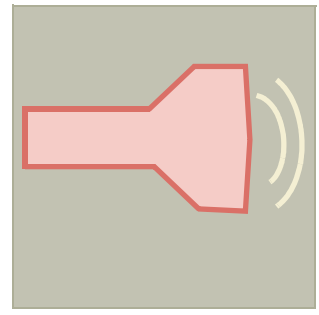
2
Understanding SMIL Code



This chapter provides a reader's guide to the structure and contents of a SMIL presentation. By walking through a complete SMIL example, we'll introduce the main concepts and concerns that you will encounter when you first start making SMIL multimedia documents.

We start with a brief review of basic XML language constructs. We then describe an example presentation (*Flashlight*) and give an overview of SMIL functionality, based on six functional categories:

- SMIL *structure*: the basic framework for encoding SMIL presentations;
- *media content*: the constructs for referencing external media objects;
- *layout*: the facilities for managing rendering resources;
- *timing*: the core control constructs for temporal structuring;
- *linking:* the constructs for link-based navigation and interaction; and
- *adaptivity*: the tools for tailoring presentations to the environment.

Together, these categories form the *what*, *where*, *when*, *how* and *for whom* of SMIL. We end with a description of *Flashlight* in various SMIL profiles.

After you finish this chapter, you should have a broad reading knowledge of SMIL. This will get you ready to build your own presentations using the in-depth information presented in the body of this book.

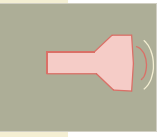
2.1 Understanding XML Structure

XML, or the *eXtensible Markup Language*, is the meta-language for SMIL (and other Web formats, including XHTML). It is a language that describes how other languages should be structured. An XML definition does not, in and of itself, tell you anything about the meaning of any part of a language — it is simply a set of common conventions for language designers.

There are three concepts to be mastered when learning an XML language:

- understanding the *elements* that the language defines,
- understanding the *attributes* that the language defines for those elements, and
- understanding the kind of *references* within (and outside) the document that the language supports.

This section will introduce you to enough of XML to let you understand the *Flashlight* SMIL example later in this chapter. It will also give you most of the details you need to read any SMIL example in our book. If you are hungry for more information on XML, see the references at the end of this chapter.



2.1.1 XML Elements

The basic unit of XML is the *element*. An XML document consists of a set of nested element definitions. The first element in an XML document, at the top of this tree, is called the *root* element.

XML elements consist of a case-sensitive name with no embedded spaces; they are delimited in XML code by *angle brackets*, the '<' and '>' characters. You may not have any white space between the left angle bracket and the name of the element.

The following fragment shows the element **<img>**.

```
<img ... >
```

Here, **img** is the element's *generic identifier*. An element's generic identifier is unique: all elements with the same generic identifier refer to the same XML element definition. An element will usually have one or more attributes associated with it; these are discussed later in this section.

> **Quick Tip**
>
> Since SMIL is strict XML, all identifiers are case-sensitive. Thus, all of the following refer to different elements:
> `<img>, <Img>, <iMg>, <imG>, <IMg>, <iMG>, <ImG>, <IMG>`.
> SMIL only allows lower-case generic identifiers for elements: `<img>`.

Technically, the **<img>** element shown above is not complete because it only indicates that the **<img>** element *begins*. This is the element's *start-tag*. To make it a complete element, you also need to define an *end-tag*. XML provides two types of end tags: one for use with empty elements that have no content, the other for (nested) elements with content.

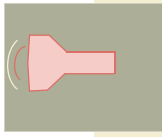
Empty Elements

When an XML element is *empty*, this means that all of the information associated with that element is specified using only the element's generic identifier and its attributes (if any). It doesn't mean that it has no attributes or that the element doesn't reference content! The end-tag for an empty element is represented by the '/' character just before the right angle bracket in the element's definition (with no embedded spaces):

```
<img ... />
```

One of the characteristics that separates SMIL from XHTML is that, for the most part, a SMIL file does not contain media content: it *references* media con-

tent. The only exception is the <smilText> element, which was introduced in SMIL 3.0 to contain text similar to how XHTML does. Media object elements in SMIL point to a file containing media data, but they do not contain the data itself. Occasional use of embedded media is allowed, but this content is placed in an element via one of its attributes. For now, consider all media object references to be empty in XML's sense of the word. The main exception to this convention is embedded timed text, discussed in Chapter 9: *Timed Text in SMIL*.



Elements with Content (Including Nested Elements)

Many elements in an XML language have content. This content is in the form of a nested containment of elements. This nested containing of elements allows the building of a hierarchical structure for a document. In the following example, our <img> element is put inside a <body> element:

```
1  <body>
2    <img attributes ... />
3  </body>
```

By placing these tags around the image element, we indicate that <body> contains the content <img>. In all cases of nesting, the nested time container is closed by repeating the parent element tag, preceded with a '/', as in line 3.

Most of SMIL's timing and control elements can contain any number of other elements, and each child element can itself contain *child* elements, continuing to any degree of depth. In this manner, the *tree*-shaped *hierarchical* containment structure of XML elements — or the element tree — can become quite large and complex. Some minimal variants of SMIL BASIC place restrictions on the depth of nesting to one level, but more recent versions of MOBILE SMIL profiles have removed this restriction.

Quick Tip

Since SMIL is strict XML, all start tags must be matched by end tags. If you don't do this, a SMIL parser must reject your file.

2.1.2 XML Attributes

Each element may have zero or more *attributes*. An attribute assigns a value to a parameter of the element. **All values for attributes must be quoted**!

In our growing SMIL example, we add a timing attribute to <img>:

```
<body>
 <img dur="3s" ... />
</body>
```

Here, we assign a value of "3s" to the attribute named dur of the <img> element. Note that XML doesn't assign a meaning to the attribute name or the value — it simply specifies the syntax. It is SMIL that assigns the meaning *duration* to the dur attribute.

Attribute names are case sensitive and are not allowed to contain embedded spaces. Not all attributes will be used in all elements; the XML specification is used to define which attributes are associated with which elements.

Complex languages like SMIL tend to have many attributes. Some of these attributes are the same as (or similar to) corresponding attributes in other XML languages, others are totally new. In the chapters below, we'll point the major differences between SMIL and other XML languages as needed.

SMIL has many attributes, so it uses long, descriptive names to help clarify the SMIL code. Unfortunately, long names — like *systemdefaultlanguage* — are hard to read. When SMIL 1.0 was designed, hyphens were used to make long attribute names more readable: *system-default-language*. By the time SMIL 2.0 was released, hyphens had fallen from favor and camel casing was used to clarify attribute names: *systemDefaultLanguage*. It is important to understand that the user can't choose between camel casing and hyphens — this is a choice that was made by SMIL's language designers.

Quick Tip

SMIL 2.0 deprecated most of SMIL 1.0's hyphenated attribute names and replaced with camel-case equivalents. You you always use the new name formats, since SMIL 3.0 and later players may no longer recognize the SMIL 1.0 encodings.

2.1.3 XML References

This section describes XML's facilities for specifying internal resources, external files and portions of external files.

Internal References

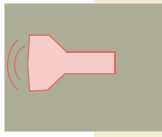
XML elements in different parts of the same document can refer to each other by using *ID (unique identifier)* and *IDREF (unique identifier reference)* attributes. An ID attribute gives its element a unique name. ID attributes typically have the name "`xml:id`", although "`id`" is allowed. An IDREF attribute allows an ID defined elsewhere in the document to be used in an element. For example:

```
<region xml:id="title" ... />
  ...
<body>
  <img dur="3s" region="title" ... />
</body>
```

In this fragment, a **`<region>`** element has the unique identifier attribute **`xml:id`** with a value `title`. Only one element in the entire document can use this particular ID attribute value. The **`<img>`** element can refer to this region by using the IDREF attribute **`region`** with the value `title`.

External Files

Sometimes external files play a role in XML documents. SMIL presentations, for example, use external files to define the media content that plays within a presentation. XML documents refer to external files by assigning their locations as values for certain attributes. For example, assume an image file named *Title.png* exists that contains the words "The GR*i*NS Flashlight". We would reference this file from our SMIL presentation as follows:



```
<region xml:id="title" ... />
  ...
<body>
  <img dur="3s" region="title" src="Title.png" />
</body>
```

The `src` attribute is recognized by SMIL as being an IRI (which is the language independent encoding of a URI, which itself was the standardized encoding of a URL). This attribute's value can be the name of a file, a pathname for the file in the local file system, or a location on the Web.

Portions of External/Internal Files

The external and internal references just discussed can be combined to refer to portions of external files. Such references are usually constructed as attribute values containing the location of an external XML file, followed by a '#' character, then followed by an IDREF of an XML element within that file. The same technique can be applied to internal references, with the exception that the file name is not used. If, for example, we wanted to define an internal link anchor reference to the *intro* section of the current SMIL file, we'd say:

```
<region xml:id="title"/>
  ...
<body>
  <a href="#intro">
    <img dur="3s" src="Title.png" region="title" />
  </a>
  ...
  <par xml:id="intro">
    ...
  </par>
</body>
```

The behavior of links and the behavior of SMIL are not determined by the XML syntax. Instead, the XML syntax is used to describe elements and attributes in a clear and unambiguous manner in an external specification that also specifies behavioral properties.

As we will see, SMIL has a wealth of options for defining temporal attributes that control the activation of elements and media content.

