Simbody 4.0 Event Architecture

Rev. 0.2

A Simbody System defines a set of Event objects representing “things that can happen” during a simulation. For example, initialization, termination, time advanced, and impacts are Events. Each Event is identified with a unique small integer EventId, assigned when the Event is added to the System. An Event has a list of zero or more EventAction objects that are invoked upon occurrence of the Event. An EventAction can be

* a **Report**, in which case no state change occurs, or
* a **Change**, in which the state may be changed in response to the Event, altering the subsequent trajectory of the simulation.

A Change Action always interrupts the time stepper so that a step has completed just prior to the action being taken. The Change then alters the state and the time stepper resumes with the modified state. A Report, on the other hand, normally does not interrupt the time stepper but instead can report an interpolated state. Optionally, a report can also require that the time stepper be interrupted so that the reported state is actually part of the integrated trajectory. This can be used to force isolation of interesting events, such as some quantity attaining a minimum or maximum value, without taking any other action.

Event occurrences are triggered by various time- and state-dependent conditions that occur during simulation. Detection of these conditions is encapsulated by EventTrigger objects. There are three general kinds of Triggers: Witnesses, Timers, and Flags, and several built-in Triggers: Initialization, TimeAdvanced, Termination, ExtremeValueIsolated. Each EventTrigger is associated with a particular Event (or Events). When a trigger is seen, the corresponding Event(s) are deemed to have occurred. Triggers interact with the time stepper’s step size algorithm to efficiently localize events. Triggers are evaluated during realization of a State.

Trigger

**Event**

Action

It is possible (and common) for several Triggers to cause the same Event. Many different contact triggers, for example, will cause the same Impact Event, since all contacts must be treated simultaneously during an Impact. Also, it is possible for one Trigger to cause multiple Events. For example, a periodic timer Trigger may be used to cause a set of Events that happen to use the same period. And it is also possible (and common) for multiple triggers to occur simultaneously, or at least indistinguishably. As a result, several Events may occur simultaneously. The resulting EventActions are sequentialized, but with no guarantee about the ordering of Events except:

* If there are both Change and Report Actions to be performed, a Change pass through all Events is made first, followed by a Report pass reporting the changed state; and
* During a pass, all Actions for one Event are performed before any Actions for the next Event; and
* The Event processing order is the same for both passes.

Trigger

Trigger

Trigger

**Event1**

Action

Action

Action

Trigger

**Event2**

Action

**Simultaneous Triggers**

**Simultaneous Events (sequentialized)**

## Witness Triggers

Witness Triggers provide a scalar method that calculate the witness’s value and up to two time derivatives if known. The “depends-on” stage is known for each of these (similar to Measures). The derivatives can be useful for prediction and localization. The sign transition that triggers the event is specified; it can be positive to negative, negative to positive, or both.

Localization requirements are given for both time and value; these are then scaled by the accuracy currently in effect by the solver. Absolute (accuracy-independent) localization requirements can optionally be provided, and localization can be specified to require *both* time and value to be localized, or either one alone.

Triggers are maintained by the SystemGlobalSubsystem and are realized *after* all other subsystems have been realized. Thus they may depend on any quantities calculated at their depends-on stage, from any subsystem.

## Timer Triggers

Like Witness Triggers, these provide a scalar method that takes a State (and time) as input. The result is the next time at which this timer’s events should occur. A PeriodicTimer is available that constructs an appropriate timer method. When attached to an Event with Change Action, these are used to limit the length of the upcoming step. When only Report Actions are indicated, Timers are used to generate interpolated states for reporting.

## Flag Triggers

TBD – are these really needed?

## Run time changes

Events and their Actions are part of the System topology and are added during extended construction of the System (that is, prior to calling realizeTopology()). Triggers, on the other hand, may also be added or removed from any writable State. Adding a new one invalidates just the Trigger value cache for the stage associated with the Trigger function; deleting does not require invalidation. There is also a discrete state variable holding the *previous* value (and derivatives if known) for each Trigger; that is extended to hold the new Trigger’s values and initialized to NaN to indicate there is no value yet known.

## Convenience classes

We retain the EventHandler and EventReporter classes from earlier Simbody releases for convenience but redefine them in the above terms. When added to a System, each creates a unique Event and a single Action for that event (a Change or Report Action, respectively). The Scheduled subclasses create Timer triggers, while the Triggered subclasses create Witness triggers.