# CUSTOM ACTIONS

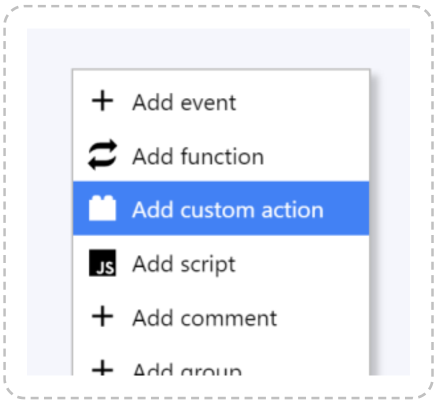
**View online:** https://www.construct.net/en/make-games/manuals/construct-3/project-primitives/events/custom-actions

**Custom actions** are special kinds of event blocks that can be called from an action in an associated object type or family. They work similarly to functions, so it is useful to understand how functions work first before reading about custom actions.

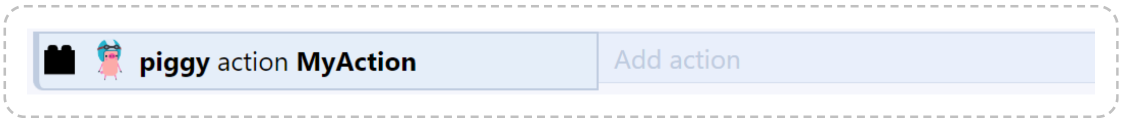
Using custom actions can help you organize event sheets and avoid having to duplicate groups of actions or events. Custom actions also have more advanced uses when added to families, allowing for members of the family to override or extend a family custom action.

## Adding custom actions

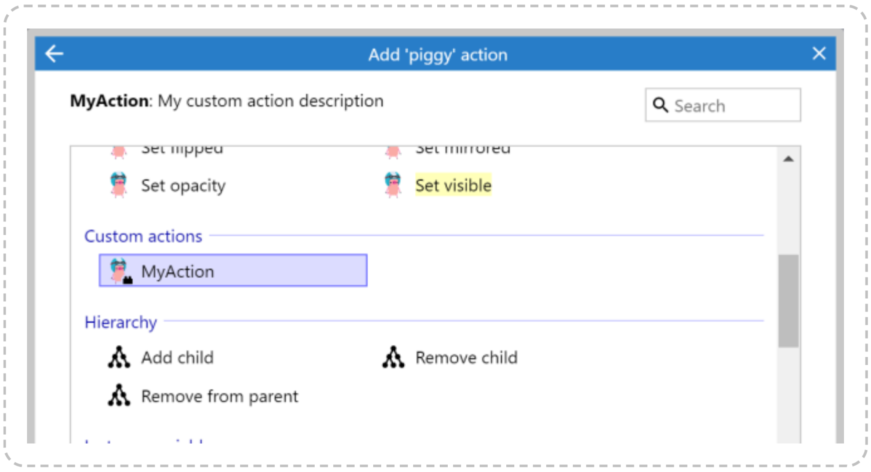
In the event sheet, custom actions are created by adding a special kind of event block. To create one, use the *Add custom action* menu option instead of *Add event*.



When you select this the Add custom action dialog will appear for you to fill in details about the custom action. Once created, the custom action appears in the event sheet similar to a normal event, but with a special icon and descriptive text at the top. This is referred to as the *custom action block*.

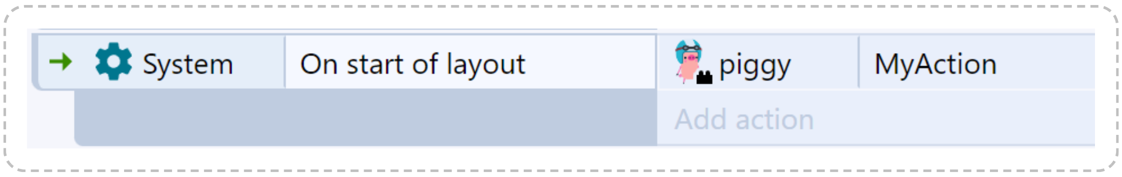


You can add conditions, actions and sub-events to custom action blocks, just like you can with normal events. However custom actions do not run unless you run them as an action in its associated object type or family. Once you've added a custom action block to your project, it will appear in the Add action dialog alongside all the other object type or family's usual actions.



