

Nokia 7710: Creating MMS Content

Version 1.0; May 24, 2005

Nokia 7710

NOKIA

Copyright © 2005 Nokia Corporation. All rights reserved.

Nokia, Nokia Connecting People and Nokia 7710 are registered trademarks of Nokia Corporation. Java and all Java-based marks are trademarks or registered trademarks of Sun Microsystems, Inc. Other product and company names mentioned herein may be trademarks or trade names of their respective owners.

Disclaimer

The information in this document is provided “as is,” with no warranties whatsoever, including any warranty of merchantability, fitness for any particular purpose, or any warranty otherwise arising out of any proposal, specification, or sample. Furthermore, information provided in this document is preliminary, and may be changed substantially prior to final release. This document is provided for informational purposes only.

Nokia Corporation disclaims all liability, including liability for infringement of any proprietary rights, relating to implementation of information presented in this document. Nokia Corporation does not warrant or represent that such use will not infringe such rights.

Nokia Corporation retains the right to make changes to this specification at any time, without notice.

License

A license is hereby granted to download and print a copy of this specification for personal use only. No other license to any other intellectual property rights is granted herein.

Contents

1	Introduction.....	5
2	Creating MMS Content.....	6
2.1	Using SMIL.....	6
2.2	Editing SMIL Files.....	6
2.3	Defining the SMIL File.....	7
2.3.1	Handling nonmultimedia content.....	8
2.4	Defining More Complex Layouts.....	9
2.5	Timing.....	13
2.6	Interactive Multimedia Messages.....	14
2.7	Linking to Other Content.....	15
2.8	Conditional Presentation Flow.....	17
2.9	Transition Effects.....	18
2.10	Text Formatting.....	21
3	File Types Supported in MMS.....	24
4	Scalable Vector Graphics.....	26
4.1	SVG Tiny Elements.....	26
4.2	Animation.....	27
5	Conclusion.....	29
6	Terms and Abbreviations.....	30
7	References.....	31
8	Evaluate This Document.....	32

Change History

May 24, 2005	V1.0	Initial document release

1 Introduction

This is the third and final document in a series that provides a detailed description of the messaging capabilities of the Nokia 7710 multimedia smartphone and describes how these capabilities can be used to create, send, and receive Multimedia Messaging Service (MMS) messages programmatically.

The three documents are:

- ◆ *Nokia 7710: Messaging APIs* — Introduces the messaging subsystem provided by Symbian OS and describes the fundamental Symbian OS messaging APIs and Message Type Modules (MTMs), as well as how to use these basic messaging APIs.
- ◆ *Nokia 7710: Using The Messaging APIs For MMS* — Describes how to use the Symbian OS messaging APIs to create, send, and receive MMS messages.
- ◆ *The Nokia 7710: Creating MMS Content* — Describes how to programmatically create the content of an MMS message and provides an introduction to Scalable Vector Graphics.

While these documents are based on the messaging and MMS capabilities of the Nokia 7710 smartphone, much of the content is applicable to both Series 60 and Series 80 Platforms and will be useful to developers wishing to create messaging or MMS applications for those platforms as well.

2 Creating MMS Content

No discussion of MMS would be complete without looking at the way in which MMS content itself is created. Unlike Short Message Service (SMS), which is a simple static message, MMS can convey information in a rich, dynamic, and interactive way. A message can present a slide show, link to other content, and even give the user control over the flow of presentation.

This document introduces the concepts used to create an MMS that does more than simply display an image or text or play a sound. It also takes a brief look at Scalable Vector Graphics (SVG), which can be used to deliver rich content in MMS messages without the data overhead that image or video files have.

Note: The techniques described in this chapter exceed the limitations of the Open Mobile Alliance (OMA) MMS specification. Different networks exhibit different behavior when dealing with such extended content. Some networks allow files that do not conform to the OMA MMS specification to be sent, while others remove such content from the message or transcode it to different, supported file types. Therefore, the examples provided in this chapter are not guaranteed to work the same way in every network.

2.1 Using SMIL

The OMA MMS specification defines the use of MMS Synchronized Multimedia Integration Language (SMIL) for the presentation part of a message. The benefit of using this SMIL profile is that all OMA-conforming MMS devices will be able to understand and play the presentation. However, there are a number of limitations of the MMS SMIL, which, while serving to ensure interoperability, at the same time reduce the expressive power of SMIL. The 3GPP suite of specifications defines the 3GPP SMIL profile. This SMIL profile is targeted primarily for streaming purposes (and is defined in the 3GPP Packet Streaming Services version 5 specification, also referred to as the PSS5 SMIL profile), but the 3GPP specification also permits use in MMS. Although 3GPP SMIL does not include all the features of SMIL 2.0, it allows much richer content to be created. The Nokia 7710 smartphone MMS client application supports both SMIL profiles.

The main additional features of 3GPP SMIL include:

- ◆ Unlimited nesting of time containers (in MMS SMIL only one level of `par` elements is allowed)
- ◆ The addition of `seq` and `excl` elements
- ◆ No limit as to the number of elements playing at a time and no restriction on their relative layout
- ◆ The ability to use conditional statements
- ◆ Interactivity using linking (both to internal and external objects)
- ◆ Event-driven timing and synchronization
- ◆ Transition effects.

As long as the content creator is certain that the receiving device is a Nokia 7710 smartphone or some other device supporting the 3GPP SMIL profile, and that the network does not modify non-OMA conforming messages in any way that influences the presentation part, the 3GPP SMIL can be used as the root part of the multimedia message.

2.2 Editing SMIL Files

SMIL files are a formatted plain text file, similar to HTML. As such, developers have a number of tools they can use to create SMIL ranging from dedicated authoring software, to XML editors, to a simple

plain text editor. In the following sections no assumption is made about the tools a developer might use.

2.3 Defining the SMIL File

The file must be set as the root part of the message and its MIME type must be `application/smil`.

The first step in creating a 3GPP SMIL file that can be played in a Nokia 7710 smartphone is to write the prologue, which consists of the XML declaration:

```
<?xml version="1.0" ?>
```

However, the DTD for the PSS5 profile has not been published, so it is not referred to in the prologue. A valid 3GPP SMIL document is also a valid SMIL 2.0 document, therefore the corresponding DTD can be used for validation purposes:

```
<!DOCTYPE smil PUBLIC "-//W3C//DTD SMIL 2.0//EN"
"http://www.w3.org/2001/SMIL20/SMIL20.dtd">
```

The document element, that is `smil`, should follow, and it should also indicate the use of the SMIL 2.0 language via the namespace attribute:

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

The head and body structures should come next.

The example presentation in this chapter will start with a simple slide presentation. Then the example will be enhanced to demonstrate the advanced features.

