Design of the Authoring System for SMIL-based Multimedia Presentations

Chun-Chuan Yang and Yi-Zheng Yang

Multimedia and Communications Laboratory
Department of Computer Science and Information Engineering
National Chi Nan University, Taiwan, R.O.C.
ccyang@csie.ncnu.edu.tw

Abstract

In this paper, design issues for the authoring system of SMIL-based multimedia presentations are discussed and the solutions to these issues are presented. The proposed authoring system accepts the SMIL file as the input and provides useful editing functions, such as clear, cut, copy, and paste. The editing result of the presentation is finally saved in the SMIL format. For the input of a SMIL file, the authoring system first parses the document to compute the playback duration for each object. All editing functions are then performed on the playback duration of objects. A converting algorithm is proposed to convert the final result of editing to a SMIL file.

Keywords: Multimedia, Authoring, SMIL, RTSM

1. Introduction

Multimedia presentation is concerning with the integration of multimedia objects, which maybe locate at remote data servers. To provide a useful tool for composing presentations, an authoring system with easy-to-use editing functions is necessary. Design issues about the authoring system for multimedia presentations include the design of the user-interface [1-2], the format of the multimedia presentations [3-6], the mechanisms for supporting the editing functions [7-9], etc. The format of the presentation plays an important role for the authoring system, since it affects the popularity of the presentation. Unfortunately, different authoring systems usually have different format of presentation. It is better to adopt a popular language as the format of multimedia presentation [6].

From the popularity point of view, HTML seems to be the best candidate. However, the lack of the ability in integrating synchronized multimedia for HTML makes it improper to be the language of multimedia presentations. *Synchronized Multimedia Integration Language (SMIL)* [10-13] was developed by the *WWW Consortium* (*W3C*) to address the lack of HTML for multimedia over WWW. It provides an easy way to compose multimedia presentations. With the efforts of W3C, SMIL is becoming the most popular language in authoring multimedia and it is currently supported by the newest versions of commercial browsers. This paper thus focuses on the SMIL-based presentations and proposes architecture of the authoring system. We make a brief introduction to SMIL in the following.

SMIL could be used to describe both the spatial relationship and temporal relationship of a multimedia presentation. The spatial relationship is concerning with the visual layout of the presentation, while the temporal relationship is concerning with the timing control of media objects. The elements for spatial relationship in SMIL include the *<layout>* element and the *<region>* element. The <layout> element determines how the elements in the document's body are positioned. The <region> element controls the position, the size, and scaling of media elements.

The synchronization elements in SMIL for temporal relationship include the *<seq>* element, the *<par>* element, and the class of media elements such

as *<img>*, *<video>*, *<audio>* and *<text>*, etc. The <seq> element defines a sequence of elements in which elements play one after the other. The <par> element defines a simple parallel time grouping in which multiple elements can play back at the same time. Both <seq> and <par> allow the nested structure that means the children element of them could be any of the synchronization elements. The media elements allow the inclusion of media objects into an SMIL presentation. Media objects are included by reference (using a *URI*). Besides, some synchronization related attributes such as "*begin*", "*dur*", and "*end*" could be associated with these synchronization elements.

The remainder of the paper is organized as follows. First of all, the overview of the authoring system is presented in section 2. The parsing process to compute the playback duration of each object in a SMIL file is presented in section 3. The editing functions are presented in section 4, and the converting algorithm for saving the editing result in the SMIL format is presented in section 5. Finally, section 6 concludes this paper.

2. Overview of the system

The major task for an authoring system is to provide useful functions for composing multimedia presentations. The authoring process is usually composed of importing an existing presentation, editing operations, and finally saving the result, as illustrated in Figure 1.

