represent a relative time and have meaning only within the context of a time container.

Clock values may be given in four general forms:

- *Full clock values*: These are times represented as a colon-separated list of hours, minutes, seconds, and fractions of a second. (If days, months, or years need to be specified, then absolute wall clock timing may be used instead.) This is a relative time and has meaning only within the context of a time container's syncbase. (Syncbases are described below.)
- *Partial clock values*: These are times represented as a shorthand notation for full clock values, containing minutes, seconds, and (optionally) fractions of a second.
- *Timecount values*: These are numbers with an optional type string and an optional fractional component. An integer clock value with no type string (such as "10") implies a timecount in seconds; it is equivalent to "10 s." Allowed type strings are "h," "min," "s," and "ms."
- *Wallclock values*: These are absolute times represented in three parts: a date field, a time field, and an (optional) timezone field. These times are absolute.

13.5.5.3 Syncbases

Except in the special case of wallclock timing, every clock value in SMIL is relative to some other part of the document. The child elements of a <par> container are started relative to the start time of that parent, while the child elements of a <seq> container are started relative to the end of their predecessor (except for the first child, which starts at the beginning of its parent). Every element in a SMIL specification has a specific temporal reference point: the *syncbase*. Most elements never have to specify their syncbase reference explicitly since the common SMIL time containers do this by default. Sometimes, however, an element may want to specify a non-default syncbase as its reference object. SMIL supports this functionality using *explicit syncbase timing*. An explicit syncbase is a named element (within the same host document) that has a temporal context, and which is not a child of the element in which it is being referenced. (This is less complex than it reads.) A

syncbase timing reference contains a temporal event that is used as the scheduling base for the referencing object. This timing reference can be further modified with a clock value. In order to fully appreciate the role of syncbase timing, we first need to consider the elements and attributes defined for less complex operations, but to give a taste of what's ahead, consider the following fragment:

```
0)   <SMIL ...>
1)   ...
2)    <video xml:id="a" src="a.mpg" />
3)   ...
4)    <img xml:id="name" begin="a.begin+10s" end="a.end" src="name.png" />
5)   ...
6)   </SMIL>
```

This element on line 2 starts an associated video object. In some other part of the document, an image containing a text label is started 10 s after the video begins. The label remains visible until the end of the video.

Syncbase timing can be very complex, and its use in simple documents is rare. Still, for certain applications, it can be a powerful construct.

13.5.6 Basic Time Containers

SMIL supports the <par>, <seq>, and <excl> elements.

Element: <par>

The most general of SMIL's timing containers is the <par>. The <par> defines a local time container that can be used to activate one or more child elements. The children of a <par> container are all rendered in "parallel." In terms of SMIL's timing model, this does not mean that they get rendered at the same time, but that they share a common syncbase defined by the <par> container. Any or all of the children may be active at any time that the parent <par> is active.

The basic timing structure of the <par> is illustrated in Fig. 13.5. The default syncbase of the <par> is the beginning of the element. That is, by default, all children of a <par> element start when the <par> itself starts. (This is illustrated by nodes "a" and "b".) As defined above, timing attributes can specify other begin times. By default, the <par> ends when the last child ends, although the exact ending behavior of the <par> is determined by the endsync attribute.

Element: <seq>

A relatively unique timing container is the <seq> element. The children of a <seq> are rendered in such a way that a successor child never can begin before its predecessor child completes. In other words, the syncbase of each child is the *end*