The following SMIL code creates a two-slide presentation with a picture and text on each slide (this and all the following examples assume that the `Content-Location` header body is identical to the name of the file). The document *Nokia 7710: Using The Messaging APIs For MMS* explains how to set the body of this header programmatically; the `Content-Location` is described in Section 2.2.2, *Message Body*. If an alternative tool is being used to send the message, the tool's documentation should be consulted for instructions on setting the headers.

```
<?xml version="1.0" ?>
<smil xmlns="http://www.w3.org/2001/SMIL20/Language">
  <head>
    <layout>
      <root-layout width="569" height="286" />
      <region id="Text" width="300" />
      <region id="Image" left="300" />
    </layout>
  </head>
  <body>
    <par dur="5000ms">
      
      <text src="text1.txt" region="Text" />
    </par>
    <par dur="5000ms">
      
      <text src="text2.txt" region="Text" />
    </par>
  </body>
</smil>
```

Figure 1 and Figure 2 show the result of playing SMIL in the MMS Player.

Text 1



Figure 1: The first slide of a SMIL presentation using fixed regions

Text 2



Figure 2: The second slide of a presentation using fixed regions

The `text` element is used to insert a text object into the presentation, while `img` is used for an image, `audio` for a music track or voice recording, `animation` for graphics containing animations (for example, animated GIF or SVG files), and `video` for video clips.

The layout coordinates and dimensions, as used in the `region` element, can be specified in terms of either pixels or percentages. It is not necessary to specify the units explicitly, but the `px` suffix can be used: `left="300"` and `left="300px"` are therefore synonymous. If percentages are used, they are always relative to the element's bounding box. Specifying `width="50%"` for a region means that the region will occupy the left-hand-side half of the root layout, because all content is top-left aligned by default.

The root layout, in turn, can also be expressed in absolute pixel values, or it can use the percentages relative to the screen size. If the `root-layout` element is omitted or its dimensions are not provided, it is assumed that the root layout occupies the whole screen. Naturally, care should be taken not to exceed the screen boundaries when defining the layout of the presentation because doing so will result in portions of the message being clipped.

2.3.1 Handling nonmultimedia content

A multimedia message may contain items that are not multimedia items, for example, Personal Information Management (PIM) objects such as vCalendars and vCards, application packages (SIS or Java™ archives), or other files including types not supported by the device. Such objects can be received by Nokia 7710 smartphone, but should not be referenced from SMIL. Even if they are

referenced, they will not be played in the MMS player. Any item that is included in the message but not referenced from SMIL will not be displayed in the MMS Viewer main view, but it will be accessible from the Objects dialog (invoked by the “Objects” menu command) or by tapping on the “attachments” icon. The dialog displayed offers the option to open these nonmultimedia files in their default application, if one exists on the device. For instance, a vCalendar attachment will be opened in the vCalendar viewer, and an application installation package will be opened in the Installer application. The dialog also offers the option to save the object in the file system. Even unsupported items may be saved and transferred to another device or a PC later.

2.4 Defining More Complex Layouts

The first example conforms to MMS SMIL and therefore has the inherent limitation that both slides have exactly the same layout. However, the Nokia 7710 smartphone MMS, being 3GPP SMIL-compliant, does not have this limitation. The message could have the image on the left and text on the right in the first slide, while on the second slide the text could be in the upper part of the screen and the image in the lower. This is easily achieved by defining two more regions. Obviously, the region names “Text” and “Image” are not sufficient in this case, but the Nokia 7710 smartphone is not limited by any specific naming requirements. The modified example is:

```
<?xml version="1.0" ?>
<smil xmlns="http://www.w3.org/2001/SMIL20/Language">
  <head>
    <layout>
      <root-layout width="569" height="286" />
      <region id="Text1" width="200" />
      <region id="Image1" left="200" />
      <region id="Text2" height="100" />
      <region id="Image2" top="100" />
    </layout>
  </head>
  <body>
    <par dur="5000ms">
      
      <text src="text1.txt" region="Text1" />
    </par>
    <par dur="5000ms">
      
      <text src="text2.txt" region="Text2" />
    </par>
  </body>
</smil>
```

Text 1



Figure 3: The first slide of a presentation using a different region layout for each slide

Text 2

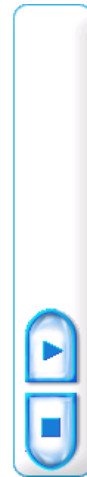


Figure 4: The second slide of a presentation using a different region layout for each slide

Now each image and text item has its own region, laid out independently, as shown in Figure 3 and Figure 4. For messages targeted at the Nokia 7710 smartphone there is no limit to the number of regions per slide (nor, for that matter, is there a requirement to have any kind of slide show). Therefore, it is possible to place several images on the screen at a time. One example where this is useful is when a bigger image is used for the background. For the background images to appear correctly on the screen, the Z-order for the images has to be set so that the background image is at the back of the stack. The Z-order is set explicitly for the image by using the `z-index` attribute on its associated region. The following example adds a background image to the slides:

```
<?xml version="1.0" ?>
<smil xmlns="http://www.w3.org/2001/SMIL20/Language">
  <head>
    <layout>
      <root-layout width="569" height="286" />
      <region id="Text1" width="200" />
      <region id="Image1" left="200" />
      <region id="Text2" height="100" />
      <region id="Image2" top="100" />
      <region id="bgimage" z-index="-1" />
    </layout>
  </head>
  <body>
    <par dur="5000ms">
      
      
      <text src="text1.txt" region="Text1" />
    </par>
    <par dur="5000ms">
      
      
      <text src="text2.txt" region="Text2" />
    </par>
  </body>
</smil>
```

Note that the `bgimage` region is not given any explicit dimension, so it is automatically set to fit its parent geometry, which is the root layout. The image to be used as the background image is assigned to the region with a negative Z-index.

There are many other uses for multiple regions in a presentation.

Sometimes the region dimensions may differ from the intrinsic dimensions of an image, especially when there are multiple different image objects displayed in the same regions. In such cases scaling, clipping, or scrolling (or a combination of them) has to be performed. The `fit` attribute of the `region` element controls the exact behavior. There are five possible values for the attribute:

- ◆ **Hidden** — No scaling. The part of the object that does not fit to the region is clipped. If the object does not fill the whole region (in either one or both dimensions), the background shows through. Figure 5 shows how an oversized image is affected by using this scaling method.
- ◆ **Scroll** — A scroll bar is displayed if the object exceeds the boundaries of the region. Figure 6 shows how an oversized image is affected by using this scaling method.
- ◆ **Meet** — Scale maintaining the aspect ratio. The scaling ratio is selected so that at least one of the object dimensions meets the boundaries of the region while the other one does not exceed them. The background color may show along two sides of the object. Figure 7 shows how an oversized image is affected by using this scaling method.
- ◆ **Slice** — Scale maintaining the aspect ratio. The scaling ratio is selected so that at least one of the object dimensions meets the boundaries of the region while the other one exceeds them. Background color will not show around the object and it will be cut in the dimension that exceeded the region boundary. Figure 8 shows how an oversized image is affected by using this scaling method.
- ◆ **Fill** — Scale without maintaining the aspect ratio. All sides of the object will meet the region boundaries. Figure 9 shows how an oversized image is affected by using this scaling method.

