Unifying Scheduled Time Models with Interactive Event-based Timing

Notes on Indeterminate Timing

Patrick Schmitz

November 29, 2000

Technical Report
MSR-TR-2000-114

Microsoft Research
Microsoft Corporation
One Microsoft Way
Redmond, WA 98052

Abstract

This note describes a model for unifying event-based indeterminate timing (also known as atemporal composition) and declarative, determinate timing. Background rationale is presented, and a mechanism is described for processing events in the timing model and scheduler. The specifics of integrating this model into the SMIL timing model are described.

This paper was originally published June 6, 1999 as a W3C (SYMM working group) internal note.

Table of Contents

Introduction
Definition of Terms
Background and Rationale
SMIL 1.0 Approach
Unifying Schedulers and Generic Event-based Timing
Applying the unified model to SMIL

Introduction

The first W3C Working Group on Synchronized Multimedia (SYMM) developed SMIL - Synchronized Multimedia Integration Language. This XML-based language is used to express synchronization relationships among media elements. SMIL 1.0 documents describe multimedia presentations that can be played in SMIL-conformant viewers.

As part of the current SYMM Activity, the Working Group is extending the SMIL Timing and Synchronization support, and generalizing the support provided in SMIL 1.0. Additional capabilities will be added to the timing model, as well as support for integration with HTML and XML languages. See also "Synchronized Multimedia Modules based upon SMIL 1.0".

Among the areas of interest is support for interactive timing, also described as atemporal composition. This document describes a model for unifying traditional scheduled time models and interactive event-based models for multimedia.

Definition of Terms

I present these terms for the sake of this discussion. These definitions may not necessarily apply to the more general context of multimedia - their significance here is only for this limited context.

Interactive content

> This is content that intrinsically supports interaction as part of the presentation. This generally means that while there may be a story line, there are many aspects of the presentation that change or are performed according to the user input (or other interaction sources). I distinguish this from linear content in an interactive medium. Static documents (i.e. the *document* is static, not the *media*) on the web are sometimes called interactive in the sense that users can navigate within and among them, by virtue of the web medium. However, the documents themselves are not really interactive, in that they are not presented or performed differently (and dynamically) in response to user input. Incidental interaction like scrolling and resizing the application display window do not count. Hyperlinking into or

out of an document does not count. Hyperlinking within a document only counts if it *activates* interactive content within the presentation.

Rendered or synthetic media

I use this to describe media that have a very abstract description, and that are rendered by "drawing" or "painting" more than by "decoding". Media of this sort are nearly always non-linear, and tend to be CPU bound rather than I/O bound.

Time-based or "played" media

I use this to describe media like audio and video that have a close association of the media and the presentation, and that are generally "decoded" and "played" over time. Media of this sort tend to be linear and I/O bound more than CPU bound.

Runtime engine

This is my general term for the application code that manages scheduling and playback of a multimedia presentation. It generally includes a scheduler of some sort, a display and audio manager, and some sort of object model or time- graph representation. It often includes a means of parsing a description of the presentation, but this is not a requirement.

Intrinsic duration

This is a concept associated with time-based media, and timeline models. For media, it is the duration of the media as defined by the underlying representation (usually a file), and is independent of any authored markup or external description. For timelines, it is the duration that would be computed from all children, ignoring any specific markup or description for duration on the timeline element itself. This is somewhat dependent upon the semantics of the timeline model, but it is generally understood to be the maximum of the extents (i.e. begin offset plus duration times repeat) of all children (SMIL 1.0 supports an explicit duration related to this with the `endsync` attribute). For file-based media, the intrinsic duration is often finite. For synthetic media and for timelines, this can be indefinite or infinite (depending upon the semantics of the model).

Scheduled or Determinate timing

This includes all time descriptions that have a specific known value relative to the global presentation timeline. This term applies to both the authoring model as well as the runtime engine time-graph. A presentation that has only scheduled, determinate timing relationships is purely linear, and has a single presentation form (i.e. it is not dynamic or interactive). In terms of the SMIL 1.0 terminology, there should be no distinction between the "desired" and the "effective" times (in the time-graph - this ignores runtime and media delivery imperfections).

