Working with Blockly

[Переключить на русский](https://docs.google.com/document/d/1Gd9xwKW9dhcgzszyvIeFQjK_mbY3qgL24qCHUs5lxZw)

Contents

[Introduction to Blockly](#_1fob9te)

[Further reading](#_3znysh7)

[Standard sections](#_2et92p0)

[Custom sections](#_tyjcwt)

[Actions](#_3dy6vkm)

[Events](#_1t3h5sf)

[Objects](#_4d34og8)

[Blocks for objects](#_2s8eyo1)

[Creating custom blocks](#_17dp8vu)

[Further reading](#_p7vab6ysjbsu)

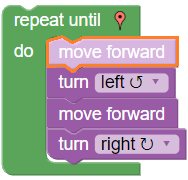
[How Blockly works on Varwin platform](#_3rdcrjn)

# Introduction to Blockly

Blockly is a visual code editor which does not require programming skills. It allows writing program scripts by connecting visual blocks. The blocks are divided into groups according to their purpose or return value. Each group has its own colour. The shapes of the blocks also vary; not all the blocks are compatible.

If blocks fit together, the program will run. Still, logical errors can arise in the process. The block causing the error will be illuminated.

Also, when the program runs in preview mode the blocks which are executed at the moment are illuminated as well. This simplifies watching the script in real time.

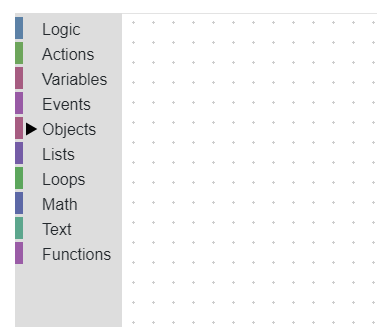


*Illuminated block (is executed at the moment)*

The blocks are managed (copied, deactivated, removed, etc.) by right-clicking on them and keyboard shortcuts Ctrl+C, Ctrl+V.

The blocks can be expanded and collapsed by right-clicking on them.

Blockly interface consists of Workspace, where logic algorithms of the project are built, and Toolbox - side menu from whence the user may select blocks or create new ones.



Blockly is used for composing scripts for VR projects powered by Varwin platform. This opens access to our platform to users with no programmer skills.

* Get acquainted with Blockly: [training games](https://blockly-games.appspot.com/?lang=en).

## Further reading

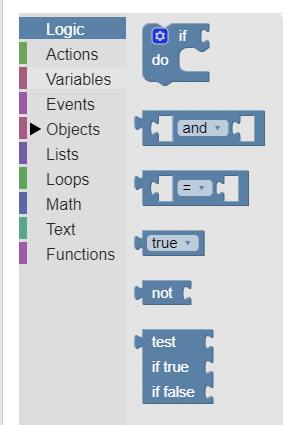
[Introduction to Blockly (official site)](https://developers.google.com/blockly/guides/overview)

# Standard sections

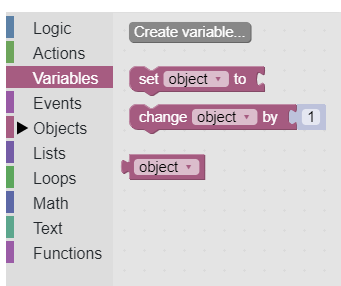
Training games have acquainted you with the standard block sections. These are used within Varwin platform:

* **Logic**

Main logic blocks.

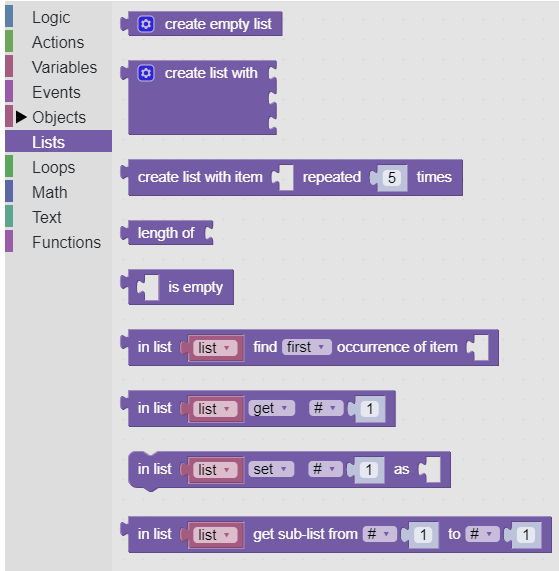


* **Variables**

A variable is an area of memory where to store a certain value to refer to it later. Strictly speaking, an object in itself is a variable.