2.2 *Flashlight*: A SMIL Example Presentation

Now that we've discussed the basics of XML, it is time to look at a detailed SMIL example: *Flashlight*. *Flashlight* is based on an interactive multimedia user's guide; it touches on all the major areas of SMIL 3.0 functionality.

The Structure of Flashlight

The presentation schematic for the *Flashlight* presentation is illustrated in Figure 2-1. Here we see that *Flashlight* consists of a set of structured segments that describe how to set up and use the device. The opening segment consists of a logo and a main image, both of which are shown in parallel with an audio track and a set of text captions. The entire content of the opening segment is accompanied by an image that contains link anchors for navigation.

The battery segment shown in Figure 2-1(b) illustrates how to put batteries in the flashlight. The segment is structured very much the same as Figure 2-1(a), except that the main image is replaced by a more complex sub-structure.

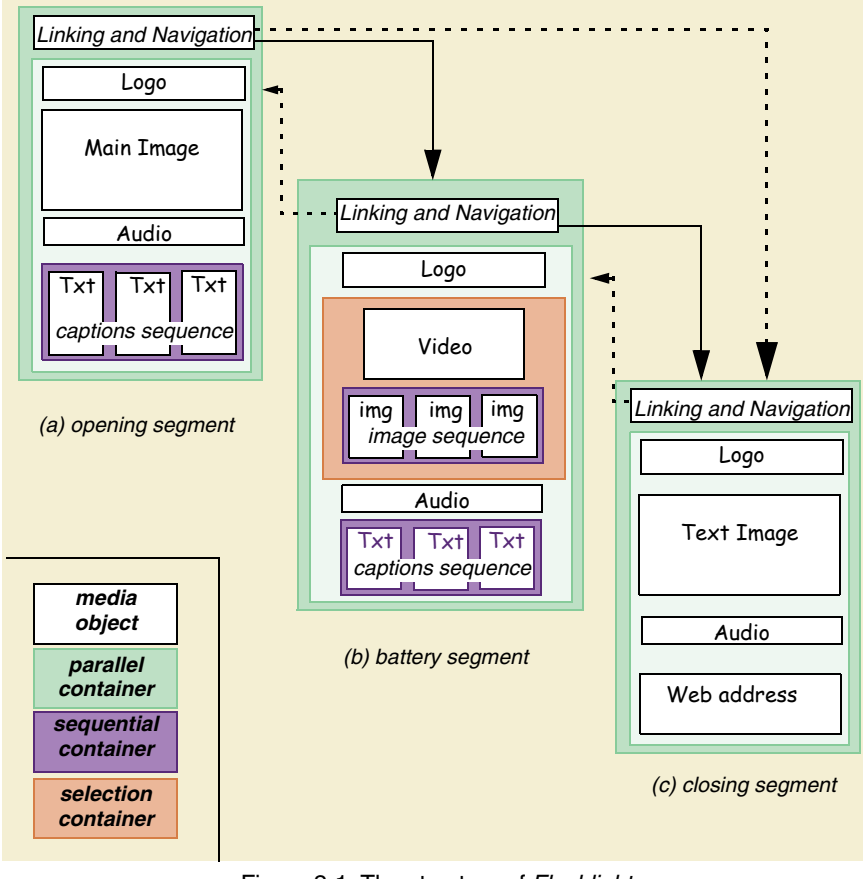


Figure 2-1. The structure of *Flashlight*.

This sub-structure, contained in a red selection box, lets the player decide whether to show a video object or replace it with a sequence of images. (In this presentation, the images will be shown instead of the video object if the bitrate of the network connection is determined by the player to be below approximately 34Kbps.) As with the opening segment, a logo is shown and a set of text captions accompany an audio voice track. Once again, a navigation bar is shown that allows the user to jump to another section.

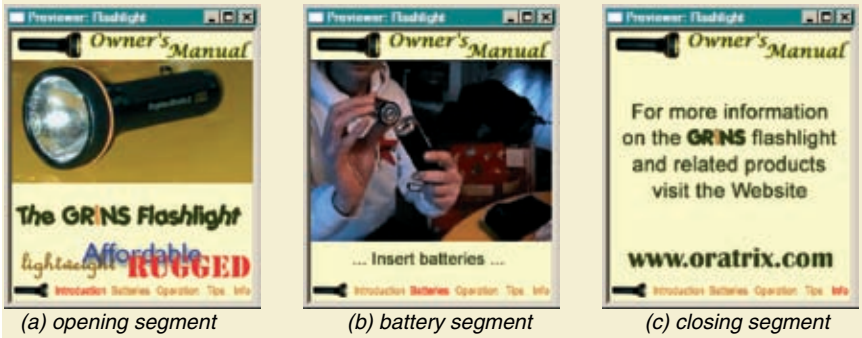
The closing segment of the presentation shows the logo and contains three interior media objects: a main image (containing styled text), a Web address and an audio track. (Since the main image gives all of the important information in text form, a separate captions sequence is not required.) The Web address has an anchor attached to it so that users can click through to the Web site for extra information (and a chance to purchase the flashlight itself.)

At any time during the presentation, a user may navigate to the section that is of interest via a set links on the linking menu bar. (Some of the links are shown in the image) When a link is followed, the presentation context changes to the new subject material — the old context goes away. This is not always necessary: sometimes a new context will augment rather than replace content.

The Runtime View of Flashlight

Once the presentation structure is defined and the media assets are created, the flashlight presentation can be rendered on a SMIL player. One such rendering is shown in Figure 2-2: this figure shows three snapshots of the presentation, one for each main segment. If we had played flashlight in another SMIL player, the outer window will be different (that is, the menu bars and the skin of the player interface will be customized by the player manufacturer), but the content of the presentation should be the same. If it is used on a different type of device, some of the content may be scaled or ignored, depending on device characteristics.

The following sections look at the general structure of the *Flashlight* example and then consider the consequences of playing the presentation on different classes of SMIL players.



(a) opening segment *(b) battery segment* *(c) closing segment*

Figure 2-2. Display of *Flashlight* using the Ambulant Player for SMIL 3.0.

Flashlight SMIL Code

Example 2-1 shows the SMIL code written for the SMIL 3.0 LANGUAGE profile for a subset of the *Flashlight* presentation. If this code seems overwhelming, look back at the presentation schematic on page 30. (If it *doesn't* seem overwhelming, perhaps you should look at it again!) In order to reduce the complexity of this example, we've only included three subsections in the code of *Flashlight*. The entire presentation is available on our book's Web site.

```
<!DOCTYPE smil PUBLIC "-//W3C//DTD SMIL 3.0 Language//EN"
      "http://www.w3.org/2008/SMIL30/SMIL30Language.dtd">
<smil xmlns="http://www.w3.org/ns/SMIL" version="3.0" baseProfile="Language">
 <head>
  <meta name="title" content="Flashlight"/>
  <meta name="generator" content="GRiNS for SMIL 3.0"/>
  <meta name="author" content="Jack Jansen and Dick Bulterman"/>
  <layout>
   <root-layout xml:id="Flashlight" backgroundColor="#ffffcc" width="240"
            height="269"/>
   <region xml:id="Title" left="0" width="240" top="0" height="29"/>
   <region xml:id="Image" left="0" width="240" top="29" height="180"
           z-index="2"/>
   <region xml:id="Text" left="0" width="240" top="209" height="42"
           fit="meet"/>
   <region xml:id="Buttons" left="0" width="240" top="251" height="15"/>
   <region xml:id="Logo" left="0" width="240" top="169" height="42"
           z-index="3"/>
   <region xml:id="Sound"/>
  </layout>
  <transition xml:id="fade1s" type="fade" dur="1s"/>
 </head>
 <body>
  <par>
   <img region="Title" src="M/title.gif"/>
   <seq>

   <!-- This is the opening segment. -->

    <par xml:id="Intro">
     <img xml:id="button-intro" region="Buttons" src="M/b-intro.gif">
      <area xml:id="AIB" href="#Bat" sourcePlaystate="stop"
           coords="99,0,139,15"/>
      <area xml:id="AIM" href="#MI" sourcePlaystate="stop"
           coords="220,0,237,15"/>
     </img>
     <par>
      <img xml:id="I-picture" region="Image" src="M/lamp1.jpg"
           transIn="fade1s"/>
      <img region="Logo" fill="freeze" src="M/FL.gif"
           transIn="fade1s"/>
      <audio xml:id="I-Welcome" region="Sound" src="M/Welcome.mp3"/>
      <seq xml:id="description" begin="2">
       <img xml:id="lightweight" region="Text" dur="1s" fill="transition"
           src="M/lw.gif" transIn="fade1s"/>
       <img xml:id="affordable" region="Text" dur="1s" fill="transition"
           src="M/af.gif" transIn="fade1s"/>
       <img xml:id="durable" region="Text" dur="2s" fill="freeze"
           src="M/du.gif" transIn="fade1s"/>
      </seq>
     </par>
    </par>
```