Event-based or Indeterminate timing

This includes all time descriptions that do not have a specific known value relative to the global presentation timeline. This term applies to both the authoring model as well as the runtime engine time-graph. A presentation based purely upon indeterminate timing has no defined synchronization relationships, and has an infinite number of possible presentation forms.

Time dependents

A time that is defined relative to another time is described as a time-dependent of the other time. Thus in a sequence timeline, the begin time for one element is a

time-dependent of the end-time of the previous element. More generally, to determine the global presentation time for any element, the sync offset for each ancestor time container must be accounted for, and so all nodes are at least indirectly dependent upon the chain of ancestor timeline containers. The notion of time-dependents at the implementation level allows for efficient updating of the time-graph when a time changes (e.g. from an indeterminate value to some determinate value).

Background and Rationale

The model I present came about as an evolution of experience with a variety of multimedia runtimes and schedulers. Many of these were supported in popular multimedia authoring tools, video and audio editing tools and some platform APIs for multimedia. In general, authoring tools tend to present an authoring model that is closely aligned to the actual implementation model of the associated runtime engine. While this is not a requirement, a unified model for scheduled and event-based timing will benefit both the runtime implementation as well as the authoring model.

For a long time, multimedia runtimes supported one of two basic models:

1. **Pure static scheduling**. A linear timeline describes the presentation. Variations are sometimes supported by "jumping" to another section of the timeline that had an alternate view. This model lends itself well to linear storytelling, using synthetic and other non-linear media (like vector graphics, text and still images). It is simple to implement and has little runtime overhead. The presentation model is generally either sampled over time, or pushes out frames at some fixed rate. This model can incorporate time-based (linear) media, but does not do a good job of managing hardware sync issues (like audio clocks) and unreliable delivery of media (as on the internet). This model cannot generally handle media of unknown duration.
   This model generally has limited or no support for interactive content.
   Examples of this general model include early CD-ROM authoring tools, and many video and audio editing runtimes.
2. **Pure event-based**. A graph of event bindings describes the presentation. There is broad support for dynamic, interactive content, but a lack of scheduling facilities. This model lends itself well to interactive models and experiential content (as opposed to storytelling). It is easy to generalize a model like this to include user interaction as well as other sources of interactive input (like broadcast or streamed events).
   The presentation model is often just a collection of independent timelines, with little or no notion of synchronization between or among elements.
   The model can handle media of unknown duration, as well as media with unreliable delivery. It does not generally support synchronization issues associated with hardware (like audio clock issues).
   Simple implementations often suffer from event propagation delay, and it can be very difficult to maintain sync as described in a document - the longer it plays, the

greater the accumulated propagation delays and the greater the slew of the elements from the described sync relationships.
Some implementations do support a master clock and some simple sync mechanisms. Better implementations allow for the event propagation delay by marking events with the virtual event time, and making event registrants respect the virtual event time rather than the observed event delivery time.
Examples of this general model include VRML2 and authoring tools for educational software.

SMIL 1.0 Approach

SMIL 1.0 presents something of a hybrid of these models, but with some constraints. The issue of performance QOS is left to the implementation, and is described as either "hard" or "soft" synchronization. In the "hard" sync model, most of the SMIL time model is scheduled and determinate. In the "soft" sync model, it seems to be undefined.

One significant point in the SMIL 1.0 model however is the ability to handle indeterminate durations for some media elements. Some media has a finite duration, but this duration is not known until the presentation of the associated media element is complete (or at least the data has all been downloaded). Other elements that are defined relative to such an element duration (e.g. successive elements in a sequence timeline) also have indeterminate timing. The timing and sync relationships for these elements is resolved when the original indeterminate time is resolved (e.g. when the movie is fully downloaded and the duration becomes known).

