Functions

An “Events Function” or simply Function is a powerful feature of GDevelop. External events, links and groups are useful to organize the events of your game, and even reuse some, in more than one place in your game. For example, you could put common events, to manage the enemies in some external events, and include these events, using a link, in various scenes.

Functions are going one step further: they allow you to declare **new conditions**, **new actions** and even **new expressions**, using events.

When you register a function, you can choose if it's a condition, an action or an expression. The Function contains events, like a scene or external events, that will be executed when the condition or the action is launched during the game.

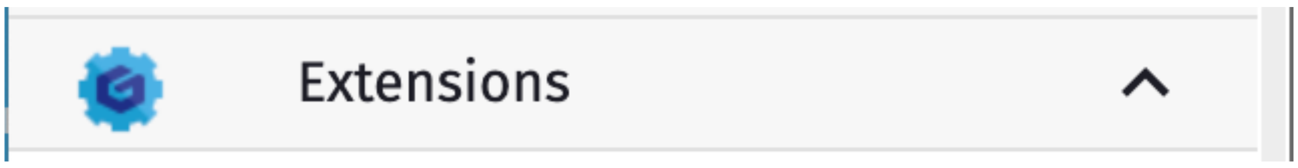
In addition to events, functions also have parameters, just like usual conditions, actions and expressions. The parameters can be objects, number or text.

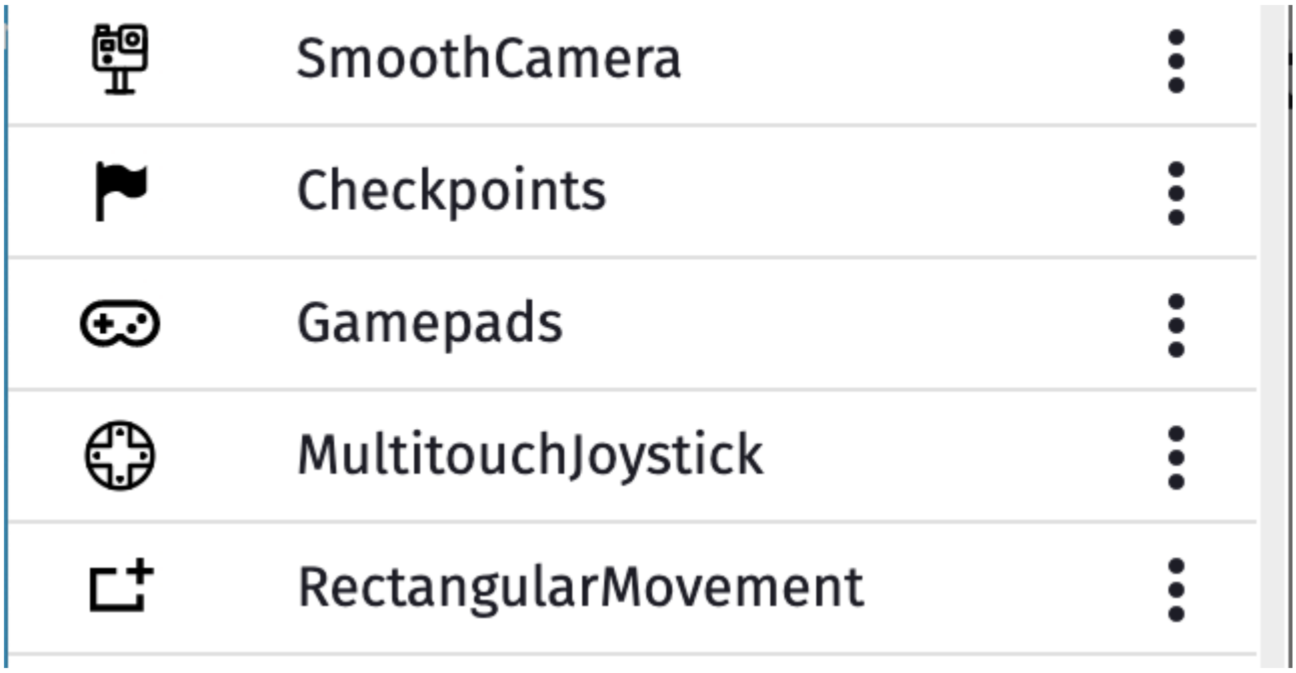
See an example of replacing external events by functions. You can also **automatically extract events to a function**.

See an example of functions being used in this video by gamefromscratch .