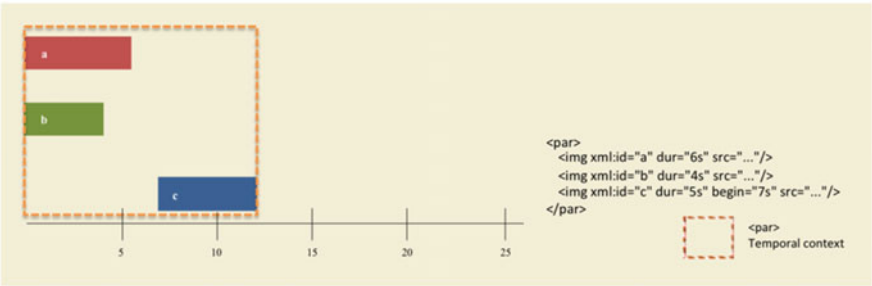


Fig. 13.5 The temporal scope of the <par> element

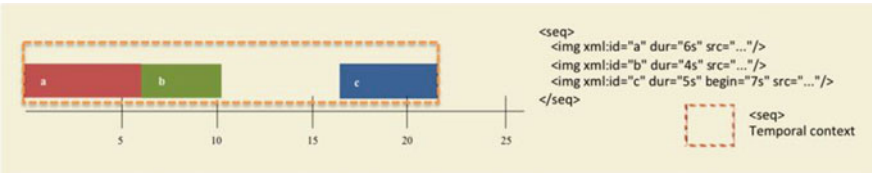


Fig. 13.6 The temporal scope of the <seq> element

of the active duration of its predecessor. A successor element may have an additional start delay, but this delay cannot resolve to be negative in relation to the end time of its predecessor. The <seq> ends when its last child has ended. Unlike the <par>, the <seq> does not support the endsync attribute: Since there is only one child active at a time in a <seq>, there is no need to select among children to determine the container's end.

The basic timing structure of the <seq> is illustrated in Fig. 13.6. The <seq> is especially useful when describing timing in a non-deterministic environment; by specifying that a set of elements logically follow one another, a delay in one element can be easily passed to its successors.

Starting with SMIL 3.0, a seemingly minor change was made to the <seq> container: The restriction that only a nonnegative offset could be used as the value for the *begin* attribute was removed from the specification. This allowed children the <seq>, just as <par> and <excl>, to have event-based starting times rather than only fixed scheduled begins. As with many seemingly simple changes, the impact of this was non-trivial.

Element: <excl>

An <excl> container will start at the temporal moment defined by its parent time container, modified by the value of the begin attribute. The begin time determination for the <excl> is identical to that of the <par>. The critical difference between a <par> and an <excl> is that at most one child of an <excl> may be

active at any one time. Nothing may violate this condition. Under certain circumstances, inactive children of an <excl> may be in a paused state and may respond to certain UI events. For a temporal perspective, they are not active during these periods.

Unlike the children of a <par>, the children have a default begin time of indefinite. This means that, unless the time gets resolved during the active duration of the <excl>, they will never begin. The children of an <excl> may make unrestricted SMIL temporal semantics to define their begin times. As with the <par>, no content may be rendered before the start of the active duration of the <excl>. If a continuous media element is scheduled to begin before the start of the parent <excl>, only that portion that falls within the temporal scope of the <excl>'s active duration will be rendered.

13.5.6.1 Dealing with Cycles and Unknown Begin Times

It is possible to define a cycle by having the begin times of a collection of objects depend entirely on the begin or end of each other. In these cases, it will be impossible to ever define a resolved begin time. Usually, none of the elements in the cycle will be activated; a SMIL player may reject to start a presentation if a cycle is detected.

If begin times are unresolved, they cannot be used to build a timing instance for the object. Elements may become resolved during the active duration of their parent, but if they are not resolved, they are ignored for timing purposes.

13.5.7 Nested Composition of Timing Elements

The basic time containers can be nested in a presentation hierarchy. That is, any child of either a <par> or <seq> (or <excl>) can be a simple media object or an embedded time container. As in SMIL 1.0, the hierarchy can represent relatively static presentation timing. The introduction of event-based activation/termination in SMIL 2.0 also allows a dynamic activation path to be defined. Most of the children of a time container will influence the timing behavior of that container, but some children—such as the <a> element, or the <switch>—are timing transparent. This means that they do not contribute to the determination of the container's duration.