Note that it is possible to construct a SMIL 1.0 runtime that essentially hands off the scheduling issues to a media server. In this case, the client runtime does not manage the inter-media sync relationships, but simply plays media as delivered by the server. The server *could* preclude all indeterminate timing by gathering duration information for media as the stream scheduler prepares a presentation. Nevertheless, in many runtimes (and even in some streaming servers), it will be a requirement that indeterminate timing be supported.

SMIL 1.0 syntax includes a means of defining a timing relationship to the begin or end of another element. The specification refers to this as event-based timing. However, there is no specified requirement on the implementation to actually use events. The semantics of the timing determine that when the "effective" time does not match the "desired" time, that the time model is based upon the "effective" time. However for many cases, this is really an issue of the QOS for synchronization in the presentation.

For any time-graph in which all durations are determinate (e.g. all durations are explicit), "hard" sync runtimes should not allow any variance between "desired" and "effective" times. In these cases, there is no need for an event based model, and a pure scheduled runtime will suffice. Only in the cases where there is a specifically indeterminate time should the need for an event system arise. Therefore, in the case of determinate timing, the semantics of event-based timing and simple declared timing relationships (e.g.

children of a <par> element) are *equivalent*. The only significant distinction between the simple timing relationships and event-based timing arises when indeterminate times are involved.

As an example, consider the following two descriptions in SMIL 1.0:

Sample 1)

```
<seq>
   <media id="m1.1" src="..." />
   <media id="m1.2" src="..." />
   <media id="m1.3" src="..." />
</seq>
```

Sample 2)

```
<par>
   <media id="m2.1" src="..." />
   <media id="m2.2" src="..." begin="id(m2.1)(end)" />
   <media id="m2.3" src="..." begin="id(m2.2)(end)" />
</par>
```

There should be no difference in the semantics of these two constructs. In practice, the runtime engine may have to use an event system for both cases, if the durations of the media elements cannot be determined when the time-graph is built. At the same time, if the durations of the elements *can* be determined, a runtime has no particular need to involve events at all. In this sense, the use of the term "event-based timing" can be seen as a means of describing the semantics, rather than any requirement on the implementation. Few authoring models would expose these kinds of timing relationships as *events*, in the same way that for example a mouse click event would be presented.

Nevertheless, the SMIL 1.0 syntax does not support timing descriptions relative to events that come from outside the timing model - in particular user interaction events and events associated with time-based media (e.g. events streamed with video). Given that the model already requires runtime support for indeterminate times, it should be a relatively simple extension to support generalized event-based timing. The key will be to incorporate user-interaction events in a manner that extends the current support for indeterminate timing.

Unifying Schedulers and Generic Event-based Timing

More recently, some hybrid models have been developed that combine scheduling support and event-based declaration. In these models, there is some form of scheduled time graph that describes the presentation and the synchronization relationships among the media elements, but there is also support for event binding mechanisms. The challenge in these models is to unify the two models in a manner that is easy to author, flexible across a broad range of content and use-cases, and relatively simple to implement.

The approach is based upon a scheduled model, with extensions to support indeterminate timing in the general case. This builds upon a known model, and does not require significant changes to the runtime model. This section describes the changes to the runtime model, and the following section describes how this model is represented in an authoring model like SMIL.

To support interactive content, the scheduled model must be more flexible in a number of ways:

Separate timing model and time values

Start and End times for media elements are described by the model, and then computed and cached by the runtime engine. Representing the scheduled times makes it possible to implement a synchronization manager that can optimize the preparation of media, and ensure that the performance is closely tracking the sync relationships as described by the author. For simple scheduled elements, there is no event propagation delay or other overhead associated with an event-based scheme. At the same time, the model separates the description from the runtime values. This makes it possible to dynamically change values, and then propagate the effects of a change by recomputing all the dependent values.