Creating a new function

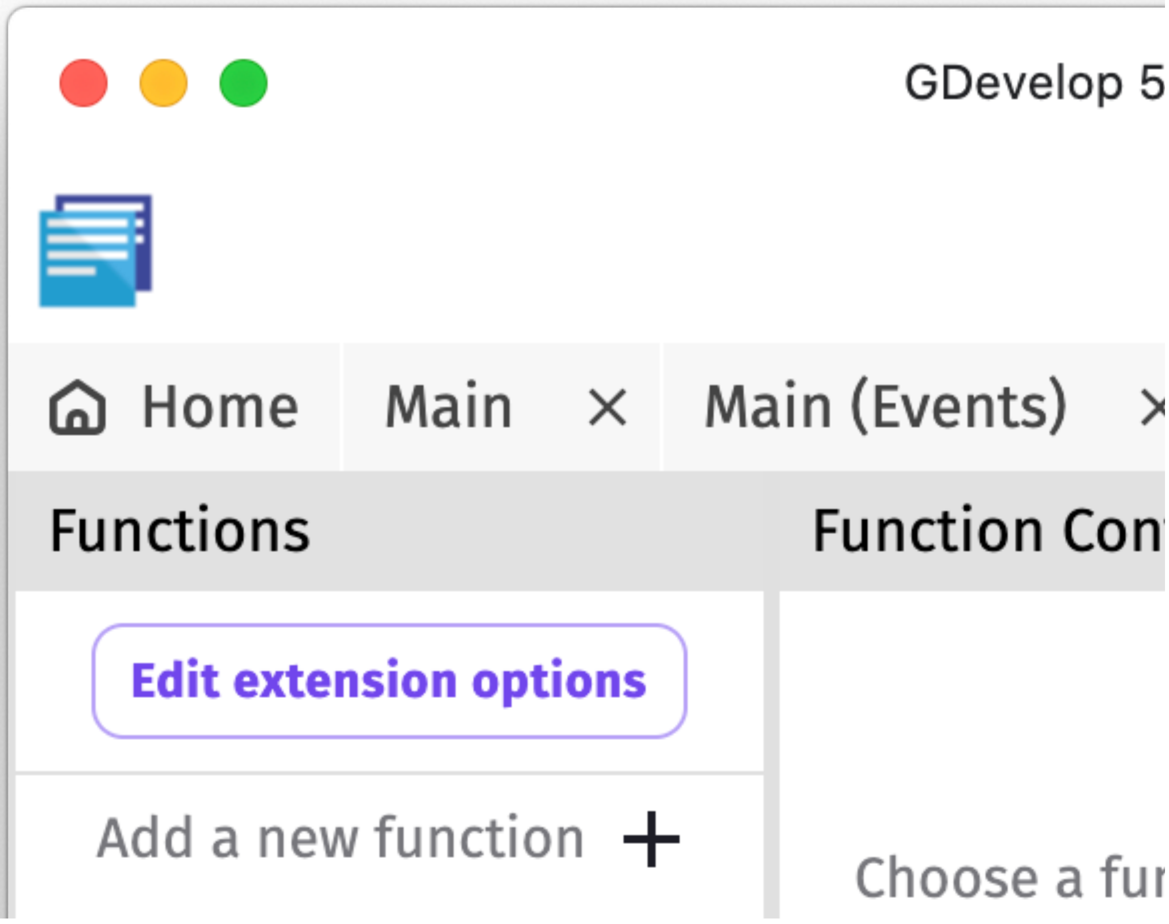
Functions are grouped into “extensions”. They are the same as the extensions that you can install for your game. You can see the list of extensions that the game has, as well as add a new extension in the Project Manager:

