

Package ‘KeyboardSimulator’

January 20, 2025

Type Package

Title Keyboard and Mouse Input Simulation for Windows OS

Version 2.6.2

Date 2023-09-13

Description Control your keyboard and mouse with R code by simulating key presses and mouse clicks. The input simulation is implemented with the Windows API.

License GPL (>= 2) | file LICENSE

Imports Rcpp (>= 1.0.0)

LinkingTo Rcpp

RoxygenNote 7.2.3

Encoding UTF-8

Depends R (>= 2.10)

OS_type windows

URL <https://github.com/ChiHangChen/KeyboardSimulator>

BugReports <https://github.com/ChiHangChen/KeyboardSimulator/issues>

NeedsCompilation yes

Author Jim Chen [aut, cre],
Jeff Keller [aut, ctb],
Garry Hopwood [ctb],
Chieh Hsu [ctb]

Maintainer Jim Chen <jim71183@gmail.com>

Repository CRAN

Date/Publication 2023-09-13 13:02:33 UTC

Contents

KeyboardSimulator-package	2
keybd.press	3
keybd.release	4

keybd.type_string	4
keyboard_value	5
mouse.click	9
mouse.get_cursor	10
mouse.move	10
mouse.release	11
Index	12

KeyboardSimulator-package

Keyboard and Mouse Input Simulation for Windows OS

Description

Control your keyboard and mouse with R code by simulating key presses and mouse clicks. The input simulation is implemented with the Windows API.

Author(s)

Maintainer: Jim Chen <jim71183@gmail.com>

Authors:

- Jeff Keller <github@vtkellers.com> [contributor]

Other contributors:

- Garry Hopwood <garryhopwood@hotmail.com> [contributor]
- Chieh Hsu <tiff103012345@gmail.com> [contributor]

See Also

Useful links:

- <https://github.com/ChiHangChen/KeyboardSimulator>
- Report bugs at <https://github.com/ChiHangChen/KeyboardSimulator/issues>

keybd.press	<i>Simulate Key Press</i>
-------------	---------------------------

Description

Simulate keyboard key presses. Multiple keys can be pressed simultaneously by using + as separator (see Examples). See [keyboard_value](#) for supported keys.

Usage

```
keybd.press(button, hold = FALSE)
```

Arguments

button	character. The key press to simulate (not case sensitive).
hold	logical. Whether the key should be held down. If TRUE, the key can be released by pressing the physical key on the keyboard or by using the keybd.release function.

See Also

[keybd.release](#)

Examples

```
## Not run:  
  
# press one key  
keybd.press("a")  
  
# press multiple keys  
keybd.press("Alt+F4")  
  
# press multiple keys using hold  
keybd.press("Alt", hold = TRUE)  
keybd.press("F4")  
keybd.release("Alt")  
  
## End(Not run)
```

keybd.release *Simulate Key Release*

Description

Simulate the release of keyboard keys held by [keybd.press](#). Multiple keys can be released simultaneously by using a + separator (see Examples). See [keyboard_value](#) for supported keys.

Usage

```
keybd.release(button)
```

Arguments

button character. The key release to simulate (not case sensitive).

See Also

[keybd.press](#)

Examples

```
## Not run:  
  
# Move to the third working window  
keybd.press("Alt", hold = TRUE)  
keybd.press("Tab")  
Sys.sleep(0.1)  
keybd.press("Tab")  
keybd.release("Alt")  
  
## End(Not run)
```

keybd.type_string *Type a raw string*

Description

Type a raw string base on a given string.

Usage

```
keybd.type_string(string)
```

Arguments

string character. The string expected to output (case sensitive).

Examples

```
## Not run:  
  
# Type 'Hello world!'  
keybd.type_string("Hello world!")  
  
## End(Not run)
```

keyboard_value	<i>keyboard_value</i>
----------------	-----------------------

Description

List of supported keyboard keys, along with their virtual-key and hardware scan codes. A field indicating whether the key is an "extended key" with a `0xE0` prefix byte is also included to differentiate between duplicate keys on the keyboard and num pad. For example, while the 1 key usually behaves the same as the 1 key on the num pad, some applications see these as two distinct keys.