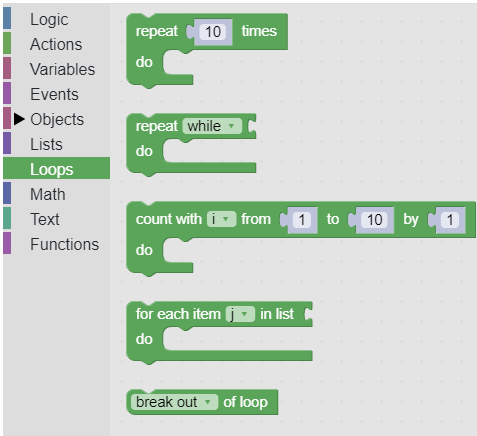
* **Lists**

Set of blocks allowing to work with lists. A list is a set of variables of any type.



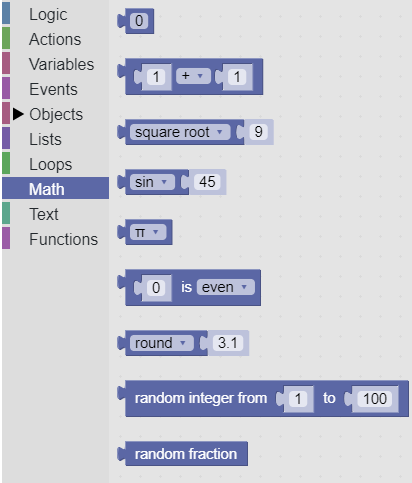
* **Loops**

Blocks allowing to set a number of actions with foregone iteration count. Can be used also to avoid lists.



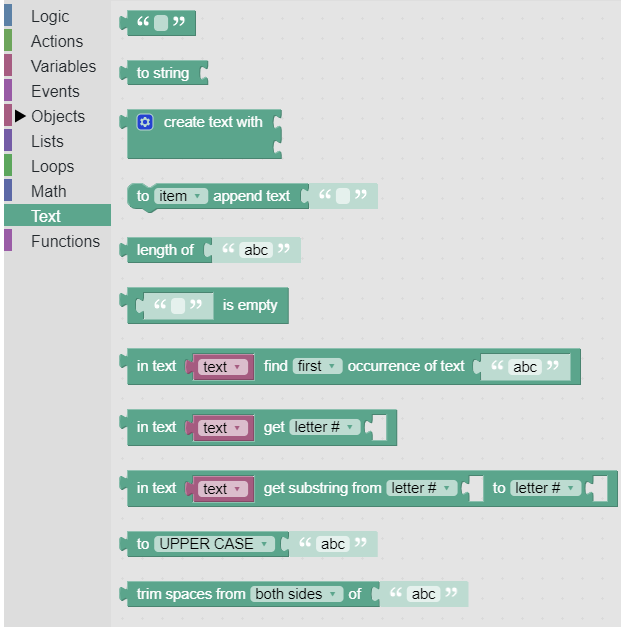
* **Math**

Set of blocks allowing to work with numbers.



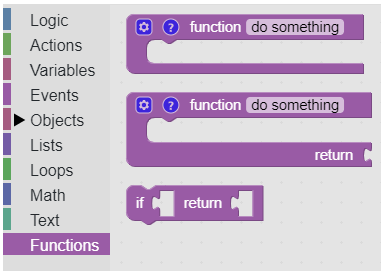
* **Text**

Set of blocks allowing to work with texts.



* **Functions**

Set of blocks allowing to create and call functions. Functions are good for structuring and multiple use of repeatable logic.

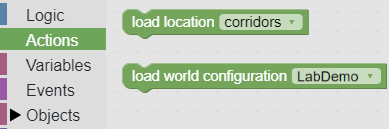


# Custom sections

These sections have been created specially for Varwin platform.

## Actions

Commands which invoke certain system actions.

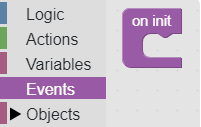


## Events

All the blocks in the workspace are executed on the frame to frame basis. The exceptions are blocks from Function and Event groups.

Blocks placed inside an event are executed only at the moment of an event.

