

SMIL and SVG in teaching

Horst Eidenberger*

Vienna University of Technology, Institute of Software Technology and Interactive Systems,
Favoritenstrasse 9-11, 1040 Vienna, Austria

ABSTRACT

This paper describes how the web standards Synchronized Multimedia Integration Language (SMIL) and Scalable Vector Graphics (SVG) are used in teaching at the Vienna University of Technology. SMIL and SVG are used in courses on multimedia authoring. Didactically, the goal is to teach students how to use media objects and timing concepts to build interactive media applications. Additionally, SMIL is applied to generate multimedia content from a database using a content management system. The paper gives background information on the SMIL and SVG standards and sketches how teaching multimedia is organised at the Vienna University of Technology. Courses from the summer term 2003 are described and illustrated in two case studies. General design problems of SMIL-based presentations are modelled as patterns. Additionally, suggestions for improvement in the standards are given and shortcomings of existing user agents are summarised. Our conclusion is that SMIL and SVG are very well suited for teaching multimedia. Currently, the main problem is that all existing SMIL players lack some properties desired for teaching applications (stability, correctness, etc.).

Keywords: Multimedia authoring, Synchronized Multimedia Integration Language, SMIL, Scalable Vector Graphics, SVG, content management, CMS, education, case study, personalisation, CC/PP

1. INTRODUCTION

In recent years universities all over the world and, especially, in the German-speaking countries, that offer computer engineering studies have established courses on multimedia and temporal media processing. So at the Vienna University of Technology. Since the year 2000 it offers a very popular media informatics curriculum (about 350 beginners per year). After three years students are graduated to Bachelors of Science. Besides theoretical foundations (mathematics, statistics, signal processing, etc.) and traditional computer engineering skills (programming, software engineering, data engineering, etc.) special courses on multimedia are offered. The curriculum includes courses on multimedia basics (media representation, compression, etc.), computer graphics, temporal media (audio, video) processing and multimedia authoring (fundamentals and application). This paper deals primarily with teaching authoring of interactive multimedia presentations based on temporal media.

Teaching multimedia authoring requires that students have understood the basics of multimedia programming. These include media types and their properties (sampling, streams, format parameters, etc.), media abstractions (binary streams, tracks, etc.) and access, time structures (e.g. timeline, time events) and synchronisation. Multimedia authoring itself includes media capturing (including sampling, compression), spatial and temporal composition and media deployment (for example, to the world wide web). To implement all these concepts in (laboratory) courses, a software platform with two – partially conflicting – properties is required. Firstly, it should provide a level of abstraction that is high enough that students can prototype multimedia presentations quickly but secondly, it should be transparent enough that students can *see* the implemented media concepts. Most commercial software packages on the market (e.g. Adobe and Macromedia products) meet with the first property but fail for the second. Software that is based on free standards (e.g. provided by the World Wide Web Consortium, W3C) is able to satisfy *both* needs.

The W3C provides several standards for the creation and management of multimedia presentations. Two of the most prominent are the Synchronized Multimedia Integration Language¹³ (SMIL) and Scalable Vector Graphics¹² (SVG). SMIL provides a set of XML tags for building interactive multimedia presentations that may also include animations and links to external documents. SVG is an XML-based standard for the description of vector graphics. Both standards have been established in recent years and are more and more adopted by the multimedia software industry (e.g. Adobe, Apple, Real; see Section 2 for details). For the reasons given above we decided to use SMIL and SVG to teach authoring of

* eidenberger@ims.tuwien.ac.at; phone 43 1 58801-18853; fax 43 1 58801-18898; www.ims.tuwien.ac.at

interactive multimedia presentations.

Below, we describe the experiences we gained in multimedia authoring courses in order to share them with lecturers that are in a similar situation. The paper is organised as follows. Section 2 gives background information on SMIL and SVG. Section 3 gives more detailed information on the scope and organisation of the proposed multimedia authoring course. Section 4 and 5 describe two case studies as they were carried out in the summer term 2003. Section 6 summarises the author's experiences in teaching multimedia authoring with SMIL and SVG. Finally, Section 7 sketches future plans for this course (next time in summer term 2004).

2. BACKGROUND

2.1 Synchronized Multimedia Integration Language (SMIL)

SMIL is an XML-based standard for the description of interactive presentations with multimedia content. A SMIL document describes *what* has to be done *when* in a presentation ("declarative media processing"). The way *how* something is done has to be implemented in the user agent (player) in the way defined in the W3C recommendation¹³. In the most recent version 2.0 SMIL is organised in ten modules. Each module encapsulates related XML-tags and attributes. For example, the timing module covers all tags for timing, synchronisation and interaction, the content control module contains all tags for personalisation, etc. Modules can be combined to profiles. The SMIL standard provides two default profiles: the basic profile (basically all tags and elements that were already available in SMIL 1.0) and the language profile (all capabilities of SMIL 2.0 except a module for manipulation of the time base of presentations). Additionally, custom profiles can be defined by interest groups (and implemented in user agents). Well-known examples are the XHTML+SMIL profile implemented in Microsoft Internet Explorer and the 3GPP SML profile used in mobile phones for the Multimedia Messaging System (also called MMS profile).

