Paper Code : MM-2
Presenter: Yongbin Ma
Date: 10/23/2003

**Multimedia Meets Computer Graphics in SMIL2.0:**

**A Time Model for the Web**

**Author: Patrick Schmitz**

# Objective

This paper present the model of SMIL 2.0.

SMIL, (pronounced "smile", *Synchronized Multimedia Integration Language*) has the following two design goals:

- Define an XML-based language that allows authors to write interactive multimedia presentations. Using SMIL 2.0, an author can describe the temporal behavior of a multimedia presentation, associate hyperlinks with media objects and describe the layout of the presentation on a screen.
- Allow reusing of SMIL 2.0 syntax and semantics in other XML-based languages, in particular those who need to represent timing and synchronization. For example, SMIL 2.0 components are used for integrating timing into XHTML and into SVG

# Introduction

Time Model: timing and synchronization

1. Video-centric model (Pure Static Scheduling)
2. Graphics/animation-centric model (Pure Event-based)

## Video-centric model

Focus on scheduling the delivery and presentation of continuous (time based) media.

**Advantage:**

1. A linear timeline describes the presentation.
2. It is simple to implement and has little runtime overhead.
3. The presentation model is generally either sampled over time, or pushes out frames at some fixed rate.

## Video-centric model

**Disadvantage:**

1. Do not do a good job of managing hardware sync issues (like audio clocks) and unreliable delivery of media (as on the internet).
2. This model cannot generally handle media of unknown duration.
3. This model generally has limited or no support for interactive content.

**Examples:** early CD-ROM authoring tools, and many video and audio editing runtimes.

## Graphics/animation-centric model

models time as an abstract notion used for purely rendered animations that have no intrinsic rate or duration

**Advantage and Disadvantage:**

- A graph of event bindings describes the presentation.
- There is broad support for dynamic, interactive content, but a lack of scheduling facilities.
- The presentation model is often just a collection of independent timelines, with little or no notion of synchronization between or among elements.

## Cont.

4. The model can be rendered at arbitrary "frame" rates.

5. It can handle media of unknown duration, as well as media with unreliable delivery.

6. It does not generally support synchronization issues associated with hardware (like audio clock issues).

**Examples:** VRML2 and authoring tools for educational software

## Requirements

Integrate traditional synchronization support with time transformations (how times are translated from an abstract representation to simple presentation time)

1. **Timelines: when aspects of a presentation should happen**
2. **Hierarchic or Structured time: break down a large presentation into constituent parts**
3. **Relative timing**
4. **Transformable time**
   **control over the pace of time for an element (including time containers) in the model**

## Cont.

- **Declarative syntax**
  1: important to content authors
  2: important for authoring tool
- **Document processing**
  XML-based syntax is a prerequisite for many document processing models which are being deployed by a growing number of content publishers.
  - **Language integration**
    XML is (again) a requirement.

## SMIL2.0

SMIL 2.0 is defined as a set of markup modules, which define the semantics and an XML syntax for certain areas of SMIL functionality.

- It combines the traditional timing and synchronization tools of hierarchic timing, relative and interactive timing, with time transformation for support of animation.
- As an XML-based language, SMIL 2.0 can be easily used in Internet document processing models.
- The modular structure and language integration semantics facilitate re-use of the SMIL 2.0 timing and animation support in other languages.

## Module

- **Timing and Synchronization**
- **Time Manipulations**
- **Animation**
- **Media Elements**
- **Transitions**
- **Layout**
- Linking
- **Content Control**

## Cont.

1. Hierarchic timing is provided by *time containers*, with local time defined as an extension of the simple cascade.
2. Relative and interactive timing are integrated directly, along with a number of other more advanced tools.
3. The framework defines how time transforms are incorporated into the local time cascade, and a simple set of time transforms is defined for authoring convenience .

