

TIMER BEHAVIOR

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

The **Timer behavior** triggers its *On timer* condition regularly or once after a delay. This is like using the system *Every X seconds* condition, or the system *Wait* action, except that times are kept for each instance individually. The rate of *On timer* triggering is affected by the time scale. The timer behavior is a more convenient alternative to adding *dt* to an instance variable every tick.

Scripting

When using JavaScript or TypeScript coding, the features of this behavior can be accessed via the `ITimerBehaviorInstance` script interface.

Tags

A single Timer behavior can keep track of multiple timers. To distinguish between them, a **tag** is used. A tag is just a string, which can be anything. For example, starting a timer with tag "attack" will trigger *On timer "attack"*, but not *On timer "defend"*.

When a timer is stopped, or after a one-off timer triggers, it no longer exists and the timer expressions cannot be used to retrieve any information about it.

Timer conditions

Is timer paused

True if a timer with the given tag has been started and then subsequently paused with the *Pause/resume timer* action.

Is timer running

True if a timer with the given tag has been started with the *Start timer* action. Once *Stop timer* is used, the timer no longer counts as running. Paused timers also count as running - use the *Is timer paused* condition to identify these timers separately.

On timer

Triggers either regularly, or once off, after a timer that was started with the same tag has reached its duration.

Note: this trigger can fire with multiple instances picked, if their timers all reach their time in the same tick. This can sometimes work unexpectedly if the actions expect there

to be just one instance picked. The workaround is to add a *For each condition after this trigger* to ensure the actions run once per instance.

Timer actions

Pause/resume timer

Set a currently running timer (started with the *Start timer* action) either paused or resumed. When a timer is paused, it will stop triggering *On timer*. When resumed, it will continue triggering *On timer*, 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, *On timer* will trigger 0.5 seconds after resuming.

Pause/resume all timers

This does the same thing as the *Pause/resume timer* action, but affecting all existing timers rather than only one with a given tag.

Start timer

Set a new timer, or if the timer exists, re-start it with new options. *Duration* is the time until *On timer* triggers. If *Type* is *Once*, then *On timer* will fire once and not again until *Start timer* is used again; if *Regular*, then *On timer* will keep firing every *Duration* seconds. The tag allows multiple timers to run at once. The corresponding *On timer* condition must use the same tag.

Stop timer

Stop a timer with a specific tag. *On timer* will no longer trigger with the given tag after this action.

Stop all timers

Stop all currently running timers regardless of their tags. *On timer* will no longer trigger for any timer after this action unless a new timer is started.

Timer expressions

CurrentTime(tag)

The time in seconds since *On timer* last triggered, for a timer with a specific tag.

Duration(tag)

The duration in seconds for a timer with a specific tag.

NormalizedProgress(tag)

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.

TotalTime(tag)

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