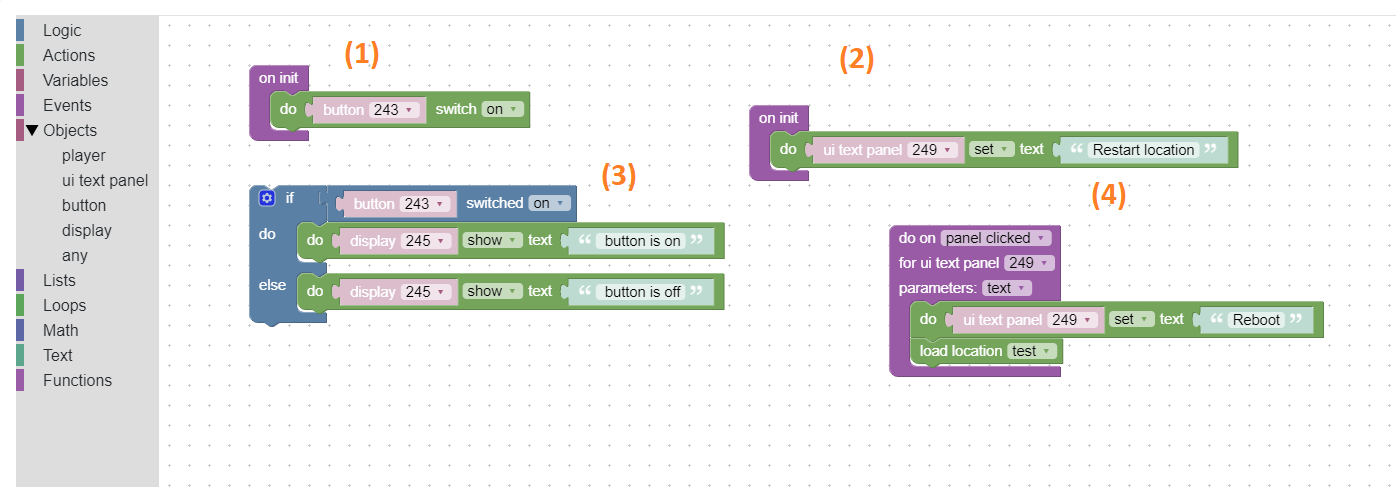
Events section stores general system events; e.g., **on init** (at the moment of initialization) event.



Blocks that are inside this event are operated only once, at the moment of scene load or of changing the mode to Preview.

This block is good for the initialization of objects state within a scene, e.g., switching on a button which is switched off by default. The block can be used multiple times.

**Example**

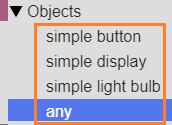
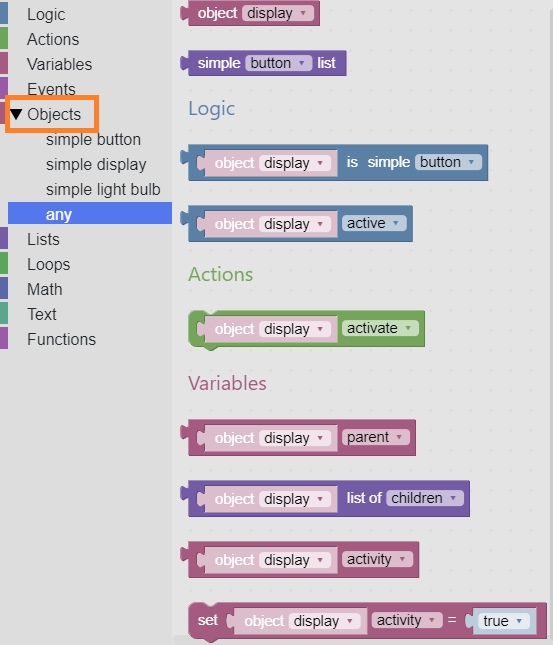


* сommands 1, 2 are executed only once at the moment of scene load or of changing the mode to Preview,
* command 3 is executed frame by frame,
* command 4 is executed at the moment of event (pressing on the panel.)