Support special value: *indeterminate*

Any cached time can have the special value *indeterminate*. This means that while there is a description for how the value is computed, the actual value is not currently known. For practical purposes, this can be thought of as equivalent to setting the value to *infinite*, which places the associated element at the theoretical end of the presentation timeline. There are several cases that lead to indeterminate times:

1. The time is based upon media information that has not yet been delivered. A common case of this is the intrinsic end of an MPEG encoded movie. Until the movie has been fully downloaded, the intrinsic duration is not known.
2. The time is indirectly computed from another indeterminate time. A common case of this is the scheduled begin time of an element that is defined to begin after (i.e. relative to) the end of an MPEG movie, as described in the case above. Another case is the intrinsic end of a container timeline with MPEG movie children.
3. The time is related to an event that is not (or cannot be) scheduled. Examples of this include times defined relative to user interaction events. While often modeled as a separate mechanism, this form of timing has the same behavior as the other cases.

Defer *indeterminate* scheduling

Indeterminate times in the model are handled by *deferring* any scheduling activity for the associated element. In most simple models, the synchronization point is defined at the

beginning of an element timeline. As such, an indeterminate end-time does not preclude the playback of the element in the presentation. However and indeterminate begin-time also implies an indeterminate synchronization relationship. Thus for an element with an indeterminate begin time, the scheduler and synchronization engine defer action for the media, and do not attempt to incorporate the element into the running presentation. Another way of thinking about this is that the associated element (which may be an entire timeline or subgraph of the presentation) is disconnected from the running presentation graph. I often say that it is floating above the running timeline (or more precisely above the *parent* timeline), in context but not attached.

Handle *indeterminate* to *determinate* transition

An indeterminate time can at some point in the presentation become determinate. When this happens, all dependent times are re-evaluated, and the scheduler incorporates all newly determinate synchronization relationships into the running model.
Considering the cases from above:

1. Media information is delivered and an associated time changes from indeterminate to determinate. In the common case of MPEG movies, as soon as the media finishes downloading, this information is available, and can be incorporated into the scheduler. The presentation engine now knows when the movie will end, and can manage it just as though it were a statically defined duration. This means that the scheduler can model media in a generic manner, independent of whether the durations is determinate or indeterminate. This can simplify implementations.
2. A time that was indeterminate becomes determinate and propagates to dependent times. For example, when the duration of an MPEG movie becomes determinate, the begin time of an element defined to follow the movie will also become determinate. At this point, the sync relationship is defined, and so the element (again, possibly an entire timeline or subgraph) can be attached to the running timeline and incorporated into the scheduler.
   Note that this transition within the model is often in advance of the observed end of the associated playout of the movie, as data is delivered in advance of the playout. By incorporating the information into the schedule graph when the data is delivered, and propagating the change to dependent times, it becomes possible to schedule the cueing and other preparation of media defined to follow something like an MPEG movie. We get the benefits of a scheduler model but the flexibility of an event model.
3. A user input event (or equivalent interactive event) is received. When the event is handled at the low level in the runtime, the current document presentation time is associated with the event. The event is then dispatched to all registrants (i.e. to all elements in the time graph that are defined relative to the event). As the dispatch proceeds, the time associated with the event is translated to the local timeline of each timeline (container) node it passes through. When the event reaches the node that is defined relative to the event, the associated node time is computed from the (translated) event time (i.e. the node time is set to the event time plus or minus

any specified delta). At this point, the determinate time is propagated just as for the cases above.
Note that if any ancestor timeline node is not currently active, the event is blocked. An element defined relative to an event is not sensitive to the event unless the parent timeline is active when the event happens.

Variant: supporting timestamped events

An additional variant can take advantage of the mechanism described. Broadcast and streaming media can often deliver events as part of, or associated with the other media. These events generally are fired immediately, in the same model as an event firing when a user clicks with the mouse. However, these events can also have an associated event-time. The events can be delivered in advance of the event-time. When a timed event is delivered, the scheduler can propagate the event to timing registrants just as though the event had fired. The event is marked with the scheduled time, rather than the current presentation time, but is otherwise handled by the standard mechanism. Any times defined relative to the event are resolved to determinate times, and the scheduler can then optimize the performance by cueing media, etc. This increases the fidelity and quality of the performance for applications like IP-based enhancement of television, HTML and XML enhancement documents associated with streaming media, etc.