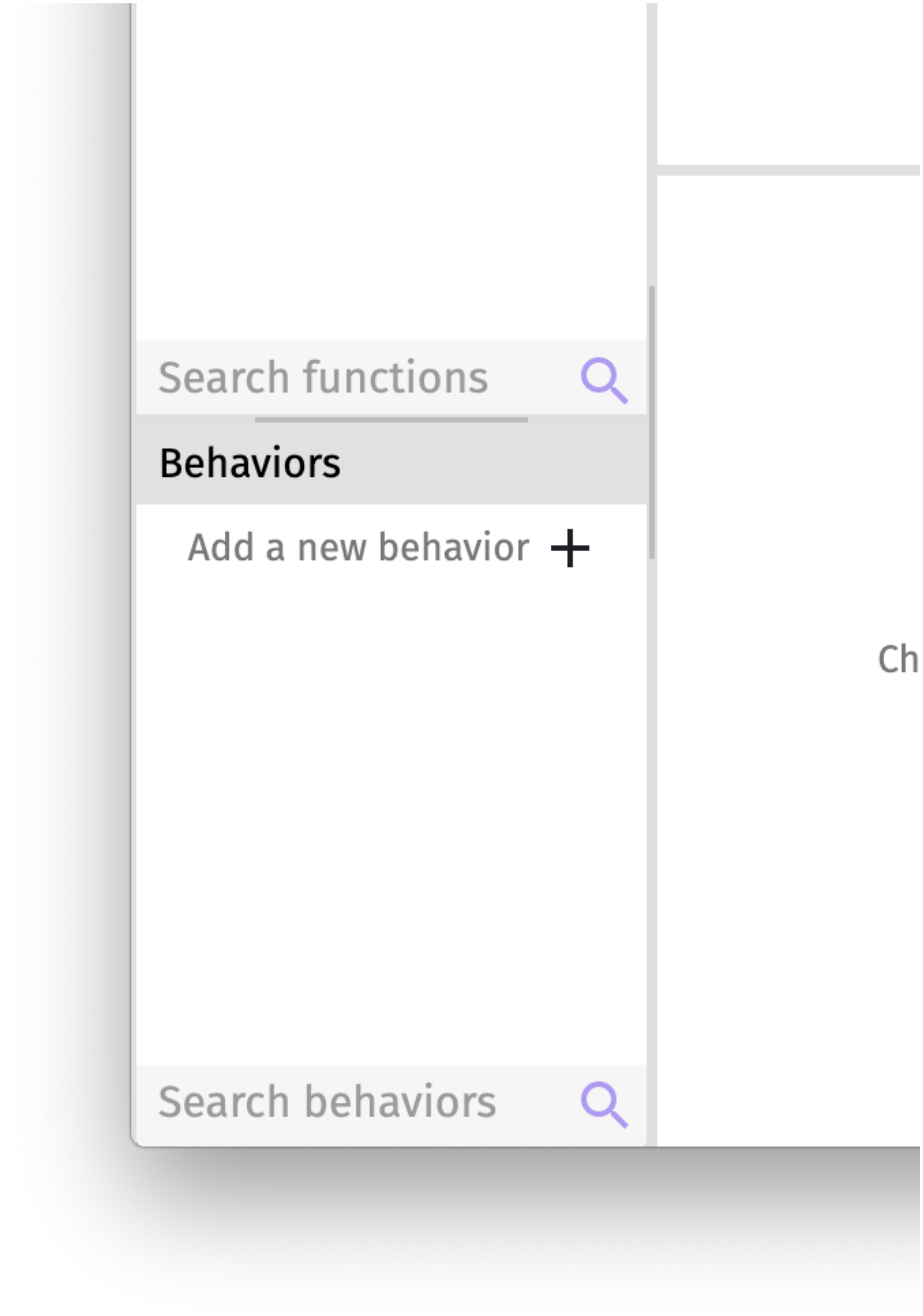




Click on **Create or search for new extensions** on the bottom then on **Create a new extension** to create a new extension. It's a good idea to have functions related to the same thing in a single extension. Careful, the name is sensitive to the case, so `DayNightEffects` and `Daynighteffects` are two different functions.

Click on the extension in the list to open its editor. By default, there are no functions in the extensions. Add one by clicking on “Add a new function”, on the left:



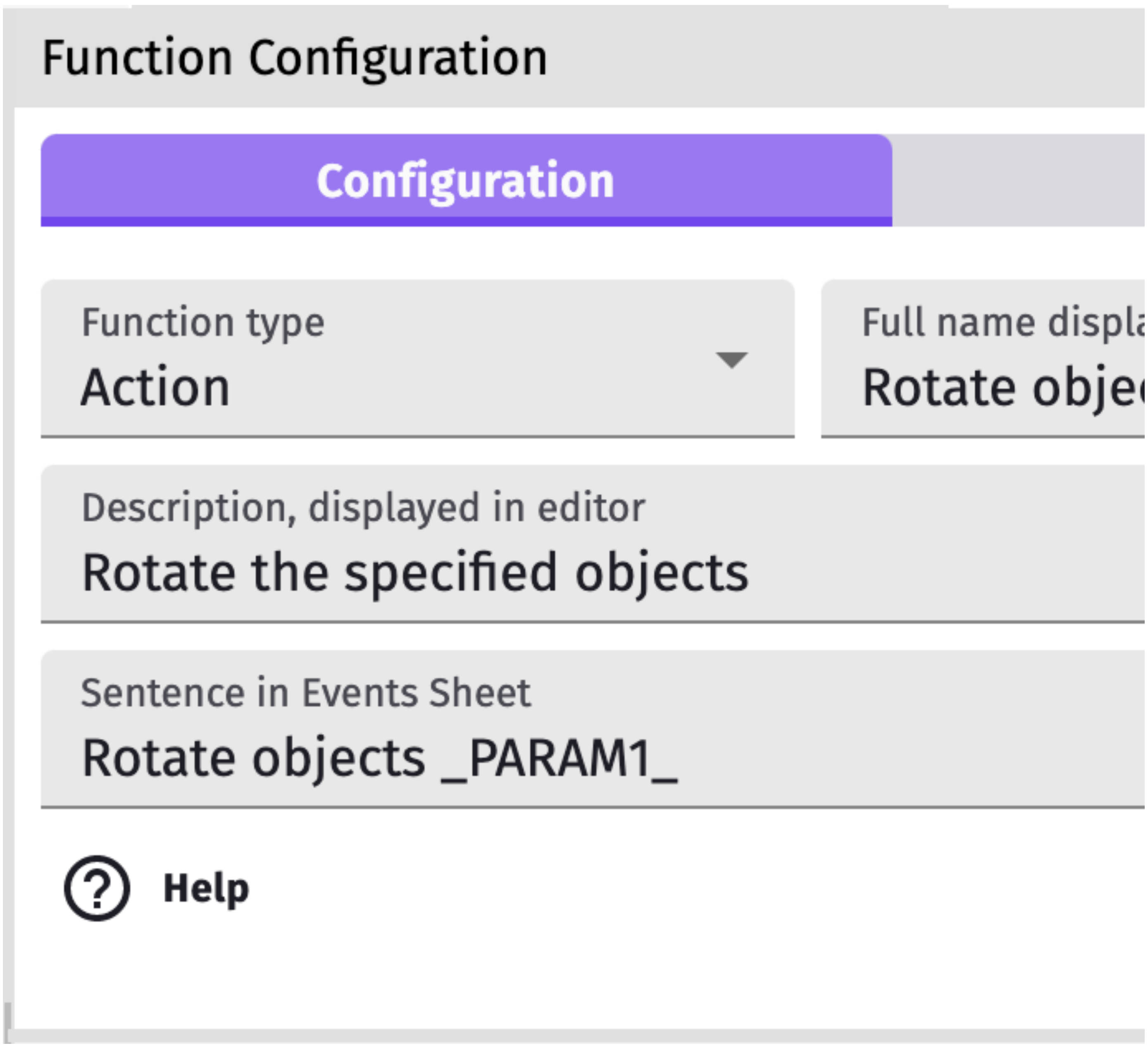


A new function is added. You can rename it, to give it a name according to what you want to do inside. For example, if your function will be a condition that checks if the object passed as parameter is ready to fight, you can call it `IsReadyToFight` (only alphanumeric characters and underscores are allowed in names).