The most relevant SMIL modules are the structure module, media objects module, layout module and timing module. These modules are needed in any presentation. The structure module defines the structure of a SMIL document including the document type, namespace and tags for head and body (similar to XHTML). The media objects module defines the basic media types: *audio*, *video*, *animation*, *img* (any image format), *text* (XHTML or plain), *textstream* (Real's text + time format⁶) and *ref* (generic media object, specified by a MIME type attribute). The elements of the layout module are used for spatial composition of media objects. Essentially, overlapping regions with height, width, z-index (depth position) can be defined and loaded with media objects. The way a media object (e.g. an image) is rendered in a *region* is defined by a fill behaviour attribute (e.g. keep aspect ratio, resize to shorter dimension, etc.).

The major quality of SMIL lies in the timing module. This module is responsible for basic timing as well as media synchronisation and interaction. SMIL supports a mixture of two fundamental time concepts: timeline and time events. In SMIL, every media object and time container (see below) has a linear timeline that can be interrupted by time and interaction events. For basic synchronisation SMIL offers three time containers: *seq* (sequential order of media objects or other time containers), *par* (parallel execution) and *excl* (parallel execution where only one media object/time container can be active at any point in time). Additionally, it is possible to define synchronisation sources, latencies and quality of service parameters. For interaction SMIL makes use of the W3C Document Object Model¹⁰ (DOM, version 2). Every user agent that implements at least the basic profile has to be able to handle all DOM events (e.g. *click*, *mouseover*). Events can be used to start, interrupt or end the execution (presentation) of media objects and time containers.

In addition to these capabilities SMIL 2.0 offers modules for content control (personalisation and media object pre-fetching), metadata definition (as simple name/value pairs or as Resource Description Framework¹¹ descriptions), linking (similar to XHTML), animation and transitions between media streams. Basically, the animation tags and attributes can be used to animate any property of a media object or region (including colour, position, size). SMIL uses key-frame animation. The frames in-between start and end frame are rendered on the basis of a linear or path interpolation. Finally, the transition module defines fades and SMPTE wipes for media objects. Since every media object in SMIL has a temporal dimension (including text and images), transitions can be applied on any type of media (of course, on audio only fades can be applied).

One current weakness of the SMIL standard is the limited availability of user agents. Even though a considerable

number of players does exist, most of them do not support the full language profile or contain bugs in the timing module. Especially, DOM-based interaction hardly works in most existing players. See Section 6.2 for more information on SMIL user agents.

2.2 Scalable Vector Graphics (SVG)

Like SMIL, SVG is a W3C recommendation and based on XML (defined by a document type definition based on a namespace). SVG is intended to be used to describe two-dimensional vector graphics. Being an open standard its major advantage over proprietary formats (like VML, Macromedia Flash) is its vendor independence.

SVG defines a set of core elements for vector graphics: geometric primitives (e.g. line, rectangle, circle), text and path elements for arbitrarily shaped objects. Path elements can be simple polylines or Bezier splines. All position, size and length parameters can be given in absolute or relative numbers. Closed graphical objects can be filled with colour shades, gradients and custom textures (images).

By using the *defs* element high-level graphical primitives can be defined and used. Additionally, SVG offers a grouping element that can be used to accelerate the workflow. Graphical constructs can be defined once and used multiple times by reference. Since SVG is a hypertext standard, any component of an SVG object may also contain links to external documents.

SVG is linked to SMIL in two ways: SMIL documents can contain scalable vector graphics as *img* elements and SVG makes use of the SMIL animation module. Using the *animate* tag any attribute of an SVG document can be animated. Therefore, it is possible to implement complex presentations in SVG. If the *animation* tag is not sufficient for a certain application, SVG allows scripting with Javascript. This option is supported by most user agents. In contrast to SMIL, a large number of professional tools and players exist for SVG (e.g. Adobe SVG viewer).

3. TEACHING MULTIMEDIA

Before two case studies from the summer term 2003 are described in sections 4 and 5, we sketch the environment of the multimedia authoring course in the following two subsections. First, scope and related courses are outlined and then, organisation and workflow of the course (one lecturer for more than 200 students) are briefly described.

3.1 Scope

Multimedia authoring at the Vienna University of Technology is a laboratory course that is held in parallel with the lecture "Multimedia 2". Multimedia 2 discusses topics from media programming concepts (based on QuickTime), visual information retrieval to video server technology and basics of embedded media systems. The lecture is held in the second year of the three year media informatics curriculum. It is preceded by the lecture "Multimedia 1" and the laboratory course "Multimedia Programming" (third term).

In Multimedia 1 students learn the basics of media processing: media types, digitisation of audio and video, formats and parameters, colour and colour models, timing and synchronisation, streaming media compression, etc. In Multimedia Programming (also called "procedural media processing" – in contrast to multimedia authoring: "declarative media processing") students learn how to apply these concepts in software applications. The course is based on Java and the Java Media Framework for audio and video processing.

After multimedia authoring students have to attend two courses on "Multimedia Production" (fifth and sixth term): one on interactive applications (e.g. based on QuickTime iShell) and one on non-interactive presentations (e.g. authoring of video clips). The latter is partially based on the multimedia authoring course discussed in this paper. Additionally, students have to do an internship in which they have to apply the knowledge and skills collected in these courses to complete larger one-person projects (approximately 250-300 hours). More information on these courses can be found on the website of the Interactive Media Systems group of the Vienna University of Technology³.