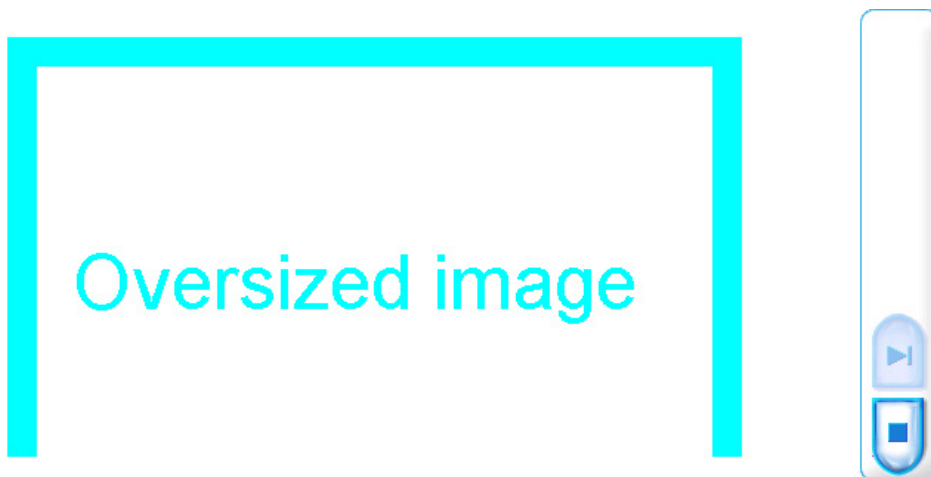


Figure 5: An oversized picture displayed in a region using the `fit="hidden"` attribute

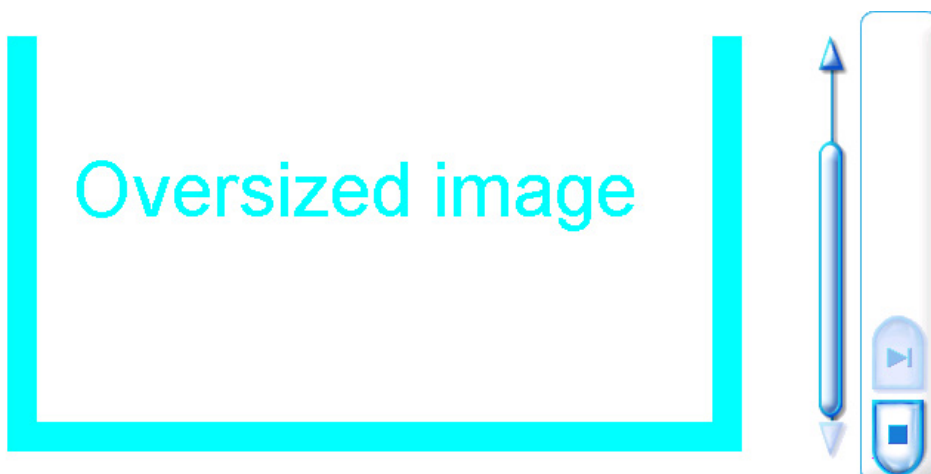


Figure 6: An oversized picture displayed in a region using the `fit="scroll"` attribute



Figure 7: An oversized picture displayed in a region using the `fit="meet"` attribute



Figure 8: An oversized picture displayed in a region using the `fit="slice"` attribute



Figure 9: An oversized picture displayed in a region using the `fit="fill"` attribute

The default value is `hidden`. It is recommended that regions that match the object sizes are used wherever possible, so that no scaling is needed, in order to maximize the performance.

If `hidden` or `meet` methods are used, and if the image contains transparency information, the background color may show through. The color of the regions background is determined by the use of

the `backgroundColor` attribute. The attribute may take either the RGB value or one of the standard textual enumerations defined in the HTML specification, such as `black`, `red`, `green`, etc. For instance:

```
<region id="Image1" left="200" bgColor="navy" />
```

and

```
<region id="Image1" left="200" bgColor="#000080" />
```

provide exactly the same behavior. The `backgroundColor` attribute can also be specified for the root layout element.

2.5 Timing

In addition to the `par` element there are two other time containers in SMIL. The first one is the `seq` element. Unlike `par`, `seq` plays all of its children in sequence, not concurrently. In other words, `seq` can be used to impose simple playback ordering. The body element of SMIL also has the same behavior. The third time container element is `excl`. The `excl` element does not impose any playback order on the child objects, but it ensures that only one of them is active at a time. It is useful in more complex presentations, especially when a higher degree of interactivity is required, as will be seen shortly.

If a media object has its own intrinsic timing (for example a GIF animation, a video clip, or an audio track), it is not necessary to specify its duration explicitly in SMIL. However, objects such as still images, graphics, or text have the duration of zero, so the duration for their display must be specified explicitly either on the media object element or on its time container. If the timing is set for the `par` element, it will also restrict the timing of all elements inside the container. No element can play longer than its time container. For instance:

```
<par dur="5000ms">
  
  <text src="text1.txt" region="Text1" />
</par>
```

will play the `par` time container for five seconds. In the slide that would be created from the example above, if there were an audio track lasting ten seconds, it would be cut in half because its parent time container would not allow it to continue past five seconds. However, the duration of the image and the text element is still zero. The reason why these elements remain visible is that after they stop playing, they remain "frozen." In the case of still images, there is no visible difference between a playing image and a frozen image. Alternatively, the timing can be specified for each object individually:

```
<par>
  
  <text src="text1.txt" region="Text1" dur="7s" />
</par>
```

If timing is specified explicitly for an object with no intrinsic duration, the "freezing" does not occur. The image object in the example above will disappear after five seconds. This behavior can be changed by using the `fill` attribute, which takes one of the following values:

- ◆ **Remove** — The media disappears. This is what happens when the media has intrinsic duration or if the timing is explicitly set.
- ◆ **Freeze** — Keeps the media object displayed after it has been "played." The parent container's time constraints determine exactly how long the element is frozen. For `par` elements it is until the end of the `par`. For `seq` elements it is until the next child starts. And for `excl` elements it is until any other element starts. This is the default value if the element has nonzero duration.
- ◆ **Hold** — Keeps the element displayed until the end of the time container.

- ◆ **Transition** — Keeps the element until the transition is over. This will be described in more detail in the Section 2.9, “Transition Effects.”

Timing attribute values are given in milliseconds (`ms` suffix), seconds (`s` suffix or no unit at all), minutes (`min` suffix), or hours (`h` suffix). Moreover, the duration can be specified in the `mm:ss` format for minutes and seconds or in the `hh:mm:ss` format to also include hours. If the element is to be rendered for an unlimited amount of time, the special value of `indefinite` can be used. In such cases, only pressing the Stop button can interrupt the playback.