Click on the function to edit it: you'll be able to change its configuration and its events.

Editing the configuration and the events of the function

When a function is selected, on the top, you can see the configuration of the function:



Function Configuration

Configuration

Function type: Action

Full name displa: Rotate obje

Description, displayed in editor: Rotate the specified objects

Sentence in Events Sheet: Rotate objects _PARAM1_

- The first parameter is the type of the function: “Action”, “Condition”, “Expression” or “String Expression”. If you choose Action or Condition, you'll find the function in the list of actions and conditions, when editing your events in the game. If you choose Expression (or String Expression), you'll find it in the list of expressions when you edit a formula.
- You can then configure the name that will be displayed in the list.

- Enter the description that will be given in the window, when choosing the parameters for the function.
- For Action or Condition, you can enter the sentence that will be displayed in the events sheet. In case your function takes parameters (see below to learn more about these), you can include them by writing PARAMx between underscores, replacing `x` by the parameter number (starting at 1):

```
Rotate objects _PARAM1_
```

This is an example for a function named “RotateObjects”, which is an action, with the description “Rotate the given objects”, and with a single parameter, the objects to be rotated.

Parameters

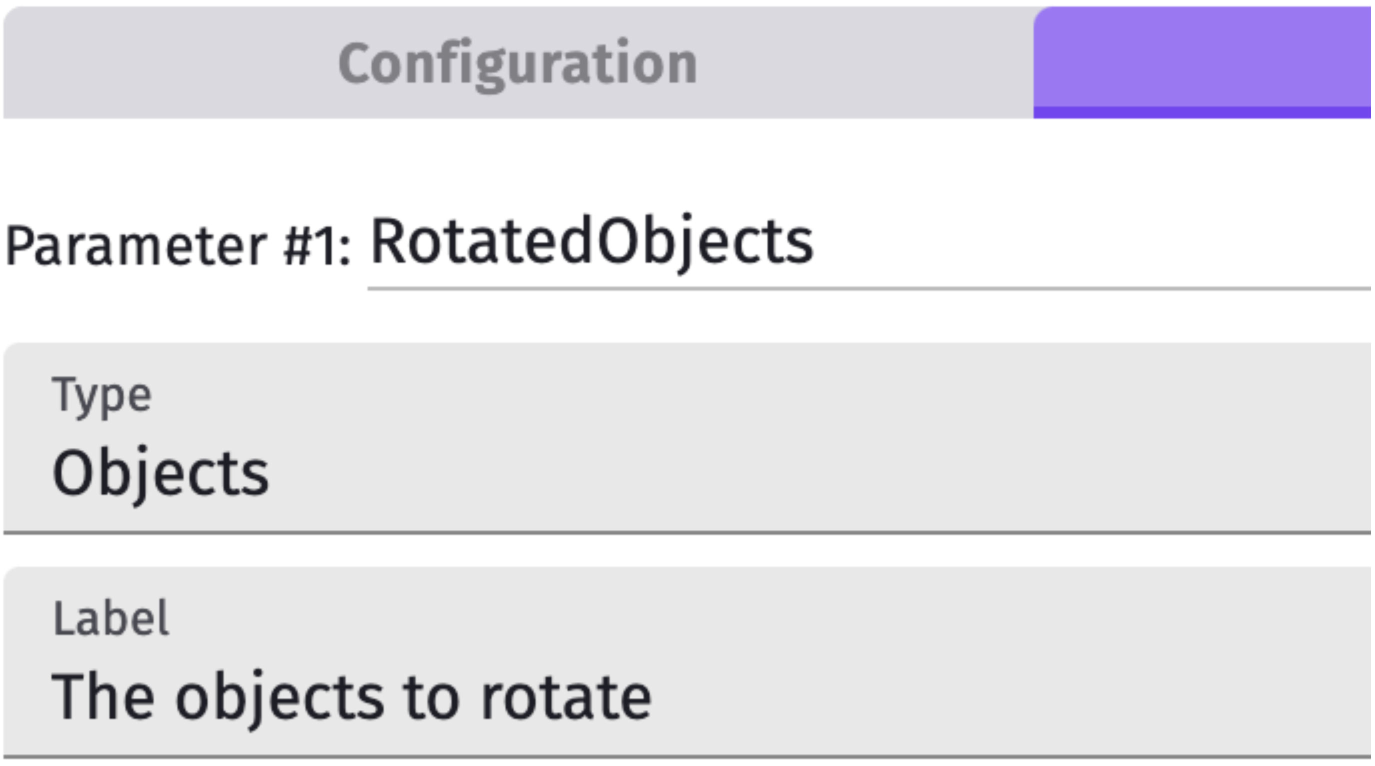
Functions are becoming really useful and powerful when you use parameters. Under the configuration of the function, you can add parameters. These parameters will be available when you use the action, condition or expression in the events sheet. It's just like the usual action/condition/expressions you're used to!