The proposed authoring system focuses on the SMIL-based presentations; however, editing functions such as cut, copy, and paste are difficult to be realized in the manner of language, since these functions always involve the manipulation of time for each object. Thus, it is better to perform the editing functions in the time domain, instead of the language domain. In other words, the editing functions in the system are timeline-based. This is the reason why the input step in Figure 1 requires computing the playback duration for each object

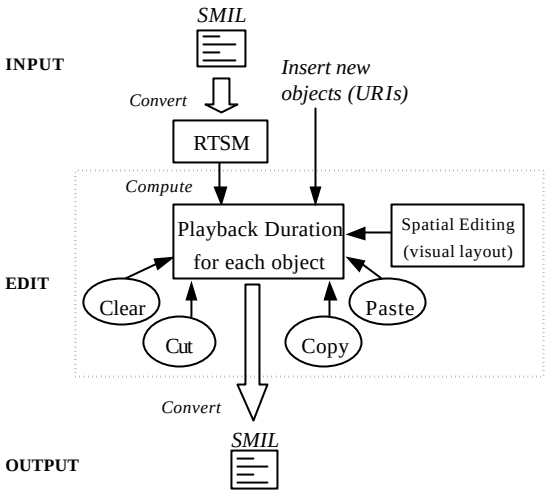


Figure 1. Overview of the authoring process

In the previous work, an algorithm calculating the playback duration of objects in a SMIL file was proposed [16]. Temporal relationship of the input script is extracted and represented by the Real-time Synchronization Model (RTSM) [15]. The playback duration for each object is then computed by traversing the model. The parsing process is briefly explained in section 3.

Editing function supported by the system should include spatial editing functions and temporal editing functions. Spatial editing is concerning with the visual layout, while temporal editing is concerning with the timing property of each object. Visual objects (such as video, image, text, etc) must be associated with a display region defined by the spatial editing functions. Hence, we could treat the display region for a visual object as one of its attributes. As shown in Figure 1, the editing functions, such as *clear*, *cut*, *copy*, and *paste* are performed on the playback duration of the object. The final result is represented in the SMIL format and is saved to a file.

3. Parsing SMIL

In this section, we make a survey for RTSM first and then briefly explain the parsing process by giving an example. For more details of the parsing process, please refer to the previous work [16].

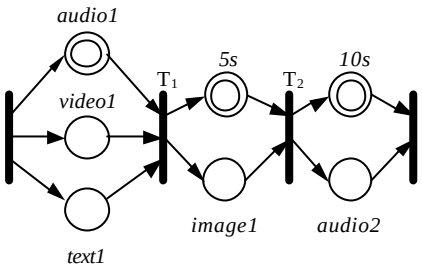


Figure 2. An example of RTSM

```
<seq>
    <par    endsync = id(URI-1)>
            <audio src=URI-1 />
            <video src=URI-2 />
            <text src=URI-3 />
    </par>
    <img src=URI-4 dur= ”5s” />
    <audio src=URI-5, dur= “10s” />
</seq>
```

Figure 3. Sample SMIL Document

3.1 Survey of RTSM

The elements in RTSM include *place*, *token*, and *transition* as in *Object Composition Petri Net* (*OCPN*) [14]. However, there are two kinds of places in RTSM, *regular places* and *enforced places*. A different firing rule for enforced places is defined. The rule specifies that once an enforced place becomes unblocked, the following transition will be immediately fired regardless the states of other places feeding the same transition.

An example of RTSM is shown in Figure 2 in which a single circle is for the regular place, a double circle is for the enforced place, and a bar is drawn for the transition. The RTSM in the figure requires that the audio segment *audio1*, the video clip *video1* and the text data *text1* be played simultaneously. Since *audio1* is an enforced place, transition *T1* is fired right after *audio1* is finished, regardless of whether *video1* or *text1* has finished or not. After firing *T1*, *image1* is displayed for 5 seconds then transition *T2* is fired. Finally, *audio2* is played for 10 seconds after

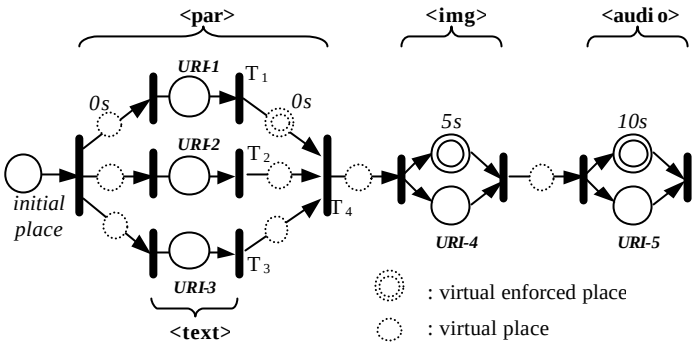


Figure 4. RTSM for the sample SMIL document

T2 fires. Note that the enforced place of “5s” in the figure is not a media object but a virtual medium that is called *Time Medium* [15]. The time medium is used to represent time duration.

