pygame documentation

Pygame Home || Help Contents || Reference Index

Color | cursors | display | draw | event | font | image | joystick | key | locals | mask | mixer | mouse | Rect | Surface | sprite | time | transform | BufferProxy | freetype | gfxdraw | midi | Overlay | PixelArray | pixelcopy | sndarray | surfarray | camera | cdrom | examples | math | music | pygame | scrap | tests | version

pygame.key

pygame module to work with the keyboard

pygame.key.get_focused	—	true if the display is receiving keyboard input from the system
pygame.key.get_pressed	—	get the state of all keyboard buttons
pygame.key.get_mods	—	determine which modifier keys are being held
pygame.key.set_mods	—	temporarily set which modifier keys are pressed
pygame.key.set_repeat	—	control how held keys are repeated
pygame.key.get_repeat	—	see how held keys are repeated
pygame.key.name	—	get the name of a key identifier

This module contains functions for dealing with the keyboard.

The event queue gets pygame.KEYDOWN and pygame.KEYUP events when the keyboard buttons are pressed and released. Both events have a key attribute that is a integer ID representing every key on the keyboard.

The pygame.KEYDOWN event has additional attributes unicode and scancode. unicode represents a single character string that is the fully translated character entered. This takes into account the shift and composition keys. scancode represents the platform-specific key code. This could be different from keyboard to keyboard, but is useful for key selection of weird keys like the multimedia keys.

There are many keyboard constants, they are used to represent keys on the keyboard. The following is a list of all keyboard constants:

KeyASCII	ASCII	Common Name
K_BACKSPACE	\b	backspace
K_TAB	\t	tab
K_CLEAR		clear
K_RETURN	\r	return
K_PAUSE		pause
K_ESCAPE	^[	escape
K_SPACE		space
K_EXCLAIM	!	exclaim
K_QUOTEDBL	"	quotedbl
K_HASH	#	hash
K_DOLLAR	$	dollar
K_AMPERSAND	&	ampersand
K_QUOTE		quote
K_LEFTPAREN	(	left parenthesis
K_RIGHTPAREN	)	right parenthesis
K_ASTERISK	*	asterisk
K_PLUS	+	plus sign
K_COMMA	,	comma
K_MINUS	-	minus sign
K_PERIOD	.	period
K_SLASH	/	forward slash
K_0	0	0

K_0	0	0
K_1	1	1
K_2	2	2
K_3	3	3
K_4	4	4
K_5	5	5
K_6	6	6
K_7	7	7
K_8	8	8
K_9	9	9
K_COLON	:	colon
K_SEMICOLON	;	semicolon
K_LESS	<	less-than sign
K_EQUALS	=	equals sign
K_GREATER	>	greater-than sign
K_QUESTION	?	question mark
K_AT	@	at
K_LEFTBRACKET	[	left bracket
K_BACKSLASH	\	backslash
K_RIGHTBRACKET	]	right bracket
K_CARET	^	caret
K_UNDERSCORE	_	underscore
K_BACKQUOTE	`	grave
K_a	a	a
K_b	b	b
K_c	c	c
K_d	d	d
K_e	e	e
K_f	f	f
K_g	g	g
K_h	h	h
K_i	i	i
K_j	j	j
K_k	k	k
K_l	l	l
K_m	m	m
K_n	n	n
K_o	o	o
K_p	p	p
K_q	q	q
K_r	r	r
K_s	s	s
K_t	t	t
K_u	u	u
K_v	v	v
K_w	w	w
K_x	x	x
K_y	y	y
K_z	z	z
K_DELETE		delete
K_KP0		keypad 0
K_KP1		keypad 1
K_KP2		keypad 2
K_KP3		keypad 3
K_KP4		keypad 4
K_KP5		keypad 5
K_KP6		keypad 6
K_KP7		keypad 7
K_KP8		keypad 8
K_KP9		keypad 9
K_KP_PERIOD	.	keypad period
K_KP_DIVIDE	/	keypad divide
K_KP_MULTIPLY	*	keypad multiply
K_KP_MINUS	-	keypad minus
K_KP_PLUS	+	keypad plus
K_KP_ENTER	\r	keypad enter

