Editing SMIL with Timelines

Cécile Roisin [1,2], Vincent Kober[1], Vincent Quint[1], Pierre Genevès [1], Patrice Navarro[1]

(1) Projet WAM, Inria Rhône-Alpes
(2) Université Pierre Mendès-France , Grenoble

Abstract

This paper presents a timeline approach to editing two of the main components of SMIL: the time and synchronization model and the animation model. This approach is illustrated with two prototypes: the LimSee tool for editing SMIL time structures, and the SVG Animation component included in the Amaya Web client. The timeline view is a faithful representation of the temporal behavior of the document where the author can perform all editing operations related to animation or synchronization between media. The advantages of this authoring approach are shown through editing session excerpts performed with LimSee and the Amaya Animation module.

1 Introduction

An increasing number of multimedia applications use the SMIL format. As a consequence SMIL authors are no longer multimedia specialists but more and more average content producers. These new users do not feel comfortable with most authoring tools available today. They need higher level tools. For example, Inria publishes now many seminars and talks on the Web (refer to the Smilthèque) under the form of SMIL presentations where the video recording of the speaker, his/her voice, the slides and the outline of the talk are synchronized. People producing this SMIL stuff are from the Communication department and have no particular programming skills. Other similar examples can be taken from the educational area or the graphic domain.

The major issue in manipulating multimedia in general, and SMIL in particular, is to figure out at editing time how the document will look like at presentation time. For tools developers, representing the time dimension in a way that any user can comprehend is a challenge. Finding the right editing paradigm for such a representation is even more challenging. Many multimedia authoring environments have addressed this issue by providing authors with timelines. Media objects are represented graphically as boxes whose length reflects the object duration and whose position on the time axis represents the starting and ending time. This graphical representation has proven to be intuitive enough to be well accepted by users. It is then tempting to use this approach for SMIL. Here we are faced with the issue that time in SMIL is represented as a hierarchy of operators, which is not exactly the timeline model. Nevertheless, we think that, to provide comfortable editing features, timelines are worth being

experimented with SMIL.

To explore the timeline approach in SMIL, we have made experiments with two of the main features of SMIL: the time and synchronization model and the animation model. For each of them we have developed a separate prototype, the LimSee tool for editing SMIL time and synchronization structures, and the SVG Animation editing component in Amaya for editing animations.

The paper presents these experiments. It is organized as follows: the next section gives a quick overview of existing SMIL authoring tools. Section 3 describes the LimSee editor, focusing on the authoring features provided through its timeline view. Then in section 4 we show how animated graphics can be created and updated using a timeline view in the Amaya editor. Section 5 gives some conclusions.

2 SMIL and SVG Animation Editors

Existing tools for SMIL documents can be divided into 4 classes:

Systems using SMIL as an export format only
: Authors specify presentations with paradigms independent from SMIL, and based on a different model. When the authoring process is considered finished the system can generate SVG or SMIL output. Examples of such systems are QuickTime, Macromedia Flash5 or SMIL generator for Powerpoint presentations. The problem with these tools is that some interesting time aspects of SMIL are not covered, and more importantly the resulting SVG or SMIL code is often of poor quality and can hardly be edited any more with a different tool.

Enhanced source-based authoring tools
: Most of these tools are extended versions of some XML source editor. Many popular SMIL editors belong to this class, for instance SMILGen (RealNetworks) or Tagfree 2000 SMIL Editor (Dasan Technology). Authors use also plain XML editors for manipulating SMIL source, such as XMLSpy or Morphon XML. Even if these tools give access to potentially any aspect of SMIL and provide nice features for creating the structure and the syntax of SMIL documents, they are very poor when it comes to visualization and to editing the dynamic behavior of documents.

Template-based authoring tools
: In a way to simplify the authoring task, some predefined presentation structures are proposed. The author just fills in the blanks with his/her own media to produce the desired result. One of the first template environments for SMIL was RealSlideShow. Aurora SmilMe is another good example of this approach. The simplicity of authoring presentations with these tools is balanced by the lack of creativity and flexibility, so their use seems to be restricted to some specific usages.

High-level SMIL-based semantic views editors

These editors allow authors to handle the genuine SMIL model, including temporal structures, through a simple user interface. The leader commercial tool of this category is certainly Grins. Among other systems we can cite LimSee presented here or EZer SMIL 1.0 from SMILmedia.

