Supporting VCR-like Operations in SMIL2.0 Players

Chun-Chuan Yang, Chih-Wen Tien, and Yung-Chi Wang

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, we deal with supporting of the VCR-like operations such as *fast forward/backward* and *sliding* for SMIL2.0 presentations. Mechanisms for supporting VCR-like operations for a SMIL2.0 player are proposed. Normal play mode for a SMIL2.0 presentation is based on *Extended Real-Time Synchronization Model* (*E-RTSM*) that provides a convenient way to represent the temporal scenario in the presentation. Fast forward/backward and sliding operations are based on the typical playback time of each object in the presentation, in which the typical playback time is defined as the playback time calculated without considering the non-deterministic events in the script. A skipping strategy is adopted to implement fast forward/backward operations. Moreover, a new design for the slider is also proposed to address some of the non-deterministic characteristics in SMIL2.0 presentations.

1. INTRODUCTION

With the development of high-speed networking and computer technologies, multimedia-rich applications over WWW have become very attractive for Internet users in recent years, and almost every web site on Internet is providing multimedia materials for visitors. To address the lack of HTML for composing multimedia documents, *Synchronized Multimedia Integration Language* (*SMIL*) [1-4] was developed by *WWW Consortium* (W3C). With SMIL, an author can create a multimedia presentation integrating audio, video, images, animations, text, etc.

There are two versions of SMIL specification that have been released by W3C. SMIL version 1 (SMIL1.0) [1] is basically a *time-based* (*schedule-based*) model. The second version of SMIL (SMIL2.0) [2] enhances SMIL1.0 by providing a strong support for user interaction with a declarative *event-based* timing. Some of the commercial multimedia players such as *GRiNs* [5], *RealOne* [6], and the latest version of *Internet Explorer* (versions 5.5 and up) support the playback of SMIL2.0 presentations. While viewing a multimedia presentation via a SMIL player, the user should be able to perform VCR-like control functions such as *play/stop*, *pause/resume*, *fast forward/backward*, and *sliding* operations that provide great flexibility over the presentation. These commercial SMIL players only support simple operations like *play/stop* and *pause/resume*, but few of them provides *fast forward/backward* and *sliding* operations for SMIL2.0 presentations (*RealOne* provides the sliding operation).

In our previous work, the mechanisms of supporting VCR-like operations for SMIL1.0 presentations had been developed and proposed [7, 8]. In this paper, extension of the previous work to support VCR-like functions for SMIL2.0 presentations is presented.

The rest of the paper is organized as follows. First of all, we briefly explain our previous work on supporting the VCR-like functions for SMIL1.0 presentations in section 2. Mechanisms supporting VCR-like operations for SMIL2.0 presentations are presented in section 3. Finally, section 4 concludes this paper.

2. PREVIOUS WORK

In order to support VCR-like operations like fast forward/backward and sliding, a SMIL1.0 player has to know before run-time the total presentation time as well as the playback time (duration) for each object under normal play mode. The first step to compute the playback time is to convert the temporal relationship of the objects in a SMIL script to *Real-Time Synchronization Model* (*RTSM*) [9]. RTSM provides a systematic view for the timing information in a presentation such that the playback time for each time can be easily obtained via some graph reduction and traversal mechanisms [7, 8]. We use an example to briefly explain the process.

RTSM for the sample SMIL1.0 code snippet in Figure 1 is displayed in Figure 2. Note that an *enforced place* (double circle) in RTSM dominates the firing time of its following transition. Thus, when the place A3 is finished playing, its following transition fires regardless of the state of place V3. Since a regular place does not have any effect on the firing time of its following transition when an enforced place is present, the player further reduces RTSM by removing the regular places that feed into the same transition with an enforced place. The reduced RTSM for Figure 2 is shown in Figure 3. The playback time is then computed by

```
<seq>
    <par endsync = “last”>
        <audio id=A1>
        <video id=V1>
    </par>
    <img dur=”10s” id=I1>
    <par endsync = “first”>
        <audio id=A2>
        <video id=V2>
    </par>
    <img dur=”10s” id=I2>
    <par endsync = “A3”>
        <audio id=A3>
        <video id=V3>
    </par>
</seq>
...
```

