

MOUSE

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

The **Mouse** object allows projects to respond to mouse input.

When designing your project, you cannot assume everyone has a mouse. Many users browse the web with touch-screen devices that have no mouse. Therefore if your project uses exclusively mouse or keyboard control, it may be impossible to use on touch devices. See the [Touch controls](#) tutorial for an alternative control system.

If you only use left clicks, consider instead using the [Touch](#) object with *Use Mouse Input* enabled. This will allow your project to work on touchscreen devices without any further changes.

Mouse hardware buttons

Most modern mouse hardware has three buttons: left, middle and right. The middle button usually is also a scroll wheel that allows more conveniently scrolling up and down (and can be detected in Construct with the *On mouse wheel* trigger). However some specialist devices have an additional two buttons, named simply button 4 and button 5, which are usually positioned on the side of the device (but some hardware may differ). You can also detect these buttons with the Mouse object, but note that not all users will have those buttons on their mouse hardware. You could use the additional buttons to provide shortcuts which can be achieved another way, such as with key presses, but you should avoid using them for unique features that cannot be accessed any other way.

Scripting

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

Mouse conditions

Cursor is over object

True if the mouse cursor is hovering over an object.

Mouse button is down

True if a given mouse button is currently being held down.

On any click

Triggered when any mouse button is clicked. This can be useful for title screens or cut-scenes where any input will continue to the next step.

On button released

Triggered when a given mouse button is released.

On click

Triggered when a given mouse button is pressed. This can also be used to detect double-clicks.

Double-clicks only support the primary (left) mouse button.

On mouse wheel

Triggered when the mouse wheel is scrolled up or down a notch. Choose *any* to detect the mouse wheel being scrolled in either direction; in this case the *WheelDeltaX/Y/Z* expressions are useful to determine the direction and magnitude of the scroll.

Some non-mouse devices, like laptop trackpads, can also fire wheel events. These inputs can have different behavior to mouse wheels, such as momentum-based scrolling with slowly reducing delta values.

On object clicked

Triggered when a given mouse button is pressed while the mouse cursor is over an object. This can also be used to detect double-clicks on objects.

Has pointer lock

True if pointer lock is currently active, i.e. after *On pointer locked* triggered.

On movement

Triggered every time the mouse moves, or any touch input moves. The *MovementX* and *MovementY* expressions provide the amount of movement. This can be used without pointer lock, but mouse movement will still be limited by the boundaries of the window or screen. Since this trigger also works with touch input, it means movement input also works with touch input on mobile devices. Usage with touch input does not require pointer lock, since that only affects the mouse cursor.

When using On movement for 3D camera mouse look, the support for touch input also means that the view can be rotated by touch input on mobile.

On pointer locked

Triggered when the pointer is successfully locked after the *Request pointer lock* action. The mouse cursor will disappear and only mouse movement values will be available via the *On movement* trigger.

On pointer unlocked

Triggered when pointer lock is released, restoring the normal mouse cursor behavior. This can happen either by the *Release pointer lock* action, or at any time if the user manually exits pointer lock, such as by pressing **Escape**.

On pointer lock error

Triggered if an error occurs attempting to use pointer lock, such as attempting to acquire pointer lock when security restrictions disallow it.

Mouse actions

Set cursor from sprite

Set the cursor image from a [Sprite](#) object. This is preferable to setting a sprite to the mouse co-ordinates, because the input lag is significantly lower. Various limitations apply: the sprite image is used as it appears in the image editor, not taking in to account size or rotation in the layout; the image cannot be too large (64x64 is usually the limit); the cursor may not be applied close to the edges of the browser window; and support varies depending on browser and OS.

Set cursor style

Set the type of mouse cursor showing. The cursor can be hidden completely by choosing **None**.

Request pointer lock

Request to acquire pointer lock, which hides the mouse cursor and only reports mouse movement values via the *On movement* trigger and *MovementX* and *MovementY* expressions. This allows continuous input without the boundaries of the window or screen stopping movement. This type of input is useful for "mouse look" with the 3D Camera object (see [First-person platformer](#) for an example). Normally this can only be used in a user input trigger, such as *On click*. *On pointer locked* is triggered if the request is successful, but the request may be denied, for example due to security restrictions, in which case *On pointer lock error* will trigger. Also note the user can also exit pointer lock at any time, triggering *On pointer unlocked*. The *Unadjusted movement* parameter can be checked to disable system adjustments to the mouse movement, such as acceleration, instead accessing the raw mouse input.

Release pointer lock

If pointer lock is active, this ends pointer lock and restores the normal mouse cursor behavior.

Mouse expressions

AbsoluteX

AbsoluteY

Return the position of the mouse cursor over the canvas area in the page. This is (0, 0) at the top left of the canvas and goes up to the window size. It is not affected by any scrolling or scaling in the project.

X

Y

Return the position of the mouse cursor in layout co-ordinates. This is (0, 0) at the top left of the layout. It changes to reflect scrolling and scaling. However, if an individual layer has been scrolled, scaled or rotated, these expressions do not take that into account - for that case, use the layer versions below.

X(layer)

Y(layer)

Return the position of the mouse cursor in layer co-ordinates, with scrolling, scaling and rotation taken into account for the given layer. The layer can be identified either by a string of its name or its zero-based index, e.g. `Mouse.X("HUD")`.

WheelDeltaX

WheelDeltaY

WheelDeltaZ

In *On mouse wheel*, these values provide the direction and magnitude of the wheel movement. Typically this will alter *WheelDeltaY* representing vertical movement, with the value being positive or negative depending on whether it is up or down movement. The magnitude can also vary depending on the speed of the movement, the system settings, and the type of device. The other values support other kind of hardware devices, such as an additional horizontal scrolling wheel on some mouse devices, or pan scrolling on a laptop trackpad.

MovementX

MovementY

In the trigger *On movement*, these values provide the amount of horizontal and vertical movement.