It is important to notice that these approaches are not mutually exclusive. On the contrary, several of them are often mixed in a single tool. For instance templates can be provided by multi-view editors, and XML editing features are often proposed along with timeline views.

While attribute and structure views become more and more sophisticated, the basic needs for editing dynamic content are not really covered. No serious effort has been put to provide high level interfaces that could truly help authors in managing complex SMIL time structures and media synchronization. Moreover the evolving usage of SMIL and the new features added in its second version bring out new editing needs that are not yet addressed by the tools listed above. We have to acknowledge that it is really boring to create presentations with animations, clipped media and complex inter-media synchronization with the tools available today.

Professional video editing tools and 3-D editors (e.g. Avid, Axial..) also have an interesting approach to timelines for editing, even though they do not support SMIL and use proprietary formats instead. They have demonstrated that a spatial metaphor for representing time is of great help to authors. However the underlying time model of each tool implies different authoring functions. Video tools for instance have a single absolute time model (the one of the physical time of the video) upon which editing operations and media annotations are based. Other editors have an event-based approach and so aim at representing the time point of events and their relationships. SMIL has a mixed time model and therefore requires to harmoniously integrate several graphical features: absolute placement, time relations, events and structural time blocks.

Following the experience of managing time through timelines gathered through such editors as Macromedia Director, Oratrix Grins or the Madeus based SmilEditor, we propose in this paper several enhanced editing features that combine in a single view (1) structure management (2) consistent time placement of media object or SMIL components and (3) direct manipulation editing.

3 Editing the SMIL Time Structures with LimSee

3.1 Context and Objectives of LimSee

Authoring techniques play a pivotal role in our research team because we consider that the definition and the implementation of authoring services are an excellent way to assert multimedia models. Following this approach we have developed the *Madeus* relation-based model together with a constraint-based formatter and several authoring and presentation tools in which direct editing features have been experimented (*Madeus-Editor* [4]). With the emergence of SMIL, we have applied and adapted our editing techniques to the specific paradigms of the SMIL time model, namely the hierarchical time structure and the mixed absolute and relative time attributes. After a first prototype called *Smil-Editor* [3] that was directly built on top of our *Kaomi* toolkit [2], we decided to redesign an independent tool with the goal to produce a lighter code and to better fit with SMIL authoring requirements.

This new tool called *LimSee* focuses on a rich timeline view synchronized with a hierarchical view. It provides the author with powerful editing features for composing the time structure of SMIL documents. The design of LimSee combines graphical objects that give a straightforward perception of time information together with simple user actions such as mouse selection and moves. The rest of this section describes more precisely the authoring features.

3.2 The Timeline View in LimSee

The timeline view is a faithful representation of the temporal behavior of the document. In this view the author can perform all editing operations related to synchronization between elements. Figure 1 shows the LimSee timeline view. The left part displays the legend of that timeline: color for object types and arrows for attributes. Under this legend the user can find the zoom buttons and the Media view displaying the content of the currently selected media object: text, image or video. A control panel is added to the latter. For audio media, the window is empty but the sound is played.

The right part of the view displays the time structure of the document. Elements are placed on horizontal axes according to their timing. The user can scale this view in two different ways: by zooming the whole view in or out and by opening or closing any node (`par`, `seq` or `switch`). In Figure 1, node Par4 is closed while nodes Par2 and Par3 are open. Moreover, it is possible to open several partial timeline views from any node (e.g. Par3 is displayed in a partial view at the bottom right of Figure 1). To prevent too much information to be displayed in the timeline, only a subset of the arrows representing time attributes for media objects or nodes are visible. As SMIL allows only local dependencies, the displayed attributes are those associated with or inside the currently selected node. Combining these three mechanisms allows users to cope with the

scalability issue. Even large and complex time graphs can be manipulated. This is particularly interesting for huge documents that require authors to work on a single fragment of the scenario at a time.