3.2 An example for the parsing process

The SMIL script in Figure 3 requires the player to play the audio object *URI-1*, the video object *URI-2* and text object *URI-3* synchronously since these three objects are contained in a <par> element. The value of the “*endsync*” attribute in the <par> element requires <par> to end with the end of the audio object *URI-1*. In other words, once the audio object *URI-1* finishes playing, the video object *URI-2* and the text object *URI-3* must also stop playing at the same time. After the <par> element, the player has to display the image object *URI-4* for 5 seconds, and then play the audio object *URI-5* for 10 seconds. The obtained RTSM for the sample SMIL document after the converting process is shown in Figure 4. The enforced place (double circle) is defined to be the dominated place for firing the following transition, and the virtual place (dashed circle) is a place with zero duration.

Simplifying process is then invoked to remove the redundancy of the obtained RTSM. The simplified RTSM for the example is displayed in Figure 5. After that, reducing process removes places that feed into the same transition with an enforced place, and the reduced RTSM for the sample is shown in Figure 6.

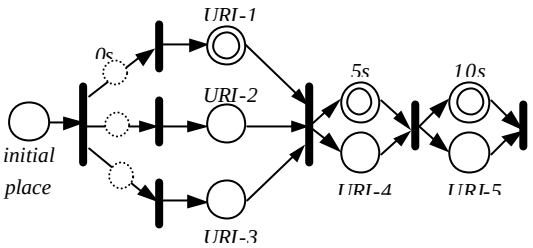


Figure 5. Simplified RTSM for the example

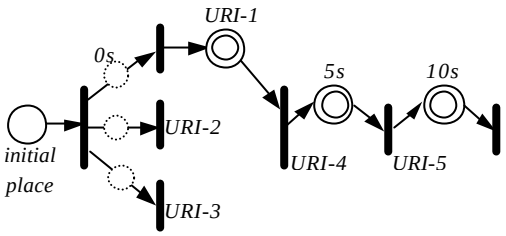


Figure 6. Reduced RTSM for the example

The playback duration for each object is actually from firing of the start transition to firing of the end transition in the reduced RTSM. The duration is computed by traversing the reduced RTSM started from the initial place. We illustrate the playback duration for objects of the example in Figure 7. The playback duration for each object is recorded for the editing functions presented in the next section.

4. Editing functions

After the parsing process, all media objects with corresponding attributes such as *URI*, *display region* (created by spatial editing), *playback duration*, etc. are stored in the *object table*. The playback duration for an object is denoted by (T^s_{obj}, T^e_{obj}) in the paper. The playback duration means that the object should be played out in the time interval after the presentation is started. The editing functions are then performed on the playback duration. In the following, we present the mechanism for each editing function. Note that only the mechanism for realization of the function is presented, the design issue of the user interface is not addressed in the paper.

4.1 Insert new objects

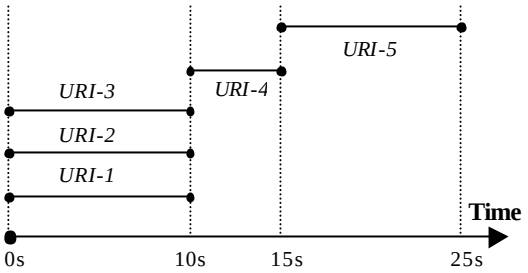


Figure 7. Object playback duration for the example.

Inserting a new object to the presentation means to add a new object to the object table. The user has to provide the values of attributes (URI, display region, playback duration, etc.) for the inserted object. For visual objects, we define the *spatial-temporal conflict* as the case that visual objects with the same display position have overlap in their playback periods. The visual objects that will be played out concurrently should not occupy the same display position. Therefore, the authoring system rejects the insertion request if there is a spatial-temporal conflict between the new visual object and those in the object table.

4.2 Modify object's attributes

The user could change the attributes of the objects in the object table. For example, the user may change the playback duration of an object by setting new values of (T^s_{obj}, T^e_{obj}) for the object, which reflects the action of moving, enlarging, or shortening the playback period of the object along the time line. Again, any modification of attributes for visual objects has to pass the test of spatial-temporal conflict.