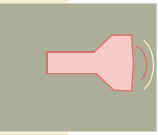
Example 2-1. The SMIL 3.0 Language profile code for *Flashlight*.

```
    <!-- This is the Batteries segment. -->

38    <par xml:id="Bat">
39     <img xml:id="buttons-battery" region="Buttons" src="M/b-batteries.gif">
40      <area xml:id="ABI" href="#Intro" sourcePlaystate="stop"
            coords="39,0,91,15"/>
41      <area xml:id="ABM" href="#More" sourcePlaystate="stop"
            coords="220,0,237,15"/>
42     </img>
43     <par>
44      <switch>
45       <video xml:id="B-vid" region="Image" src="M/Bat.avi" transIn="fade1s"
           transOut="fade1s" systemBitrate="34400" />
46       <seq>
47        <par>
48         <img xml:id="BI-ccw" region="Image" dur="5s" src="M/cCW.jpg"/>
49         <audio xml:id="BA-CCW" region="Sound" src="M/cCW.mp3"/>
50        </par>
51        <par>
52         <img xml:id="BI-insert" region="Image" dur="8s"
             src="M/Insert.jpg"/>
53         <audio xml:id="BA-insert" region="Sound" src="M/Insert.mp3"/>
54        </par>
55        <par>
56         <img xml:id="BI-CW" region="Image" src="M/cw.jpg"
             transIn="fade1s"/>
57         <audio xml:id="BA-CW" region="Sound" src="M/cw.mp3"/>
58        </par>
59       </seq>
60      </switch>
61      <seq>
62       <img xml:id="B-ccw" region="Text" dur="5s" src="M/ccw.gif"/>
63       <img xml:id="B-in" region="Text" dur="8s" fill="transition"
           src="M/insert.gif" transIn="fade"/>
64       <img xml:id="B-cw" region="Text" src="M/cw-sm.gif"
           transIn="fade1s"/>
65      </seq>
66     </par>
67    </par>

   <!-- This is the Closing segment. -->

68    <par xml:id="More">
69     <img xml:id="b-MI" region="Buttons" dur="50s" src="M/b-moreinfo.gif">
70      <area xml:id="AMI" href="#Intro" sourcePlaystate="stop"
          coords="39,7,91,26"/>
71      <area xml:id="AMB" href="#Bat" sourcePlaystate="stop"
          coords="99,6,139,25"/>
72     </img>
73     <par>
74      <img xml:id="moreinfo" region="Image" src="M/moreinfo-F.gif"/>
75      <audio xml:id="Thanks" region="Sound" src="M/Thanks.mp3"/>
76      <a href="http://www.oratrix.com/" >
77       <img xml:id="wlink" region="Text" dur="indefinite" src="M/www.gif"/>
78      </a>
79     </par>
80    </par>
81   </seq>
82  </par>
83 </body>
84 </smil>
```

Example 2-1. The SMIL 3.0 Language profile version of *Flashlight (continued)*.

2.2.1 The Head and Body Sections

Example 2-2 gives an outline form of the *Flashlight* document. All SMIL host-language compliant documents adhere to this general structure, so it can be used as a standard text template — although you'll have to include the correct DOCTYPE and namespaces for particular SMIL versions.



The basic declarations used in a SMIL 3.0 LANGUAGE document are shown in Example 2-2. Line 1 defines the document type to be used by the XML parser for the SMIL document. Note that line 1 forms a single XML statement, but that it ends with a single right angle bracket — not a "/>"! Line 2 identifies the host language of this document as SMIL. The `<smil>` element is used in every type of SMIL document in which SMIL is the host language. In order to differentiate between current and future versions, SMIL 3.0 introduced the use of the `version` and `baseProfile` attributes. If you specify a DOCTYPE with a reference to a valid DTD for a given profile, then default values for `version` and `baseProfile` will be defined in the DTD. If these are specified on the `<smil>` element, they must match those in the DTD. A DOCTYPE isn't required; if one is missing, the two attributes `version` and `baseProfile` must be specified to ensure that the correct version of SMIL is used.

```
1 <!DOCTYPE smil PUBLIC "-//W3C//DTD SMIL 3.0 Language//EN"
        "http://www.w3.org/2008/SMIL30/SMIL30Language.dtd">
2 <smil xmlns="http://www.w3.org/ns/SMIL" version="3.0" baseProfile="Language">
3  <head>
     ...
17  </head>
18  <body>
     ...
83  </body>
84 </smil>
```

Example 2-2. The basic structure of a SMIL presentation.

Earlier versions of SMIL did not have a `version`/`baseProfile` attribute set. SMIL 2.0 used a common DOCTYPE for all host-language versions, and identified the relevant profile by a unique `xmlns` namespace. A typical declaration for our example document would be:

```
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">
```

This namespace specification is required in SMIL 2.0 host language documents, whether they are based on the SMIL BASIC profile or the full SMIL 2.0 LANGUAGE profile (or somewhere in between). Some SMIL files may contain additional namespace declarations; these are used to identify any extensions to SMIL that are required by the document. If an additional or non-standard namespace is used in a presentation, then the DOCTYPE declaration must either be removed or be updated to point to a new DTD that will recognize the namespace additions. The default DOCTYPE declaration may only be used with the standard SMIL namespace definition.

Lines 3 through 17 contain the SMIL <head> section. This section contains definitions that are not part of the temporal flow of the presentation. Among the elements you can expect to find in the <head> are: <meta>, <layout> and <transition>. The <head> section is optional.

Lines 18 through 83 define the SMIL <body> section. This section contains the media and timing control elements in the SMIL specification. The <body> is technically optional, but a SMIL presentation without a <body> is seldom useful. While most XML languages contain a <body> section, SMIL's <body> has an extra feature: it also defines a sequential timing container, just like a SMIL <seq> element. This means that all of the top-level contents of the <body> are evaluated as a temporal sequence by the SMIL player.

The following sections discuss the contents of the <head> and <body> sections in detail. Since, however, the <transition> element introduced in SMIL 2.0 is difficult to place in our partitioning of concepts, we will consider it here before moving to the rest of SMIL. Line 16 of our example contains a definition of a transition resource:

```
16    <transition xml:id="fade1s" type="fade" dur="1s"/>
```

This line defines a named transition (in this case, `fade1s`) which can be used in any media object reference in the body of the presentation. The definition on line 16 does not cause a transition to occur — it simply defines all of the parameters associated with a potential transition in one place. The transition can be applied to a particular media object by adding an attribute for an input or output transition (or both) to an object. For example, line 45 contains the following video reference:

```
45          <video xml:id="B-vid" region="Image" src="M/Bat.avi" transIn="fade1s"
                transOut="fade1s" systemBitrate="34400" />
```

Line 45 defines both an input (`transIn`) and an output transition (`transOut`). Since both reference the same transition definition on line 16 (`fade1s`), both will have the same transition behavior. This form of indirect attribute assignment reduces the complexity of the specification. Indirect attribute assignment is also used for SMIL layout and for aspects of SMIL content control.

Quick Tip

SMIL allows you to define transitions on all types of media; keep in mind that on some platforms, placing transitions on certain types of intensive media (such as video) can be computationally expensive: it may look great on the desktop, but it may put off users on mobile devices.

2.2.2 Media Content

The 'I' in SMIL stands for *"Integration"* — SMIL rarely defines media, but instead integrates multiple media that already exist into a single presentation. The primary SMIL construct for retrieving media content is the <ref> element.

The <ref> specifies a general object reference. In order to aid reading of a SMIL document, most profiles allow <ref> to have the synonyms <img>, <video>, <audio>, <text>, <animation> and <textstream>. The SMIL language does not enforce that an object referenced via a synonym is of the type suggested by the tag (for example, that an <img> tag actually references an image) — that's why all of these element identifiers are only synonyms to <ref>.[1]

Line 20 contains a simple reference to a media object:

```
20    <img region="Title" src="M/title.gif"/>
```

This line requests that the media file **title.gif** is shown in the layout region `Title`. One interesting aspect of this example — from a timing and synchronization perspective — is that the display request does not contain any start or end times, or any duration! Unlike a lot of presentation languages, SMIL can often compute these times based on other information in the specification. This makes creating and maintaining a presentation easier than if every object had its timing explicitly specified.

Note that in this example, we don't use the SMIL 3.0 <smilText> functionality for defining headlines and captions. We do this to maximize compatibility with earlier versions of the language. For an example of how <smilText> might have been used, read Chapter 9: *Timed Text in SMIL*.

Lines 20, 23-26, 28-30, 32-34, 39-42, 45, 48-49, 52-53, 56-57, 62-64, 69-72 and 74-75 and 77 all define media object references. Most references will have attributes to define the source of the media (**src**) and a layout region for rendering the media (**region**). They may also include attributes for specifying transitions or alternative content, or explicit timing information when necessary.

It is important to remember that the SMIL file contains no actual media content — it only contains pointers to content. This makes the SMIL file small and it allows it to be processed independently of any content.

Quick Tip

When an *alt* attribute is used in a SMIL document, its content is never rendered directly in the presentation. Instead, the contents may be used by some players to identify content in a mouse-over balloon, but this is optional behavior. SMIL content control elements should be used to specify alternative content for media objects.

2.2.3 Layout

Most SMIL host-language players (including MMS and SMIL Daisy) use SMIL Layout. Players that implement other profiles usually support CSS layout. Both approaches provide control over object placement and scaling.

1 Versions of MMS require that only the synonyms be used instead of <ref> and that the type of the element (i.e., `img`) match the encoding of the media.