Figure 1. A SMIL1.0 code snippet

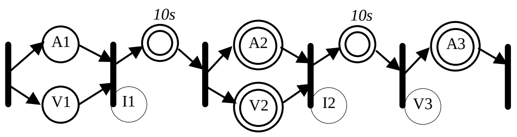

Figure 3. Reduced RTSM for Figure 2

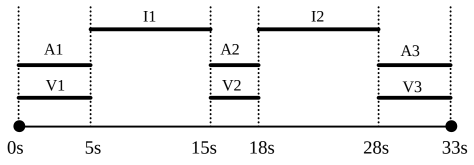

Figure 4. Playback time for each object in Figure 1

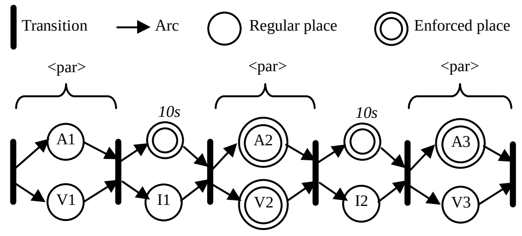

Figure 2. RTSM for the sample in Figure 1

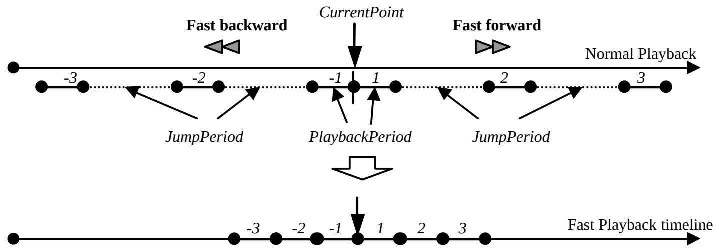

Figure 5. Playback pattern for fast forward/backward

traversing the reduced RTSM. For example, we assume that the intrinsic duration of A1, A2, and A3 is 5s, and V1, V2, and V3, 3s, the playback time for the sample SMIL1.0 document is displayed in Figure 4.

A skipping strategy was adopted to implement fast forward/backward operations. As illustrated in Figure 5, we defined *JumpPeriod* as the period to be skipped and *PlaybackPeriod* as the period to be played in fast forward/backward operations. Thus, the playback speed of fast forward/backward = (*JumpPeriod* + *PlaybackPeriod*) / *PlaybackPeriod*. The PlaybackPeriod is used to locate the media objects (or part of an object) that compose the new playback pattern.

Sliding operation allows the user to change the playback point of the presentation. When the user operates the slider, parameters for determining the next playback point are passed to the player so that the player can determine the objects that should be played next for the user’s action. Please refer to our previous paper [8] for more details of mechanisms supporting VCR-like operations over SMIL1.0 presentations.

3. VCR-LIKE OPERATIONS FOR SMIL2.0

3.1. Normal play operation

To play a SMIL2.0 presentation, the player has to know the temporal scenario presented by the script. We have proposed *Extended RTSM* (*E-RTSM*) [10] to model the temporal relationship of SMIL2.0. E-RTSM provides a systematic view of the temporal scenario in a SMIL2.0 presentation and it is helpful to use E-RTSM for the player to more easily determine the media objects to be played under normal play mode. For example, a sample SMIL2.0 code snippet is displayed in Figure 6, and the E-RTSM for the sample is shown in Figure 7. Note that in Figure 7, a *non-deterministic event* (an enforced place with a “?” in it) represents an event with an unknown occurring time, and the *dynamic arc* following a non-deterministic event will change to a normal arc when the event occurs at run-time, which means a dynamic arc does not have any effect on its following transition until it has occurred at run-time.

Moreover, it is easy to implement the *pause* operation since the player only has to record the run-time state of E-RTSM of the presentation when the pause action is made and waits for the user to make the *resume* action for continuation of the playback.

3.2. Fast Forward and Fast Backward operations

Event-based timing in SMIL2.0 presents the non-deterministic synchronization behavior such that the player cannot get the accurate playback time (and duration) for a media object as well as the total length of a presentation before run-time. Although E-RTSM provides a helpful tool for representing the temporal scenario in the presentation, it is difficult to predict beforehand the

```
<seq>
  <par endsync = "A1">
    <audio id=A1>
    <video begin="Btn1.Click" end="Btn2.Click" id=V1>
  </par>
  <img dur="10s" begin="Btn3.Click" end="Btn4.Click" id=I1>
  <par endsync = "A2">
    <audio id=A2>
    <video begin="Btn1.Click" end="Btn2.Click" id=V2>
  </par>
  <img dur="10s" begin="Btn3.Click" end="Btn4.Click" id=I2>
</seq>
...
```