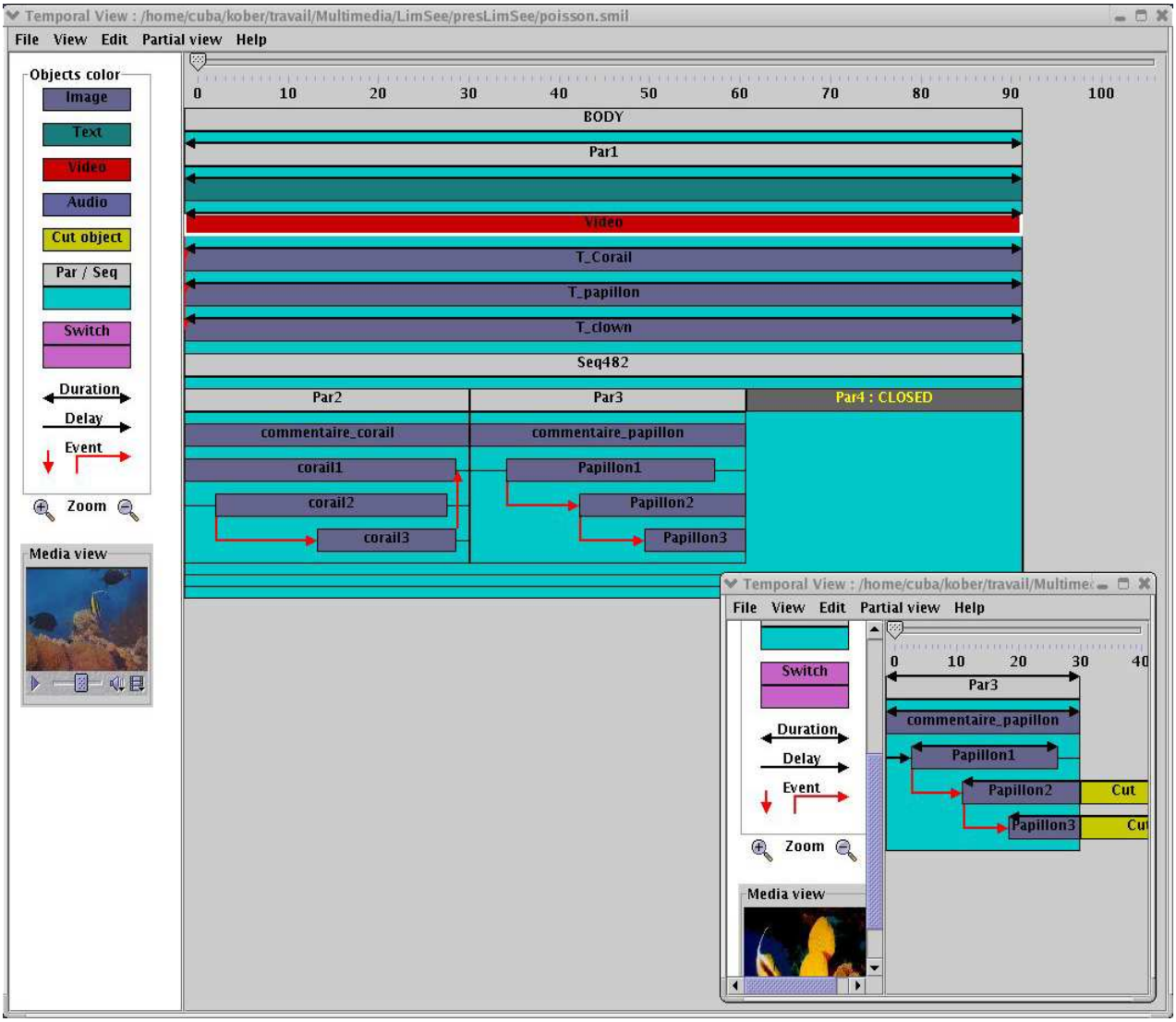

Figure 1: Main timeline view of LimSee

Every basic editing operation is available in this view as well as in the hierarchical view: Copy, Cut, Paste inside, Paste before, Insert and Node properties.

As shown in the figure, additional information carried by the attributes `begin`, `end` and `dur` is displayed when the user moves the mouse over the elements. The system displays then the arrows representing these attributes. A lighter color is used to indicate that the duration of an object is partially caused by a `fill=freeze` attribute. Moreover the view displays also dependencies between elements as expressed by the attributes. Red arrows are used for that purpose. Finally the user is aware of truncated (Cut) elements of the current composite node as in Figure 2 below. Note that when resizing a media element, a truncated part of that element may automatically appear to warn the user that its total duration is greater than the composite in which it is inserted.