The multimedia authoring course is split in three parts: (1) media capturing (e.g. with microphones and video cameras), (2) media composition (editing, spatial and temporal organisation, event handling, etc.) and (3) media deployment (e.g. compression for streaming over the Internet). The didactic intention is threefold: (1) Repetition and practical application

of the basic multimedia concepts taught in the lectures Multimedia 1 and 2, (2) provision of an introduction to multimedia content production (handling of video cameras, usage of software tools for video editing, wave editing, etc.) and (3) introduction of students to web-(XML-)based standards for multimedia applications.

The first goal is mainly treated in the second step of the course. Especially, SMIL is ideally suited for teaching the properties of media types, teaching timing concepts and explaining synchronisation. Additionally, knowledge on media formats and (compression) parameters can be used in tools for media editing and deployment. The second goal is the main intention of the course. Students should get a foundation for the more detailed courses in the consecutive terms. The third goal is satisfied in the second step of the course. By learning SMIL and SVG students also learn the philosophy of XML and its dialects as well as the role of the W3C in further developing the global hypertext network.

3.2 Organisation and workflow

Every year about 350 students start the media informatics program at the Vienna University of Technology. About 100 to 150 of these drop out during the first year (introduction to programming, basic education in information science, mathematics, etc.). The remaining 200 to 250 attend the introductory multimedia lectures and the courses on multimedia programming and multimedia authoring. Of these students, about 75% have received only a fundamental education in computer engineering in high school (text processing, using the Internet, etc.). About 25% have graduated from technical high schools where they have already learned programming, software engineering and other computer engineering-related skills. Unfortunately, only one lecturer is available for the multimedia laboratory courses (assisted by three student assistants). In consequence, well-considered organisation and clear guidelines for the students are crucial success factors for the courses.

The multimedia authoring course is split in two parts. In the first (at the beginning of the term), students are taught the basic concepts of SMIL, SVG, video camera handling, filming, video editing and audio processing. Then, students begin to work (in groups of three) on topics chosen from a given list. Each presentation has to consist of SMIL documents with synchronised video, audio, text and images (SVG and others) and has to allow user interaction. While working on their presentation, they can ask questions through an electronic forum system. Their questions are answered by the lecturer, the student assistants and very often by colleagues who ran into the same problem and found a solution. The forum system means a considerable relief for the lecturer and, by the way, has a documenting character. After each course, a frequently asked questions document is created from the students' questions.

During the last month of the term students have to upload their presentations to a server. Then, in a first step the presentations are formally evaluated by the student assistants and – if no further refinement is needed – the students are invited to an oral examination with the lecturer. This procedure allows to give laboratory courses for more than 200 students at minimal costs (in terms of human resources). All administrative tasks (course registration, topic selection, etc.) are performed using the forum system.

4. CASE STUDY 1: MULTIMEDIA AUTHORING

In the following two sections we will outline two case studies. The conclusions derived from these case studies are summarised in Section 6. Firstly, the multimedia authoring course from the summer term 2003 will be described. The students' problem definition is described in Subsection 4.1. Subsection 4.2 shows some example screenshots of solutions. Finally, Subsection 4.3 abstracts essential components of solutions to patterns.

4.1 Problem definition

In the multimedia authoring course during the summer term 2003 every group (three students) had to build a presentation of one line of the Vienna public transport system (only subway lines were excluded). About seventy groups attended. Each presentation had to consist at least of an introduction page, a menu, the line's route, descriptions of sights along the route and an animated timetable. Figure 1 shows the presentation flow. Sub presentations, like the timetable, should always return to the main menu. Every presentation had to be available in German and English and for clients with low (< 1 Mbit) and high bandwidth to the server hosting the presentation. Every page of the presentation had to include a background image and looping audio. It was a requirement to switch on/off audio at any time. Presentations had to consist exclusively of SMIL (e.g. menu) and SVG (e.g. route plan) documents.