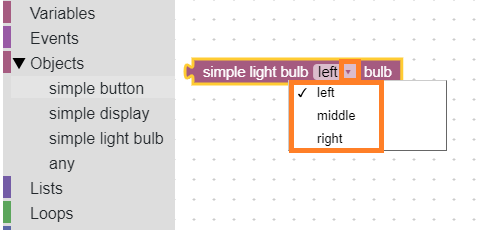
## Objects

Objects spawned on the scene in VR are displayed in this section. Available states (variables) for each object are shown as well (ill. 1)

* **object:any** is a section with blocks that can be applied to any object.
* Objects added to a scene are grouped by type (ill.2: area, button, display, etc.) The specific object can be selected from the drop-down list (ill.3)



*ill.1 ill.2*



*ill.3*

### Blocks for objects

|  |  |  |  |
| --- | --- | --- | --- |
| **Section** | **Name** | **Purpose** | **Picture** |
| Logic | checker | check true/false |  |
| Action | action | Actions with object |  |
| Variables | getter | Get a state |  |
| setter | Set a state |  |
| Events | event | Events with object |  |

### Creating custom blocks

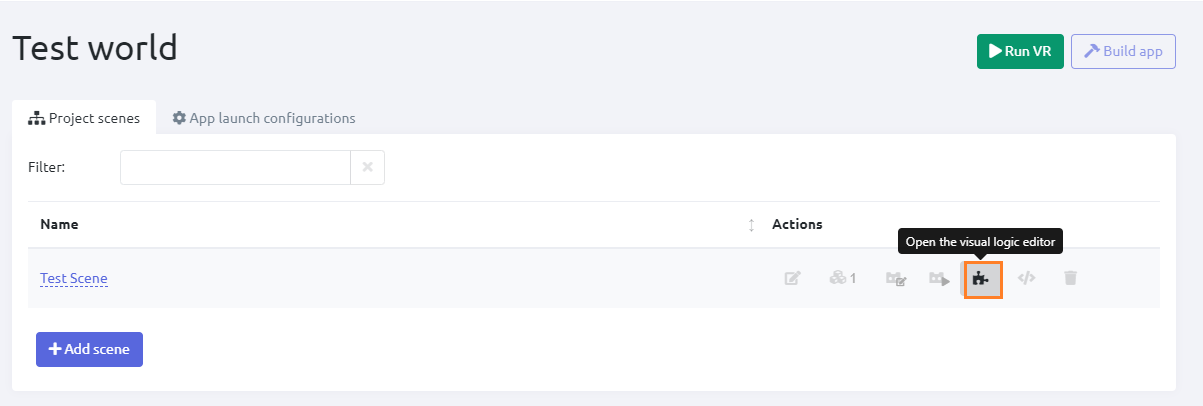
Custom blocks can be created along with object code. See an example here: [Creating objects for Varwin in Unity - Example](https://docs.google.com/document/d/1hsiaY3RPNxpEiCmi49X1WuThOj3C-a0yksD43CJXfFc/edit#heading=h.30j0zll)

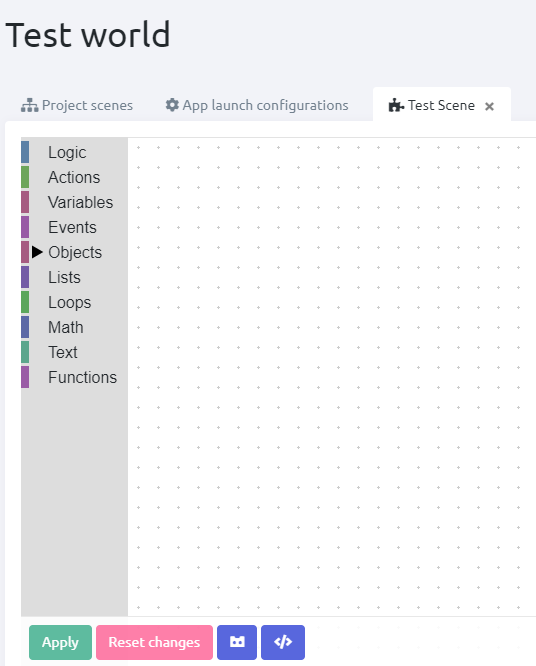
## Further reading

[Custom blocks (official site)](https://developers.google.com/blockly/guides/configure/web/custom-blocks)

# How Blockly works on Varwin platform

* To open Blockly,
  + Open relevant project,
  + Click “Open visual logic editor”.

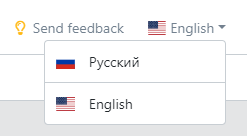




* The new algorithm can be immediately checked in VR. To do it, click ‘Run in VR.’



* Languages: English, Russian.



* Logic set in the script is executed in VR frame by frame, except Function and Event blocks.