Dividable Dynamic Timeline-based Authoring for SMIL2.0 Presentations

Chun-Chuan Yang, Chen-Kuei Chu, and Yung-Chi Wang
Multimedia and Communications Laboratory
Department of Computer Science and Information Engineering
National Chi Nan University, Taiwan, R.O.C.
cyang@csie.ncnu.edu.tw

ABSTRACT
Timeline-based editing provides users an intuitive and friendly way for multimedia authoring, but the schedule-based and deterministic property of timeline results in the lack of the ability for supporting non-deterministic temporal behavior, which is one of the key features of SMIL2.0. This paper presents the effort of supporting non-deterministic temporal behavior by timeline-based editing. The concept of Dividable Dynamic Timeline (DDTL) is proposed in the paper, which includes two novel features: dividable timeline and dynamic section. With DDTL, authors can create interactive multimedia presentations while enjoying the convenience of timeline. Mechanisms of converting from DDTL editing results to SMIL2.0 are also presented in the paper.

1. INTRODUCTION
Synchronized Multimedia Integration Language (SMIL) [1-3] developed by WWW Consortium (W3C) provides Internet users a mechanism to compose multimedia documents. With SMIL, authors can create multimedia presentations integrating video, audio, animation, image, text, etc. There are two versions of SMIL specification that had been released. The current version of SMIL (SMIL2.0) enhances the previous scheduled-based version (SMIL1.0) timing.
Normally authors must specify the exact playback time and duration of each media object as well as the total length of the presentation cannot be determined before run-time. Apparently, the original schedule-based scheme cannot support authoring of non-deterministic temporal behavior. Thus, two novel features, namely dividable timeline and dynamic section, are proposed in this paper to support non-determinism in timeline-based editing. The new editing scheme is thus called Dividable Dynamic Timeline-based (DDTL-based) authoring.

The rest of the paper is organized as follows. First of all, the concept of DDTL as well as the kernel mechanisms in DDTL-based authoring is presented in section 2. Converting of DDTL data to SMIL2.0 format is explained in section 3. Finally, section 4 concludes this paper.

2. DIVIDABLE DYNAMIC TIMELINE (DDTL)
Normally authors must specify the exact playback time and duration of each object in timeline-based editing. According to an object’s playback time and duration, a timeline segment representing that object is displayed at corresponding position on the timeline. In order to introduce non-deterministic temporal behavior in timeline and decide the extent that timeline-based editing can do for SMIL2.0 authoring, we need to investigate the synchronization characteristics in SMIL2.0.

To systematically deal with SMIL2.0, an extension of our SMIL1.0 modeling technique had been proposed in our previous work [7, 8]. The new model was called Extended Real-Time Synchronization Model (E-RTSM). Figure 1 shows a sample SMIL2.0 code snippet and its corresponding E-RTSM model. It is easy to understand from the model that non-deterministic temporal behavior in the code snippet comes from the non-deterministic events (denoted by a double circle with a “?”). Moreover, two types of the non-deterministic events were identified in the E-RTSM model: splitting event and non-splitting event [8]. An event is a splitting event when removing the event results in two separate parts in the timeline. The player must wait for the occurrence of a splitting event before it can continue playing the rest of the presentation. Event Btn3.Click in Figure 1 is a splitting event. Events Btn1.Click and Btn2.Click are non-splitting events.

In this paper, inspired by the two different types of non-deterministic events, we introduce two novel features in timeline-based editing: dividable timeline (from the idea of splitting event) and dynamic section (from the idea of non-splitting event), which are explained respectively in the following subsections.

2.1. Dividable Timeline
We define TL-Divide operation enabling users to divide a timeline table into two sequentially separated timeline tables and associate a non-deterministic splitting event (e.g., user’s mouse action) with the beginning of the latter timeline table. An example of TL-Divide operation is shown in Figure 2. To divide a timeline table, the author...
has to specify a proper cutting point on the time axis such that no objects will be cut into two pieces. Moreover, in order to properly display information of the time axis for the latter timeline table, we need to choose a preset (default) value (5s in Figure 2) for the occurrence time of the splitting event. The preset value is only for reference in authoring stage since the exact occurrence time of the event is unknown before run-time. For more flexibility, the authoring system should enable users to change the reference occurrence time of the event as user’s wish.

The reverse of TL-Divide is TL-Merge operation. The author uses the operation to remove the non-deterministic event and merge two separated timeline tables. Figure 3 gives an example of TL-Merge operation.