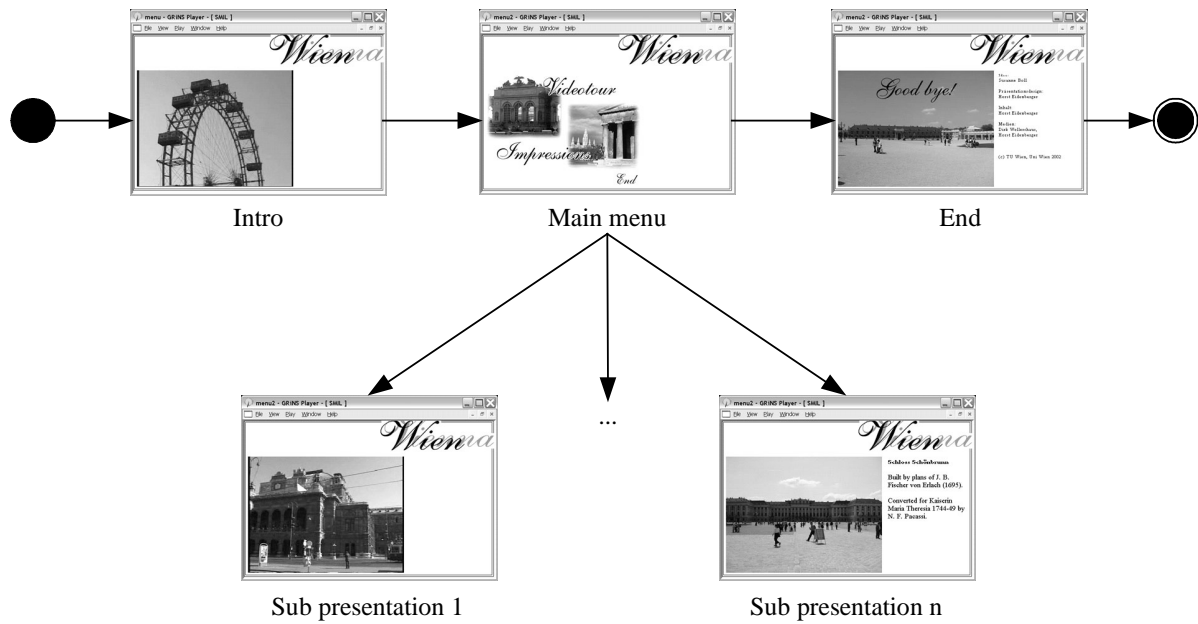


Figure 1: Organisation of presentations. Every presentation consists of an intro, a main menu, several sub presentations and a final page with a short disclaimer.

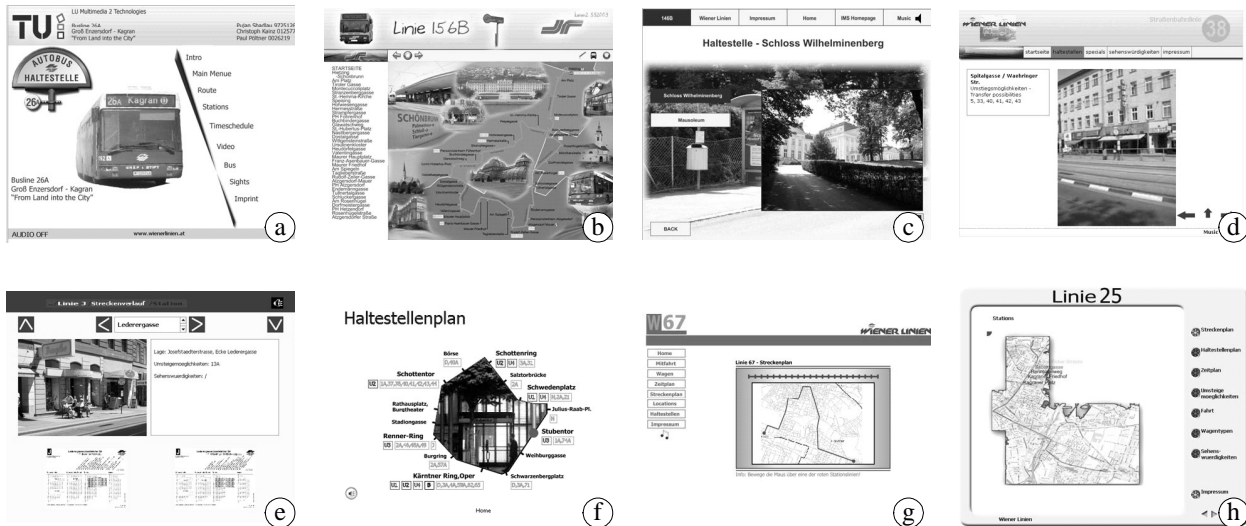


Figure 2: Design examples from the summer term 2003 (rendered in X-Smiles). Element *a* and *b* show menus, *c-e* show sub presentations and *f-h* show embedded SVG documents.

To meet all these requirements students had to record and cut videos, prepare audio tracks and background images and assemble all elements by SMIL and SVG documents. Since most students had no prior experience in these tasks, short introductions to video camera handling and recording as well as audio processing were added to the course. As it turned out, providing students with elementary knowledge on good recording (camera positions, good panning, etc.) and common software tools (Adobe Premiere for cutting, Audacity for wave processing) could be accomplished quickly. Since most students had already worked with XML standards, they were able to learn SMIL and SVG easily.

The major problem of the course was finding a user agent that was able to display SMIL and SVG documents and would be available at no costs. Our choice was the X-Smiles player provided by the Helsinki University of Technology². X-Smiles is a free Java-based browser for a variety of W3C standards (including SMIL, SVG, XHTML, XForms, X3D).

Audio and video processing in X-Smiles are based on the freely available Java Media Framework⁷. Therefore, X-Smiles should have been the tailor-made user agent for the course. Unfortunately, X-Smiles had two major shortcomings: Firstly, at that time it did not implement essential parts of the SMIL 2.0 standard (including full DOM interaction, time events, transitions) and secondly, it was still in a beta stadium. The player was generally unstable and its behaviour (especially when communicating with the embedded SVG renderer) was often non-deterministic. This turned out to be very annoying for students. Still, since no other choice was available, X-Smiles was used in the multimedia authoring course.

4.2 Design examples

Styling decisions were left to the students. It was their task to choose the colours, fonts, spatial organisation, navigation style, etc. Only a basic design (screen size, media formats, media resolution) was provided and a guideline was defined that the design should follow the design of the website of our workgroup. Additionally, since X-Smiles is able to work with Cascading Style Sheets⁸ (CSS) students should collect all styling tags in a single CSS document. It should be possible to change the appearance of a presentation by changing the style sheet.

