

# FOLLOW BEHAVIOR

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

---

The **Follow behavior** allows an object to follow another object on a time or distance delay. More generally, it records a short history of an object's changes, allowing it to also perform tasks like record and replay, or rewinding time.

Only adding the Follow behavior to an object does not do anything. You must use an action like *Follow object* before it starts changing the object the behavior has been added to.

You can find a number of examples using the Follow behavior in Construct's Example Browser, such as the [Follow behavior example](#) and [Follow record/replay](#).

## Scripting

When using JavaScript or TypeScript coding, the features of this behavior can be accessed via the [IFollowBehaviorInstance script interface](#).

## Follow properties

---

### Mode

The following mode to use. *Time* mode follows on a time delay. *Distance* mode follows with a distance delay. In *Distance* mode the X and Y properties are always followed, as it is necessary for determining the distance.

---

### Delay

The delay on which to follow the tracked object. In *Time* mode this is the delay in seconds to follow. In *Distance* mode this is the distance in pixels at which to follow. The delay cannot exceed the max delay.

---

### Max delay

Determines the amount of data remembered in the same units as *Delay*. For example in time mode, if the delay is 1 second but the max delay is 3 seconds, then the behavior is remembering 3 seconds of history but following at a 1 second delay. The delay can be increased up to but not past the max delay. Usually the max delay can be the same as the delay, as additional data does not need to be remembered, but having a higher max delay can be useful if the delay may be increased. Note that the higher the max delay the more data is remembered and so the more memory will be used, so it's best to use the shortest max delay possible.

## History rate

The rate in entries per second at which data about the object being followed is saved. For example if the history rate is 10, then the state of the followed object is saved every 100ms. When following, values in between entries are interpolated. Higher rates use more memory and have a higher performance overhead, but have a smoother movement; for efficiency it is best to choose the lowest rate which produces acceptably smooth movement. The default of 30 usually produces good results.

## Follow X

## Follow Y

## Follow Z elevation

## Follow width

## Follow height

## Follow angle

## Follow opacity

## Follow visibility

## Follow destroyed

Choose the built-in properties to follow. For example if following X and Y is enabled, but not width and height, then if the followed object both moves and changes size, the following object will only move but not change size. Following more properties uses more memory. Following the destroyed state means that if the followed object is destroyed, then the following object will also be destroyed at the same point.

## Enabled

Whether the behavior is initially enabled.

## Follow conditions

## Has follow data

True if the behavior has enough data to be able to start following on a delay. For example if the behavior starts following an object on a 5 second time delay, then for the first 5 seconds there is no follow data and so the object will not be updated, and *Has follow data* will be false. Once 5 seconds has elapsed it then starts updating and *Has follow data* will be true. Note that if following starts with *From current position* enabled, then that counts as having follow data immediately.

## Is following object

True if any object has been set to be followed. If false then the behavior is not recording any information.

## **Is paused**

True after using the Set paused action to pause following.

## **Is following custom property**

Test if a particular custom property, specified by a string, is currently enabled for following.

### **Compare delay**

### **Compare max delay**

### **Compare history rate**

### **Compare mode**

### **Is enabled**

## **Is following property**

Test the current values of the behavior properties. See *Follow properties* for more details.

# **Follow actions**

## **Follow object**

Begin following the specified object. This starts recording the changes over time of the specified object, and after the delay period has passed, it will then start following the changes for the enabled properties. Until the delay has passed, *Has follow data* will be false as there is not yet any data to follow. Alternatively the *From current position* setting can be enabled, which allows immediate following. When enabled this creates an initial history entry based on the following object's current state in the past at the delay time. Therefore *Has follow data* is immediately true and the object is able to immediately start updating. This has the effect of interpolating from the following object's starting position to the followed object's starting position over the delay time.

## **Follow self**

Begin following the object the behavior belongs to. This records the changes over time of the current object. In this mode, the behavior does not update the object; it merely records the history. Note that as with following a different object, data is only retained up to the max delay. The history can then be saved to JSON and later replayed, or it can stop following and then set the delay to move to a previous position.

## **Stop following**

Stops recording the history of a followed object. Any recorded history of the object that was followed is still preserved, and it will still continue following changes up until the time that this action was used.

## Clear history

Erases any recorded history about the object being followed. This will cause *Has follow data* to become false and stop updating the object until enough data has been collected again. This is useful for resetting the behavior.

## Set paused

Set whether following is paused. While paused, no further history is recorded, but it also stops advancing the follow time. Upon resuming, the behavior will restart recording the history of the followed object. If the followed object has substantially changed while following was paused, then it will skip to the new position as it follows the history, as no changes in between will have been saved.

## Rewind history

Rewinds the follow time, deletes history entries past that time, and then continues recording history. Note that it is not possible to rewind further than the max delay, as data beyond that time is erased. This action allows for implementing a 'rewind time' feature where an object can go backwards in time and then continue from a different location, while preserving the history prior to the time it continued from.

## Load history JSON

Load the recorded history of the object being followed from a string of data in JSON format previously saved by the *HistoryAsJSON* expression. This also sets the delay to the time of the oldest history entry loaded, so it then immediately follows the amount of data originally saved. This allows for creating a record/replay feature.

## Start following custom property

### Stop following custom property

Start or stop following a custom property. This allows the Follow behavior to track a custom value other than one of the built-in properties such as the X and Y position. Multiple custom properties can be followed, each identified by a case-insensitive string. The interpolation mode of the custom property determines how values in between history entries are determined. *Step* does not interpolate and just uses the previous history entry. *Linear* uses linear interpolation, suitable for linear values like position and size. *Angular* uses angular interpolation, suitable for rotational values like angles. Note that if a custom property value is a string, then only *Step* mode is supported. The value to be recorded must be set with *Set custom property value*, and then the value to be followed can be retrieved with the *DelayedCustomPropertyValue* expression.

## Set custom property value

Set the current value of a custom property that is being followed. The custom property is identified by a case-insensitive string. The value can be either a string or a number, but if it is a string then it will only use *Step* interpolation mode. This action should be used every tick while following a custom property, so that the latest value is available when the behavior decides to add a history entry.

---

**Set delay****Set enabled****Set following property****Set history rate****Set max delay****Set mode**

Set the corresponding behavior properties. See *Follow properties* for more details.

---

**Set property interpolation**

Change the interpolation mode of one of the built-in properties. Generally this is used to change one of the built-in properties from smooth interpolation to step interpolation. For example mirroring an object with the Platform behavior should always update the width instantly, and not interpolate any in-between values.

## Follow expressions

---

**FollowUID**

The UID of the current object being followed, or -1 if no object to follow has been set.

---

**HistoryAsJSON(MaxDelay)**

Save the current recorded history of the followed object to a string in JSON format. This can then be loaded again later using the *Load history JSON* action. The *MaxDelay* parameter can be used to save only a portion of the most recent history, rather than all the recorded history, suitable for a record/replay feature when the recording duration is less than the behavior's max delay. If *MaxDelay* is 0, then it saves all the history.

---

**DelayedCustomPropertyValue(CustomProperty)**

Retrieve the current value to be followed for a custom property, specified by a case-insensitive string.

---

**Delay****MaxDelay****HistoryRate**

Return the corresponding behavior properties. For more information, see *Follow properties*.