2.2. Dynamic section

A dynamic section (DS) is defined for a dynamic object whose beginning and/or ending are triggered by non-splitting events. As shown in Figure 4-(a), to create a DS for an object, the user has to specify the position (anchor) and the length of the DS on the time axis. Two events are associated with beginning and ending of the object in the DS. That is, DS defines the time range of playback for the object, but the actual starting time and ending time of the object are determined until run-time. Thus, the positions of the two events on the time axis are only for reference in authoring stage. As illustrated in Figure 4-(b), DS can also control a group of parallel objects such that the group acts just like a single object in the DS. The authoring system should provide the option to disable/enable events in a DS for more flexibility.

Given that the author may expect a dynamic object depending on a former dynamic object. We propose a DS operator called connector to support dependency between different dynamic sections. As illustrated in Figure 5, beginning of DS2 depends on DS1’s ending, and DS1’s ending depends on event Btn2.Click. By using DS connector, the author can easily create a sequence of dependent dynamic objects.
3. CONVERTING DDTL TO SMIL2.0

The editing result of DDTL needs to be converted to SMIL2.0 format and saved as a SMIL document for future playback on a SMIL player. We had developed a converting algorithm from deterministic timeline data to SMIL1.0 in our previous work [6]. We only present the converting mechanisms for dividable timeline and dynamic section in this paper, and the mechanisms can be easily integrated into the original converting algorithm in our previous work.

3.1. Converting algorithm for DDTL

Since the two separated timeline tables resulted from TL-Divide operation have a sequential relation, a parent `<seq>` time container is created for the two timeline tables, and the root time container of the latter timeline table must set its `<begin>` attribute the non-deterministic event. Figure 6 shows the code snippet for the result of TL-Divide in Figure 2.

To convert a DS to SMIL2.0, we need to know that there are two run-time cases ending a DS as well as the dynamic object in the DS: (1) the viewer triggers the ending event for the object, or (2) no ending event triggered but the preset ending time of the DS (which is according to the length of the DS) is reached. Therefore, in the conversion, we need to create a parent time container `<par>` for the object and set the `<end>` attribute of the `<par>` to reflect the above two cases. Figure 7-(a) shows the code snippet for the DS in Figure 4-(a), in which the `<end>` attribute of the parent element `<par>` is a list of two cases to end the DS (i.e. event `Btn2.Click` or time `T_1` is reached). The `<begin>` attribute in the parent `<par>` specifies the location of the DS on the time axis, and the `<begin>` attribute of the object defines the triggered event for starting the object. Similarly, the conversion of the DS containing a group of objects (Figure 4-(b)) is shown in Figure 7-(b).}

For a sequence of dynamic sections connected by DS connectors, we need to create a parent `<seq>` for all DS in the sequence. The code snippet for a sequence of DS (Figure 5) is displayed in Figure 8. Please note that the time base of the first DS (DS1, with an anchor) and the time base of the other DS (DS2 and the following) are different, which is reflected in the calculation of the ending time of the root `<par>` element for each DS.

Due to the limit of paper length, the detailed algorithm for converting DDTL to SMIL2.0 is not presented in the paper. Instead, we give a typical example of the conversion. A sample DDTL editing result is
DDTL-based authoring is addressed in another paper [12]. Apparently, DDTL (and other timeline-based editing schemes) cannot cover the whole set of non-deterministic temporal behaviors.

In this paper, we present our first effort towards supporting of SMIL2.0 authoring by timeline-based editing. The concept of Dividable Dynamic Timeline (DDTL) is proposed, which includes two novel features, divisible timeline and dynamic section, to extend the original timeline scheme with the support of non-deterministic temporal behavior. Mechanisms for converting DDTL data to SMIL2.0 format are also proposed in the paper. In summary, DDTL features the easy-learning characteristic of timeline and to some extent allows authors to compose interactive multimedia presentations. The implementation of extending our previous SMIL1.0 authoring system to support DDTL is currently under way. Apparently, DDTL (and other timeline-based editing schemes) cannot cover the whole set of non-deterministic temporal behaviors supported by SMIL2.0. Thus, in addition to the implementation, we will also explore more about the potential of DDTL in supporting SMIL2.0 authoring. To make authoring more efficient, an authoring system needs to provide reuse of SMIL scripts, which involves the conversion of SMIL2.0 scripts to DDTL. Reuse of SMIL scripts in DDTL-based authoring is addressed in another paper [12].

REFERENCES