By default, the duration of the `par` element is determined automatically so that the element ends when its longest playing child ends. This behavior can be modified by the `endSync` attribute.

If the duration specified in SMIL is shorter than the intrinsic duration of the multimedia object, the playback is clipped. If it is longer, the playback is filled with silence or the last frame of the clip.

Apart from its duration, an object's beginning and end can also be specified. In the simplest form, the beginning (the `begin` attribute) is given in terms of time offset relative to the time container. For example, the following SMIL code displays a slide where the text is shown immediately and the image appears five seconds later:

```
<par dur="10s">
  
  <text src="text1.txt" region="Text1" />
</par>
```

Similarly, the `end` attribute defines when to stop. Note that the timing for the multimedia objects is always relative to their time container (the `par` element in this case).

To permit looping, SMIL defines the `repeatCount` attribute. It defines how many times a multimedia object or a time container is repeated. For instance, the following lines alternate two images in the same region 10 times:

```
<seq repeatCount="10">
  
  
</seq>
```

2.6 Interactive Multimedia Messages

Thus far the examples given have involved only more or less complicated playback, but not any user interaction. However, it is very easy to write a presentation that reacts to user input. The following SMIL code displays an image. Tapping it activates the audio track playback:

```
<smil>
  <head>
    <layout>
      <root-layout width="569" height="286" />
      <region id="Image" height="100%" width="100%" z-index="1"
left="0%" fit="meet" />
    </layout>
  </head>
  <body>
    <par dur="40s">
      
      <audio src="woosh3.amr" begin="i1.activateEvent" />
    </par>
  </body>
</smil>
```

This example uses `activateEvent` to start playback when the object with the ID “i1” is activated, that is, clicked. The same technique can be employed for ending the playback (for example, `end="i1.activateEvent"`). There are also other events on which timing can be based:

- ◆ `beginEvent` — When a multimedia object or a time container begins.
- ◆ `endEvent` — When a multimedia object or a time container ends.
- ◆ `repeatEvent` — When the second or subsequent playback of a multimedia object or a time container is started as the result of a `repeatCount` or `repeatDur` attribute.
- ◆ `focusInEvent` — When a multimedia object gains the keyboard focus (in the case of Nokia 7710 smartphone without a keyboard, an element gets focus when it is tapped).
- ◆ `focusOutEvent` — When a multimedia object loses the keyboard focus.

To make the interactivity example a little bit more interesting, the next sample will demonstrate how to use the `excl` element together with the events. The message will contain two images, and tapping on each will start the playback of a different audio track. The `excl` element will ensure that the tracks are not played simultaneously.

```
<smil>
  <head>
    <layout>
      <root-layout width="569" height="286" />
      <region id="Image1" height="100%" width="50%" z-index="1"
left="0%" fit="meet" />
      <region id="Image2" height="100%" width="50%" left="50%" z-
index="1" left="0%" fit="meet" />
    </layout>
  </head>
  <body>
    <par dur="40s">
      
      
      <excl dur="indefinite">
        <audio src="woosh3.amr" begin="i1.activateEvent" />
        <audio src="teletuppi.amr" begin="i2.activateEvent" />
      </excl>
    </par>
  </body>
</smil>
```

In the example above, the images and the `excl` element are all on the same level inside the `par` element, meaning that all of them are active at the same time. The `excl` element ensures switching between the audio tracks. Whenever a child of the `excl` element is started, all its siblings are stopped. Note that the `excl` element has the explicit indefinite duration to ensure that it is always active.

2.7 Linking to Other Content

Another way to provide interactivity in Nokia 7710 smartphone multimedia messaging is by using links to other content. This feature, which is similar to linking in HTML, allows a URL to be associated with a visible element. On activating the element (tapping on it), the default application for the link will be launched. For example, consider a circumstance where a multimedia message is being sent to notify a user about a new service on a Web site. Rather than requiring the user to copy the address from the message and paste it to the browser application, the user can open the Web page directly from the MMS player with a single tap of the stylus. Suppose that the presentation contains an image `click_to_browse.gif` inviting the user to the Web site. The image element has to be enclosed by the `anchor` element tags and its `href` attribute must contain the address to navigate to. SMIL can look like this:

```
<par dur="indefinite">
  <text src="some text.txt" region="Text" />
  <a href="http://www.my.web.page.address.com">
    
  </a>
</par>
```

In addition to HTML pages, the same approach can be used to open a video stream. To have the RealOne Player application launched after clicking on an object, the `http:` scheme should be replaced with `rtsp:`.

The MMS application in Nokia 7710 smartphone is not limited to `http:`, `https:`, or `rtsp:` — it has the capability to handle many other URL schemas. Another possibility includes interworking with the e-mail application. For example, an image can be used as a trigger to start up the mail editor. For this purpose, the `mailto:` scheme should be used:

```
<a href="mailto:name.surname@company.com">
  
</a>
```

Launching an empty editor is not as attractive as prefilling the different message fields with content. With the `mailto:` scheme such a multimedia message can be composed that on tapping an image, the e-mail editor with the subject is opened, and body and recipients are filled in automatically. In the `mailto:` scheme, the additional headers are separated from the main recipient with the question mark character, and the headers are separated from each other with the ampersand character.

```
<a
href="mailto:name.surname@company.com?cc=the.boss@company.com&bcc=the.pr
esident@company.com&subject=In%20reply%20to%20your%20MMS&body=Yes,%20I%2
0would%20like%20to%20participate.">
  
</a>
```

Note that the space character, as well as many other characters, is reserved in URIs, so it has to be included using the escape sequence `%20`.

Another URI scheme that can be used in the presentation part of MMS is `smsto:`. This scheme is quite similar to `mailto:`, although the subject, the CC and the BCC recipients are not supported, and, of course, phone numbers are used instead of mailboxes. For instance, the following code will send a text message to +1234567890 after the user taps on an image:

```
<a href="smsto:+1234567890?body=Hi,%20this%20is%20a%20semi-
automatic%20reply.">
  
</a>
```

Last, but not least, the Nokia 7710 smartphone is a mobile device, so interactivity would not be complete without messages that can also place a call. This uses the `tel:` URL scheme. In the following example a phone will be dialed when the recipient of the message taps on the text object. Note, however, that clicking on the object pops up a telephone banner. The phone call is placed only after the banner is clicked on. Therefore, it is not possible to dial a number without the user's approval.

```
<a href="tel:+123456789">
  <text src="text1.txt" region="Text" />
</a>
```

In this example a text element was used instead of an image. In the Nokia 7710 smartphone, any clickable object can be used as the anchor for a hyperlink.