13.5.8 Special Timing Values

SMIL defines two special timing values that can be used to specify the temporal context of an element. These are *indefinite* and *media*.

Value: indefinite

The *indefinite* value can be used to indicate that a timing dependency (either a begin/end time or duration) is to be determined outside of the element in which the indefinite value appears. While many SMIL authors (and some SMIL implementers!) assume that *indefinite* is synonymous with "forever," it is really synonymous with "I have no local idea …". As we will see later in this chapter, an attribute assignment of *dur* = "indefinite" simply means that some other part of the SMIL specification—either the parent time container or some other element—will determine the element's duration. Only if no other part of the document constrains the duration will an indefinite value result in an unlimited duration, but even then, a value of *indefinite* can never result in a presentation that a user (or user agent) cannot end.

Value: media

The attribute value *media* can be applied to obtain the desired timing value directly from the associated media item. Although it is not often required to use the media value explicitly (mostly because this will be the default behavior), it is sometimes useful to set duration or end time to *media* when you want to exert explicit control in complex timing situations.

13.5.9 Interactive Timing and Events

In the normal course of processing, the activation hierarchy of a SMIL document determines the rendering of document elements. The user can influence the elements selected by using SMIL events. The event architecture allows document components that are waiting to be activated or terminated to actually start or stop. There are several uses of events allowed in SMIL 2.0, but perhaps the most important new semantic introduced in SMIL 3.0 was the combination of events and the begin/end attributes. In further combination with the <excl> element, events provide a very powerful mechanism for conditional content activation.

SMIL also supports a rich hyperlinking architecture. Unlike links in 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 intervening event-based activation is ignored.) This means that all nodes temporally between the source and destination of the link need to be evaluated to see if they would have contributed to the final target play state. The temporal seeking and activation facility allow very polished presentation construction—but its implementation in the agent is not for the fainthearted.

Figure 13.7 illustrates the temporal composition of the example presentation in Fig. 13.1. Note that the <par> and <seq> elements are nested as green and blue

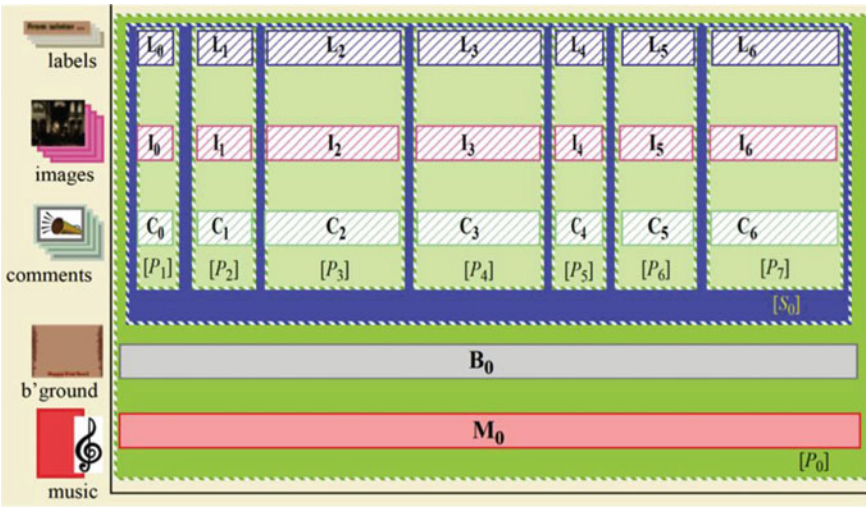


Fig. 13.7 Pure structural representation of the presentation in Fig. 13.1

boxes, respectively. (The node labels of the structure containers are placed in brackets at the bottom of each container.)

13.5.10 Applying SMIL Timing Attributes

SMIL was designed so that obvious structural relationships between elements—and the intrinsic duration of continuous media objects—could be used to specify most timing and synchronization relationships explicitly. There are many occasions, however, when the default synchronization behavior might not be expressive enough to build a particular media message. For this reason, SMIL provides a host of timing control attributes.