4.3 Clear

The clear function could be used to clear (part of) an object in the object table. In addition, the system also allows the user to clear a zone in the time line. A new object type named *zone* is defined to differentiate from the normal objects. When the user specifies a time zone to clear, all objects within the zone are cleared. The algorithm for the clear function is shown Figure 8 and an example is given in Figure

Function **Clear** *(objID,* T^s_{clear} *,* T^e_{clear} *)*
If objID = zone then perform the following clear action on the objects whose (T^s_{obj} *,* T^e_{obj} *) overlaps with (* T^s_{clear} *,* T^e_{clear} *). (two periods do not overlap when* $T^s_{obj} > T^e_{clear}$ *or* $T^e_{obj} < T^s_{clear}$ *)*
if ($T^s_{obj} >= T^s_{clear}$ *and* $T^e_{obj} <= T^e_{clear}$ *) then delete the object*
else
if $T^s_{obj} < T^s_{clear}$ *and* $T^e_{obj} > T^e_{clear}$ *then split the object into two objects with periods (* T^s_{obj} *,* T^s_{clear} *) and (*T^e_{clear} *,* T^e_{obj} *)*
else if $T^s_{clear} < T^s_{obj} < T^e_{clear}$ *then new* $T^s_{obj} = T^e_{clear}$
else if $T^s_{clear} < T^e_{obj} < T^e_{clear}$ *then new* $T^e_{obj} = T^s_{clear}$
Else
Perform the above action on objID only

Figure 8. The algorithm for the clear function

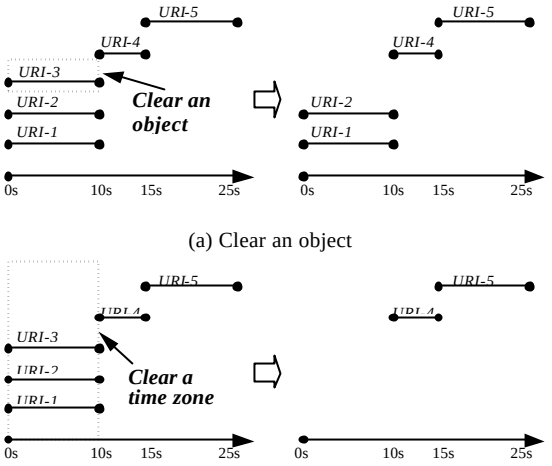
9, in which case (a) shows the case of clearing an object, and case (b) shows the case of clearing a time zone. Notice that the clearing action would sometimes result in the division of an object. For static objects like image and text, the division is merely reflected by setting new values for the playback periods. However, for continuous objects like video and audio, the division requires the authoring system to relocate the corresponding part of the medium data. Same situation happens in other editing functions.

4.4 Cut and Copy

The cut function provides a way to cut (part of) an object or a time zone and to save the cutting part in the clipboard for pasting. The cutting action for an object is similar to that of the clear function except that the cut part of the object is saved in the clipboard. However, cutting a time zone not only moves all objects within the specified time zone to the clipboard but also advances the playback periods by the length of the time zone for the objects, which are in back of the time zone. The algorithm of the cut function is displayed in Figure 10, and an example is given in Figure 11.

The copy function is similar to the cut function in the algorithm of saving objects in the clipboard, except that the copy function does not result in any change in the object table.

4.5 Paste



(a) Clear an object

(b) Clear a time zone

Figure 9. E.g., The <Clear> function

The paste function provides the user to paste the objects in the clipboard at some time point of the presentation. The paste action depends on the types of object stored in the clipboard. If only one single object in the clipboard, the paste action is similar to that of inserting a new object, in which the user could specify the time point and the display position for inserting the object. On the other hand, if a time zone object is stored in the clipboard, the paste function inserts the time zone (with all objects saved in the zone) at the specified time point. Figure 12 shows examples of the two cases for the paste function. The algorithm of paste function is shown Figure 13. Note that the algorithm does not allow the paste point T_{paste} within the playback period of some objects to reduce to complexity of paste function.

