TIMER BEHAVIOR SCRIPT INTERFACE

View online: https://www.construct.net/en/make-games/manuals/construct-3/scripting/scripting-reference/behavior-interfaces/timer

The `ITimerBehaviorInstance` interface derives from IBehaviorInstance to add APIs specific to the Timer behavior.

Example

Below is a sample code snippet demonstrating setting two timers on `inst` (assumed to be an instance with the Timer behavior). One is a regular timer and the other is a one-off timer.

Note that many browsers combine identical log messages in the console, so repeat messages may just increment a number by the message rather than showing it again.

```
// Handle "timer" event and log to the console that the timer ran.
inst.behaviors.Timer.addEventListener("timer", e =>
{
        console.log(`Timer "${e.tag}" ran!`);
});

// Set a regular timer running every 1 second
inst.behaviors.Timer.startTimer(1, "myRegularTimer", "regular");

// Set a one-off timer to run after 2.5 seconds
inst.behaviors.Timer.startTimer(2.5, "myOneOffTimer", "once");
```

Timer behavior events

See behavior instance event for standard behavior instance event object properties.

"timer"

Fired when a timer period has elapsed. The event object has a `tag` property which is a string of the tag for the timer that has elapsed.

Timer behavior APIs

startTimer(duration, tag, type = "once")

Set a new timer, or if a timer with the same `tag` exists, re-start it with new options. `duration` is the time in seconds until the `"timer"` event fires. If `type` is `"once"`, then the timer event will fire once and not again until `startTimer` is called again. If `type` is set to `"regular"`, then the timer event will keep firing every `duration` seconds. The `tag` is a string that allows identifying different timers.

---

setTimerPaused(tag, isPaused)

Set a currently running timer either paused or resumed. When a timer is paused, it will stop firing timer events. When resumed, it will continue firing timer events, resuming from the time that it was paused at. In other words if a timer is set for 1 second, it is paused after 0.5 seconds, and then after some time it is resumed again, the next timer event will fire 0.5 seconds after resuming.

---

setAllTimersPaused(isPaused)

This does the same thing as the `setTimerPaused()` method, but affecting all existing timers rather than only one with a given tag.

---

stopTimer(tag)

Stop a timer with a specific tag. The timer event will no longer fire for the stopped timer after this call.

---

stopAllTimers()

Stop all currently running timers regardless of their tags. The timer event will no longer fire for any timer after this method unless a new timer is started.

---

isTimerRunning(tag)

Returns a boolean indicating if a timer with the given tag has been started. Once stopped, the timer no longer counts as running. Paused timers also count as running - use `isTimerPaused()` to identify these timers separately.

---

isTimerPaused(tag)

Returns a boolean indicating if a timer with the given tag has been started and then subsequently paused with `setTimerPaused()`.

---

getCurrentTime(tag)

Returns the time in seconds since the `"timer"` event last fired, for a timer with a specific tag.

getTotalTime(tag)

Returns the time in seconds since a timer with a specific tag was started. This is only useful with regular timers, since it will always equal `getCurrentTime()` for one-off timers (after which they fire and the timer no longer exists, so these expressions return 0).

getDuration(tag)

Returns the duration in seconds for a timer with a specific tag.

getNormalizedProgress(tag)

Returns the current progress for a timer with a specific tag represented as a number between 0 and 1, regardless of the duration of the timer. For example a normalized progress of 0.5 means half way through the duration.

hasFinished(tag)

Returns a boolean that is true for the tick that the `"timer"` event fires in. This is the way the *On timer* condition of the Timer behavior checks for finished timers, and this method allows script callers to make the same check, such as by polling the timer in the `"tick"` event.