13.5.11 General Timing Control Attributes

The general timing control attributes supported by SMIL are *begin*, *dur,* and *end*. The *begin* attribute controls when a particular element starts relative to its syncbase, the *dur* attribute controls the element's simple duration, and the *end* attribute controls the end of the active duration. (In general, *begin* + *dur* = *end*, but this is not always true.)

All three of these attributes retain their behavior from SMIL 1.0. The principal SMIL 2.0 extension to *begin* and *end* is that they attributes support a list of *begin*/

end times instead of a single *begin*/*end* value. This is not particularly useful for simple SMIL applications, but it is very useful if a combination of SMIL's scheduled and event-based timing is used.

Attribute: begin

Each of SMIL's time containers defines a default begin time for its children. The default for <par> is the start of that <par>, while the default for a <seq> depends on the lexical order of its children: The first child starts at the start of the <seq>, and each successive child starts after the preceding child has ended. SMIL allows this default behavior to be overridden by the begin attribute. Usually, this clock value contains a nonzero positive temporal offset, such as:

```
0)  begin="10"
1)  begin="10s"
2)  begin="0:10.0"
3)  begin="0:00:10.0"
```

All of these values represent a begin delay of 10 s. Given a choice, the second version, line 2, is preferred as it gives a reasonable balance between clarity and brevity. Recall that each of these notations defines a 10-s delay from the implied syncbase of the element containing the begin attribute. (This is usually either the start of a parent <par> or the end of a predecessor child in a <seq>.)

The begin attribute and the <seq> element.

There is one restriction on the use of the begin attribute that is important for all SMIL versions: A <seq> time container may only have a single begin attribute value and that value must contain a nonnegative clock value. Multiple begin time lists are not allowed in a <seq>, and neither is explicit syncbase timing. In the following set of statements, line 4 is not valid SMIL, but lines 5 through 7 are valid and provide the same functionality as was intended in line 4:

```
0)   <par>
1)   <audio xml:id="snarf" ... />
2)   <seq>
3)       <img begin="2s" dur="5s" ... />
4)       <img begin="snarf.end+2s" ... />    <!-this is invalid SMIL! -->
5)       <par>
6)           <img begin="snarf.end+2s" ... />    <!-this is valid SMIL! -->
7)       </par>
8)   </seq>
9)  </par>
```

Attribute: dur

The *dur* attribute establishes the simple duration of an element. When used with a media object, it defines how much of that media object will be used. When used with a time container, it defines a limit on the simple duration of that container. If the defined duration is longer than the time container or media object, temporal and

rendering padding will result. The various types of duration qualifiers supported by SMIL were shown in Fig. 13.4.

Attribute: end

A SMIL element ends at the end of its active duration. This is determined by either the element's content or the parent time container. An element can also specify a non-default end time via the *end* attribute. The *end* attribute is very similar to the *begin* attribute.

The *end* attribute provides explicit control over the end of the active duration. As with *begin*, the *end* attribute can specify a single clock value or an explicit syncbase (with optional clock value offset), or a list of values. Unlike the *begin* attribute, there is no restriction on the use of *end* with <seq> elements. Note that even if multiple end times are specified, the associated element will end only once—this is when the first end condition is met.

The following fragments give examples of the use of the *end* attribute:

```
0)  <video src="one-hour-video.mpg" end="25s" .../>
1)  <video src="one-hour-video.mpg" end="1:00:15" .../>
2)  <video src="one-hour-video.mpg" end="snarf.end" .../>
3)  <video src="one-hour-video.mpg"end="35s; snarf.end+5s" .../>
```

The first two of these statements have obvious behavior. Line 2 says that the video will end whenever the temporal element *snarf* ends. (This is an example of syncbase timing.) Line 3 states that the video will end at either 35 s after the start of its rendering or 5 s after the end of the syncbase element *snarf*.