Add a parameter with the button “Add a parameter”:

For each parameter, you can enter:

- The name, that will be used to access the parameter in the *events of the function*.
- The type. The parameter can be objects, a number or a text. If you choose objects, you'll also be able to choose the type of object to receive.
- A description, that will be shown in the window when configuring the action/condition or expression.

For example, we can add a parameter, that would return the objects to be rotated:



Parameter #1: RotatedObjects

Type
Objects

Label
The objects to rotate

These parameters can be used in a Javascript event, see here how to use parameters in Javascript.

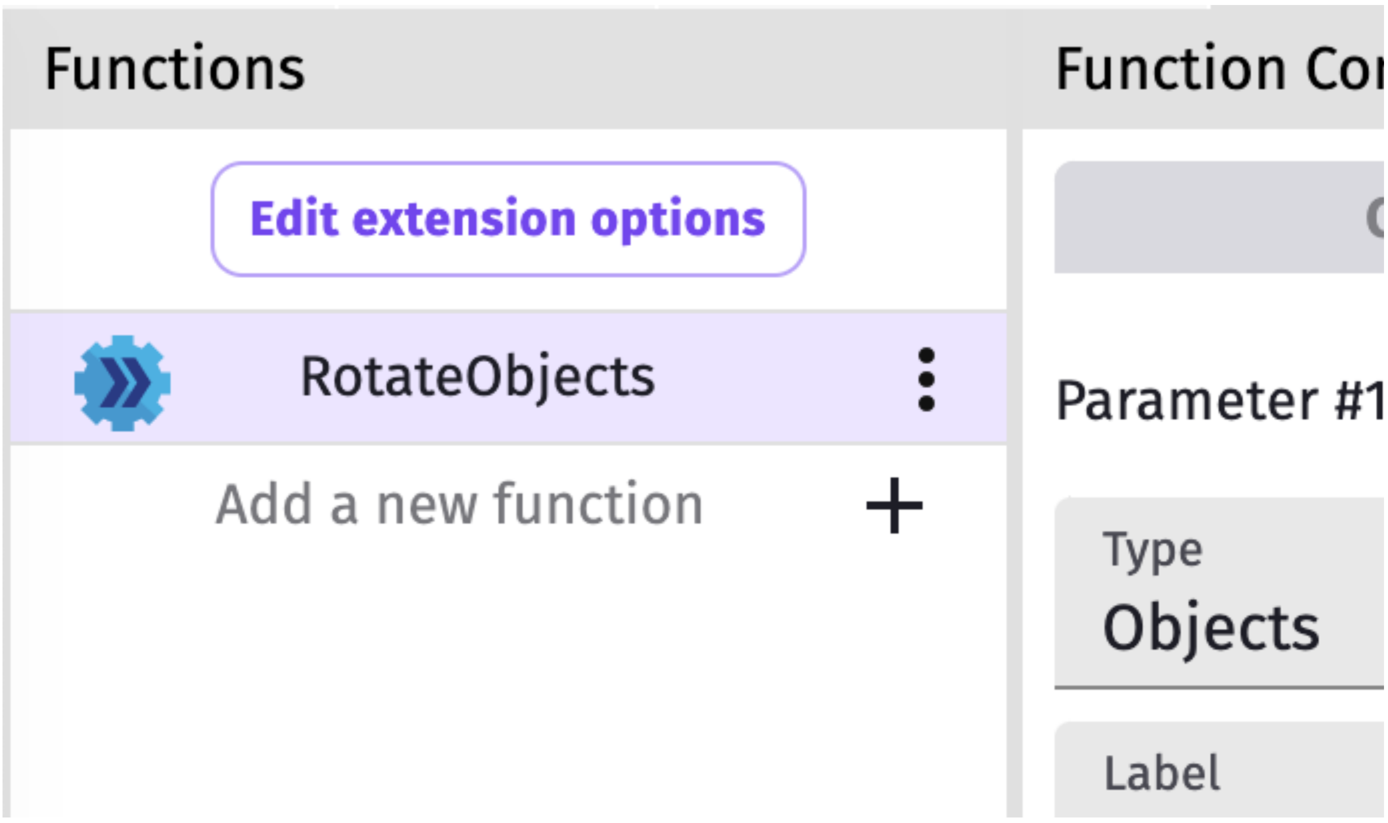
Adding the events to the function and using the parameters