**MyAction**: My custom action description

Choosing the custom action from the *Add action* dialog adds an action that calls (runs) the associated custom action block.



Running the custom action action will run the corresponding custom action block, including testing its conditions, running actions, and running any sub-events, and then return to the original action and continue from where it left off.

*Custom actions are global. This means you can use custom actions anywhere in your event sheets, even if the corresponding custom action block is in a different event sheet that is not included in the event sheet you call it from.*

## Parameters

Much like functions, custom actions can also use parameters. Since these work the same as with functions, refer to the section on *Parameters* in the Functions manual entry for more details.

## Picking

When running a custom action, the custom action block is run with the same instances picked as the calling event block. For example this means running a custom action in a *On object clicked* trigger will run the custom action block with just the clicked instance picked. This means custom actions automatically alter just the picked instances, much like normal actions. However when the custom action block finishes running, any changes to the picked instances it made are discarded, so it does not affect the running of the original event that called it.

The *Copy all picked* setting of the custom action block can alter how this works. Normally only instances of the custom action block's object are automatically picked. However if *Copy all picked* is checked, the custom action block will inherit *all* picked instances from the calling event block - including other object types and families. This makes it work similarly to a function with *Copy picked* enabled.

## Asynchronous custom actions

Much like functions, custom actions can also be made asynchronous, so they can be used with the system *Wait for previous actions to complete* action. Since this feature works the same as with functions, refer to the section on *Asynchronous functions* in the Functions manual entry for more details.

## Family custom actions

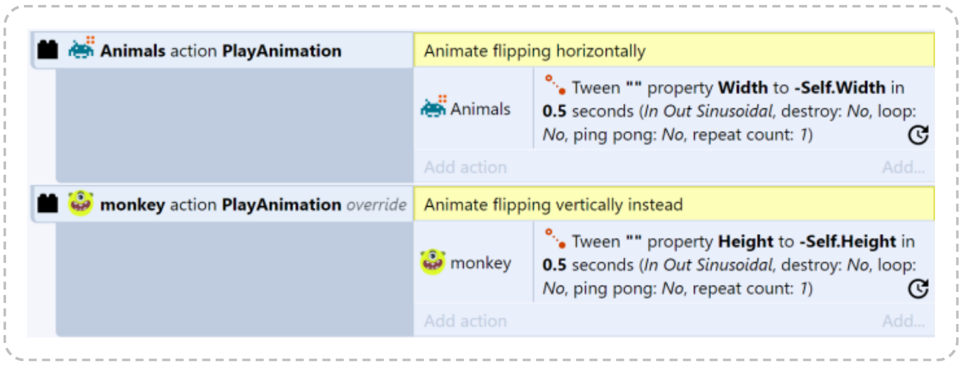
A custom action can be created for a family. This allows some more advanced uses of custom actions.

Much like with inheriting family instance variables, behaviors and effects, family custom actions can also be used as actions for every object type in the family. This allows every member of the family to share the same custom action.

### Overrides

When a family has a custom action, it's still possible to create a custom action with the same name for a specific object type in that family. In that case, the object type's custom action *overrides* the family custom action.

For example suppose there are three sprite object types named *Piggy*, *Octopus* and *Monkey* in a family named *Animals*, and there is a custom action named *PlayAnimation* for the family *Animals*. A custom action named *PlayAnimation* can also be added for *Monkey*. Then when running the family custom action *PlayAnimation*, the family custom action block will run for *Piggy* and *Octopus* instances, but the *Monkey* custom action block will run instead for *Monkey* instances.