Relationship to timeline container semantics

The sync relationship, begin and end times are always subject to the semantics of the parent timeline. As such, any local timeline that has indeterminate sync will still be cropped by the begin and end of the parent. Other mechanisms can force the presentation timeline to reset if so desired, but this is orthogonal to the basic model of indeterminate times and synchronization relationships. E.g. the SMIL hyperlinking model specifies that when linking to an element within a document, the document timeline is advanced to the presentation time associated with that element. If that element had an indeterminate begin time, an implementation can seek the presentation timeline to a point on the parent timeline and then resolve the indeterminate time to be determinate (e.g. by emulating a user-click). *How would SMIL handle a link to an element defined to follow an MPEG movie in a sequence? Since the begin-time for the link destination cannot be determined other than by fully loading the MPEG movie, what should an implementation do?*

Applying the unified model to SMIL

To have useful application to the current SYMM Activity, we must be able to apply this model to the SYMM timing model, originally described in SMIL 1.0. This is my understanding of the requirements for such an integration:

1. The model must integrate SYMM scheduled timing and event-based timing
2. The model should support events in the most generic sense (e.g. it should not be constrained to user-interface events).

3. The integration should not require significant changes to the existing timing semantics
4. The model should be aligned to the proposed DOM Level 2 Events specification

Based upon these requirements, the integration is based upon the following precepts:

Events in general are an extension of the SMIL model for timing events

SMIL 1.0 has already introduced the notion of time-model events (i.e. begin and end events) and a syntax for defining a time relative to these events. As discussed above, the notion of indeterminate timing was also introduced in SMIL 1.0. Although the event model described herein extends these concepts, there is no reason not to leverage the SMIL 1.0 syntax and definitions.

Events are an activation mechanism

Events are used in the model only to describe a means of activating a given element time. There are no additional semantics associated with the event timing model. In particular, event timing does not carry the semantics defined in SMIL for timed hyperlinking, in which the link activation can seek the presentation timeline.

This principle simplifies the event semantics, and reduces the impact of event timing on existing semantic constructs. In particular, event-timed elements are still subject to all the semantic constraints defined by timeline containers. These include:

- **`<par>` containers:** children with event-based begin and/or end times are still constrained to begin no earlier than the `par` container begins, and to end no later than the `par` container ends. Activation of an event-based begin or end time does not directly affect the `par` container (other than via the definition of the `endsync` specifier, which remains unchanged from SMIL 1.0). See also the event routing discussion.
- **`<seq>` containers:** children of a `seq` are constrained to begin no earlier than preceding sibling element ends. Due to the indeterminate nature of event timing, it must be *illegal* to specify an event-timed begin for a child of a `seq`. A similar constraint is defined in SMIL 1.0. It is legal to specify an event-based end time for a child of a `seq`. Just as in SMIL 1.0, the effective end of such a child will be the earlier of the event time, and an explicit end of the parent `seq` container. Activation of an event-based begin or end time does not directly affect the `seq` container (other than the contribution to the implicit end of the `seq`, which remains unchanged from SMIL 1.0).
- **`<choice>` or `<excl>` containers (proposed for SYMM 2.0):** These new timeline constructs support the semantic that only one child of the container may be active at one time. If a child is activated by any means, any sibling that is already active is made inactive. Children of a `choice` are subject to the same constraints as children of a `par` container. If a child element with an event-timed

begin receives the associated event, the child begins and the container manages the specific choice/exclusive semantics.

Elements that define the a begin or end time relative to an event outside the timing model (e.g. user events, stream events) are said to have an explicitly indeterminate time. If the time is a begin time, the element has an explicitly indeterminate sync relationship. If the time dependency chain for an element time traces through any time that is explicitly indeterminate, then the element time is also explicitly indeterminate. It may be desirable to exclude elements with explicitly indeterminate sync relationships from certain calculations, such as the implicit duration of a par container. Conversely, the current SMIL model for the definition of implicit end could be applied, and the implicit begin could be defined to be 0 (zero) on the parent timeline. This needs further discussion

Event Restart

If an event-timed element is active (i.e. it is playing), and it receives a (second) begin-event, two behaviors are possible, and are resolved with the use of an attribute: "eventRestart". If "eventRestart=true", then the element will restart. This is useful for running graphic animation in response to a button click. If "eventRestart=false", then the additional begin events are ignored. This is useful to prevent restart of things like audio, while they are playing.