Figure 2 shows some examples of presentation designs. Element *a* and *b* show different menu designs. Most groups decided on a clear arrangement as in *a*. The arrangement in element *b* consists of SMIL and embedded SVG. Element *c*, *d* and *e* show route sub presentations. Most groups have created an active route plan: if a link is activated detailed information on the selected stop is presented (photo, interchanges, etc.). Element *f*, *g* and *h* show outstanding applications of SVG. Element *f* shows the route of an inner city tramway line. The route is a closed path and the students have taken the opportunity to fill it with an image texture. In *g* and *h* SVG is embedded in SMIL and path primitives are laid over images of city maps.

Most groups invested significantly more time than requested in their presentations. Especially, most groups made use of scripting in SVG and enhanced their graphics with Javascript interaction. Unfortunately, this option is not available in SMIL. Javascript would be a good workaround for the DOM events missing in X-Smiles.

4.3 Result patterns

During the course it turned out that the main didactic topics were easy to solve for the students. Spatial organisation and synchronisation of video, audio and other media types with SMIL time containers were easy to understand. The major implementation problems turned out to be the following three: (1) Personalisation of the presentation (language, bandwidth), (2) navigation (based on the presentation flow in Figure 1) and (3) switching the background audio on and off at any point in time. Since the author observed similar problems in earlier courses as well, below, these three problems are described and standard solutions (patterns) are sketched.

At the time when this paper was written, SMIL did not support the Composite Capabilities / Preference Profiles⁹ (CC/PP) W3C standard for content personalisation based on user preferences and system properties. The *switch* element is the only available mechanism for personalisation. It allows to activate parts of a presentation (including links to external documents) depending on certain conditions (so-called system tests). In SMIL 2.0 it is possible to test, for example, the screen depth, system bandwidth and the language preferred by the user. Working with *switch* has two general disadvantages: its capabilities are very limited and it soon makes SMIL documents unreadable.

The first problem is beyond the users influence but for the second a simple solution can be provided. Figure 3 shows the idea. All *switch* statements (in our case: language, bandwidth) are gathered in a start page that has no visual output but – depending on the evaluation of the system tests – calls personalised presentations through links (SMIL links can also be triggered by system events!). These presentations are derived from a template and differ only in the media objects (high/low resolution, etc.) they use. This pattern minimises the effort needed for personalisation. Additionally, any object in a SMIL document can be used as link target. For example, it is possible to jump over the intro video if the bandwidth is low.

The navigation problem can be solved by using the *area* attribute of SMIL links and the *excl* time container. With *area* it is possible to associate spatio-temporal regions in media objects with link targets. For example, in Figure 4 link areas *A*, *B* and *C* are associated with different sub presentations. If the media object containing the *area* links (e.g. the image in

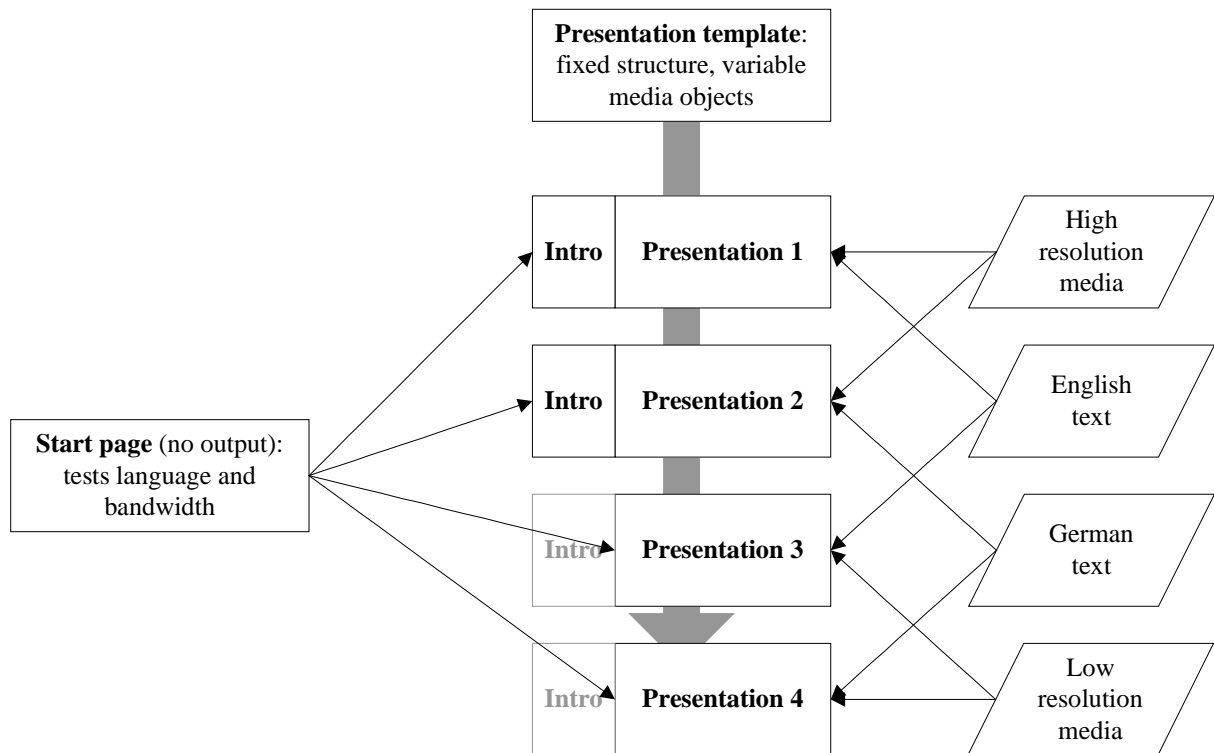


Figure 3: Pattern for personalised SMIL presentations. The actual presentations are derived from a template and called by a start page that tests relevant presentation properties. The only difference between presentations are the used media objects. If the available bandwidth is low, the intro is skipped.