5. Save the result in SMIL

When user finishes the editing process and asks the authoring system to save the result, the system converts the objects in the object table to a SMIL file. The converting algorithm has to deal with both the spatial and temporal information of objects in the object table. The spatial information created by the spatial editing functions is concerning with the layout of display region for visual objects, and it is easy to convert the spatial information to the layout-related

Function **Cut** *(objID,* T^s_{cut} *,* T^e_{cut} *)*

If objID = zone then

For ? objects whose (T^s_{obj} *,* T^e_{obj} *) overlaps with (* T^s_{cut} *,* T^e_{cut} *).*

if ($T^s_{obj} >= T^s_{cut}$ *and* $T^e_{cut} <= T^e_{obj}$ *) then save the whole object in the clipboard and delete the object from the object table.*

else

if $T^s_{obj} < T^s_{cut}$ *and* $T^e_{obj} > T^e_{cut}$ *then split the object into two objects with periods (* T^s_{obj} *,* T^s_{cut} *) and (* T^e_{cut} *,* T^e_{obj} *) in the object table, and save (* T^s_{cut} *,* T^e_{cut} *) of the object in the clipboard.*

else if $T^s_{cut} < T^s_{obj} < T^e_{cut}$ *then*

save (T^s_{obj} *,* T^e_{cut} *) of the object in the clipboard, and new* $T^s_{obj} = T^e_{cut}$ *in the object table.*

else if $T^s_{cut} < T^e_{obj} < T^e_{cut}$ *then*

save (T^s_{cut} *,* T^e_{obj} *) of the object in the clipboard and new* $T^e_{obj} = T^s_{cut}$ *in the object table.*

Else

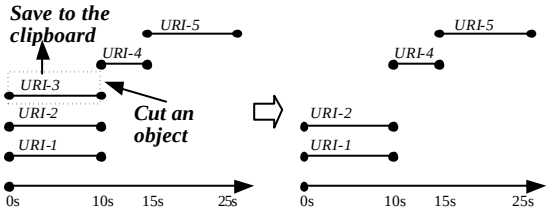
Perform the above action on objID only

If objID = zone then advance the playback periods which are in back of the zone.

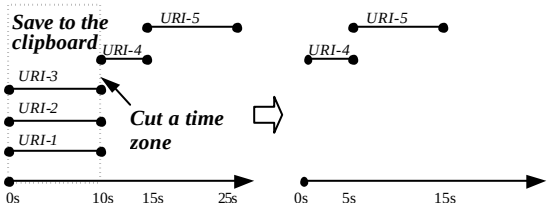
For all objects in the object table with $T^s_{obj} > T^e_{cut}$ *,*

Set $T^s_{obj} = T^s_{obj} - (T^e_{cut} - T^s_{cut})$ *and* $T^e_{obj} = T^e_{obj} - (T^e_{cut} - T^s_{cut})$

Figure 10. The algorithm for the cut function



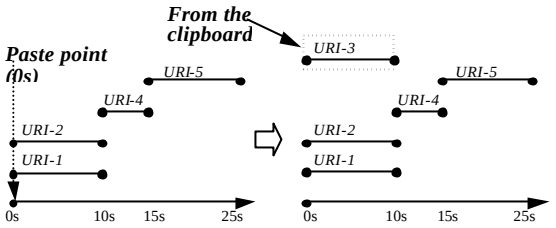
(a) Cut an object



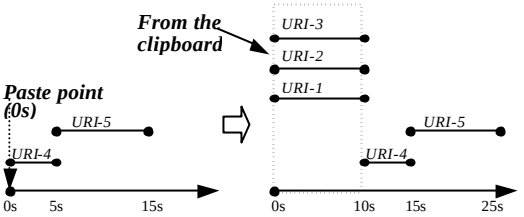
(b) Cut a time zone

Figure 11. E.g., The <Cut> function

elements in SMIL, such as <layout> and <region>. On the other hand, since the playback periods of objects may spread over the time line, the temporal information is much more complicated than the spatial information. The converting algorithm thus



(a) Paste: an object in the clipboard



(b) Paste: a time zone in the clipboard

Figure 12. E.g., The <Paste> function

Function **Paste** *(Clipboard,* T_{paste} *)*

If Clipboard = zone then

If exist an object with $T^s_{obj} < T_{paste} < T^e_{obj}$ *, reject the request*