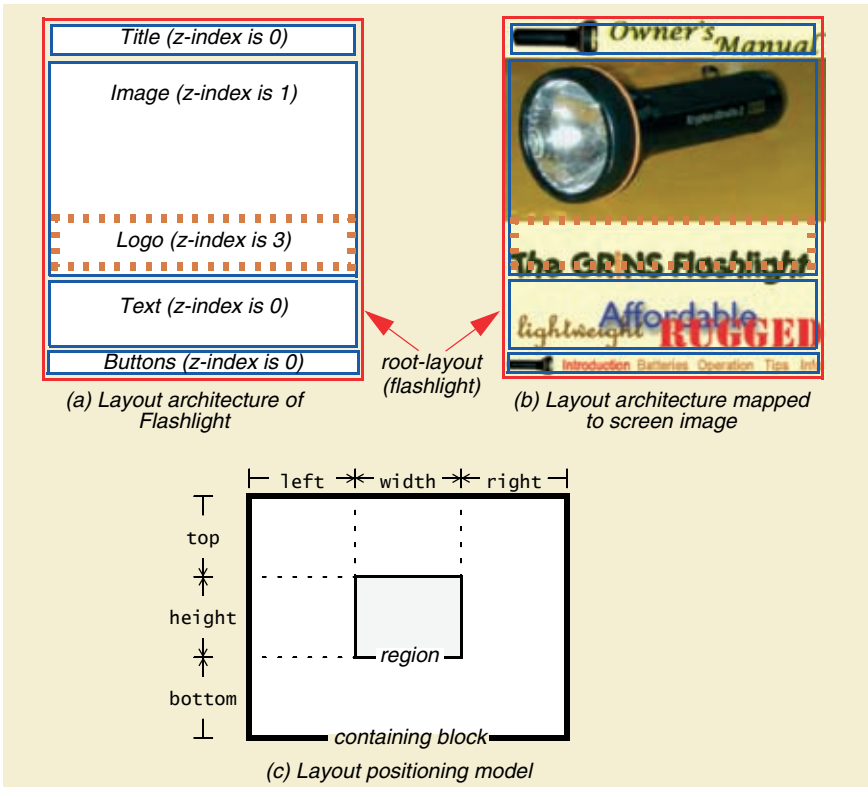


Figure 2-3. SMIL layout and positioning model.

The general layout and positioning model used in the *Flashlight* example is shown in Figure 2-3. SMIL supports the notion of a top-level container window (called either the *root-layout* or *topLayout*, which contains one or more regions. Each region defines a rectangular collection of pixels and a region stacking order (called the *z-index*). In profiles supporting hierarchical layout, the layout section may also support a region hierarchy within parent regions. (This is useful if the parent's position is animated.) Each region can have default behavior specified for media object clipping and scaling, and, in most profiles, it is also possible to specify centering and registration points for media in regions. SMIL also provides support for audio layout, but this support is not extensive.

SMIL layout is geared to the placement of media objects in a fixed viewing environment. That is, unlike an HTML page which can be resized without affecting content, a SMIL presentation layout is usually very media-centric. As a result, SMIL Layout has only indirect support for specifying window border widths and text placement options. (CSS has much more extensive support for these.) While this limitation can sometimes be frustrating, it does help keep SMIL Layout simple enough to be used on low-power devices.

```
<layout>
 <root-layout xml:id="Flashlight" backgroundColor="#ffffcc" width="240"
           height="269"/>
 <region xml:id="Title" left="0" width="240" top="0" height="29"/>
 <region xml:id="Image" left="0" width="240" top="29" height="180"
         z-index="2"/>
 <region xml:id="Text" left="0" width="240" top="209" height="42"
         fit="meet"/>
 <region xml:id="Buttons" left="0" width="240" top="251" height="15"/>
 <region xml:id="Logo" left="0" width="240" top="169" height="42"
         z-index="3"/>
 <region xml:id="Sound"/>
</layout>
```

Example 2-3. Layout section for *Flashlight*.

Example 2-3 shows the layout specification for our sample presentation. The layout contains the root-layout container window (called `Flashlight`) and six media rendering regions. Some of the regions are assigned an explicit stacking order, others take the default of their parent (in this case, the default for the root is '0'). Each region contains positioning information when screen space is used; audio regions do not have positioning information in SMIL. Note that the region `Logo` is spatially embedded on top of the region `Image`. Any content rendered in `Logo` will overlay the content in the overlapping part of `Image`.

Most of the attributes associated with regions are self-explanatory. (If you can't explain them to yourself yet, take a look at Chapter 8: *SMIL Basic Layout*.) One attribute that is new is **`fit`**: this attribute specifies how over- and under-size media objects are rendered. (A value of `meet` means: scale the media in such a way that it fills the region completely.)

Quick Tip

SMIL supports dynamic resizing of media content to the shape of the rendering window. This can be very convenient for multi-platform support, but it can also be computationally expensive for a SMIL renderer. Use this facility with caution if you plan to target low-performance devices.

2.2.4 Timing

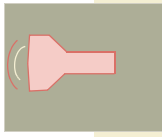
The timing and synchronization functional group represents the core of SMIL functionality. The group is divided into 19 modules, each of which defines a collection of XML elements and attributes that control some aspect of timing.

For any content within a timing element (whether a media object or structure container), the primary issues to be addressed are:

- when does the element begin?
- how long is it active?

- what happens to it when it is no longer active?
- are there other non-timing conditions that cause an element to begin/end?

These questions are "answered" by specifying a set of timing attributes and elements. The timing attribute values are applied to SMIL's temporal elements or to other elements and their children.



Timing Attributes

SMIL 2.0 has an extensive set of attributes to control timing, most of which carry reasonable defaults so that basic timing and synchronization operations can be accomplished easily. Note that not all of the SMIL profiles support all of the attributes, and the syntax of defining and setting the attribute values may vary. This being said, a design objective of SMIL was that each attribute should have a well defined semantic that remains constant across profiles.

Timing Elements

Although SMIL defines a large set of timing attributes, it only introduces three timing elements: `<seq>`, `<par>` and `<excl>`. All three elements represent timing containers: they influence how the timing of their children is defined.

A `<seq>` (sequential) container specifies that its children get processed sequentially: each child, whether it contains media or hierarchical structure, is processed when its predecessor ends. The following excerpt from Example 2-1 shows the top-level sequential behavior in our *Flashlight* presentation:

```
21    <seq>
22      <par xml:id="Intro">
        ...
37      </par>
38      <par xml:id="Bat">
        ...
67      </par>
68      <par xml:id="More">
        ...
80      </par>
81    </seq>
```

The `<seq>` container on line 21 indicates that each of its children (in this case, three `<par>` containers) are to be processed sequentially.

A `<par>` element specifies that its children play in *parallel*. Actually, this is a simplification of SMIL behavior: a `<par>` really says that the default behavior of its children is that they all share a common parent timeline with a common default begin at the start of the `<par>`, but that this default can be changed via SMIL's timing attributes. The `<par>` is the most general SMIL container. The following excerpt from Example 2-1 shows the parallel behavior of one of the `<par>` groups in *Flashlight*. Two images and one audio fragment are rendered starting at the default time of the beginning of the container. Two seconds later, an embedded sequential container becomes active. The `<par>` ends when the last element in the container ends.

```
27      <par>
28       <img xml:id="I-picture" region="Image" src="M/lamp1.jpg"
              transIn="fade1s"/>
29       <img region="Logo" fill="freeze" src="M/FL.gif"
              transIn="fade1s"/>
30       <audio xml:id="I-Welcome" region="Sound" src="M/Welcome.mp3"/>
31       <seq xml:id="description" begin="2">
32        <img xml:id="lightweight" region="Text" dur="1s" fill="transition"
               src="M/lw.gif" transIn="fade1s"/>
33        <img xml:id="affordable" region="Text" dur="1s" fill="transition"
               src="M/af.gif" transIn="fade1s"/>
34        <img xml:id="durable" region="Text" dur="2s" fill="freeze"
               src="M/du.gif" transIn="fade1s"/>
35       </seq>
36      </par>
```

An **<excl>** (or *exclusive*) element is very much like a **<par>**, with one major exception: at most one of the children of the **<excl>** can be active at one time. The easiest way to think about an exclusive container is that it is a **<par>** in which all of the children have conditional begin times. (Usually, this condition is a mouse-click.) When one child is selected, any other child that was playing is immediately de-activated. If another child is selected, it replaces its predecessor. When no child is active, the **<excl>** container ends (unless it has attributes that extend its life). Since **<excl>** is advanced SMIL functionality that is not supported by all SMIL profiles, we defer a discussion until Chapter 15: *Advanced SMIL Timing Behavior and Control*.

2.2.5 Linking

The **<excl>** construct illustrates one form of interactivity in a SMIL presentation. The **<excl>** is useful for controlling the activation of a piece of code conditionally, but there are other useful forms of interaction as well. SMIL linking provides facilities for presentation *navigation* and *branching*: users can move around in the presentation, and they can grab extra information (sometimes from outside the scope of the document) for conditional inclusion into the presentation.

SMIL's primary linking constructs are, as in HTML, the **<a>** element (and its **href** attribute) and the **<area>** element (which gives linking control within an object). Both have very similar meanings to their HTML counterparts except for one major extension: both can add the notion of time to a link.

Links — or more correctly, *anchors* — can have a duration specified during which they are active. In this way, an object can have several links associated with it while the underlying object is active. Timed anchors also control the state of the entire source presentation: when a link is followed, the original document can be paused, replaced, or simply left to keep on running.