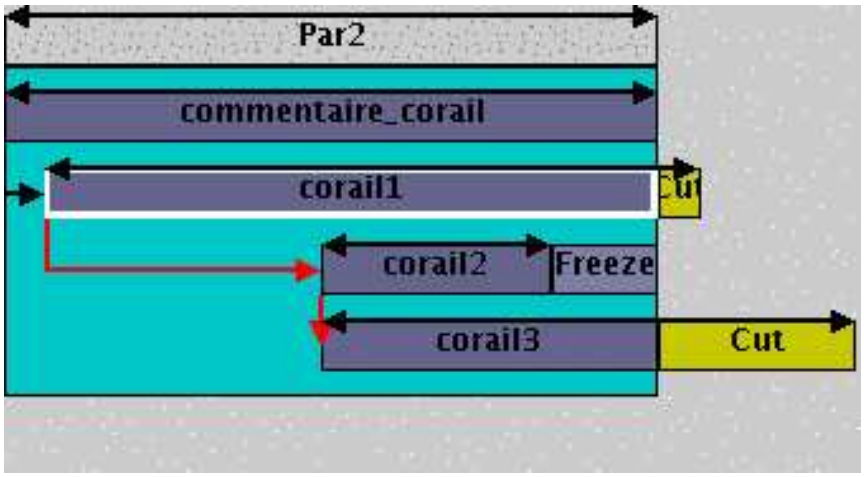

Figure 2: Visualization of truncated elements as defined by a SMIL scenario

The mouse cursor informs the user on the editing operations he/she can perform on the selected SMIL element. When the cursor takes the shape of a cross, an user action moves the object along the time axis (thus changing the `begin` and `end` attributes); if on the other hand it looks as a double arrow, the action changes the length of the object (`duration` attribute). Durations expressed as relative time placements between elements and represented by red arrows can be modified directly by moving the mouse while it is on these arrows. It is important to notice that every editing action implies a consistent update on all the elements of the document. For instance changing the duration of an object can induce changes of time positions of other objects of the same composite or of ascendent composites. This consistency checking is continuously performed during user actions thanks to the use of the Cassovary constraint solver [1]. This feature allows users to perceive on the whole document and in real time all the consequences of the modification they are performing.

3.3 An Editing Scenario

Consider the following document to be composed from scratch: a 3 minute video must be played in parallel with a sequence of three groups of three images each where two groups of images are displayed in parallel and the third in sequence. Each parallel image must be displayed with some particular duration.

Editing such a document with Limsee is very simple: when the user launches the editor, a default SMIL document is opened with a basic structure in the hierarchical view as shown in Figure 3. Similarly, every time a media element is added to the document, a region is automatically created with default attributes. The author simply has to modify those values or to link the element to another region.

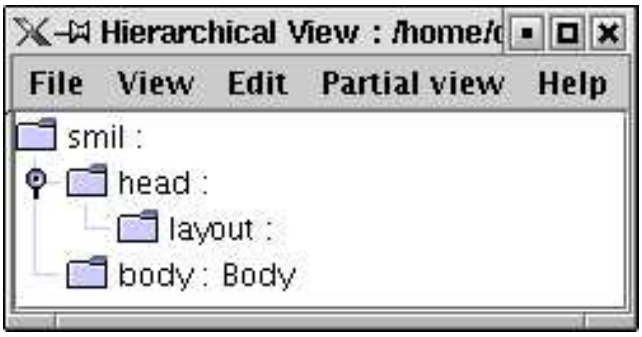

Figure 3: Default structure of a SMIL document

In our example, after launching LimSee, the best way to continue is to create a `par` element in the hierarchical or the timeline view. We can then immediately define the duration of this element by right-clicking the `par` element and by opening the attributes view.

Zooming in the timeline view allows us to adjust the scale of the presentation.

We can now add in the same way the video element and a `seq` component. The default duration of the newly created objects can easily be resized using the timeline view as shown in Figure 4. As we can see, information about the truncated elements (Cut) is important during this editing step.