Combined Use of begin, dur, and end

It is possible to combine the use of the *begin*, *end,* and *dur* attributes on an element. Combining *begin* and *dur* usually results in obvious behavior, since they both relate to the simple duration of an object. Since the *end* attribute overrides the default end of the active (not simple) duration, the resulting behavior is not always obvious.

13.5.12 Object Persistence Attributes

Two attributes control the persistence of objects after they have completed their active duration: *fill* and *fillDefault*. In this chapter, we consider simple uses of *fill* behavior.

Attribute: fill

SMIL allows an object to remain visible after the end of its active duration by specifying a *fill* attribute. If *fill* is set to "freeze," the object will remain visible until the end of its parent time container. If it is set to "remove," the object is removed as soon as its active duration ends. The default for visual media is effectively "freeze"; *fill* has no meaning for audio media. There are some special cases for *fill* that allow

the visibility of an object to extend past the active duration of its parent (such as when transitions exist). The value of *fill* also determines how long an object remains "clickable" for linking and interaction.

If the *fill* attribute is applied to a time container, then the rendering state of all of the children (and descendants of the children) is set to their respective fill behavior at the moment the active duration of the container ends.

13.5.13 Extended Timing Control Attributes

SMIL 2.0 introduced two attributes that provide extra duration control: *min* and *max*. These attributes can be used to define a lower or upper bound on the active duration of the containing element, regardless of that element's other timing characteristics. SMIL also supports the ending of a time container via the *endsync* attribute.

Attribute: min

The *min* attribute can be used to specify an explicit minimum active duration on an element. It accepts either a clock value or "media" as values. The default value of *min* is "0," which is the same as saying that the value is unconstrained. The value *media* may only be used on media objects; its use says the minimum duration is the implicit value of the media object.

Attribute: max

The *max* attribute can be used to constrain the maximum active duration of an element. It is similar to *min*, except that an explicit value of indefinite can be specified. (This is also the default.)

The behavior of the *max* attribute is predictable, although finding relevant use cases can be a challenge. For example, consider the following statements:

```
0)  <videoxml:id="a"src="one-hour-video.mpg"max="30s"end="bazinga.end".../>
1)  <video xml:id="b" src="one-hour-video.mpg" end="30s; bazinga.end" ... />
```

Line 0 states that the element *a* will play for a maximum of 30 s or until the element *bazinga* ends (if this is earlier). The same behavior applies to video element *b* (line 1).

Attribute: endsync

Where *min* and *max* provide clock value-based constraints on element timing, SMIL provides the *endsync* attribute to provide logical control on timed objects. As in SMIL 1.0, if a <par> has multiple children, it can specify that the entire <par> ends when the first child ends, when the last child ends, or when a named child ends using the *endsync* attribute. This attribute can be assigned to the <par> and <excl> time containers.

The following fragment illustrates endsync behavior:

```
0)   <par endsync="snarf" >
1)       <video xml:id="bazinga" src="video.mpg" ... />
2)       <audio xml:id="snarf" src="audio.mp3" ... />
3)   </par>
```

The <par> ends when the *snarf* ends; if this is later than the end of *bazinga*, the last frame of *bazinga* will be frozen. The default value of *endsync* is "last." An element with the assignment *endsync* = "all" waits for all of its children to play to completion at least once, regardless of whether they have scheduled or interactive begin times.

The behavior of the *endsync* attribute is intuitive, except when combined with the *dur* and/or *end* attributes. This is because having both an *endsync* and a *dur* or *end* will cause conflicting definitions of element duration. In order to resolve these conflicts, the following rules are applied by a SMIL agent:

- If *dur* and *endsync* are specified on the same element, the *endsync* is ignored.
- If *end* and *endsync* are specified on the same element, the *endsync* is ignored.