Linking is a very powerful feature for adding navigation and conditional content activation facilities to a presentation. Unfortunately, not all SMIL implementations support full SMIL linking functionality completely, so make sure you test your SMIL file carefully on all candidate players.

Two types of linking are shown in *Flashlight*. The following fragment shows internal linking using the **<area>** element:

```
23      <img xml:id="button-intro" region="Buttons" src="M/b-intro.gif">
24        <area xml:id="AIB" href="#Bat" sourcePlaystate="stop" coords="..."/>
25        <area xml:id="AIM" href="#MI" sourcePlaystate="stop" coords="..."/>
26      </img>
```

Line 23 contains a reference to an image. There is no **begin** or **end** attribute defined for this object, so it lasts as long as other objects in the same **<par>** element.[2] Lines 24 and 25 contain **<area>** anchors that are defined as children of this object. Each anchor is defined to cover a certain part of the buttons bar at the bottom of the display area. If an anchor is selected, control is passed to the named part of the document as if a fast-forward or fast-reverse operation had taken place. The behavior of area-based links is similar to the equivalent HTML construct, with the exception that the state of the initial presentation can be explicitly manipulated via the **sourcePlaystate** attribute. Following the link also has a defined temporal behavior and thus influences the presentation timeline.

A second type of link is shown in the following code fragment:

```
76        <a href="http://www.oratrix.com/" >
77          <img xml:id="wlink" region="Text" dur="indefinite" src="M/www.gif">
78        </a>
```

Line 76 shows an anchor that is wrapped around the entire object rather than being defined as part of an object's rendering space. There are no spatial constraints and no explicit playstate settings.

Links in SMIL are similar to those in HTML in most respects, except that SMIL adds a description of the temporal consequences of links. It also provides a set of timing attributes that can be applied to links. Consider the following fragment, which extends the behavior of *Flashlight*:

```
<img xml:id="button-intro" region="Buttons" src="M/b-intro.gif" dur="20s" >
    <area xml:id="AIB" href="#Bat" begin="5s" dur="10s"
     sourcePlaystat="stop" coords="99,0,139,15"/>
</img>
```

The anchor named *AIB* now starts its active period 5 seconds after the image begins; the anchor is then active for 10 seconds. If someone clicks on the image anytime before the first 5 seconds, nothing happens. If someone clicks on the image after 15 seconds (that is, after the 5 second start delay and the 10 second active period of the anchor), nothing happens. However, if the anchor is selected during its 10 second active period, the control will flow to the target of the anchor (in this case, the anchor *Bat*).

2.2.6 Adaptivity

On the Web, each document is available to the entire world. It is thus available to users that have special needs. SMIL contains a suite of content control ele-

2 This is a simplification of what really goes on when elements have no timing.

ments that simplify the task of supporting a worldwide, diverse audience by integrating multiple content streams in a single document.

The <switch> element is the most important SMIL element for adaptivity. At most one child of each <switch> element may be selected for inclusion in the presentation. Each child of the <switch> is evaluated by checking the state of a SMIL *test attribute* against the value of that variable in the SMIL player. The first child that is determined appropriate for playing is played. If none of the children is appropriate, then none is played.

To see how the <switch> works, consider the following fragment from Example 2-1:

```
44        <switch>
45          <video xml:id="B-vid" region="Image" src="M/Bat.avi" transIn="fade1s"
              transOut="fade1s" systemBitrate="34400" />
46          <seq>
47            <par>
48              <img xml:id="BI-ccw" region="Image" dur="5s" src="M/cCW.jpg"/>
49              <audio xml:id="BA-CCW" region="Sound" src="M/cCW.mp3"/>
50            </par>
51            <par>
52              <img id="BI-insert" region="Image" dur="8s" src="M/Insert.jpg"/>
53              <audio xml:id="BA-insert" region="Sound" src="M/Insert.mp3"/>
54            </par>
55            <par>
56              <img id="BI-CW" region="Image" src="M/cw.jpg" transIn="fade1s"/>
57              <audio xml:id="BA-CW" region="Sound" src="M/cw.mp3"/>
58            </par>
59          </seq>
60        </switch>
```

The <switch> on line 44 contains two top-level children: a <video> element and a <seq>. The <video> element contains the systemBitrate attribute, which says: if the system bitrate has been determined (by the player) to be at least 34Kbps, then consider this switch child to be available. The SMIL player will use the first <switch> child that it determines is available for activation. Note that the <seq> does not contain a system test attribute: this means that it is always an acceptable candidate. In this presentation, the image/audio sequence will only be played if the bitrate test for video failed.

System test attributes are usually found on children of a <switch>, but SMIL also supports *in-line test attributes*. In-line test attributes provide a means for conditional acceptance of individual elements (or element trees) of SMIL, while switch-based content control provide a way of specifying alternatives within a document.

2.2.7 Putting it All Together

Now that we've looked at all the major components of the SMIL presentation, we can take a look at the presentation as a whole. You may want to refer back to Example 2-1 while we walk through the total presentation.

Lines 3 through 17 define the SMIL head section, in which the layout of the presentation is defined and in which any transitions are given. This informa-

tion is used by reference in the rest of the presentation. The layout consists of five main windows and one audio region. The transition specifies a one-second fade.

The body of the presentation starts on line 18. The entire presentation consists of a sequence of three parallel elements. The first parallel element defines an image that is presented in parallel with two other images, an audio file and a sequence of three images. This structure is repeated with minor modifications in the other two parallel components. The structure group on line 38 contains the linking image, and then either a video sequence or a set of images and audio. (The choice depends on the evaluation of the switch element.) This is played in parallel with a sequence of three images on line 61. The final parallel group contains a linking image in parallel with a combination of an image and audio on line 73, and an external linking image on line 78.

One interesting way to look at a SMIL presentation is to focus on its structure. Here, we are interested in the big picture, rather than any of the timing details of the individual media objects. Figure 2-4 shows a structure-only representation of *Flashlight*, as rendered in the GR*i*NS Editor. This view uses colors to show the nested presentation components, but it doesn't show any of the timing details. An advantage of the GR*i*NS approach is that, for example, the red block in the center of the image can show the state of a `<switch>` element. In this case, the video is selected and the image/audio sequence is darkened to indicate that it is inactive.

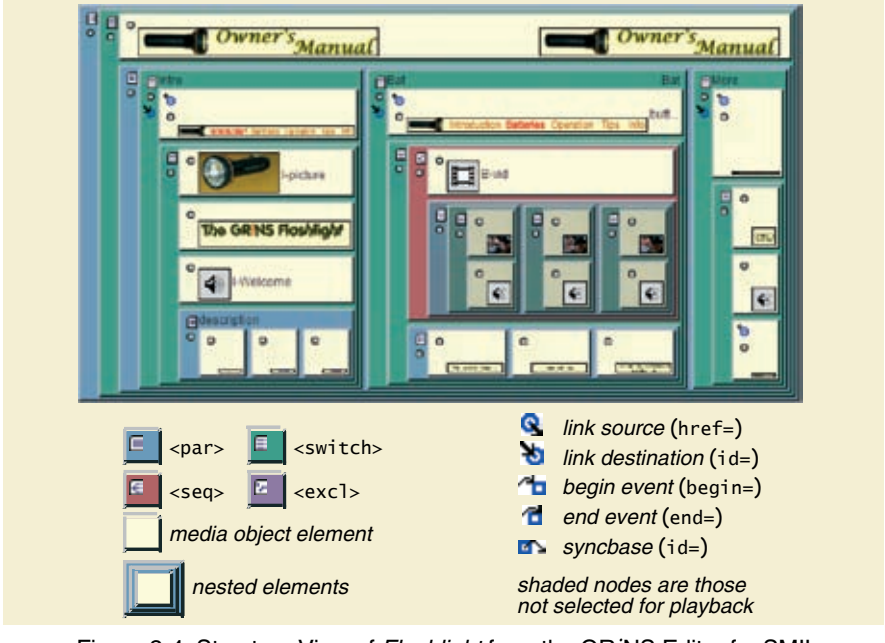


Figure 2-4. Structure View of *Flashlight* from the GR*i*NS Editor for SMIL.

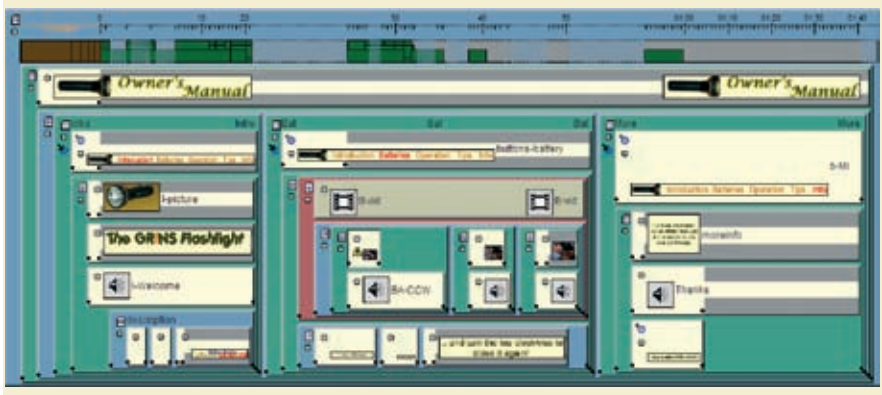


Figure 2-5. StructureTimeline View of *Flashlight* from the GR*i*NS Editor

You should compare the structured view provided by the GR*i*NS Editor with the presentation schematic diagram presented in Figure 2-1 on page 30. Both use an encapsulated view of the structured presentation as a collection of nested time containers and media objects.

