mouse_event function

12/05/2018 5 minutes to read

In this article

The **mouse_event** function synthesizes mouse motion and button clicks.

Note This function has been superseded. Use **SendInput** instead.

Syntax

Copy

```
void mouse_event(
  DWORD     dwFlags,
  DWORD     dx,
  DWORD     dy,
  DWORD     dwData,
  ULONG_PTR dwExtraInfo
);
```

Parameters

`dwFlags`

Type: **DWORD**

Controls various aspects of mouse motion and button clicking. This parameter can be certain combinations of the following values.

Value	Meaning
MOUSEEVENTF_ABSOLUTE	The *dx* and *dy* parameters contain normalized absolute coordinates. If not set, those parameters contain relative

0x8000	data: the change in position since the last reported position. This flag can be set, or not set, regardless of what kind of mouse or mouse-like device, if any, is connected to the system. For further information about relative mouse motion, see the following Remarks section.
MOUSEEVENTF_LEFTDOWN 0x0002	The left button is down.
MOUSEEVENTF_LEFTUP 0x0004	The left button is up.
MOUSEEVENTF_MIDDLEDOWN 0x0020	The middle button is down.
MOUSEEVENTF_MIDDLEUP 0x0040	The middle button is up.
MOUSEEVENTF_MOVE 0x0001	Movement occurred.
MOUSEEVENTF_RIGHTDOWN 0x0008	The right button is down.
MOUSEEVENTF_RIGHTUP 0x0010	The right button is up.
MOUSEEVENTF_WHEEL 0x0800	The wheel has been moved, if the mouse has a wheel. The amount of movement is specified in *dwData*
MOUSEEVENTF_XDOWN 0x0080	An X button was pressed.
MOUSEEVENTF_XUP	An X button was released.

0x0100	
MOUSEEVENTF_WHEEL 0x0800	The wheel button is rotated.
MOUSEEVENTF_HWHEEL 0x01000	The wheel button is tilted.

The values that specify mouse button status are set to indicate changes in status, not ongoing conditions. For example, if the left mouse button is pressed and held down, **MOUSEEVENTF_LEFTDOWN** is set when the left button is first pressed, but not for subsequent motions. Similarly, **MOUSEEVENTF_LEFTUP** is set only when the button is first released.

You cannot specify both **MOUSEEVENTF_WHEEL** and either **MOUSEEVENTF_XDOWN** or **MOUSEEVENTF_XUP** simultaneously in the *dwFlags* parameter, because they both require use of the *dwData* field.

`dx`

Type: **DWORD**

The mouse's absolute position along the x-axis or its amount of motion since the last mouse event was generated, depending on the setting of **MOUSEEVENTF_ABSOLUTE**. Absolute data is specified as the mouse's actual x-coordinate; relative data is specified as the number of mickeys moved. A *mickey* is the amount that a mouse has to move for it to report that it has moved.

`dy`

Type: **DWORD**

The mouse's absolute position along the y-axis or its amount of motion since the last mouse event was generated, depending on the setting of **MOUSEEVENTF_ABSOLUTE**. Absolute data is specified as the mouse's actual y-coordinate; relative data is specified as the number of mickeys moved.

`dwData`

Type: **DWORD**

If *dwFlags* contains **MOUSEEVENTF_WHEEL**, then *dwData* specifies the amount of wheel movement. A positive value indicates that the wheel was rotated forward, away from the user; a negative value indicates that the wheel was rotated backward, toward the user. One wheel click is defined as **WHEEL_DELTA**, which is 120.

If *dwFlags* contains **MOUSEEVENTF_HWHEEL**, then *dwData* specifies the amount of wheel movement. A positive value indicates that the wheel was tilted to the right; a negative value indicates that the wheel was tilted to the left.

If *dwFlags* contains **MOUSEEVENTF_XDOWN** or **MOUSEEVENTF_XUP**, then *dwData* specifies which X buttons were pressed or released. This value may be any combination of the following flags.

If *dwFlags* is not **MOUSEEVENTF_WHEEL**, **MOUSEEVENTF_XDOWN**, or **MOUSEEVENTF_XUP**, then *dwData* should be zero.

Value	Meaning
XBUTTON1 0x0001	Set if the first X button was pressed or released.
XBUTTON2 0x0002	Set if the second X button was pressed or released.

`dwExtraInfo`

Type: **ULONG_PTR**

An additional value associated with the mouse event. An application calls GetMessageExtraInfo to obtain this extra information.

Return Value

This function has no return value.

Remarks

If the mouse has moved, indicated by **MOUSEEVENTF_MOVE** being set, *dx* and *dy* hold information about that motion. The information is specified as absolute or relative integer values.

If **MOUSEEVENTF_ABSOLUTE** value is specified, *dx* and *dy* contain normalized absolute coordinates between 0 and 65,535. The event procedure maps these coordinates onto the display surface. Coordinate (0,0) maps onto the upper-left corner of the display surface, (65535,65535) maps onto the lower-right corner.

If the **MOUSEEVENTF_ABSOLUTE** value is not specified, *dx* and *dy* specify relative motions from when the last mouse event was generated (the last reported position). Positive values mean the mouse moved right (or down); negative values mean the mouse moved left (or up).

Relative mouse motion is subject to the settings for mouse speed and acceleration level. An end user sets these values using the Mouse application in Control Panel. An application obtains and sets these values with the SystemParametersInfo function.

The system applies two tests to the specified relative mouse motion when applying acceleration. If the specified distance along either the x or y axis is greater than the first mouse threshold value, and the mouse acceleration level is not zero, the operating system doubles the distance. If the specified distance along either the x- or y-axis is greater than the second mouse threshold value, and the mouse acceleration level is equal to two, the operating system doubles the distance that resulted from applying the first threshold test. It is thus possible for the operating system to multiply relatively-specified mouse motion along the x- or y-axis by up to four times.

Once acceleration has been applied, the system scales the resultant value by the desired mouse speed. Mouse speed can range from 1 (slowest) to 20 (fastest) and represents how much the pointer moves based on the distance the mouse moves. The default value is 10, which results in no additional modification to the mouse motion.

The **mouse_event** function is used to synthesize mouse events by applications that need to do so. It is also used by applications that need to obtain more information from the mouse than its position and button state. For example, if a tablet manufacturer wants to pass pen-based information to its own applications, it can write a DLL that communicates directly to the tablet hardware, obtains the extra information, and saves it in a queue. The DLL then calls **mouse_event** with the standard button and x/y position data, along with, in the *dwExtraInfo* parameter, some pointer or index to the queued extra information. When the application needs the extra information, it calls the DLL with the pointer or index stored in *dwExtraInfo*, and the DLL returns the extra information.

Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Target Platform	Windows
Header	winuser.h (include Windows.h)
Library	User32.lib
DLL	User32.dll

See Also

Conceptual

GetMessageExtraInfo

Mouse Input

Other Resources

Reference

SystemParametersInfo