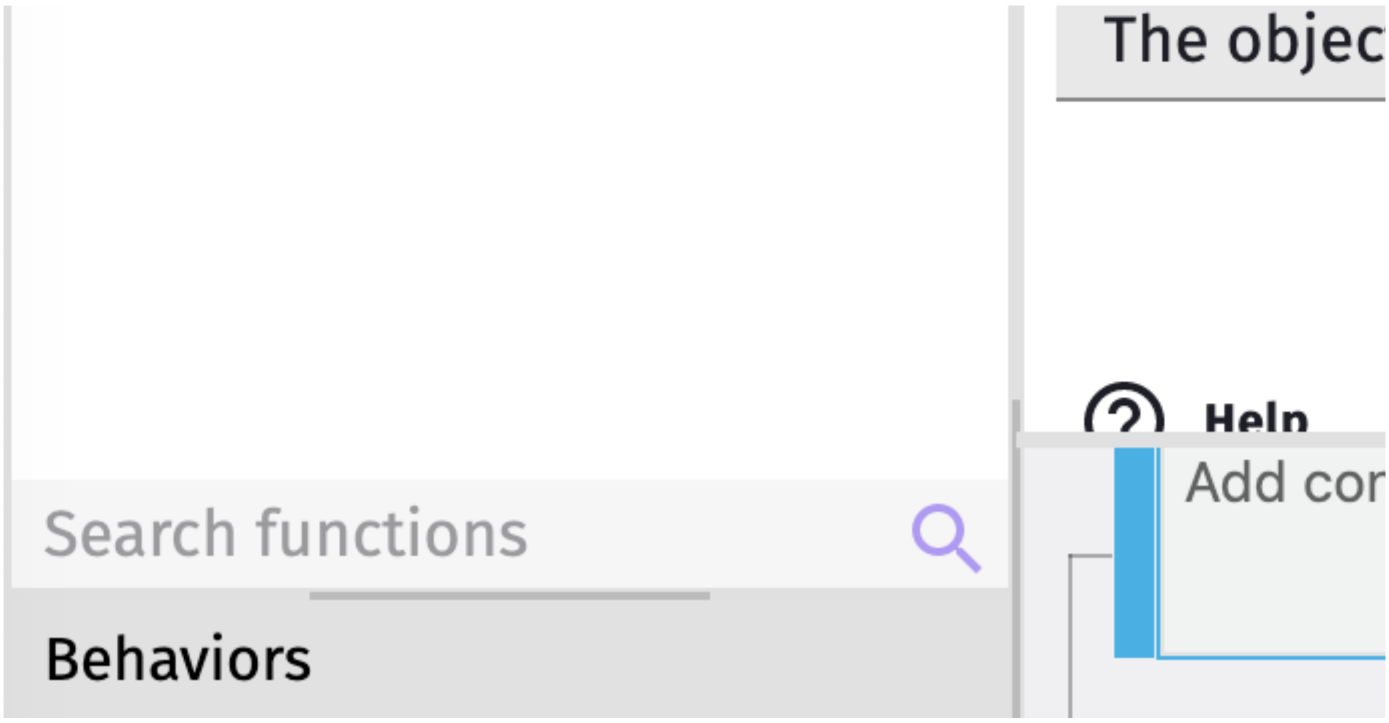
When your function is configured, you can add events to it. These events will be launched when the condition, action or expression is used in the rest of the game.

- You can use all the existing events, actions, conditions and expressions, but you are limited to the **objects that you entered as parameters**. This is to ensure that your function is only acting on them, and has no “side-effects” on the rest of the game - which would be a bad practice and make functions hard to reuse and to generate.
- Note that functions can be reused everywhere, and are not limited to a scene. You **won't have the list of variables of your scenes** in them. You can still manipulate them by using the usual actions and manually writing the variable you want to modify.
- **Links** are not available in functions, because a function is autonomous and is in theory not even tied to a project.

Object parameter

Here is an example of a function to rotate some objects:





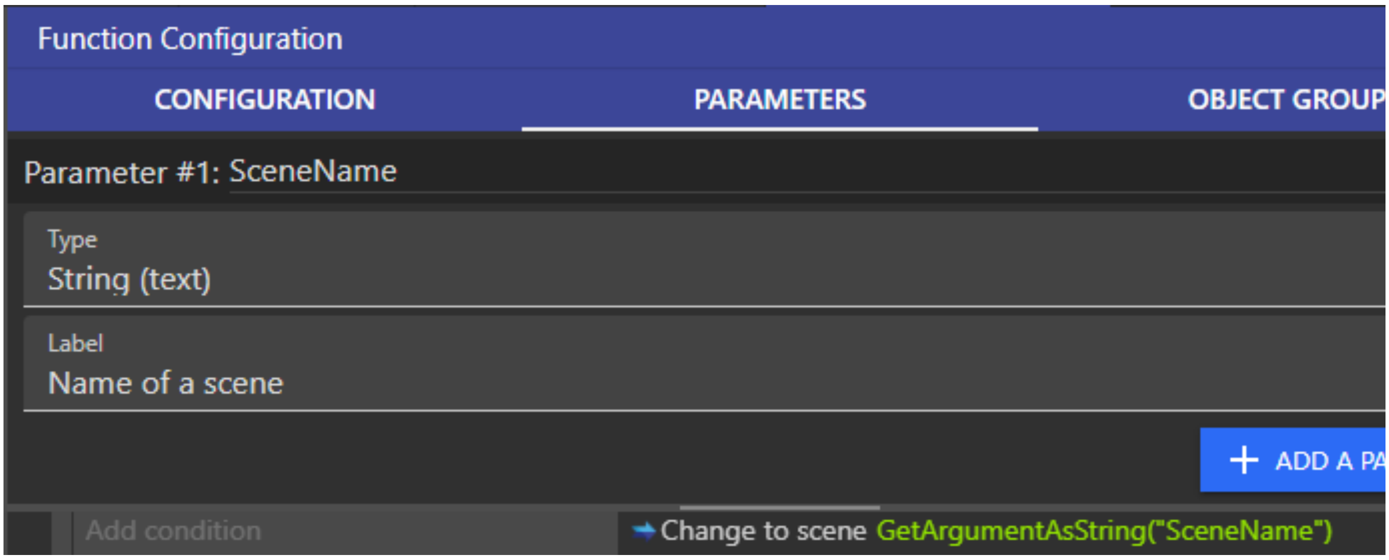
This is a really simple and not really useful example of a function (you could as well use the action to rotate objects directly without writing a function). But, when you add more complex logic inside, a function's strength can be seen. It's then super easy to reuse this logic from the rest of the game!