Figure 4: Main menu and link areas for navigation (rendered in GRiNS player).

Figure 4) is embedded in an *excl* time container, it is guaranteed that only one sub presentation is active at any point in time. Depending on the parameters of *excl* it is also possible to assure that sub presentations always return to the menu. Unfortunately, *excl* is not available in X-Smiles. Therefore, students had to use workarounds based on time and interaction events.

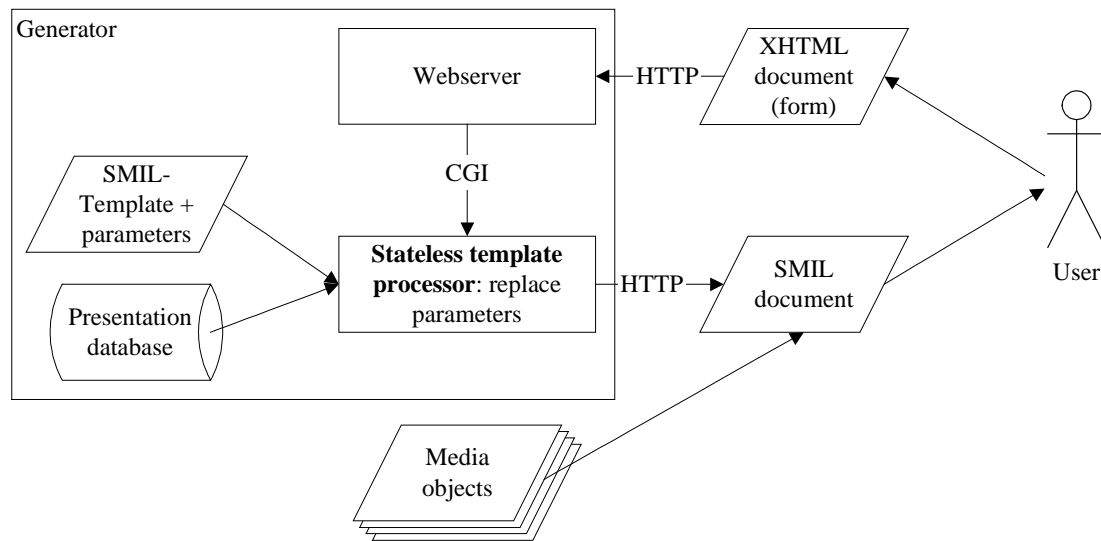


Figure 5: Flow chart of SMIL document generation.

The third problem, switching audio on and off at any point in time, should be easy to solve in SMIL. A button on every SMIL page and associations of the click event with start and stop time of the audio track should be sufficient. Unfortunately, even though this behaviour is well-defined in the SMIL recommendation, it does not work reliably in any of the currently available SMIL user agents. Therefore, the only applicable solution is associating a reload event with the audio button. Depending on the current status of the presentation it is reloaded and started with or without background audio respectively.

5. CASE STUDY 2: GENERATING SMIL FROM A CMS

The second case study describes a student's internship. He had developed a portal server for young people. This portal was technically based on a relational database (MySQL) and a self-developed content management system (CMS) based on PHP. Now, he wanted to extend his portal server to interactive presentations of (partially time-based) media objects (e.g. video shows, etc.). Since he had attended the multimedia authoring course he wanted to use SMIL for the multimedia presentations.

For this problem SMIL has the advantage that the structure of the CMS does not need to be changed. Figure 5 shows the structure of the resulting multimedia CMS. As in any CMS, data is held in a database and on request fed into pre-defined templates. Here, templates are SMIL documents with variable IDs for media objects and template processor-specific parameters. The template processor is – like all other scripts in the CMS – a PHP script that is started through a webserver CGI call. As for XHTML the resulting SMIL document is interpreted by a user agent on client side. In this solution media objects are held outside the media database. Of course, they could also be managed inside the database as BLOBs.

For this project it was decided to use the RealONE⁵ player as user agent. RealONE has several advantages over other players: it is widely available on Windows systems and it supports the largest quantity of the SMIL 2.0 recommendation of all available players (e.g. QuickTime supports only the basic profile). Unfortunately, RealONE has several bugs that lead to unstable behaviour of the player (especially, if multiple SMIL presentations are played in a sequence).

Figure 6 shows a typical screenshot of a generated SMIL presentation. In this template the title is set through a parameter. The template processor replaces the placeholder in the template with the actual presentation title. The second element shows the selected image of a set of images. The selected image changes over time or on user interaction. The row of images at the bottom represents the navigation. Four images are shown at the same time (the number can be controlled by a parameter). The left ("<<") and right (">>") buttons allow navigation through the entire media collection.

This implementation has two major advantages: Firstly, the presentation data can be visualised by arbitrary templates.



Figure 6: Example screenshot (rendered in RealONE player) of a presentation based on a simple template with three sections.