Time can also be integrated into a structured representation. Figure 2-5 shows the GR*i*NS structured timeline: here, objects are scaled to their duration (when known), but it is still possible to focus on the presentation structure. Note that the status of the `<switch>` element has changed between Figure 2-4 and Figure 2-5: now the image/audio sequence is active and the video is inactive.

2.3 Encoding *Flashlight* in Different Profiles

The examples of *Flashlight* in the previous section focused on the SMIL 3.0 Language profile. In this section, we look at transformations of *Flashlight* for use with other profiles.

2.3.1 SMIL 3.0 Extensions

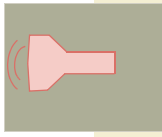
While the DOCTYPE, `xmlns` and the `version`/`baseProfile` attributes all identified the *Flashlight* example as a SMIL 3.0 presentation, the SMIL code does not make use of any extensions introduced in SMIL 3.0 (or even SMIL 2.1). This has been done to keep a relative complex example as simple as possible.

2.3.2 SMIL 2.0/2.1 Language Profiles

Since no SMIL 3.0 extensions were used in *Flashlight*, the only change required to make the presentation compatible with SMIL 2.1 or SMIL 2.0 is to use the correct namespace and DOCTYPE for these versions. Everything else will work without alteration.

2.3.3 SMIL Mobile Versions

There are several SMIL profiles developed for use on mobile telephones. While these provide general guidance to manufacturer and industry standardization groups, most handset vendors do not directly implement SMIL. Instead, the embed SMIL functionality in layered standards that include signally, packaging and transport protocols that are optimized for mobile handsets. One such layer set of protocols is developed by 3GPP: the third-generation partnership platform. One common packaging of SMIL in this environment is via MMS, the multimedia messaging system.



Many early multimedia telephones support the MMS language, version 2. This language restricts a presentation to containing a sequence of media objects, with each object containing a single image, a single text caption and a single audio file. (Not all media objects are required.) This is too restrictive for our *Flashlight* example, but it does work for less complex applications.

The entire *Flashlight* example of Example 2-1 can be viewed without major modification on an MMS player using a 3GPP SMIL compliant version (version 5 or later). 3GPP SMIL version 4 and earlier imposes substantial restrictions on SMIL structure and should probably be avoided. Note that 3GPP used to label their SMIL support with an explicit version number (such as 3GPP PSS5 SMIL), but the version number has been removed in current 3GPP nomenclature. We refer to these more modern versions generically as MMS.

The MMS client is essentially a SMIL 2.0 MOBILE engine with several restrictions. (The differences among profiles are given in Appendix A: *SMIL Profile Architecture Reference*.) The changes that would need to be made to *Flashlight* in order to be fully MMS compliant are discussed in the following sections.

Head and Body Sections

All MMS documents require the SMIL 2.0 namespace declaration shown below:

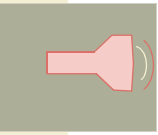
```
<smil xmlns="http://www.w3.org/2001/SMIL20/Language" >
```

If a specific document needs a particular feature from SMIL 2.0 (such as transitions, which may not be implemented in all versions of mobile clients), the following expanded namespace definition could be used:

```
<smil xmlns="http://www.w3.org/2001/SMIL20/Language"
      xmlns:pss5="http://www.3gpp.org/SMIL20/PSS5/"
      systemRequired="pss5" >
```

This definition defines a namespace variable (`pss5`) that is shorthand for the entire 3GPP/PSS5 namespace string. The `systemRequired` attribute informs the SMIL player that it must understand all of the definitions of the 3GPP SMIL namespace to process the document correctly. This can be handy if your SMIL document may be used on a variety of clients (not all of which comply with the full standard). In general, however, it is not necessary to include a specific 3GPP namespace declaration.

The body section includes a transition definition. MMS supports a limited set of transition types: only `barWipe`, `irisWipe`, `clockWipe`, `snakeWipe`, `pushWipe`, `slideWipe` and `fade`. Full-screen transitions are also supported. (*Flashlight* only uses `fade`.)



The rest of the body section of Example 2-1 conforms to MMS. Keep in mind, however, that mobile terminals benefit from small file sizes: the smaller the size, the faster the setup within the handheld device. The only place to really be able to save on SMIL file size is in the use of SMIL's meta-data. *Flashlight* only uses three lines of meta-data — for a total of about 150 characters — but since this meta-data is typically not used by a MMS, other files with lots of meta-data could be stripped before being published. (This is true for PC players, too, but the setup time on desktop systems is rarely a factor.)

Media Content

MMS supports audio, video, images and text, but it does not support all types of media within these types. All MMS players should support AMR audio, H.263 Video, MIDI, GIF, JPEG and plain text. *Flashlight* uses GIF and JPEG images, so these present no problems. For the most restrictive implementations, the AVI video and MP3 audio would need to be converted to the MMS corresponding formats.

Layout

Flashlight uses layout facilities that are compatible with MMS. The outer window size in the original presentation is 269 pixels high and 240 wide. This is acceptable for many modern devices, but it may be too large for the smaller resolutions of older mobile telephones.

There are several alternatives available for making the presentation compatible with all devices:

- *Resize the base presentation*: you could survey all mobile devices and create a layout that is no bigger than the most restrictive presentation environment. This is very safe, but it can lead to frustration among users who have purchased fancier devices (or users of desktop SMIL players).
- *Use device default layout*: if no layout section is defined in a presentation, the SMIL player can place objects where it thinks they fit best. While this sounds attractive, it usually results in all content being stacked on top of all other content. The result is rarely pleasing (or useful).
- *Explicitly allow objects to be scaled*: SMIL provides values for object placement that allow media to be scaled. While this is potentially useful, small devices will often have trouble when scaling images and (especially) video. (With lots of video, the presentation will probably last longer than the battery!)
- *Define multiple layouts using* SMIL *content control*: this is the most useful way of handling multiple devices. SMIL provides a wealth of facilities for providing alternative layouts within one presentation.

```
  <switch>
   <layout systemScreenSize="269X240">
    <root-layout xml:id="Flashlight" backgroundColor="#ffffcc"
                 width="240" height="269"/>
    <region regionName="Title" left="0" width="240" top="0" height="29"/>
    <region regionName="Image" left="0" width="240" top="29"
                height="180" z-index="1"/>
    <region regionName="Text" left="0" width="240" top="209"
                height="42"fit="meet"/>
    <region regionName="Buttons" left="0" width="240" top="251"
                height="15"/>
    <region regionName="Logo" left="0" width="240" top="169"
               height="42" z-index="3"/>
    <region regionName="Sound"/>
   </layout>
   <layout systemScreenSize="200X175">
    <root-layout xml:id="Flashlight-A" backgroundColor="#ffffcc"
                  width="175" height="200"/>
    <region regionName="Title" left="0" width="100%" top="0"
                 height="15%"/>
    <region regionName="Image" left="0" width="100%" top="15%"
                height="60%" z-index="1"/>
    <region regionName="Text" left="0" width="100%" top="75%"
                height="20%" fit="meet"/>
    <region regionName="Buttons" left="0" width="100%" top="95%"
                height="5%"/>
    <region regionName="Logo" left="0" width="100%" top="70%"
                height="20%" z-index="3"/>
    <region regionName="Sound"/>
   </layout>
  </switch>
...
 <img region="Title" src="M/title.gif"/>
```

Example 2-4. Mobile layout section for *Flashlight*.

While a complete discussion of using multiple layouts within a presentation will have to wait until Chapter 17: *Advanced Layout Topics*, the code fragment in Example 2-4, provides an overview of how SMIL handles layout diversity. This example contains two layout sections within a common switch statement. The first section contains the standard layout, which has been designed for a display area of 269 pixels high and 240 pixels wide (or greater). If this room is available, then the base layout will always be used. If a smaller display is available, a second layout is used. This layout is defined for a 4::3 display and uses relative values instead of absolute positioning. It also allows certain media to be scaled. Note that rather than using the `id` attribute to define region names, the `regionName` attribute is used. This allows multiple layouts to be defined that create regions with the same name.

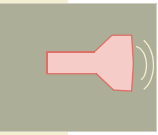
In the body section, a reference to a region is made. The player will select the appropriate layout for the device used during playback.

Timing

Flashlight is fully MMS compatible: it only uses the `<par>` and `<seq>` timing elements (and not the unsupported `<excl>`) and it makes use of only standard timing attributes.

Linking

MMS supports the internal and external linking used in *Flashlight*. The only modification necessary might be to the content displayed in the external Web page:



```
<a href="http://www.oratrix.com/" >
  <img xml:id="wlink" region="Text" dur="indefinite" src="M/www.gif">
</a>
```

Not all MMS players will have full HTML support, and not all will have screens that are large enough to handle generic Web content.

One other potential problem for supporting *Flashlight* is the use of multiple linking targets within `<area>` elements:

```
<img xml:id="button-intro" region="Buttons" src="M/b-intro.gif">
  <area xml:id="AIB" href="#Bat" sourcePlaystate="stop" coords="..."/>
  <area xml:id="AIM" href="#MI" sourcePlaystate="stop" coords="..."/>
</img>
```

The problem is not a lack of MMS support for `<area>`, but rather a (potential) problem with the mobile device's UI: it may not have a pointer device that can easily be navigated over items. In general, use of the `<a>` element is safest, but this does introduce extra overhead in handling (and positioning) multiple images for linking use.