A hyperlink does not need to point to an external resource. Links inside the same SMIL document can also be used to provide additional functionality, for example, to create a slideshow that runs

backward as well as forward in time, so that the presentation is interactive and nonlinear. To achieve this, an ID has to be assigned to the timed object. Then, the #ID notation has to be used with the href attribute of the anchor:

```
<smil>
  <head>
    <layout>
      <root-layout width="569" height="286" />
      <region id="Image" height="50%" width="100%" z-index="0" left="0%"
top="50%" fit="meet" />
      <region id="Text" height="50%" width="100%" z-index="1" left="0%"
top="0%" fit="scroll" />
    </layout>
  </head>
  <body>
    <par dur="5s" id="slide1">
      <text src="slide_text.txt" region="Text" />
    </par>
    <par dur="40s">
      <a href="#slide1">
        
      </a>
    </par>
  </body>
</smil>
```

The hyperlink does not necessarily need to point backward in time. If used properly, this mechanism can be used to create very complicated and interactive presentations. There may be, however, cases where the event-based timing (described earlier in Section 2.5, "Timing") is more appropriate.

Sometimes it is unnecessary to create a hyperlink using the anchor element. If a URL, an e-mail address, or a telephone number appears inside running text, the MMS player will detect the link automatically, as shown in Figure 10.

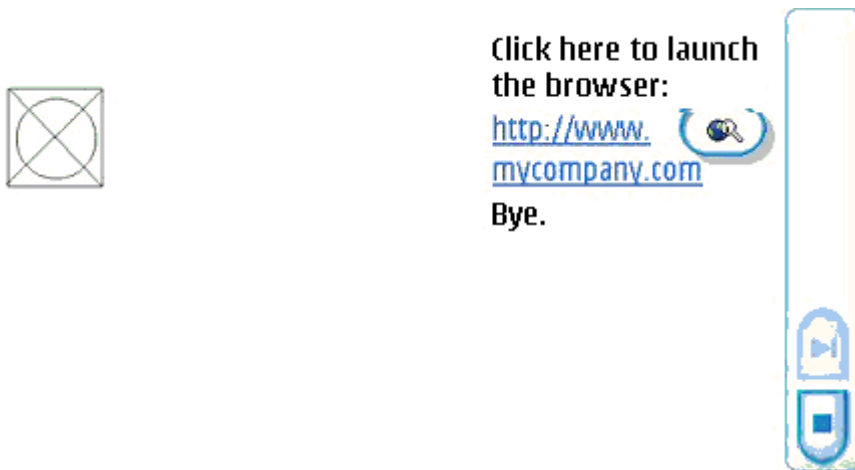


Figure 10: The text attachment in this message is just plain text, but the player automatically detects the hyperlink in it.

2.8 Conditional Presentation Flow

Unlike MMS SMIL, in 3GPP SMIL it is possible to use the `switch` element to provide alternative flows in a presentation based on certain conditions. The `switch` element selects and executes the first SMIL statement matching the criteria. For instance, if the native language of the recipient is unknown the message can be created to present text in the language that matches the device's active language.

To test for the currently used language, the `systemLanguage` attribute is used. The following example will display the same image for all users, but different text files for a French and an English multimedia message recipient:

```
<par dur="10s" >
  
  <switch>
    <text src="text_french.txt" region="Text" systemLanguage="fr-fr" />
    <text src="text_english.txt" region="Text" systemLanguage="en-gb" />
  </switch>
</par>
```

The first statement fulfilling the criteria is executed and all the others are skipped. Therefore, the default element, the one to be executed when no criteria is met, should be placed last. Alternatively by excluding the `switch` element all the statements that meet the match criteria are executed.

The other attributes that can be tested to provide self-adjusting presentations include:

- ◆ `systemCPU` — Evaluates to true if the name of the processor family matches. In the Nokia 7710 smartphone the processor string is `arm` (or `x86` on the emulator).
- ◆ `systemOperatingSystem` — Checks for a specific OS. In the Nokia 7710 smartphone it is always `symbian`.
- ◆ `systemRequired` — Checks if the given module or group of modules is supported by the player. For instance, `systemRequired="http://www.w3.org/2001/SMIL20/BasicTransitions"` evaluates to true if transition effects are supported by the player (as is the case with the Nokia 7710 smartphone). The Nokia 7710 smartphone MMS player can handle all modules required in 3GPP PPS SMIL. The full list of strings that can be used with the `systemRequired` attribute is included in the document *Synchronized Multimedia Integration Language*, available from the World Wide Web Consortium (W3C) Web site at <http://www.w3.org/AudioVideo>, and *Transparent End-to-End Streaming Service; Protocols and Codecs*, available from the 3rd Generation Partnership's Web site at <http://www.3gpp.org/>.
- ◆ `systemColorDepth` — Evaluates to true if the device is able to display graphics with the specified bit depth. May be used to show different images to the user depending on the display capabilities.
- ◆ `systemScreenSize` — Checks if the presentation of the given dimensions can be displayed. For instance, `systemScreenSize="600X200"` will be true if at least 600 horizontal pixels and 200 vertical pixels are available for the presentation. This test attribute can be used inside the layout element in the presentation head to provide alternative layouts for different devices.
- ◆ `systemComponent` — Checks if the capability referred to by this attribute's value is available. Any example of usage appears in Section 2.10, "Text."

2.9 Transition Effects

Thus far, the examples given in this chapter have indicated how to set timing for elements so that objects appear and disappear to be replaced by other objects. However, changes do not need to be abrupt — they can occur gradually by using transition effects. With the transition effects module, sophisticated visual effects can be created without the need to code them as part of a video clip or animation, which can help considerably in constraining the size of the MM.

To use a transition, it first has to be defined in the `head` section of SMIL, which is done using the `transition` element:

```
<smil>
  <head>
    <layout>
      <region id="Image" />
```

```

    </layout>
    <transition type="clockWipe" subtype="clockwiseNine" dur="3s"
id="tansion1"/>
  </head>
  <body>
  <seq repeatCount="10">
    
    
  </seq>
  </body>
</smil>

```

The `transIn` attribute is then used on any objects where the defined transition is to be used to transition in the object during a presentation. If the transition is to occur at the end of the element's activity time (transition out), then the `transOut` attribute is used instead. The transition has its own duration, which is independent from the object (image, text) duration. The time taken by the transition effect does not add to the normal playing time of the object. In the example above, the `image1.jpg` is shown for five seconds, then it disappears and the `image2.jpg` transitions in with the clock wipe effect. However, in most cases it would be better if the first image were not to disappear and the other one appear out of an empty background, but rather the images' transition is used to move from one to the other. To achieve this, the `fill="transition"` attribute has to be used with the first object. It will cause the object to be frozen for the duration of the transition.

The following code causes the first image to transition directly into the second one:

```

<smil>
  <head>
    <layout>
      <region id="Image"/>
    </layout>
    <transition type="clockWipe" subtype="clockwiseNine" dur="3s"
id="tansion1"/>
  </head>
  <body>
  <seq repeatCount="10">
    
    
  </seq>
  </body>
</smil>

```

An example of two images transitioning using the snake effect is shown in Figure 11.

