KEYBOARD

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

The **Keyboard** object allows projects to respond to keyboard input.

Scripting

When using JavaScript or TypeScript coding, the features of this object can be accessed via the IKeyboardObjectType script interface.

About keyboards

When designing your project, you cannot assume everyone has a keyboard. Many users browse the web with touch-screen devices that have no keyboard. (The Keyboard object also does not respond to input from on-screen keyboards on any modern touch devices.) Therefore if your project uses exclusively mouse or keyboard control, it is impossible to use on touch devices. See the Touch controls tutorial for an alternative control system.

Also note that there are a variety of keyboard layouts used internationally. For example if you only provide "WASD" as direction controls, your project may be difficult to control on AZERTY keyboards. "ZQSD" controls covers the AZERTY layout, but there are many other possible keyboard layouts. In this case, also supporting arrow keys for direction controls will cover most international keyboards, but remember the same problem applies for any other controls depending on a specific key layout.

Key codes

It's possible to detect key presses by numerical *key codes* with the Keyboard object. A key code is simply a number assigned to every possible key on the keyboard. This can be useful for implementing custom controls, since key codes can be stored in variables.

Key ghosting

You may notice you cannot reliably detect three or more simultaneous key presses on the keyboard. This is a limitation of common keyboard hardware, not Construct. The circuitry in common keyboards exhibits an effect called *key ghosting* where only certain combinations of a certain number of keys can be reliably detected. You can get special gamer keyboards that support anti-ghosting, but since these are rare it's probably a better idea to design your project around the limitations of common keyboards, such as by avoiding having to hold down lots of keys.

Keyboard conditions

Key code is down

True if a given key by its key code is currently being held down.

On key code pressed

Triggered when a specific key code is pressed.

On key code released

Triggered when a specific key code is released.

Key is down

True if a given keyboard key is currently being held down.

On any key pressed

Triggered when any keyboard key is pressed. Useful for title screens or cutscenes. The corresponding key code is set in the *LastKeyCode* expression.

On any key released

Triggered when any keyboard key is released. The corresponding key code is set in the *LastKeyCode* expression.

On key pressed

Triggered when a specific keyboard key is pressed.

On key released

Triggered when a specific keyboard key is released.

Is keyboard lock supported

True if the current browser/platform supports using keyboard lock with the *Lock keyboard* action.

On keyboard locked

On keyboard lock error

Triggered after the *Lock keyboard* action depending on whether the attempt to lock the keyboard completed successfully. If the keyboard is now locked, *On keyboard locked* is triggered; if locking the keyboard failed (including the user declining a permission prompt), *On keyboard lock error* is triggered.

Left/right key is down

On left/right key pressed

On left/right key released

As per *Is key down*, *On key pressed* and *On key released*, but is able to identify the left or right Shift, Control, Alt or Meta keys separately.

Keyboard actions

Lock keyboard

Request to lock the keyboard so key presses are always received by the project, rather than the browser/application (such as to activating browser keyboard shortcuts). For example if WASD controls are used to move an object, and the Control key is also used for another purpose, it is easy to accidentally press Ctrl+W, which is a keyboard shortcut to close the current browser tab. If the WASD keys are locked, then Ctrl+W will act as a normal keyboard input to the project and not activate the browser shortcut. The keys to lock are specified by a string of key names separated by commas, e.g. "KeyW,KeyA,KeyS,KeyD" for the WASD keys - a full list of key names can be found on this MDN page. Pass an empty string to request to lock all keys. Requesting to lock the keyboard may show a permission prompt to the user, as it is a security-sensitive feature. Further, it only works in fullscreen mode, i.e. after using *Request fullscreen* in the Browser plugin. If keyboard lock is successfully enabled, *On keyboard locked* triggers; otherwise *On keyboard lock error* triggers.

> *The condition Is keyboard lock supported checks if this action is supported by the current browser/platform.*

Unlock keyboard

If the keyboard was previously locked, this action unlocks it again, restoring the normal handling of key presses (where the browser/application may handle some key presses).

Keyboard expressions

LastKeyCode

Retrieve the key code of the last key press. This is useful in *On any key pressed* or *On any key released* to determine the key code of the key the user pressed, which is useful when setting up custom controls.

StringFromKeyCode

Convert a numerical key code back in to a string representation. For example this turns the key code 65 in to the string "A".

TypedKey

Return the last key press as the character that would have been entered in to a text field. For example when pressing A, this could be "a", "A", "á" or something else, depending on which other keys are held down. If the last key press is not a typed character, like Shift, then the expression is set to the name of the key.