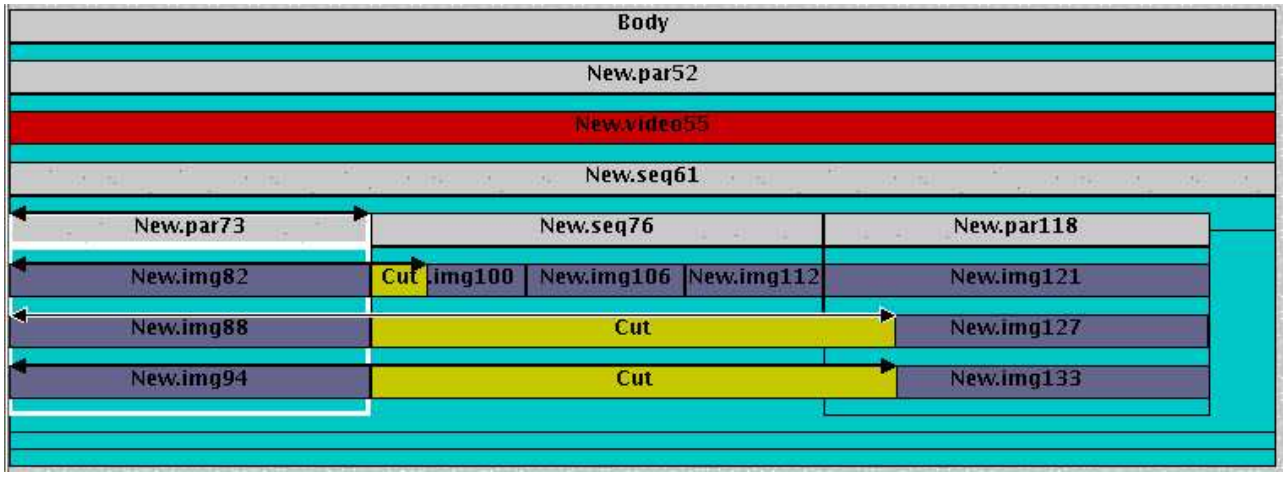

Figure 4: Directly adjusting the duration of elements

Note that thanks to the `id` attribute available with all elements in SMIL, every component of the presentation can be labelled.

After completing this editing process, the SMIL document can be saved and played with any SMIL player.

4 Editing SMIL Animations with Amaya

While the initial development of LimSee has put the emphasis on the SMIL 2.0 timing and synchronization model, another effort was focusing on a

complementary aspect of SMIL 2.0, animations. SMIL Animation is based upon the SMIL timing model and provides an animation functionality. It was designed to animate any XML format that requires this functionality. In particular, it is used to enliven SVG, the vector graphics language for the Web.

The main goal of our experiment with SMIL Animation in Amaya was to validate the timeline approach when manipulating another timed aspect of multimedia documents.

4.1 Amaya, SVG and SMIL Animation

Amaya is a Web client that acts both as a browser and as an authoring tool, the two features being seamlessly integrated. Web pages are edited in WYSIWYG mode, i.e. the user interacts on a formatted document, but the editor maintains simultaneously a structured representation of the document, a DOM tree. Every user command is first performed on this tree, and the part of the tree that has been modified is immediately reformatted and redisplayed. Several views of each document may be open simultaneously, to provide a more complete representation of the document being edited: in addition to the formatted view, Amaya can display the source code, the DOM tree, the hypertext links, and the outline of the document. All these views can be edited at any time, and all of them are synchronized to always present the current status of the document.

To allow users to really edit the Web, Amaya provides direct access to remote Web sites through the HTTP 1.1 protocol, both for reading and writing Web pages remotely (the HTTP Get and Put methods are used). Thus users can edit pages that are stored on servers exactly in the same way they work on local files.

Several document formats are supported natively in Amaya: HTML, XHTML, MathML, and SVG. This allows authors to edit various types of Web pages, including compound documents mixing structured text (XHTML or XML), mathematical expressions (MathML), and structured graphics (SVG). Here we focus on SVG graphics, but it is worth noting that all other aspects of a document can be edited simultaneously in a single and consistent environment. SVG elements are edited through the same model as the rest of the document: a DOM tree is manipulated through several views, but prominently through a final presentation view. A few specific editing commands are provided for graphics, as well as an additional view, a timeline, for manipulating animations.

As opposed to SMIL, where timing is represented in a single structure, the body of a SMIL document, SVG represents animation as elements interspersed through the main structure which represents the organization of the graphics. Animation elements appear as children of the graphics elements they animate. To allow the author to focus on animation, a timeline view shows all animation elements and group them together according to the graphics element they animate.