1. Expand the timeline (from T_{paste} *) by the length of the zone.*

Assume the zone is (T^s_{zone} *,* T^e_{zone} *) and it's length =* L_{zone}

For all objects with $T^s_{obj} > T_{paste}$ *,* set $T^s_{obj} = T^s_{obj} + L_{zone}$ and $T^e_{obj} = T^e_{obj} + L_{zone}$

2. Add all objects in the clipboard to the object table, by setting $T^s_{obj} = T_{paste} + (T^s_{obj} - T^s_{zone})$, $T^e_{obj} = T_{paste} + (T^e_{obj} - T^s_{zone})$

Else

Insert the object in the clipboard to the object table.

(Visual object must pass the test of spatial-temporal conflict)

Figure 13. The algorithm for the paste function

focuses on the conversion of the temp oral information to SMIL.

The temporal information consists of a set of media objects each with its playback period. The most straightforward way to convert the temporal information is to treat all media objects as the children of a root <par> element. The "begin" attribute for each object is assigned to the playback time T^s_{obj} of the object and the "dur" attribute is assign to the length of the playback duration, i.e. $T^e_{obj} - T^s_{obj}$. The straightforward conversion is simple but introduces more overheads to the browser while presenting the SMIL file. The reason is that the

straightforward conversion makes all media objects the children of a <par> element, and the <par> element, by the definition, requires the browser to deal with all its children concurrently. Hence, more processing overhead and more buffers are required for browsing the resulted SMIL file of the straightforward conversion.

Therefore, from the processing point of view of the browser, more sequential parts in the resulted SMIL file make the browsing more efficient. Unfortunately, it is not easy at all to find as fewer as possible sets of objects with disjoint playback periods from the temporal information after the editing process. Thus, we try to find some clues from the semantic level.

First of all, since the number of medium used in a presentation is limited, we could first classify the objects by their medium type. Objects of each medium type form a child element of the root <par> element in the SMIL file. Furthermore, by observing that the authoring system does not allow the spatial-temporal conflict for visual objects, it implies that the visual objects with the same display position form a set of disjoint playback periods. Therefore, if we further classify the objects of a visual medium by their display position, we could determine all the sets of disjoint objects for that medium. More specifically, a set of disjoint objects forms a <seq> element for the same display position, and all the sets of the same medium type further forms the children of a <par> element which is one of the children of the root <par>. We illustrate the idea by the example in Figure 14.

The classification by the display position does not work for non-visual objects like audio, so we developed the *Scan2SMIL* algorithm to convert the non-visual objects. The *Scan2SMIL* does not consider any semantic relationship among the objects, but only provide a rule to determine a set of disjoint objects in each scan (iteration). The first step in