Once an element completes (including any repeat behavior), it can be restarted by another begin event, subject to the normal parent-timing constraints.

Resolving Event times

If an element with an indeterminate begin-time is activated (e.g. it received a begin event like a click), the begin time is resolved to a determinate begin time that matches the event-time (runtimes could use the current time in place of the event-time, but will slew as events propagate through the system). This determinate begin time is used to calculate the end time, and is propagated to all time dependents (e.g. to other elements specified relative to this one). This is described in somewhat more detail above.

Events and Repeat

For the case that an element is in a parent (or ancestor) timeline that repeats: for each iteration of the parent or ancestor, the element is played as though it were the first time the parent timeline was playing. This may require a reset of some sort in the implementation to ensure that the media is restarted, and that any event sensitivity is reset. Any state associated with eventRestart is reset. Any indeterminate begin-time (or end-time) that was resolved to a determinate time during the just completed repeat iteration, is reset to be indeterminate again.

Event Routing

According to the DOM event model, events are dispatched from the DOM root to the target node, following the DOM parent hierarchy. In general events may be captured during the dispatch phase, and can be cancelled, precluding further processing. Events also bubble up from the target node, again according to the DOM containment hierarchy.

The current discussion for a SYMM DOM includes the idea that the DOM will represent containment (i.e. parenting) based upon the timeline containers (e.g. par and seq). This model would support the semantics for event dispatch to event-timed elements in the time graph. In particular, an timeline container element that is not active can capture events targeted at any descendent, and cancel dispatch to any descendents. This enforces the semantic that an element cannot begin or end beyond the bounds of the parent timeline extent. It also makes explicit the semantic that an event-timed element is not sensitive to events if the parent timeline container is not active.

Another related semantic specifies that if an element is not currently active, and it receives an event included as part of an end-time specification, the event is ignored by the element.

This model is easy to implement based upon the DOM event model, and has minimum impact upon the existing time-graph semantics.

Event Sets

Many common authoring use-cases associate a given interaction result with several different events. In some cases, the set of events include different event types targeted at a given node (e.g. onClick and onDoubleClick). In other cases, the set of events includes a given event type targeted at one of a number of elements (e.g. button1.onclick, button2.onclick and button3.onclick). With event bubbling, it is possible to implicitly associate to an event on a number of elements, by specifying the focus as a parent (e.g. "parent.onClick" will actually associate to all click events on the descendents of "parent", unless event bubbling is cancelled).

It is a minor extension to support the explicit association of an element time to more than one event, by specifying a set of events. A possible syntax is a comma separated list of qualified event names.

This functionality is also described in the note: "Event Handling in SMIL and XHTML" [SYMM-EVENTS].

References

[DOM2EVENTS]

"Document Object Model Events", Tom Pixley.
Available at: http://www.w3.org/TR/DOM-Level-2-Events/.

[SMIL]
"Synchronized Multimedia Integration Language (SMIL) 1.0 Specification W3C Recommendation 15-June-1998 ".
Available at: http://www.w3.org/TR/REC-smil.

[SYMM-EVENTS]
"Event Handling in SMIL and XHTML", Ted Wugofski.
Available (members only) at:
http://www.w3.org/AudioVideo/Group/Timing/symm-events.htm.

[SYMM-MOD]
"Synchronized Multimedia Modules based upon SMIL 1.0", Patrick Schmitz, Ted Wugofski, Warner ten Kate.
Available at: http://www.w3.org/TR/NOTE-SYMM-modules.

[SMIL-TIMING]
"The SMIL 2.0 Timing and Synchronization Module", Patrick Schmitz, Jeff Ayars, Bridie Saccocio, Muriel Jourdan.
Available at: http://www.w3.org/TR/smil20/smil-timing.html.