4.2 The Timeline View in Amaya

The timeline view in Amaya looks roughly the same as in LimSee, but there is an important difference. As time in SVG gets in play only for animating graphics elements, the timeline does not represent media objects, but animations associated with graphics objects. Each animated object of the document is represented there, with a graphical representation of its animation elements.

Figure 5, shows the three animated objects of a document. Each object is represented on the left side of the view by a label with a white background. Clicking this label highlights the corresponding element in the formatted view, thus providing the user with the context of that element. The animation elements associated with each graphics element are displayed in a box next to its label. If the graphics element has a single animation element (like element Rectangle in the figure), this animation element is displayed as a colored bar. If there are several elements, a single gray bar represents the whole animation (Circle, at the bottom of the figure), and a '+' button in the label allows the user to get a expanded representation. Element MyText, is an example of such an expanded representation where each animation element is represented by a colored bar. The button becomes a '-' that allows the user to get back to the condensed representation.

The color of each bar reflects the type of animation (`animate`, `set`, `animateMotion`, `animateColor`, `animateTransform`) and their position on the timeline depends on when they start and stop acting.

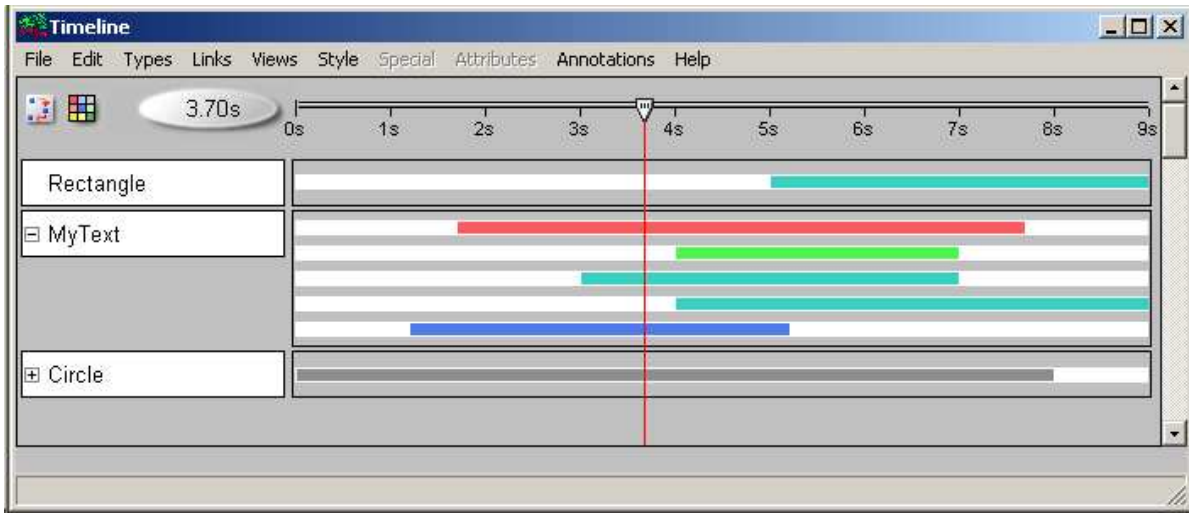

Figure 5: Animation view in Amaya

4.3 Editing Animations

The timeline view is helpful to quickly perceive the animation of all graphics in a document, but that is not its only role. It also allows an author to edit animation in a intuitive way. New animation elements can be created and existing elements can be modified at any time. Most manipulations are done

directly on the timeline, such as moving a bar or changing its length. This is immediately reflected in other views where the corresponding attributes of the animation element are updated (attributes `begin` and `dur` in that case).

In some cases, other views are also involved in editing an animation. As an example, when creating a movement for an existing graphics element, the user starts by selecting the element of interest in the main view, he/she then clicks on the top left button in the timeline view. This creates a new animation element in that view. The user then points at the starting position and at the ending position of the movement in the main view. Doing so, he/she can control the key positions of the animated element in the context of the other graphics elements. In fact, the user draws the motion path it as if he/she would draw the shape of any curve belonging to the document. Finally, going back to the timeline view, he/she can move and/or resize the new bar to adjust the timing. Manipulating timing in the timeline view is more comfortable, as this allows the user to better understand the synchronization of a particular element with the other animated elements. The user can also modify the key positions in the main view, just by moving points with the mouse. But the other views are still there, and some parameters may be adjusted in the structure view by editing attributes, as well as in the source code if necessary.