### Time Containers and the Local Time Cascade

Support for hierarchic time is provided by three time containment primitives: parallel, sequence and exclusive grouping

**par:** often used simply as a grouping construct for temporal elements.

**seq:** provides sequential activation of child elements (with delays).

**excl:** Provides exclusive activation of child elements.

## Cont.

- Time layer : *simple time*, *segment time*, *active time*.
- Local time cascade: a recursive function that derives a child's active time from the parent time container's simple time. From the local active time, the segment and simple times are derived

### Relative and Interactive Timing Support

- Relative timing is supported both in the implicit sense of hierarchic timing, as well as support for sync-arcs, wall-clock timing, and timing relative to a particular repeat iteration of another element.
- Interactive timing is supported via event-based timing, DOM activation and hyperlink activation

## Time Transform Support

Four simple abstractions for time transformation to control the pace of element simple time.

1. Speed,
2. Accelerate
3. Decelerate
4. AutoReverse

## Cont.

- **speed**

  Modifies the pace of local time relative to parent simple time. The value is a multiple of the rate of parent time, with negative values used to reverse the normal pace of time.

## Cont.

- **accelerate and decelerate**

  Define a simple acceleration and deceleration of element time, within the simple duration. The values are expressed as a proportion of the simple duration (i.e., between 0 and 1), and are defined such that the length of the simple duration is not changed by the use of these attributes. An illustration of the progress of time with different accelerate and decelerate values is provided in Figure below.

## Cont.

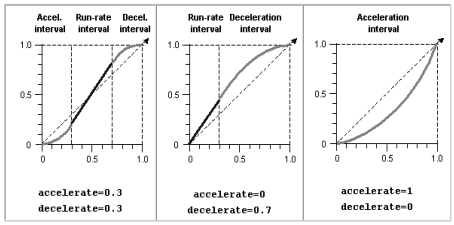


Figure: Effect of acceleration and deceleration upon progress, as a function of time. The x-axis is input time (as a proportion of the simple duration), and the y-axis is the progress/transformed time.

## Cont.

- **autoReverse**

  This causes the simple duration to be played once forward, and then once backward. It causes the segment duration to be twice the simple duration, but this side effect is sensible and easy for authors to understand.

## Example

```
<html>
<head>
<style>
   <!-- Bind the IE timing behavior to the needed elements -->
   t\:*, p, span { behavior:url(#default#time2); }
  .highlight { background-color:yellow; font-weight:bold }
</style>
<XML:NAMESPACE PREFIX="t"/>
</head>
<body>
<!-- We wrap everything in a "par" so that it all starts together 1 second
     after the page is loaded. -->
<t:par begin="1s">
  <t:audio src="media/talk.wma" syncBehavior="locked" />
  <!-- We set a timeAction to apply a highlight to the spans.
       The p becomes a sequence time container for the highlights,
       and the spans just need durations -->
  <p timeContainer="seq" timeAction="none" >
    <span dur="2.9s" timeAction="class:highlight" >
       SMIL Timing syntax consists of a set of </span><br>
    <span dur="3s" timeAction="class:highlight" >
        <em>attributes</em> for controlling the behavior of media, </span><br>
    <span dur="2s" timeAction="class:highlight" >
        and several types of <em>time containers</em> </span><br>
    <span dur="4s" timeAction="class:highlight" >
        that group media together for coordinated presentation. </span>
  </p>
</t:par>
</body>
</html>
```

## CONCLUSION

SMIL 2.0 defines powerful, flexible syntax and semantics for timing and synchronization of media. It combines support for traditional media scheduling and interactive timing to provide a state of the art timing model. The model scales well from basic presentations with simple syntax, to complex and finely tuned multimedia experiences that integrate a range of media, animation, and user interaction. Unlike other models, SMIL 2.0 is designed specifically for web delivery, and allows authors to manage the unreliable nature of the Internet, without sacrificing a structured scheduling model.

## Critical on the paper

• SMIL will provide authors with the tools they need to author web pages with rich media, animation and interaction. However, it is more subjective.
• SMIL has much more complex *Tags* than HTML
• SMIL does not concentrate on describing the actual media.

## Reference


•http://www.w3.org/TR/smil20/smil-timing.html#Timing-ParSyntax
•ftp://ftp.research.microsoft.com/pub/tr/tr-2001-01.doc
•http://www.ludicrum.org/plsWork/papers/UnifyingNote.html

# Quiz Question

1. What is SMIL ?
2. What's the requirements for integrated time model?
3. How is hierarchic timing supported in SMIL 2.0?
4. Explain 4 simple abstractions for time transform in SMIL 2.0.
5. What's fallback semantic?