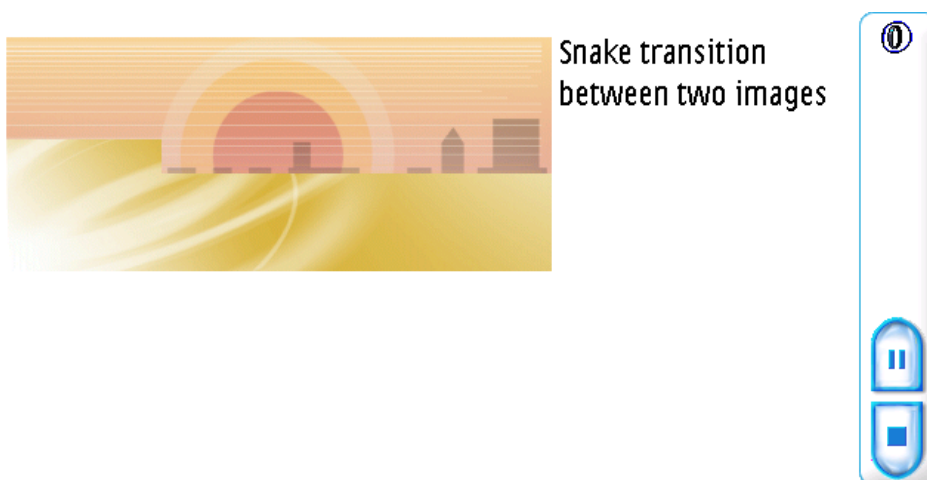


Figure 11: The effect of using the `type="snakeWipe"` attribute with the transition element

The transition-effect types and subtypes that can be used with the Nokia 7710 smartphone MMS are listed in Table 1. To select the main type, use one of the types listed in the table as the value of the

`type` attribute of the transition element. Use the `subtype` attribute to choose a transition subtype, otherwise the default will be used. The `direction="reverse"` attribute can be used to make the transition occur in the opposite direction (this only affects the geometric appearance of the transition effect, and not the order of the transitioned objects).

Type	Available Subtype
BarWipe	leftToRight (default) topToBottom
clockWipe	clockwiseTwelve (default) clockwiseThree clockwiseSix clockwiseNine
Fade	crossfade (default) fadeToColor fadeFromColor (a synonyme of fadeToColor)
pushWipe	fromLeft (default) fromRight fromTop fromBottom
slideWipe	fromLeft (default) fromRight fromTop fromBottom
triangleWipe	up (default) right left down
arrowHeadWipe	up (default) right left down
pentagonWipe	up (default) down
hexagonWipe	horizontal (default) vertical
irisWipe	diamond (default) rectangle
ellipseWipe	circle (default) vertical horizontal
EyeWipe	horizontal (default) vertical
roundRectWipe	horizontal (default) vertical

starWipe	fourPoint (default) fivePoint sixPoint
miscShapeWipe	heart (default) keyhole
spiralWipe	topLeftClockwise (default) topRightClockwise bottomRightClockwise bottomLeftClockwise topLeftCounterClockwise topRightCounterClockwise bottomLeftCounterClockwise bottomRightCounterClockwise
snakeWipe	topLeftHorizontal (default) topLeftVertical topLeftDiagonal topRightDiagonal bottomLeftDiagonal bottomRightDiagonal

Table 1: The available transition effects

2.10 Text Formatting

According to OMA MMS specifications only plain text can be used in MMS. This is a serious limitation for an otherwise rich messaging technology. Fortunately this limitation does not apply to the Nokia 7710 smartphone, which allow the use of technologies such as XHTML rich text in MMS. But even with plain text, the Nokia 7710 smartphone allows MMS developers to go a step further and add life to the text — by using colors. In the previous section, Section 2.9 “Transition Effects,” the region background coloring was explained. The foreground of text can also be colored. Doing so requires the `param` element, a generic element used to pass media type-specific parameters to the media object. In the case of colored plain text, the `param` element is used to pass the `foreground-color` property to the text element. For instance, the following displays red text on a yellow background:

```
<smil>
  <head>
    <layout>
      <region id="Text" fit="scroll" backgroundColor="yellow"/>
    </layout>
  </head>
  <body>
    <par dur="4s">
      <text src="Red_text.txt" region="Text">
<param name="foreground-color" value="red" />
      </text>
    </par>
  </body>
</smil>
```

The Nokia 7710 smartphone MMS viewer is not limited to the choice of background and global foreground colors. Other text formatting can be achieved as well, thanks to support for the XHTML Mobile Profile. To create rich text content for a multimedia message, an XHTML-specific authoring tool may be used, but a generic XML or even plain text editor will suffice. The XHTML file that will result must be saved with the `.html` extension and encoded as part of the outgoing multipart/related structure with the `application/vnd.wap.xhtml+xml` MIME type. The

rich text part has to be referenced from SMIL just as a plain text part would, that is with the `text` element.

If a message is targeted at several different receiving devices, it can be composed so that it displays rich text content on a Nokia 7710 smartphone and plain text content on devices that support plain text only, provided that all of the receiving devices support 3GPP SMIL. The `switch` element can be used to select the most appropriate content and the `systemComponent` attribute can be used to perform the test for the support for XHTML as follows:

```
<switch>
  <text src="text_rich.html" region="Text"
systemComponent="ContentType:application/vnd.wap.xhtml+xml" />
  <text src="text_plain.txt" region="Text" />
</switch>
```

In this case the rich text version will be displayed on a Nokia 7710 smartphone, and the plain text version will be displayed on devices that do not support XHTML in MMS. The `ContentType: URI` schema can also be used together with the `systemComponent` attribute to test for the support of any other media type.

An XHTML file begins with the XML prologue (just like a SMIL file), the document type declaration with a private identifier, and the root `html` element:

```
<?xml version="1.0" ?>
<!DOCTYPE html PUBLIC "-//WAPFORUM//DTD XHTML Mobile 1.1//EN"
"http://www.wapforum.org/DTD/xhtml-mobile11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <title>Rich text content for Nokia 7710 device MMS</title>
  </head>
```

Note that the markup used for the rich text formatting has to use the `http://www.w3.org/1999/xhtml` namespace. The `head` element has to contain a `title` element in order to satisfy the requirements of the XHTML specification. However, the Nokia 7710 smartphone MMS application does not utilize the document title information in any way.

After the `header`, the `body` element follows, and it contains all the textual information that is to appear in the MMS presentation. Nokia 7710 smartphone formatting features that can be used to enrich the text are listed in Table 2.