4.4 Implementation aspects

The timeline view is a structured graphics object itself. For that reason, it makes sense to implement it as an SVG document, by reusing all the SVG features already available in Amaya. In addition, SVG is a very complete graphics language which can represent everything a graphic designer may dream of. This gave us a lot of possibilities in designing the timeline view, whereas a GUI tool box would have been a pain, due to the many limitations of this kind of tool. Also, by using SVG, we take advantage of the full power of the language. In particular, depending on the complexity of the animations to be represented or the available screen space, the user may zoom in and out. Finally, the implementation work is done only once for all platforms. No need to adapt to several GUIs.

Interaction in the timeline view is handled directly by the graphic part of the Amaya editor. This is a benefit for the implementer (no new code is necessary) but also for the user. Editing commands are exactly the same in all views, including the timeline view. The user can manipulate a bar representing an animation in the same way he/she manipulates a rectangle in the document. Obviously some constraints are put in the timeline view. For instance colored bars can move only horizontally along the time axis and their height can not be changed individually. Those constraints represent the semantics of the timeline graphic language.

This approach will make further developments easier. As an example the `keySplines` attribute defining the cubic Bézier function that controls interval

pacing in an animation can be specified by editing a curve in exactly the same way as any Bézier curve that is part of a drawing.

5 Results and Concluding Remarks

These two experiments in developing authoring services for dynamic presentations have convinced us that time structures and time dependencies must be shown to the author as completely as possible and manipulations must be as intuitive as possible. Moreover providing several view points (graphic, time, structure, syntax) on the same document through different, coordinated views has proven very efficient when it comes to edit such complex objects as time structures and animations.

LimSee is available on several platforms and is freely distributed to both users and developers under an open source license. LimSee is still a prototype, and even if a substantial part of the SMIL timing model is covered, there are some basic elements or attributes that can not yet be edited directly (editing them is only possible through the hierarchical and attribute views). In particular the `excl` element is not yet implemented. Once LimSee offers a complete editing environment for all features of the SMIL2.0 timing and synchronization module, the plan is to work on the authoring aspects of other SMIL2.0 modules, with a higher priority for the linking, animation and transition modules. In fact we want to continue to investigate new authoring methods for handling SMIL features that are related to a timed behavior. The timeline views described here need to be enhanced in particular to give the author some hints about the schedule of his/her presentation in case of indeterministic behavior resulting from event-based synchronization and links. Finally we plan to cover other modules later and to address the problem of editing multiple profiles.

The SVG animation module of Amaya is planed to be integrated in a future public version of Amaya. At the same time this tool will gain in usability as some more SVG features are added.

6 Acknowledgements

LimSee results from the contributions of many WAM team members. In particular the Madeus model and constraint-based techniques for handling documents have been introduced by Nabil Layaïda, timelines and direct editing have been developed by Muriel Jourdan and Laurent Tardif. Patrice Navarro was the main developer of the LimSee prototype presented here. Daniel Weck is developing the production version.

The overall development of Amaya is coordinated by Irène Vatton and the SVG features are implemented by Paul Cheyrou-Lagrèze. Both of them helped us a lot in developing the animation module.

References

[1] G. J. Badros, A. Borning., "The Cassowary Linear Arithmetic Constraint Solving Algorithm: Interface and Implementation", Technical Report UW-CSE-98-06-04 (1998).

[2] M. Jourdan, C. Roisin, L. Tardif, "Kaomi, A Scalable Toolkit for Designing Multimedia Authoring Environments", *Multimedia Tools and Applications*, Kluwer Academic Publishers, num. 12, vol 2-3, pp. 257-279, November 2000.

[3] M. Jourdan, C. Roisin, L. Tardif, L. Villard, ''Authoring SMIL documents by direct manipulations during presentation'', World Wide Web, Balzer Science Publishers, vol. 2, num. 4, December 1999.

[4] M. Jourdan, N. Layaïda, C. Roisin, L. Sabry-Ismail, L. Tardif, ''Madeus, an Authoring Environment for Interactive Multimedia Documents'', ACM Multimedia'98, pp. 267-272, ACM, Bristol (UK), September 1998.