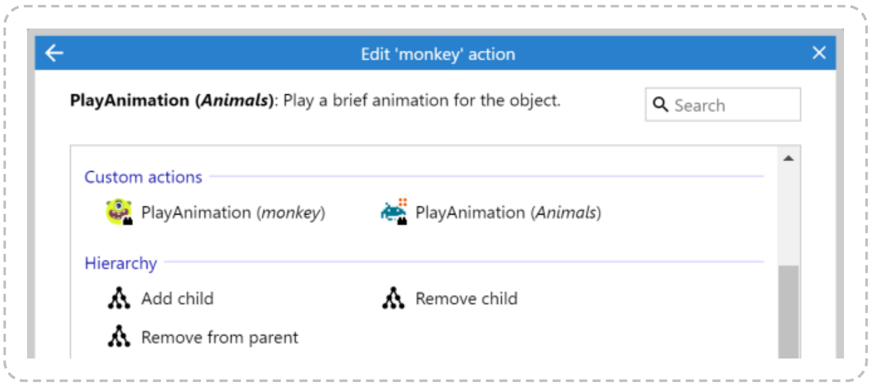
**Animals** action **PlayAnimation**
Animate flipping horizontally
Animals: Tween "" property **Width** to **-Self.Width** in **0.5** seconds (*In Out Sinusoidal*, destroy: *No*, loop: *No*, ping pong: *No*, repeat count: *1*)
Add action
Add...

**monkey** action **PlayAnimation** *override*
Animate flipping vertically instead
monkey: Tween "" property **Height** to **-Self.Height** in **0.5** seconds (*In Out Sinusoidal*, destroy: *No*, loop: *No*, ping pong: *No*, repeat count: *1*)
Add action
Add...

This allows specific object types in the family to override what a custom action will do for instances of that object type.

### Choosing overrides

Consider the previous example with the *Monkey* object type having an override for the family custom action *PlayAnimation*. In this case, the action list for *Monkey* includes both actions, listing *PlayAnimation (Monkey)* and *PlayAnimation (Animals)*.



**PlayAnimation (*Animals*)**: Play a brief animation for the object.

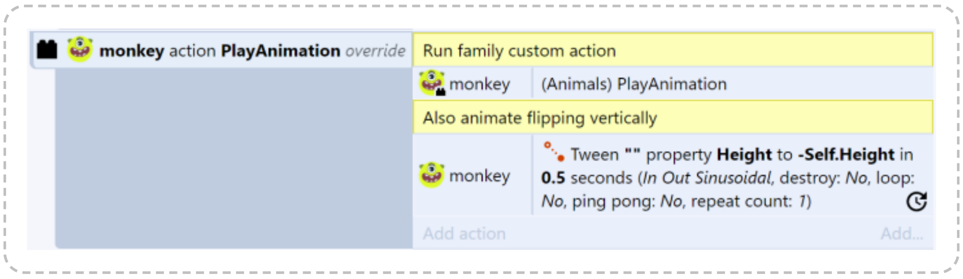
PlayAnimation (*monkey*)    PlayAnimation (*Animals*)

This allows choosing which custom action is run: the *Monkey* variant will run the override custom action for *Monkey*, and the *Animals* variant will run the original family custom action ignoring the override. This allows you to explicitly choose whether to run the override or the original family custom action for the *Monkey* object type.

## Extending a family custom action

By default, an override custom action will entirely replace the family custom action. However it's also possible to make it *extend* what the family custom does. This can be done by using the ability to choose overrides to add an action to the override custom action block that calls the original family custom action block. (In programming languages, this is sometimes referred to as a "super" call.)

Continuing the previous example, the *Monkey* custom action block can add a *Monkey* action to run *PlayAnimation (Animals)*, which is the original family custom action.



**monkey** action **PlayAnimation** *override*

Run family custom action

monkey    (Animals) PlayAnimation

Also animate flipping vertically

monkey    Tween **""** property **Height** to **-Self.Height** in **0.5** seconds (*In Out Sinusoidal*, destroy: *No*, loop: *No*, ping pong: *No*, repeat count: *1*)

Therefore when running the *Monkey* custom action override, it will first do the original family custom action, and then do its own actions after that. This allows extending what the family custom action does to do additional things for specific members of the family, rather than completely replacing the custom action.

*Be sure to get the right action in this case. If you accidentally call the Monkey custom action again from the custom action block, it will create an infinite loop of calling the custom action repeatedly. To run the family custom action, be sure to add an action for the same object type as the custom action block, and then choose the family variant of the custom action.*