Figure 6. A SMIL2.0 code snippet

```
<seq>
  <par endsync = "A1">
    <audio id=A1>
    <video id=V1>
  </par>
  <img dur="10s" id=I1>
  <par endsync = "A2">
    <audio id=A2>
    <video id=V2>
  </par>
  <img dur="10s" id=I2>
</seq>
...
```

Figure 8. Reduced SMIL2.0 script for Fig. 6

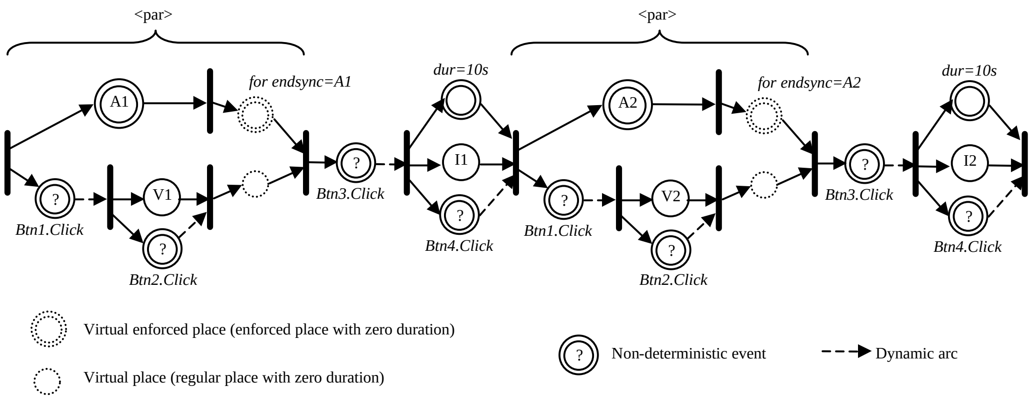

Figure 7. E-RTSM for the sample in Figure 6

occurring time of a non-deterministic event. However, the timeline information for each object in a SMIL2.0 presentation is still required for supporting fast forward/backward operations. For that reason, we define the *typical playback time* of an object to be the playback time calculated without considering the non-deterministic events, and fast forward/backward operations are implemented on the basis of the typical playback time.

In order to obtain the typical playback time, the player first reduces a SMIL2.0 script by removing non-deterministic part in the script. That is, the attributes with value of a non-deterministic event (event with unknown occurring time, such as *Mouse-Click*, *Key-Pressed*) in an element are removed, and the temporal relationship of the resulted script (a SMIL1.0-like script, for example, Figure 8 is the resulted script for Figure 6) can be represented by RTSM. Typical playback time of each object is then computed by processing the RTSM of the reduced script as in the case of SMIL1.0. The way of skipping for fast forward/backward operations are therefore based on the typical playback time, which means all non-deterministic events are ignored in the fast play mode.

3.3. Sliding operation

As the user is operating the slider to change the playback point in the presentation, the player has to display on the screen the corresponding part in the presentation according to the position of the slider. The corresponding part is called a *scene* in the paper. Only visual objects (video, image, text, etc) are displayed in a scene as the user is operating the slider. The position of the slider indicates a time point in the typical playback timeline. A visual media object with playback time covering the time point indicated by the position of the slider should be displayed in the scene. There will be no problem if the object that should be displayed in a scene is a static object like image. For a streaming visual object like video, the player has to determine the corresponding frame to be displayed in the scene. The playback time of the streaming visual object and the time point of the scene are used to locate the corresponding frame in a video object for the scene.

When the sliding operation is finished, the player should continue playing the presentation (in the normal play mode) starting from the time point indicated by the final position of the slider. The last scene for the sliding operation is then mapped to the corresponding state of E-RTSM for the normal play mode of the presentation as mentioned in section 3.1.