K_KP_EQUALS	=	keypad equals
K_UP		up arrow
K_DOWN		down arrow
K_RIGHT		right arrow
K_LEFT		left arrow
K_INSERT		insert
K_HOME		home
K_END		end
K_PAGEUP		page up
K_PAGEDOWN		page down
K_F1		F1
K_F2		F2
K_F3		F3
K_F4		F4
K_F5		F5
K_F6		F6
K_F7		F7
K_F8		F8
K_F9		F9
K_F10		F10
K_F11		F11
K_F12		F12
K_F13		F13
K_F14		F14
K_F15		F15
K_NUMLOCK		numlock
K_CAPSLOCK		capslock
K_SCROLLOCK		scrollock
K_RSHIFT		right shift
K_LSHIFT		left shift
K_RCTRL		right control
K_LCTRL		left control
K_RALT		right alt
K_LALT		left alt
K_RMETA		right meta
K_LMETA		left meta
K_LSUPER		left Windows key
K_RSUPER		right Windows key
K_MODE		mode shift
K_HELP		help
K_PRINT		print screen
K_SYSREQ		sysrq
K_BREAK		break
K_MENU		menu
K_POWER		power
K_EURO		Euro

The keyboard also has a list of modifier states that can be assembled by bitwise-ORing them together.

```
KMOD_NONE, KMOD_LSHIFT, KMOD_RSHIFT, KMOD_SHIFT, KMOD_CAPS,
KMOD_LCTRL, KMOD_RCTRL, KMOD_CTRL, KMOD_LALT, KMOD_RALT,
KMOD_ALT, KMOD_LMETA, KMOD_RMETA, KMOD_META, KMOD_NUM, KMOD_MODE
```

pygame.key.get_focused()

true if the display is receiving keyboard input from the system

```
get_focused() -> bool
```

This is true when the display window has keyboard focus from the system. If the display needs to ensure it does not lose keyboard focus, it can use pygame.event.set_grab() to grab all input.

Search examples for pygame.key.get_focused | **Add a Comment** | **Comments 2**

pygame.key.get_pressed()

get the state of all keyboard buttons

`get_pressed() -> bools`

Returns a sequence of boolean values representing the state of every key on the keyboard. Use the key constant values to index the array. A True value means the that button is pressed.

Getting the list of pushed buttons with this function is not the proper way to handle text entry from the user. You have no way to know the order of keys pressed, and rapidly pushed keys can be completely unnoticed between two calls to pygame.key.get_pressed(). There is also no way to translate these pushed keys into a fully translated character value. See the pygame.KEYDOWN events on the event queue for this functionality.

Search examples for pygame.key.get_pressed | **Add a Comment** | **Comments 6**

pygame.key.get_mods()

determine which modifier keys are being held

`get_mods() -> int`

Returns a single integer representing a bitmask of all the modifier keys being held. Using bitwise operators you can test if specific shift keys are pressed, the state of the capslock button, and more.

Search examples for pygame.key.get_mods | **Add a Comment** | **Comments 6**

pygame.key.set_mods()

temporarily set which modifier keys are pressed

`set_mods(int) -> None`

Create a bitmask of the modifier constants you want to impose on your program.

Search examples for pygame.key.set_mods | **Add a Comment**

pygame.key.set_repeat()

control how held keys are repeated

`set_repeat() -> None`

`set_repeat(delay, interval) -> None`

When the keyboard repeat is enabled, keys that are held down will generate multiple pygame.KEYDOWN events. The delay is the number of milliseconds before the first repeated pygame.KEYDOWN will be sent. After that another pygame.KEYDOWN will be sent every interval milliseconds. If no arguments are passed the key repeat is disabled.

When pygame is initialized the key repeat is disabled.

Search examples for pygame.key.set_repeat | **Add a Comment** | **Comments 9**

pygame.key.get_repeat()

see how held keys are repeated

```
get_repeat() -> (delay, interval)
```

When the keyboard repeat is enabled, keys that are held down will generate multiple pygame.KEYDOWN events. The delay is the number of milliseconds before the first repeated pygame.KEYDOWN will be sent. After that another pygame.KEYDOWN will be sent every interval milliseconds.

When pygame is initialized the key repeat is disabled.

New in pygame 1.8.

Search examples for pygame.key.get_repeat **Add a Comment**

pygame.key.name()

get the name of a key identifier

```
name(key) -> string
```

Get the descriptive name of the button from a keyboard button id constant.

Search examples for pygame.key.name **Add a Comment** **Comments 7**