If your function is a condition or an expression, use the actions in "Functions" category to set the expression/condition value (also called the “return value”).

Other types of parameters

A parameter can be an object, a text, a number and so ones, for texts and numbers there is two expressions that can be used in events.

Get a number from a parameter: `GetArgumentAsNumber(string)` Get a text from a parameter: `GetArgumentAsString(string)` Here is an example of a text parameter that will get the name and use it in an action for change scene.

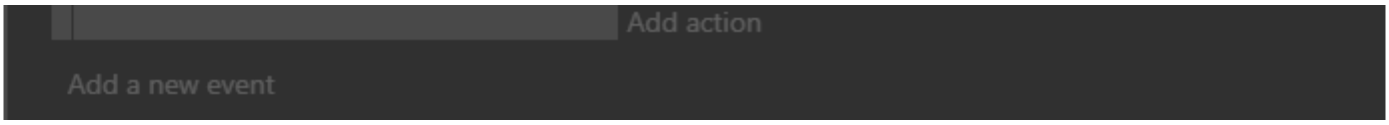


Parameter #1: SceneName

Type
String (text)

Label
Name of a scene

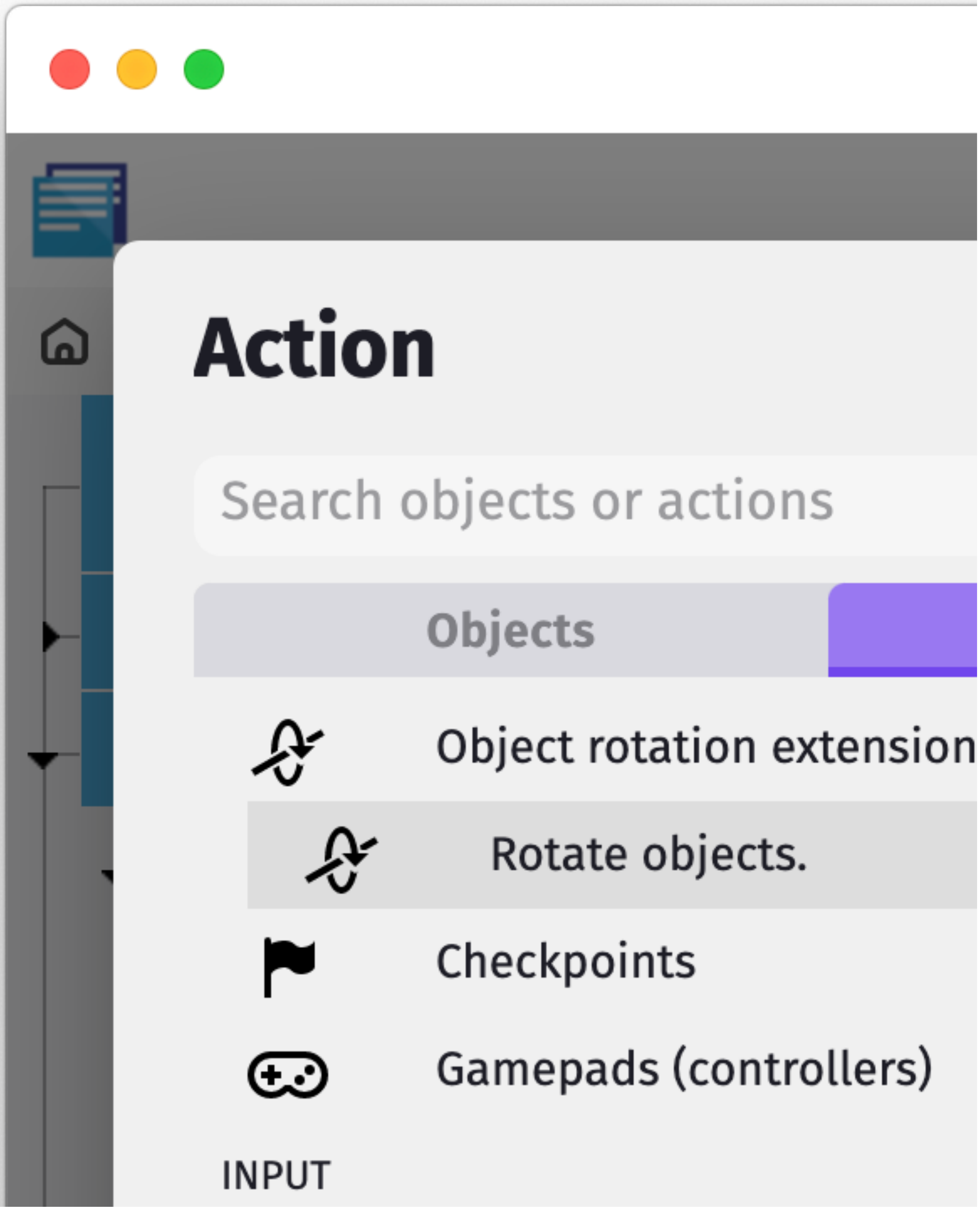
Change to scene GetArgumentAsString("SceneName")