3.4. New design for the slider

The idea behind the method of using the typical playback time to implement the sliding operation is to treat the non-deterministic events as a supplement to the presentation for the reason of more user interaction. Thus,

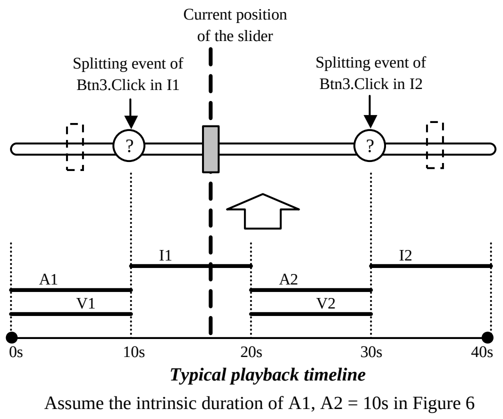
Current position of the slider
Splitting event of Btn3.Click in I1
Splitting event of Btn3.Click in I2
?
?
I1
I2
A1
A2
V1
V2
0s
10s
20s
30s
40s
Typical playback timeline
Assume the intrinsic duration of A1, A2 = 10s in Figure 6

Figure 9. New slider for the sample SMIL2.0 script

these supplementary non-deterministic events can be ignored when performing the sliding operation. However, the idea is not always true since some of the non-deterministic events may play an important role in the playback of a presentation. For example, as we take a look at the E-RTSM in Figure 7, we find that there are actually two types of non-deterministic events in the model: the *splitting events* and the *non-splitting events*. An event is a splitting event when removing the event results in two separate parts in the model. The player must wait for the occurrence of a splitting event before it can continue playing the presentation. Event *Btn3.Click* for I1 (and I2) in Figure 7 is an example of a splitting event. On the other hand, events *Btn1.Click*, *Btn2.lClick* and *Btn4.Click* are all non-splitting events.

A splitting event implies that the author of a presentation would like the user (viewer) to trigger (or wait for) some event before continuing the playback of the presentation. Thus, it is better to let the user know the author’s intention even when the user is operating a slider. Therefore, a new design of the slider emerges. The new slider includes information about splitting events in a presentation. In our design, a *round button* marked “?” is added on the slider bar to represent a splitting event. The position of a round button on the slider bar is according to the typical playback time of the element containing that splitting event. Figure 9 shows an example of the new design for the sample SMIL2.0 script in Figure 6. Since there are two splitting events in the sample SMIL2.0 script, two round buttons are added on the slider bar. Positions of the two round buttons are according to the playback time of the corresponding elements (i.e. I1 and I2) of the splitting events.

Operation of the new slider is similar to the one presented in section 3.3 except when the user clicks on a round button, the current position of the slider moves to the round button and the player displays the corresponding scene to wait for the occurrence of the splitting event.

4. CONCLUSION

VCR-like operations such as fast forward/backward and sliding provide users more flexibility of playback control over a presentation. In our previous work, mechanisms of supporting VCR-like operations for SMIL1.0 presentations had been proposed. Extension of the mechanisms for SMIL2.0 presentations is presented in this paper. *Extended Real-Time Synchronization Model* (*E-RTSM*) is used to represent the temporal scenario in a presentation and for supporting the normal playback mode. The typical playback time, which is defined as the playback time calculated without considering the non-deterministic events in the script, is used to support fast forward/backward and sliding operations. Moreover, the type of splitting events is defined as the non-deterministic events that the player should wait for before continuing the playback of the presentation. A new design of the slider considering the splitting events is proposed.

REFERENCES

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

[2] *Synchronized Multimedia Integration Language (SMIL) 2.0 Specification*, W3C Recommendation, 2001, http://www.w3.org/TR/smil20

[3] D.C.A. Bulterman, “SMIL 2.0 part 1: overview, concepts, and structure,” IEEE Multimedia, Volume: 8 Issue: 4, Oct.-Dec. 2001, Page(s): 82 –88

[4] D.C.A. Bulterman, “SMIL 2.0. 2. Examples and comparisons,” IEEE Multimedia, Volume: 9 Issue: 1, Jan.-March 2002, Page(s): 74 –84

[5] Oratrix, GriNS for SMIL-2.0, http://www.oratrix.com /GRiNS /index.html.

[6] RealNetworks, RealOne, http://www.realnetworks.com /solutions/leadership/realone.html.

[7] C. C. Yang, “Design of the Data-Retrieving Engine for Distributed Multimedia Presentations,” Proceedings, IEEE International Conference on Communications, 2001 (ICC2001), pp. 3237-3243.

[8] C. C. Yang, “User-Interaction Supported Data- Retrieving Engine for Distributed Multimedia Presentations,” Proceedings, IEEE International Conference on Communications, 2001 (ICC2001), pp. 3244-3250.

[9] 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.

[10] C. C. Yang, C. W. Tien, and Y. C. Wang, “Modeling of the Non-deterministic Synchronization Behaviors in SMIL2.0 Documents,” also in Proceedings of IEEE International Conference on Multimedia and Expo (ICME) 2003.