13.5.14 Repeating Objects and Substructures

The default behavior of all elements is that they play once, for either an implicit or explicit duration. The SMIL *repeatCount* attribute allows an iteration factor to be defined that acts as a multiplier of the object's simple duration. (The resulting duration is the active duration.) A special value *indefinite* is used to specify that an element repeats continually until its (parent) timing context ends. The *repeatDur* attribute defines duration for all of the repeated iterations.

Attribute: repeatCount

The *repeatCount* attribute makes the element's content sub-presentation or media object to repeat the stated number of times. Its value can be either a number or *indefinite*. A numeric value indicates the media object plays that number of times. Fractional values, specified with decimal points, can be used as well.

The following statement specifies that *snarf* is to be repeated 4.5 times:

```
0)   <audio xml:id="snarf"  src="audio.mp3" repeatCount="4.5" ... />
```

If the associated audio file was 4 s long (its implicit duration), then the active duration of *snarf* is 18 s.

A repeat count may also be applied to time containers. As is shown in the following fragment, the simple duration of the container is repeated for the number of iterations specified by the repeatCount.

```
0)   <par repeatCount="3" >
1)       <video xml:id="bazinga" src="video.mpg" ... />
2)       <audio xml:id="snarf" src="audio.mp3" ... />
3)   </par>
```

In this example, the active duration will depend on the implicit durations of the media items; the simple duration will be the maximum of *bazinga* and *snarf*. The active duration will be the simple duration times three.

Attribute: repeatDur

The *repeatDur* attribute is similar in most respects to *repeatCount*, except that duration is given instead of a multiplier. It states that the element is to repeat its playback until the specified duration has passed. The following statement specifies that *snarf* is to be repeated for 15 s:

```
0)   <audio xml:id="snarf"     src="audio.mp3" repeatDur="15" ... />
```

The actual number of times that the audio object is repeated depends on its duration. If the audio object's implicit duration is 3 s, it will repeat times. If its implicit duration is 15 s, it will be played once. If its implicit duration is 25 s, only the first 15 s will be played.

As with all elements, a timing constraint on a parent container will limit the active duration of the children. As a result, in the following fragment, the audio object will be rendered for 25 s:

```
0)   <par dur="25s">
1)       <audio xml:id="snarf" src="audio.mp3" repeatDur="100s" ... />
2)   </par>
```

13.5.14.1 Combing Repeat Behavior with Other Timing Attributes

It is possible to combine either *repeatCount* or *repeatDur* with other timing attributes. The resulting timing usually results in predictable behavior for simple cases, but sometimes the differences in manipulating the simple, active, and rendered durations within one element can lead to SMIL statements that are not always easy to understand.

The composite behavior is nearly always clear if the differences between the various types of SMIL durations are well-understood. This becomes vital if use is made of interactive behavior in SMIL, or if multiple begin times on objects are used. If you plan to use SMIL Animation—or if the SMIL repeat attribute is used—then a complete understanding of the SMIL timing model is essential. Interested readers are encouraged to consult the standard text on SMIL [6]. As a last resort, the SMIL specification can be consulted [x]: There you will find 156 pages of information on the timing model alone.

13.6 Advanced Timing and Synchronization Attributes

SMIL provides a collection of advanced attributes to control synchronization behavior and to facilitate integration of SMIL into other XML languages. While these attributes are rather esoteric for a complete discussion here, we introduce them briefly in the following sections.

13.6.1 SMIL Synchronization Control Attributes

In a perfect world, all of the defined timing in a specification would be implemented perfectly by a SMIL agent. Unfortunately, the world is not only imperfect—it is also unpredictable. In order to provide some measure of control in the face unpredictably, SMIL provides three high-level synchronization control attributes: *syncBehavior*, which allows a presentation to define whether there can be slippage in implementing the composite timeline of the presentation; *syncTolerance*, which defines how much slip is allowed; and *syncMaster*, which allows a particular element to become the "master" timebase against which all others are measured.