The presentation designer has full control over presentation layout and behaviour. Secondly, the template processor can be used interactively but, as well, to generate presentations off-line (e.g. for export on CD).

6. RESULTS

Below, we try to generalise the insights from our teaching experiences. Subsection 6.1 points out areas where SMIL and SVG could possibly be improved in future revisions of the standards. Subsection 6.2 discusses some bugs and shortcomings of the user agents (players) used in the courses. Finally, Subsection 6.3 repeats technically and/or didactically relevant feedback of students.

6.1 Suggestions for further improvements in SMIL and SVG

SMIL and SVG are both sophisticated standards that offer the user a great variety of functions for a wide-spread range of applications. However, there is still some space for improvement left. Using the standards in teaching we discovered a few shortcomings of SMIL and SVG. Below, we will discuss three of them: (1) Code reuse in SMIL, (2) personalisation in SMIL and (3) embedding large text blocks in SVG.

As briefly described in Subsection 2.2 SVG offers mechanisms for code reuse. With *defs* and *use* it is possible to define and use complex objects that consist of geometric primitives. Additionally, with the *g* element groups can be defined and re-used (referenced by a unique *id* attribute). SMIL does not offer such mechanisms. For example, for the navigation pattern described in Subsection 4.3 it would be desirable to have a code macro for the execution of sub presentations. Additionally, a grouping mechanism would be useful for groups of media objects that are always used in combination (e.g. background image, background music and navigation buttons).

Personalisation in SMIL means using *switch* and system tests. Even though SMIL offers an extension mechanism for the limited number of system tests (so-called custom tests), this solution is hardly applicable. The implementation of the extension mechanism is not standardised but left to the user agent. Therefore, it differs from player to player. Generally, it would be desirable that SMIL made use of CC/PP⁹ and RDF¹¹ for the definition of user preferences and for system description. With CC/PP and the personalisation pattern from Subsection 4.3 a two-step procedure could easily be

User agent	Advantages	Disadvantages
GRiNS (Oratrix)	<ul style="list-style-type: none"> - full SMIL 2.0 support - authoring environment available - very stable 	<ul style="list-style-type: none"> - not free - only Windows version available - partially surprising event handling
Internet Explorer (Microsoft)	<ul style="list-style-type: none"> - SMIL user agent integrated in browser 	<ul style="list-style-type: none"> - just XHTML+SMIL profile supported: no support for the layout module - bugs in event handling
QuickTime (Apple)	<ul style="list-style-type: none"> - platform-independent - very stable - additional QuickTime-specific controls 	<ul style="list-style-type: none"> - only SMIL 2.0 basic profile supported
RealONE (Real)	<ul style="list-style-type: none"> - full SMIL 2.0 support (including transitions in real-time) - free 	<ul style="list-style-type: none"> - many bugs in event handling and animations - unstable
X-Smiles (Helsinki University of Technology)	<ul style="list-style-type: none"> - integration of players for SVG, XHTML, XForms and X3D - platform-independent (Java) - free as open source 	<ul style="list-style-type: none"> - SMIL 2.0 not fully implemented (e.g. no transitions module) - important DOM events are missing - unstable (no failure tolerance) - difficult to install (needs Java Media Framework) - only a few media formats are supported (e.g. no MP3 audio)

Table 1: Advantages and disadvantages of evaluated SMIL user agents.

implemented: firstly, the personalisation parameters would be negotiated with CC/PP and then, a personalised presentation (based on a template) would be presented.

The only shortcoming we found in SVG was handling long text segments. Text is embedded in SVG with the *text* element. This is a convenient method for short text segments (and, for example, much more elegant than embedding short text segments in SMIL) but not sufficient for longer text. A method for importing large text blocks (e.g. as XHTML) could be based on the *tref* linking mechanism. With *tref* it is possible to reference text segments *within* an SVG document. If it was allowed to address external text documents as well and render them within SVG presentations, modular vector graphics with text would become possible.

6.2 Problems in SMIL user agents

Table 1 summarises all SMIL players we used in courses. In contrast to SVG, where several bug-free and easy-to-use players do exist, the available SMIL players have their advantages and disadvantages. One major problem in all SMIL players is robust resolution of start and end time of media objects and time containers. Begin and end time depend on absolute time values, time events (e.g. end of a previously played media stream) and interaction events. To the author's experience even the players of established multimedia companies have problems with calculating the correct time values. This is especially true if the complexity of a SMIL presentation increases.

From Table 1 it can be seen that the most powerful currently available players are RealONE and GRiNS. Unfortunately, the GRiNS player is not free of charge. RealONE supports the full SMIL 2.0 recommendation but is, especially for complex presentations, unstable. QuickTime offers the most stable SMIL support but supports only the basic profile and no user interaction.

For the future the Centrum voor Wiskunde en Informatica (CWI) of the University of Amsterdam – one of the driving forces behind SMIL – plans to release a new SMIL player: AMBULANT¹ will be based on the GRiNS player (developed by the CWI spin-off Oratrix), support the full SMIL 2.0 recommendation and be free of charge. A first release is announced for December 2003.

6.3 Feedback of students

The feedback of students on the multimedia authoring course was mostly positive. Students appreciated the chance to work creatively in groups on a project. Their main complaints were on the user agents used in the courses. Since all used players have problems with event handling and resolution of start and end times they were always uncertain whether a certain misbehaviour resulted from an error in the SMIL document or a bug in the user agent.