Adaptivity

MMS supports the `<switch>` element as used in *Flashlight* (and in the enhanced layout section just discussed). No further modifications are necessary. One consideration that may be worthwhile is to add additional content control to address the needs of small devices. For example, the presentation could be modified to augment the bandwidth related `<switch>` statement with one that considers display size as well:

```
<switch>
  <video xml:id="B-vid" region="Image" src="M/Bat.avi" transIn="fade1s"
         transOut="fade1s" systemBitrate="34400" systemScreenSize="269X240"/>
  <seq>
    ...
  </seq>
</switch>
```

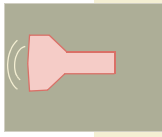
Here, the `<video>` will only be activated if the bitrate is 34400 (or greater) and the screen size is at least 269x240.

Other Considerations

The use of MMS can definitely be classified as emerging. It is likely that there will be a high degree of non-uniformity across devices until the market settles on various devices and screen sizes. You should experiment carefully or use authoring tools that explicitly support a diverse set of devices.

2.3.4 XHTML+SMIL Profile

In order to make a version of *Flashlight* in the XHTML+SMIL profile, we need to change the basic encoding of the document and to replace SMIL layout with CSS layout functions. Example 2-5 (placed on the following two pages) shows one transformation from the SMIL 2.0 LANGUAGE profile to HTML+TIME.[3]



At first glance, the contents and structure look very different from that shown in Example 2-1, but on closer inspection, the similarities are much greater than the differences. The elements and attributes share essentially the same names and the method of defining and manipulating temporal objects within the presentation is also very similar.

The following paragraphs provide an overview of the modifications and limitations of XHTML+SMIL for *Flashlight*.

Head and Body Sections

As with most XML languages, the head section contains general declarations that are used for file parsing and processing, while the body section contains the 'meat' of the content. The new host language for the document will become HTML, since the only implementation of this profile is Microsoft's HTML+TIME. The `<html>` element contains a reference to Microsoft's `time` namespace; this namespace (which is given the shorthand definition '`t`') contains all of the information required to parse the HTML+TIME structure.

```
2 <html xmlns:t="urn:schemas-microsoft-com:time">
3  <head>
4   <style>
5    .time {behavior: url(#default#time2)}
...
12   </style>
13   <?IMPORT namespace="t" implementation="#default#time2">
14  </head>
```

In order to make *Flashlight* HTML+TIME compliant, several fundamental declarations need to be made in the `<head>` section. The most important ones are the CSS declaration of the `time` behavior on line 5 and the association of the `time2` implementation of HTML+TIME with the `t` namespace on line 13. (The `time2` behavior is the second implementation of HTML+TIME; this implementation supports the bulk of SMIL 2.0 functionality, but not the SMIL 2.1 or 3.0 extensions.) The `<head>` section also contains a number of CSS definitions that are used for layout — more on this in the Layout section below. Note that HTML+TIME does not place transition definitions in the head section.

The structure of the body section is a mix of standard HTML and HTML+TIME extensions. Each of the elements in the body that are to be interpreted as HTML+TIME have the prefix `<t:>`. This lets the parser (and the browser) know that SMIL 2.0 functionality should be associated with the element rather than HTML behavior. In order to understand the distinction

3 The bulk of this version was generated by the GRiNS/SMIL 2.0 Pro editor. Hand-editing may yield a different structure.

between HTML and SMIL (HTML+TIME) code, consider the following fragment:

```
17    <t:seq>
18     <t:par>
19      <t:img src="M-HT/title.gif" class="R-Title"/>
20      <t:seq >
21       ...
130     <t:seq >
131    <t:par >
132   <t:seq >
```

All of these statements are preceded with the `t` namespace qualifier. The `<t:par>` and `<t:seq>` elements are obviously SMIL related; they have no HTML equivalents. The `<t:img>` element on line 19 is an example of an element which has an HTML equivalent type (that is, HTML also has its own `<img>` element, with an HTML-defined structure and behavior). The `<t:img>` specification says that the SMIL definition should be used instead of the HTML one.

```
1  <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
2  <html xmlns:t="urn:schemas-microsoft-com:time">
3   <head>
4    <style>
5     .time {behavior: url(#default#time2)}
6     .R-root-layout {position:absolute;overflow:hidden;
                left:0;top:0;width:240;height:269;background-color:#ffffcc}
7     .R-Title {position:absolute;overflow:hidden;
                left:0;top:0;width:240;height:29;}
8     .R-Image {position:absolute;overflow:hidden;
                left:0;top:29;width:240;height:180}
9     .R-Buttons {position:absolute;overflow:hidden;
                left:0;top:251;width:240;height:15}
10    .R-Text  {position:absolute;overflow:hidden;
                left:0;top:209;width:240;height:42}
11    .R-Sound {}
12   </style>
13   <?IMPORT namespace="t" implementation="#default#time2">
14  </head>
15  <body>
16   <div id="Flashlight" class="R-root-layout">
17    <t:seq>
18     <t:par>
19      <t:img src="M-HT/title.gif" class="R-Title"/>
20      <t:seq >
21       <t:par id="Intro" >
22       ...
45       <t:/par>
46       <t:par id="Bat">
47        <t:par>
48         <t:switch>
49          <div class="time" timeContainer="par">
50           <t:video id="B_vid" src="M-HT/Bat.avi"
                 systemBitrate="56120" class="R-image"/>
51           <t:transitionFilter type="fade" subtype="crossfade"
                 dur="1" targetElement="B_vid" begin="B_vid.begin"
                 mode="in" from="0" to="1"/>
52           <t:transitionFilter type="fade" subtype="crossfade"
                 dur="1" targetElement="B_vid" begin="B_vid.end-1"
                 mode="out" from="0" to="1"/>
53          </div>
```

Example 2-5. Excerpt from *Flashlight* as an HTML+TIME presentation.

```
54            <t:seq>
55             <t:par>
56              <t:img id="BI_ccw_m" src="M-HT/cCW.jpg" class="R-image"/>
57              <t:audio id="BA_CCW" src="M-HT/cCW1.mp3" class="R-Sound"/>
58             </t:par>
59             <t:par>
60              <t:img id="BI_insert_m" fill="transition"
                    src="M-HT/Insert.jpg" class="R-Image"/>
61              <t:audio id="BA_insrt" src="M-HT/Insert1.mp3" class="R-Sound"/>
62             </t:par>
63             <t:par id="m31">
64              <div class="time" timeContainer="par">
65               <t:img id="BI_CW" src="M-HT/cw.jpg" class="R-Image"/>
66               <t:transitionFilter type="fade" subtype="crossfade"
                   targetElement="BI_CW" dur="1" begin="BI_CW.begin" ... />
67              </div>
68              <t:audio id="BA_CW" src="M-HT/cw1.mp3" class="R-Sound"/>
69             </t:par>
70            </t:seq>
71           </t:switch>
72           <t:seq>
73            <t:img id="B_ccw" dur="6s" src="M-HT/ccw.gif" class="R-text"/>
74            <div class="time" timeContainer="par">
75             <t:img id="B_in" src="M-HT/insert.gif" dur="8s"
                   fill="transition" class="R-text"/>
76             <t:transitionFilter type="fade" subtype="crossfade"
                   targetElement="B_in_m" dur="1" begin="B_in.begin" .../>
77             <t:par id="m31">
78              <div class="time" timeContainer="par">
79               <t:img id="BI_CW" src="M-HT/cw.jpg" class="R-Image"/>
80               <t:transitionFilter type="fade" subtype="crossfade"
                   targetElement="BI_CW" dur="1" begin="BI_CW.begin" ... />
81              </div>
82              <t:audio id="BA_CW" src="M-HT/cw1.mp3" class="R-Sound"/>
83             </t:par>
84            </t:div>
85           </t:seq>
86           <t:seq>
87            <t:img id="B_ccw" dur="6s" src="M-HT/ccw.gif" class="R-text"/>
88            <div class="time" timeContainer="par">
89             <t:img id="B_in" src="M-HT/insert.gif" dur="8s"
                   fill="transition" class="R-text"/>
90             <t:transitionFilter type="fade" subtype="crossfade"
                   targetElement="B_in_m" dur="1" begin="B_in.begin" ... />
91            </div>
92            <div class="time" timeContainer="par">
93             <t:img id="B_cw" src="M-HT/cw-sm.gif" dur="3s"
                   fill="transition" class="R-text"/>
94             <t:transitionFilter type="fade" subtype="crossfade"
                   targetElement="B_cw" dur="1" begin="B_CW.begin" ... />
95            </div>
96           </t:seq>
97          </t:par>
98         </t:par>
99         <t:par id="More">
           ...
130        </t:par>
131       </t:seq>
132      </t:par>
133     </t:seq>
134    </div>
135   </body>
136 </html>
```

Example 2-4. Excerpt from *Flashlight* as an HTML+TIME presentation (*Continued)*.