Formatting	Element or Attribute
Italic	<code>i</code> or <code>em</code> elements
Bold	<code>b</code> or <code>strong</code> elements
Big font	<code>h1</code> element
Medium font	<code>h2</code> element
Color	<code>style="color: ..."</code> attribute

Table 2: MMS rich text styling features and their mapping to XHTML

The example below shows a full XHTML file that utilizes all the formatting techniques, and Figure 12 shows how it is rendered in MMS Player:

```
<?xml version="1.0" ?>
<!DOCTYPE html PUBLIC "-//WAPFORUM//DTD XHTML Mobile 1.1//EN"
"http://www.wapforum.org/DTD/xhtml-mobile11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <title>Rich text content for Nokia 7710 device MMS</title>
```

```

</head>
<body>
  <div>
    <h1>This line uses the biggest font.</h1>
    <h2>This line uses the medium font.</h2>
    <p>
      This is normal text.<br/>
      <em>This line is italicized (emphasized)</em><br/>
      <strong>This line is bolded (strong)</strong><br/>
      <span style="color: red">Red words </span>
      <span style="color: green">Green words </span>
      <span style="color: blue">Blue words </span>
    </p>
  </div>
</body>
</html>

```

This line uses the biggest font.

This line uses the medium font.

This is normal text.
This line is italicized (emphasized)
This line is bolded (strong)

Red words
 Green words

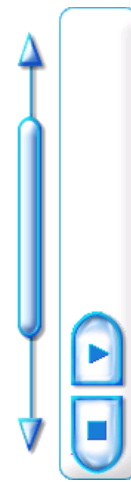


Figure 12: The XHTML rich text formatting used in MMS

The coloring information for the foreground text (using the `style` attribute) can be used in conjunction with the background color of the SMIL region.

3 File Types Supported in MMS

Apart from supporting advanced technologies like SVG, XHTML, and SMIL, the MMS application in the Nokia 7710 smartphone supports many other formats. Table 3 through Table 7 shortlist the types that can be received by MMS.

Type	MIME Type Designation	File Name Extension	Notes
Joint Photographic Experts Group (JPEG)	image/jpeg	.jpg	
Graphics Interchange Format (GIF) 87a and 89a	image/gif	.gif	Also supports transparency and animations.
Portable Network Graphics	image/png	.png	
Windows BMP	image/x-bmp	.bmp	
Scalable Vector Graphics (Tiny Profile)	image/svg+xml	.svg	
Tagged Image File Format (TIFF) [†]	image/tiff	.tif	Only class 0 (fax black and white) is supported.
Wireless Bitmap (WBMP)	image/vnd.wap.wbmp	.wbmp	

Table 3: Graphics formats supported for receiving by the Nokia 7710 smartphone MMS

Type	MIME Type Designation	File Name Extension	Notes
WAVE	audio/wav	.wav	Only PCM is supported.
Adaptive Multi Rate	audio/amr	.amr	
Adaptive Multi Rate Wide Band	audio/amr-wb	.awb	
Motion Picture Experts Group Layer 3 Audio (MP3)	audio/mpeg	.mp3	
Advanced Audio Coding	audio/mp4	.aac	
Third Generation Partnership (3GPP) audio [†]	audio/3gpp	.3gp	3GPP format is a container format that can have many different audio-visual objects inside. For audio, the AMR format is supported.
Musical Instrument Digital Interface	audio/midi	.mid	
Scalable Polyphony MIDI	audio/sp-midi	.mid	
Nokia Ring Tone	application/vnd.nokia.ringing-tone	.rng	

Table 4: Audio formats supported for receiving by the Nokia 7710 smartphone MMS

Type	MIME Type Designation	File Name Extension	Notes
Third Generation Partnership Video Format (3GPP)	video/3gpp	.3gp	3GPP format is a container format that can have many different audio-visual objects inside. For video, the H.263 Profile 0 Level 10 and MPEG-4 Visual Simple Profile Level 0 are supported.

Table 5: Video formats supported for receiving by the Nokia 7710 smartphone MMS

Type	MIME Type Designation	File Name Extension	Notes
Plain Text	text/plain	.txt	
Extensible Hypertext Markup Language Mobile Profile (XHTML MP)	application/vnd.wap.xhtml+xml	.html	
vCalendar	text/x-vcalendar	.vcs	
vCard	text/x-vcard	.vcf	

Table 6: Text and PIM formats supported for receiving by the Nokia 7710 smartphone MMS

Type	MIME Type Designation	File Name Extension	Notes
Synchronized Multimedia Integration Language (SMIL)	application/smil	.smil	Can only be used as the root part of the message for the purpose of the scene/presentation description.
Java application	application/java application/java-archive	.jar	Will be opened by the Application Installer.
Symbian Application Installation Package	application/vnd.symbian.install	.sis	Will be opened by the Application Installer.

Table 7: Other object formats supported for receiving by the Nokia 7710 smartphone MMS

In addition to the files that can be opened with the built-in Nokia 7710 smartphone applications and viewers, there may be additional applications installed in the device. These applications may be able to handle file formats other than those listed in Table 3 through Table 7. The MMS client application allows for the opening any file that has registered a default viewer. Note that the nonmultimedia file types such as PIM objects or application binaries should not be referenced from SMIL.

4 Scalable Vector Graphics

A rich variety of multimedia object types can be inserted into an MMS. The SMIL presentation part, plain text, and rich text do not require a lot of space in the message to express very complex content. However, images and audio tracks often contain tens of kilobytes of data. As of this writing, the vast majority of networks do not support sending multimedia messages larger than around a hundred kilobytes. This implies that only a few medium-quality image objects can be inserted into a message. Therefore, the need for a compact graphic format with rich capabilities is obvious.

One such format is Scalable Vector Graphics (SVG), an XML-based language defined by the W3C. The Nokia 7710 smartphone MMS client application supports the Tiny (SVG-T) profile of this language, which is aimed at mobile devices and is one of two SVG Mobile profiles. Details of the SVG Mobile profiles are provided on the *Mobile SVG Profiles: SVG Tiny and SVG Basic* page of the W3C's Web site (<http://www.w3.org/TR/SVGMobile/>).

The SVG language is a format for encoding 2-D vector graphics, that is, graphics consisting of basic primitives such as points, lines, rectangles, circles, ellipses, and other geometrical shapes. Moreover, text and raster graphics can also be used inside an SVG file. The SVG standard supports animations, too. Because the SVG format is an XML-based language it offers an ideal solution for computer-generated graphics such as charts, maps, and diagrams. The SVG file can be automatically converted from other XML data formats by means of technologies like XSL Transformations (XSLT).

The following sections will show how to create simple SVG graphics for use with the Nokia 7710 smartphone MMS. It is not intended to be a complete SVG tutorial. For the full SVG standard specification, please refer to the *Scalable Vector Graphics (SVG) 1.0 Specification*, which is available from the W3C's Web site (<http://www.w3.org/TR/SVG10>).