One X-Smiles-specific problem that turned out to be very annoying for the students is the limited number of supported audio and video formats. X-Smiles is based on the Java Media Framework (JMF). Unfortunately, in recent years SUN had to remove support for enhanced MPEG media formats from the JMF for legal reasons. Therefore, students had to work with older formats like Intel Indeo and Radeon Cinepak. In return media capturing, cutting and compression took more time and media files became significantly larger and awkward to handle.

7. FUTURE WORK

The student's major – scientifically not usable – complaint, on the multimedia authoring course was that it is generally very interesting but that spending hours with cutting and compressing audio and video is practically annoying. As these procedures will later partially determine the students' daily work, the lecturer cannot agree to this complaint. On the other hand he can understand their feelings very well. Therefore, an additional aspect will be introduced to the multimedia authoring course in the summer term 2004 that opens the door for completely new applications.

In the future the course will be based on SMIL, SVG and dynamic HTML (DHTML) programming with Javascript (ECMAScript) and the Document Object Model of the W3C. Using this framework students will implement old-fashioned 2D arcade games (like Pacman, Frogger, etc.) as they were very popular in the eighties (for example, on the Commodore C64, Atari ST, Amstrad CPC, etc.). Each game will consist of an intro, the game, winner ceremony (video, audio, text, based on SMIL) and a scoreboard (based on DHTML and cookies). In the game, an animated icon (sprite, an animated GIF derived from an SVG animation) jumps and runs through a matrix-like world of levels (described as an HTML table of image elements, manipulated through DOM). To make it easier for the students, they are provided with a Javascript template of an event queue for both interaction and time events. Their main tasks will be game design (including SMIL and SVG authoring) and event programming. Through temporal events, for example, opponents of the user-controlled sprite can be animated. The games will be developed for the Netscape browser with a RealONE media player plug-in. Game ideas will be taken from old computer magazines the author collected (and together with his brother hacked into their Amstrad CPC 464) in the early eighties.

Didactically, this course should teach the same multimedia concepts as described above as well as provide the students with valuable skills in webpage programming. After all, it should be much more fun and result in a suite of funny games. The course results will be presented in a small "retro-game convention".

8. CONCLUSIONS

This paper summarises the author's experience with teaching SMIL and SVG in multimedia courses at the Vienna University of Technology. Using SMIL and SVG students learn and apply basic multimedia concepts to create interactive multimedia presentations. Additionally, SMIL has been used in internships to generate multimedia presentations from a database by a content management system.

The paper gives background information on the SMIL and SVG standard. It describes the didactic scope of the multimedia lectures and the organisation of the laboratory courses. In two case studies courses from the summer term 2003 are sketched. General problems of SMIL-based multimedia presentations are modelled as patterns. Additionally, suggestions for improvements in SMIL and SVG are given and shortcomings of existing user agents are summarised.

Our experience is that SMIL and SVG are well-suited for being used in multimedia courses. SVG is easy to learn and apply. SMIL is more complex to understand (especially timing and synchronisation) but still very transparent. The main problem of using SMIL in courses turned out to be the selection of a SMIL player. Unfortunately, at the current point in time, no player exists that would satisfy all needs (stable, bug-free, easy to use, free of charge). In the future, the AMBULANT player¹ could close this gap.

ACKNOWLEDGEMENTS

The author would like to thank his student assistants Katharina Weislein, Gabriel Wurzer and Matthias Zeppelzauer for their enthusiasm and excellent input to the multimedia courses and Christian Breiteneder for his valuable comments and suggestions for improvement!

REFERENCES

1. CWI, University of Amsterdam, Ambulant SMIL player website, <http://www.cwi.nl/projects/Ambulant/> (last visited 2003-10-25).
2. Helsinki University of Technology, X-Smiles website, <http://www.xsmiles.org/> (last visited 2003-10-25).
3. Interactive Media Systems group, teaching website, <http://www.ims.tuwien.ac.at/> (last visited 2003-10-25).
4. Oratrix, GRiNS player website, <http://www.oratrix.com/GRiNS/> (last visited 2003-10-25).
5. Real Networks, RealONE player website, www.realnworks.com (last visited 2003-10-25).
6. Real Networks, Real textstream media format, <http://service.real.com/help/library/guides/realtext/realtext.htm> (last visited 2003-10-25).
7. SUN, Java Media Framework website, <http://java.sun.com/products/java-media/jmf/> (last visited 2003-10-25).
8. World Wide Web Consortium, Cascading Style Sheets, <http://www.w3c.org/Style/CSS/> (last visited 2003-10-25).
9. World Wide Web Consortium, Composite Capabilities / Preference Profiles, <http://www.w3c.org/Mobile/CCPP/> (last visited 2003-10-25).
10. World Wide Web Consortium, Document Object Model, <http://www.w3c.org/DOM/> (last visited 2003-10-25).
11. World Wide Web Consortium, Resource Description Framework, <http://www.w3c.org/RDF/> (last visited 2003-10-25).
12. World Wide Web Consortium, Scalable Vector Graphics, <http://www.w3c.org/Graphics/SVG/> (last visited 2003-10-25).
13. World Wide Web Consortium, Synchronized Multimedia Integration Language, <http://www.w3c.org/AudioVideo/> (last visited 2003-10-25).