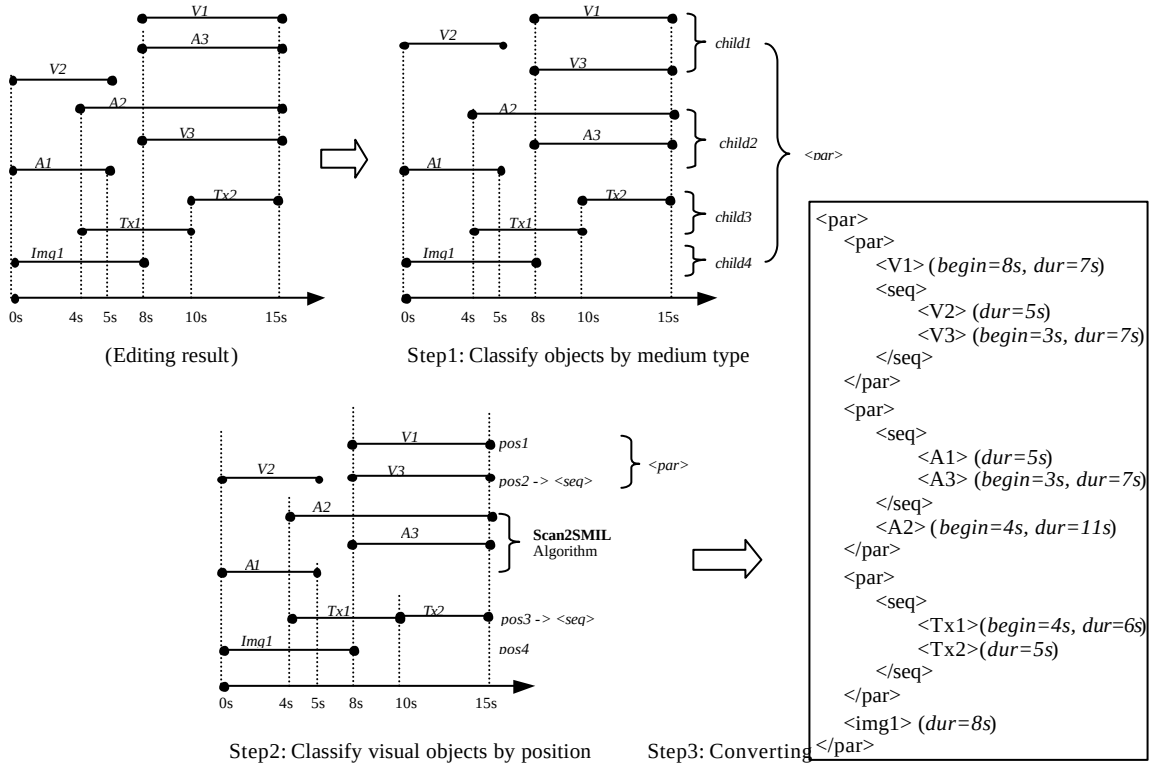


Figure 14. E.g., Convert playback duration to SMIL

Algorithm ***Scan2SMIL*** *(for non-visual objects)*
1. Create a <par> element for the medium type
The <par> element is a child of the root <par> in the SMIL file
2. Sorting all objects by T^s_{obj}
3. While object exists in the object table do {
Select the object with smallest T^s_{obj}
Add the object to a new set S.
Find the nearest object that is disjoint with all objects in S.
Add the found object to S and continue to add disjoint objects to S until all objects are scanned.
Remove the objects in S from the object table.
The objects in S compose a <seq> element which is the child of the <par> element created in step 1.
}

Figure 15. The *Scan2SMIL* algorithm

Algorithm ***Convert2SMIL***
1. Create the root <par> element
2. Classify objects by the medium type
Objects of each type compose one child of the root <par>
3. For visual objects
Classify objects by the display position
Objects of each display position form a <seq> element
All <seq> elements of different display positions form a <par> element.
The <par> element is a child of the root <par>.
4. For non-visual objects
Perform the Scan2SMIL algorithm

Figure 16. The *Convert2SMIL* algorithm

Scan2SMIL is sorting the objects by T^s_{obj}. Next, the algorithm selects the object with smallest T^s_{obj} as the first object of a new set of disjoint objects. The algorithm then searches the nearest disjoint object for the set of disjoint objects, i.e. the object with the smallest value of T^s_{obj} such that T^s_{obj} of the object >= T^e_{obj} of the first object in the set. The nearest object is added to the set of disjoint objects as the second object. The scanning process continues to search the next nearest disjoint object for the second object, the third object, etc. until all objects are scanned. The objects in the set obtained from the iteration obviously form a <seq> element and are removed from the object table. Similarly, following iterations create other sets of disjoint objects and form more <seq> elements. The algorithm stops when no objects in the object table. All <seq> elements from all iterations form the children of a <par> element of the non-visual medium.

The *Scan2SMIL* algorithm is displayed in Figure 15, and the converting algorithm (*Convert2SMIL*) for the temporal information is displayed in Figure 16. In fact, if the authoring system allows the existence of the spatial-temporal conflict in the presentation, the *Convert2SMIL* algorithm should adopts *Scan2SMIL* for objects of each medium type, instead of considering the semantic level conversion.

6. Conclusion

The architecture of the authoring system for SMIL-based presentations is proposed in the paper. Kernel mechanisms of the authoring process, such as input, editing, and output, are presented. Useful timeline-based editing functions such as clear, cut, copy, and paste are proposed, and the algorithms for them are also included in the paper. In order to support the timeline-based editing functions, the input SMIL file is converted to RTSM and the playback period of each object is computed. The editing result is finally saved in the format of SMIL. An algorithm for converting the timeline-based editing results to the SMIL format is proposed.