As was the case with the XML-based formats described earlier in this document (SMIL, XHTML), the SVG file can be edited with a dedicated editor, a general-purpose XML editor, or even a plain text editor. After the file has been created, it has to be included as a part of the multimedia message with the `image/svg+xml` MIME type. The part has to be referenced from SMIL using the `Content-ID` or `Content-Location` header and the `img` or `animation` elements. As with any other XML-based language, an SVG-T file starts with the XML prologue:

```
<?xml version="1.0" ?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.0//EN"
"http://www.w3.org/TR/2001/REC-SVG-20010904/DTD/svg10.dtd">
```

Following the prologue, there is the root `svg` element using the `http://www.w3.org/2000/svg` namespace. The `version` attribute indicates the version of the SVG specification used. It must be version 1.0:

```
<svg xmlns="http://www.w3.org/2000/svg" version="1.0">
```

4.1 SVG Tiny Elements

The graphic elements are specified using the `rect`, `circle`, `ellipse`, `line`, `polyline`, and `polygon` elements. Every element can have its brush color specified with the `stroke` attribute and the closed objects can have the fill color specified with the `fill` attribute. The width of the brush is determined with the `stroke-width` attribute.

The `rect` element creates a rectangle. The geometry of the rectangle is specified with the `x` and `y` attributes (the origin), and the `width` and `height` attributes (dimensions). Additionally, the corners of the rectangle can be rounded by specifying the radii of the ellipses used for drawing the corners with the `rx` and `ry` attributes. The following example draws a 200 x 100 rectangle with slightly rounded corners, blue edges of increased width, and a red interior:

```
<rect x="10" y="10" width="200" height="100" rx="10" ry="10" fill="red"
stroke="blue" stroke-width="2" />
```

The `circle` element creates a circle. The attributes `cx` and `cy` specify the coordinates of the center of the circle, and the `r` attribute defines the radius. An ellipse (the `ellipse` element) also has the `cx` and `cy` attributes, which define the coordinates of the center, and two radii defined with the `rx` and `ry` attributes.

A line is drawn simply by specifying the coordinates of its beginning and end — `x1` and `y1`, and `x2` and `y2` attributes — for the `line` element. A polyline can be drawn with the `polyline` element by specifying a list of points. The `points` attribute contains a space-separated list of coordinate pairs, and the `x` and `y` coordinates in a pair are separated with commas. For instance, the following will draw a blue V-shaped polyline:

```
<polyline points="0,0 50,100 100,0" stroke="blue"/>
```

Using the `polygon` element is very similar, except that the shape will be automatically closed — the last point in the polyline will be connected with the first one.

Text in the simplest form can be drawn using the `text` element, and the `x` and `y` attributes to define the starting coordinates. The typeface family is selected with the `font-family` property.

```
<text x="10" y="10" font-family="serif">
  This is sample text using serif font.
</text>
```

To rotate, scale, move, or skew a graphic primitive, their `transform` attribute is used. The value of the attribute can be the `scale` (`<scale ratio X>[, <scale ratio Y>]`), the `rotate` (`<rotate angle>[cx, cy]`), the `translate` (`tx[, ty]`), or `skewX` (`<angle>`) or `skewY` (`<angle>`) transformation. For instance, to draw a rectangle rotated at a 45° angle, the `rotate` transform is used:

```
<rect x="10" y="10" width="80" height="80" transform="rotate(45)" />
```

To group several objects into one and then apply various transformations to the whole group, make the primitives children of the `g` element.

4.2 Animation

Another interesting feature of the SVG-T is the support for animation. Animation is achieved by either changing the value of a single property or attribute of an element over time (using the `animate` or `animateColor` elements), or by moving or transforming the whole object (using the `animateTransform` or `animateMotion` elements).

For example, to make the radius of a circle change with time, the `animate` element is used as the child of the animated element:

```
<circle cx="100" cy="100" r="10">
  <animate attributeName="r" from="1" to="10" dur="5s"
repeatCount="indefinite" />
</circle>
```

To make an object change its colors, `animateColor` is used in a very similar way. The following example draws a rectangle whose interior color keeps changing:

```
<rect x="10" y="10" width="80" height="80" fill="blue" >
  <animateColor attributeName="fill" from="blue" to="navy" dur="5s"
repeatCount="indefinite" />
</rect>
```

The `animateMotion` element is used to make an object move. For instance, the following SVG-T code moves a vertical line diagonally on the screen:

```
<line x1="0" y1="0" x2="0" y2="100">
  <animateMotion from="0,0" to="100,100" dur="5s"
  repeatCount="indefinite" />
</line>
```

As with transformations, animations can operate on complex graphics, as well as on the primitives. To create a composite object, the graphic primitives or other composite objects are enclosed with the `g` element. The following example rotates a group of circles around the axis at the coordinate (50,50):

```
<g transform="rotate(0,50,50)">
  <circle cx="0" cy="0" r="5" fill="blue"/>
  <circle cx="0" cy="100" r="5" fill="blue"/>
  <circle cx="100" cy="0" r="5" fill="blue"/>
  <circle cx="100" cy="100" r="5" fill="blue"/>
  <animateTransform attributeName="transform" type="rotate" from="0"
  to="360" dur="10s" repeatCount="indefinite" />
</g>
```

The examples given in this section were merely a basic introduction to the world of Scalable Vector Graphics and its use in MMS. To learn more about SVG, please refer to the *Scalable Vector Graphics 1.0* specification. In addition, the sample accompanying this document provides a full example of a working SVG Tiny file for use with MMS that utilizes all the features that have been described here.

5 Conclusion

The final installment in a three-part series, this document has described how to create the body of an MMS using SMIL and reviewed the creation of SMIL files to control the presentation of elements within a multimedia message. It also has provided an introduction to Scalable Vector Graphics. Developers should now have all the tools they need to create multimedia messages either from within a Nokia 7710 smartphone application or from a server application for delivery to a Nokia 7710 smartphone.

This series of documents has provided a detailed description of the messaging subsystem on the Nokia 7710 smartphone. Readers now have all the information they need to create, send, and receive rich, fully featured MMS messages on a Nokia 7710 smartphone.

6 Terms and Abbreviations

Term or abbreviation	Meaning
Glossary text	Glossary text
Glossary text	Glossary text
Glossary text	Glossary text

7 References

Nokia 7710: Messaging APIs

Nokia 7710: Using The Messaging APIs For MMS

[22.140] “Multimedia Messaging Service (MMS); Stage 1 (service aspects),” 3rd Generation Partnership, <http://www.3gpp.org>.

[23.140] “Multimedia Messaging Service (MMS); Stage 2 (technical realization),” 3rd Generation Partnership, <http://www.3gpp.org>.

[26.140] “Multimedia Messaging Service (MMS); Media formats and codecs,” 3rd Generation Partnership, <http://www.3gpp.org>.

[RFC 2882] “Internet Message Format,” ed. P. Resnick, 2001.

8 Evaluate This Document

In order to improve the quality of documentation, we kindly ask you to fill in the [document survey](#).