Usage

```
keyboard_value
```

Format

An object of class `data.frame` with 137 rows and 5 columns.

Details

Supported keys:

- a
- b
- c
- d
- e
- f
- g
- h
- i
- j
- k
- l
- m

- n
- o
- p
- q
- r
- s
- t
- u
- v
- w
- x
- y
- z
- A
- B
- C
- D
- E
- F
- G
- H
- I
- J
- K
- L
- M
- N
- O
- P
- Q
- R
- S
- T
- U
- V
- W
- X

- Y
- Z
- 0
- 1
- 2
- 3
- 4
- 5
- 6
- 7
- 8
- 9
- num0
- num1
- num2
- num3
- num4
- num5
- num6
- num7
- num8
- num9
- f1
- f2
- f3
- f4
- f5
- f6
- f7
- f8
- f9
- f10
- f11
- f12
- backspace
- tab
- enter

- shift
- ctrl
- alt
- capslock
- esc
- pageup
- pagedown
- end
- home
- left
- up
- right
- down
- insert
- space
- del
- numlock
- win
- ‘
- ~
- !
- @
- #
- \$
- %
- ^
- &
- *
- (
-)
- -
- _
- +
- =
- [
- {
-]

- }
- \
- |
- ;
- :
- '
- "
- ,
- <
- .
- >
- /
- ?

References

[Hardware Scan Codes, Virtual-Key Codes](#)

<code>mouse.click</code>	<i>Simulate Mouse Clicks</i>
--------------------------	------------------------------

Description

Simulate left, right, and middle mouse clicks.

Usage

```
mouse.click(button = "left", hold = FALSE)
```

Arguments

<code>button</code>	character. Allowed values are "left", "right", and "middle".
<code>hold</code>	logical. Whether the button should be held down.

See Also

[mouse.release](#)

Examples

```
## Not run:  
  
# left mouse click  
mouse.click(button = "left")  
  
# left mouse click and hold  
mouse.click(button = "left", hold = TRUE)  
  
## End(Not run)
```

mouse.get_cursor	<i>Get Current Cursor Coordinate</i>
------------------	--------------------------------------

Description

Get current cursor coordinate of screen .

Usage

```
mouse.get_cursor()
```

Examples

```
## Not run:  
  
mouse.get_cursor()  
  
## End(Not run)
```

mouse.move	<i>Move Cursor to Specific Location</i>
------------	---

Description

Move cursor to specific coordinate of screen .

Usage

```
mouse.move(x, y, duration = NA, step_ratio = 0.01)
```

Arguments

x	numeric. X-axis of screen.
y	numeric. Y-axis of screen.
duration	numeric. Cursor movement time in seconds, there might be some delay on different computer.
step_ratio	numeric. Ratio of total distance in each step, only available when duration is not NA.

Examples

```
## Not run:  
  
# Move cursor to middle of screen in 1080FHD monitor  
mouse.move(x = 960, y = 540)  
  
# Move cursor to middle of screen in 1080FHD monitor within 3 seconds  
mouse.move(x = 960, y = 540, duration = 3)  
  
## End(Not run)
```

mouse.release	<i>Simulate Mouse Click Release</i>
---------------	-------------------------------------

Description

Simulate the release of mouse button held by [mouse.click](#).

Usage

```
mouse.release(button = "left")
```

Arguments

button character. Allowed values are "left", "right", and "middle".

See Also

[mouse.click](#)

Examples

```
## Not run:  
  
# right mouse click and hold  
mouse.click(button = "right", hold = TRUE)  
  
# release right click  
mouse.release(button = "right")  
  
## End(Not run)
```

Index

* **data**

 keyboard_value, [5](#)

keybd.press, [3](#), [4](#)

keybd.release, [3](#), [4](#)

keybd.type_string, [4](#)

keyboard_value, [3](#), [4](#), [5](#)

KeyboardSimulator

 (KeyboardSimulator-package), [2](#)

KeyboardSimulator-package, [2](#)

mouse.click, [9](#), [11](#)

mouse.get_cursor, [10](#)

mouse.move, [10](#)

mouse.release, [9](#), [11](#)