Reference

[1] J. Song, M.Y. Kim, G. Ramalingam, R. Miller, and B.K. Yi, "Interactive authoring of multimedia documents," Proceedings, IEEE Symposium on Visual Languages, 1996, pp. 276 – 283.

[2] M. Jourdan, C. Roisin, and L. Tardif, "Multiviews interfaces for multimedia authoring environments," Proceedings, Multimedia Modeling, 1998 (MMM '98), pp. 72 – 79.

[3] L. Hardman, G. van Rossum and Dick C. A. Bulterman, "Structured multimedia authoring," Proceedings, 1st ACM international conference on Multimedia, 1993, pp. 283 – 289.

[4] J. Emery and A. Karmouch, "A time-based multimedia document architecture," Proceedings, IEEE ICC 1995, vol.1, pp. 555 – 559.

[5] M. Jourdan, N. Layaïda, C. Roisin, L. S.-Ismaïl and L. Tardif, "Madeus, an authoring environment for interactive multimedia documents," Proceedings, 6th ACM international conference on Multimedia, 1998, pp. 267 – 272.

[6] Shi-Kuo Chang, "Perspectives in multimedia software engineering," Proceedings, IEEE International Conference on Multimedia Computing and Systems, 1999, pp. 74-78.

[7] S. Hudson and C.-N. His, "The walk-through approach to authoring multimedia documents," Proceedings, the second ACM international conference on Multimedia 1994, pp. 173 – 180.

[8] J. Kelner, D. Hadj Sadok, F. Marques, and A. Neves, "The role of parametrization in the multimedia authoring process," Proceedings, IEEE Conference on Protocols for Multimedia Systems - Multimedia Networking, 1997, pp. 142 –149.

[9] D. Del Corso, G. Morrone, E. Ovcin, A. Truzzi, C. Scrizzi, and M. Gastaldi, "Interactive educational multimedia: a quick design and development tool," Proceedings, IEEE International Conference on Multimedia Computing and Systems, 1999, vol. 2, pp. 841 –845.

[10] *Synchronized Multimedia Integration Language (SMIL) 1.0 Spec*. W3C Recommendation, June 1998, http://www.w3c.org/TR/REC-smil.

[11] *Synchronized Multimedia Integration Language (SMIL) Boston Specification*, W3C Working Draft 20-August-1999, http://www.w3.org/TR/smil-boston.

[12] Larry Bouthillier, "Synchronized Multimedia on the Web- A New W3C Format is All Smiles," Web Techniques Magazine, September 1998, Vol. 3 Issue 9.

[13] Philipp Hoschka, "An introduction to the Synchronized Multimedia Integration Language," IEEE Multimedia, Oct.-Dec. 1998, pp. 84-88.

[14] T. D. C. Little, and A. Ghafoor, "Synchronization and storage models for multimedia objects," IEEE Journal of Selected Area in Communications, vol. 8, no. 3, pp. 413-427, Apr. 1989.

[15] C. C. Yang and J. H. Huang, "A Multimedia Synchronization Model and its Implementation in Transport protocols," IEEE Journal of Selected Area in Communications, vol. 14, No. 1, pp. 212-225, Jan. 1996.

[16] Chun-Chuan Yang, "On the Design of the Data-Retrieving Engine for Distributed Multimedia Presentations", accepted by IEEE ICC2001, http://www.csie.ncnu.edu.tw/~ccyang/Publication/ICC2001-1.pdf.

Biographies

Chun-Chuan Yang received his B.S. degree in computer and information science from National Chiao-Tung University, Taiwan, in 1990 and Ph.D. degree in computer science from National Taiwan University in 1996. Since 1998, he has been an assistant professor in the department of computer science and information engineering, National Chi-Nan University, Taiwan. His research area of interests includes multimedia network protocols, multimedia synchronization control, and multimedia applications.

Yi-Zheng Yang received his B.S. degree in industry education from National Changhua University of Education, Taiwan, in 1999. He is currently a master student in the department of computer science and information engineering, National Chi Nan University. His current research topic is the design and implementation of the multimedia authoring system for distance learning.