One of the challenges of constructing a hybrid HTML/SMIL implementation is the definition of overall temporal control of a presentation. In documents written for the SMIL LANGUAGE profile, it is clear that a single temporal scope governs the entire document. HTML has no corresponding single temporal scope — each cluster of HTML+TIME specification defines an independent local timeline. Various parts of the HTML code are delineated by the HTML `<div>` element. This element is used to define the scope of an encapsulated timeline with the HTML file.

Media Objects

Media objects are referenced using the standard SMIL media elements. As expected, the use of each media reference (such as `audio` or `video`) is prefaced with the "`t:`" namespace qualifier. (See lines 19 and 57 for examples.) If transitions are used on media items, these are implemented using XHTML+SMIL's transition filters. Each transition (input or output) gets its own transition filter specification, which are bundled together with the associated media object in a parallel element.

```
49          <div class="time" timeContainer="par">
50           <t:video id="B_vid" src="M-HT/Bat.avi"
                 systemBitrate="56120" class="R-image"/>
51           <t:transitionFilter type="fade" subtype="crossfade"
                 dur="1" targetElement="B_vid" begin="B_vid.begin"
                 mode="in" from="0" to="1"/>
52           <t:transitionFilter type="fade" subtype="crossfade"
                 dur="1" targetElement="B_vid" begin="B_vid.end-1"
                 mode="out" from="0" to="1"/>
53          </div>
```

In this fragment, a video object (on line 50) is given an input transition (on line 51) and an output transition (on line 52). All three of these statements are bundled into a separate timeline defined by the `<div>` on line 49. (This `<div>` defines a `<par>` time container; while it is valid to use a `<t:par>` element instead, this does not work in HTML+TIME's implementation.)

Layout

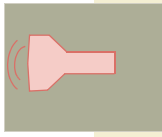
HTML+TIME does not use SMIL Layout. Instead, it uses CSS to position objects. The use of CSS is especially handy for text, but CSS absolute positioning can also be applied to emulate SMIL's Basic Layout functionality. In CSS terms, each of these definitions defines a class that can be used when positioning a media item. The class is referenced in a manner similar to the SMIL region attribute:

```
19          <t:img src="M-HT/title.gif" class="R-Title"/>
```

In our example, several CSS definitions are grouped in the style section; these define areas that correspond to SMIL's layout regions. This is shown in the fragment at the top of the following page.

Matters of syntax aside, there are no fundamental differences in layout capabilities for the translation of *Flashlight* into XHTML+SMIL.

```
13  <style>
14    ...
15    .R-root-layout {position:absolute;overflow:hidden;
                      left:0;top:0;width:240;height:269;background-color:#ffffcc}
16    .R-Title {position:absolute;overflow:hidden;
                      left:0;top:0;width:240;height:29;}
17    .R-Image {position:absolute;overflow:hidden;
                      left:0;top:29;width:240;height:180}
18    .R-Buttons {position:absolute;overflow:hidden;
                      left:0;top:251;width:240;height:15}
19    .R-Text  {position:absolute;overflow:hidden;
                      left:0;top:209;width:240;height:42}
20    .R-Sound {}
21  </style>
```



Timing

XHTML+SMIL provides full support for SMIL 2.0 timing elements. This means that the **<par>**, **<seq>** and **<excl>** elements and their attributes are all supported. XHTML+SMIL supports interactive, event-based timing and also all other major aspects of SMIL time functionality.

This being said, HTML+TIME's implementation of XHTML+SMIL does not always manage the functionality within time containers in a predictable manner. Very often, local timelines need to be defined (using the ** ** element to define a **<par>**, **<seq>** or **<excl>** time container) instead of being able to use the **<t:par>**, **<t:seq>** or **<t:excl>** elements. If you experience timing problems in implementing an HTML+TIME document, we suggest you experiment with both forms until you get the desired behavior.

Linking

The XHTML+SMIL specification provides full support for SMIL's temporal linking capabilities. Unfortunately, the HTML+TIME implementation provides no support for temporal linking. This means that all linking within a document to '**t:**' qualified elements will not work. Only the HTML **<area>** element (not the **<t:area>** element) are supported in HTML+TIME, but even these elements cannot be used to jump inside a time container. The reasons for this are complex: SMIL linking requires that substantial state information gets saved (or generated) regarding the state of the target element at the time the link is activated. Without having some notion of global time, it is very difficult to construct the appropriate state description.

The following fragment gives a description of how linking could be handled within an XHTML+SMIL application. (We use the '**xs:**' namespace qualifier in order not to confuse the fragment with the HTML+TIME implementation.)

```
<xs:img id="buttons_battery_m" src="M-HT/b-batteries.gif"
    class="R-Buttons" dur="20s"/>
   <xs:area id="ABI" href="#Intro" sourcePlaystate="stop"
    shape="rect" coords="39,0,91,15">
   <xs:area id="ABM" href="#More" sourcePlaystate="stop"
    shape="rect" coords="220,0,237,15">
<xs:/img>
```

(Note that SMIL does not require a map element definition, since all of the information needed is contained in the area elements that are children of the `<t:img>` element.)

The lack of linking support in HTML+TIME can, in part, be compensated by structuring the presentation as multiple children of an `<t:excl>` element and then applying event-based timing. We discuss this further in Chapter 15: *Advanced SMIL Timing Behavior and Control*.

Adaptivity

As XHTML+SMIL (and HTML+TIME) provide support for SMIL's basic content control functionality, no substantial changes need to be made to the *Flashlight* application. (See line 48 for an example of the use of the `<switch>` element.) In-line use of text attributes on individual HTML elements is not allowed because HTML has no notion of the `<switch>` element.

One area where SMIL and HTML layout differ is the ability to provide aliases for layout region/class names using the `regionName` attribute. This has a consequence for supporting multiple layouts within a single HTML+TIME document: since the name of the CSS class definition is fixed, it is not possible to define alternative layouts in a style section that use the same class names in each definition. As a result, it is not possible to have the document select a layout at parse time that is then applied to the document at execution time. This means that if multiple layouts need to be supported, multiple specifications of media objects need to be made each time a variable class would be referenced, all of which would have to be wrapped inside of a SMIL `<switch>` element.

Other Considerations

Microsoft's HTML+TIME implementation (using the `time2` behavior) is a solid attempt at integrating SMIL functionality within an existing HTML framework. Our impression is that while the implementation is not fully debugged, the general XHTML+SMIL approach is very useful. The fact that it is integrated into the world's most wide distributed browser means that it is certainly worthwhile to investigate SMIL support within HTML using HTML+TIME. One potential pitfall: at the time this book was written, we could not get clarification of Microsoft's intentions of supporting HTML+TIME in versions of Internet Explorer beyond IE7. You should check Microsoft's pages and documentation.

Quick Tip

Example 2-5 focuses on the overall specification of the presentation and zooms in on the detailed specification of the Battery section; if you want to see the complete presentation, visit `http://www.XmediaSMIL.net`.

2.4 Summary and Conclusions

The purpose of this chapter has been to introduce the structure of SMIL by walking though a complete SMIL example. The application we selected represents a non-trivial collection of elements and attributes. It supports complex SMIL layout structures and it makes use of a variety of SMIL time containers.

The chapter also presented a translation of this example in to various SMIL profiles, including the full SMIL LANGUAGE profile and its SMIL TINY, mobile SMIL and SMIL 1.0 versions. We also looked at creating an XHTML+SMIL version of the demonstration by looking at an implementation using Microsoft's HTML+TIME language.

The first SMIL applications you will write will probably only use a small subset of the features and concepts presented in this chapter. Still, by looking at a large SMIL presentation, we expect that you will have a better idea of how the various pieces of the SMIL language fit together.

Parts Two and Three of this book review each of the elements and attributes provided in this section — plus several constructs that we didn't cover here. Now that you have a general idea of how SMIL works, you can work through the various chapters in this book to get a complete idea on how you can work with SMIL.

Before jumping into the details of the language, we provide an additional background chapter next. This chapter reviews the essential elements of the Web architecture that are relevant to most SMIL presentations. If you are familiar with the Web, you can start with Chapter 4, but a review of common terminology and architecture can be helpful if you aren't exactly sure of the differences between language and protocols, or the distinctions among servers, networks and clients.

2.5 Further Resources

HTML

XHTML 1.0: The Extensible HyperText Markup Language (Second Edition), Steven Pemberton, (ed. chair), W3C Recommendation, 1 August 2002, `http://www.w3.org/TR/xhtml1`.

IRI

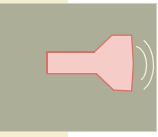
RFC 3987: Internationalized Resource Identifiers (IRIs), M. Duerst, M. Suignard, IETF (Internet Engineering Task Force), January 2005. `http://www.ietf.org/rfc/rfc3987.txt`.

Namespaces in XML

Namespaces in XML, W3C Recommendation, `http://www.w3.org/TR/REC-xml-names/`.

SVG

Scalable Vector Graphics (SVG) 1.0 Specification, Jon Ferraiolo (ed.), W3C Recommendation, 4 September 2001, `http://www.w3.org/TR/SVG`.



XML

Extensible Markup Language (XML) 1.1, Tim Bray, Jean Paoli, C.M. Sperberg-McQueen, Eve Maler, François Yergeau and John Cowan (Eds.), W3C Recommendation, 04 February 2004.
`http://www.w3.org/TR/2004/REC-xml11-20040204/`.

XML Base

XML *Base*, Jonathan Marsh (ed.), W3C Recommendation, 27 June 2001, `http://www.w3.org/TR/xmlbase/`.