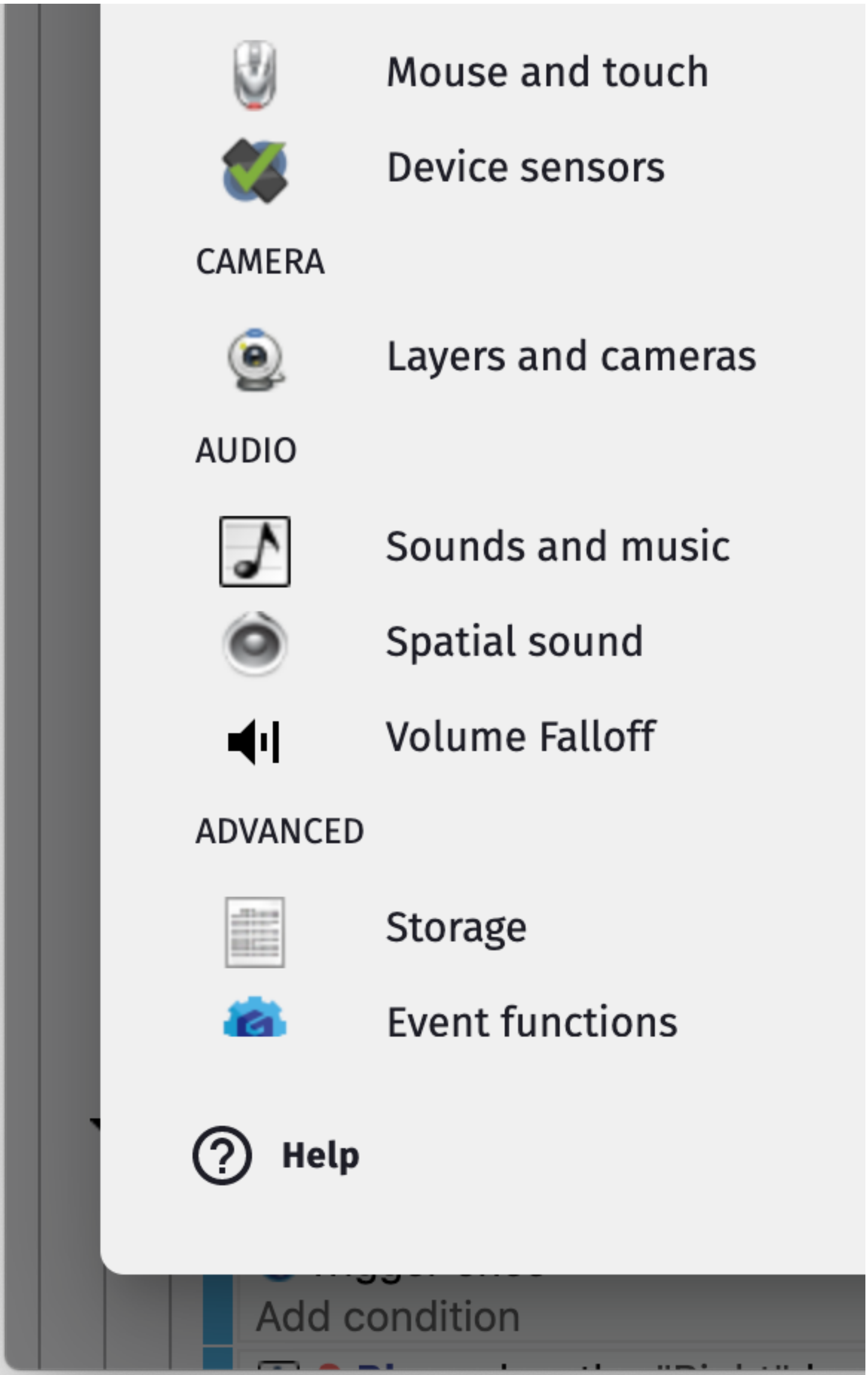


Using the function in your game

When you have configured and created the events for your function, you can use it in the rest of your game.

Simply create a new action (or condition) and find in the list the name of your extension, then choose the function that you've created:




Mouse and touch
Device sensors
CAMERA
Layers and cameras
AUDIO
Sounds and music
Spatial sound
Volume Falloff
ADVANCED
Storage
Event functions
Help
Add condition

That's it! The function is used like any other condition, action or expression in the Events Editor.

The **Object Groups** feature, while creating functions, helps grouping similar object parameters. This way you can apply an action/condition to a group of object parameters (parameters pointing at objects) at once.

Advanced usages

This page gave a basic overview of what functions are. They are one of the more powerful features of GDevelop, as you can extend the events by using them, enabling to create very readable and concise events sheets. By using them, you can reduce the amounts of events that you write for your game, avoid copy-pasting them and even reduce bugs by ensuring that functions are always used for common tasks on your objects.

Recursive functions

A function can call itself! In the events of a function, you can use the same action/condition/expression as in the rest of the game. This is called a “recursive” function.

Be very careful when writing a recursive function. If you don't add conditions, the function could call itself infinitely, blocking your game.

You can use the actions in "Functions" category to return expression/condition values.

Sharing functions

Functions can be shared across projects (like actions/conditions that are built in GDevelop). The simplest way to do it is just to copy/paste them. But if you've created a useful set of functions, you may consider sharing them with the community.

See an example of replacing external events by a function, which is reusable and shareable. You can also **automatically extract events to a function**.