13.6.1.1 XML Integration Support

When used in a native SMIL 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, a mechanism is required to identify timing containers. The SMIL specification does this using the *timeContainer* and *timeAction* attributes.

13.7 Summary and Conclusion

Support for a general model for timing and synchronization SMIL's most important contribution. The basic time container elements <seq> and <par> specify SMIL's simplest temporal relationships: those of playing in sequence and playing in parallel. The basic in-line timing values of the *inline* timing attributes *begin*, *end,* and dur set an element's timing as temporal offsets from the begin time it gets from its time container parent. Syncbase values for the *begin* and *end* attributes synchronize an element with the beginning or ending of other elements. The attributes *repeatCount* and *repeatDur* cause an element's media object to repeat its playback. If, after its start time, duration, and repeated duration, the element ends before the end of its parent, the *fill* attribute defines the persistence of the element in SMIL's timing tree.

All of the various options for timing and synchronization control have given SMIL the reputation—in some circles—to being overly complex. Certainly for developers of HTML5 [10], the impression existed that simpler was better. This reputation is largely undeserved. In SMIL, very simple facilities exist for crafting complex presentation based entirely on implicit timing control. The value of SMIL is that complexity is available when it is needed without having to resort to scripting. SMIL also provides a unified timing control model in which all elements of a document have a common temporal basis. This "complexity" actually simplifies the task of creating interactive presentations.

This book summarizes a number of timing formats that have been applied to controlling content across the World Wide Web. SMIL was the first of these formats to be captured in a set of standards that, for a long time, formed the basis of timing in Web documents. There have been many uses of SMIL in embedded applications, and nearly every smartphone on the planet carries a partial SMIL implementation. Still, in terms of mass adoption at the end-user level, SMIL has never been a great success. This has little to do with SMIL's timing model or its use of structure-based timing. SMIL has suffered from a lack of universally accepted media codecs for video and audio content. This has crippled any hope of interoperability. The irony of this situation is that such interoperability forms one of the core potential successes of SMIL, at least as far as the development of an abstract timing model is concerned.

References

1. W3C, World Wide Web Consortium. http://w3.org/
2. Bugaj, S., Bulterman, D.C.A., Butterfield, B. et al.: Synchronized Multimedia Integration Language (SMIL) 1.0 Specification, June 1998. https://www.w3.org/TR/1998/REC-smil-19980615/
3. Ayers, J, Bulterman, D.C.A., Cohen, A. et al.: Synchronized Multimedia Integration Language (SMIL 2.0), 2nd edn., Jan 2005. https://www.w3.org/TR/2005/REC-SMIL2-20050107/
4. Bulterman, D.C.A., Grassel, G., Jansen, J. et al.: Synchronized Multimedia Integration Language (SMIL 2.1), Dec 2005. https://www.w3.org/TR/2005/REC-SMIL2-20050107/
5. Bulterman, D.C.A., Jansen, J., Cesar, P.S. et al.: Synchronized Multimedia Integration Language (SMIL 3.0), Dec 2008. https://www.w3.org/TR/SMIL/
6. Bulterman, D.C.A., Rutledge, L.: SMIL 3.0: Interactive Multimedia for the Web, Mobile Devices and Daisy Talking Books. Springer (2008)
7. W3C Synchronized Multimedia Working Group. https://www.w3.org/AudioVideo/
8. Bulterman, Dick C.A.: Specification and support of adaptable networked multimedia. Multimed. Syst. **1**, 68 (1993). https://doi.org/10.1007/bf01213485
9. Bos, B. et al.: Cascading Style Sheets Level 2 Revision 1 (CSS 2.1) Specification, June 2011. https://www.w3.org/TR/CSS2//
10. Hickson, I. et al.: HTML-5: A vocabulary and associated APIs for HTML and XHTML, October 2014. https://www.w3.org/TR/html5/