MOUSE

The mouse is a pointing device with one or more buttons that allows users to interact with computers.



Despite the emergence of alternative input devices like touchscreens and light pens, the mouse remains the primary input device for desktop computers.



This is due to its versatility, ease of use, and affordability.

Windows 98 supports mice with one, two, or three buttons. It can also use joysticks or light pens to mimic mouse input. In the early days of Windows, applications were designed to work with one-button mice, as many users didn't have two-button mice.



However, two-button mice have become the standard, and most applications now utilize the second button for various functions, such as invoking context menus or performing special drag operations.

Determining Mouse Presence and Button Count

To determine if a mouse is present, you can use the GetSystemMetrics function with the SM\_MOUSEPRESENT parameter.

However, this function always returns TRUE in Windows 98, regardless of whether a mouse is attached. To get accurate information, use this function in Microsoft Windows NT.

To determine the number of buttons on the installed mouse, use the GetSystemMetrics function with the SM\_CMOUSEBUTTONS parameter.

This function should also return 0 if a mouse is not installed. However, in Windows 98, the function returns 2 if a mouse is not installed.

Left-Handed Mouse Users

Left-handed users can switch the mouse buttons using the Windows Control Panel. While an application can determine this by calling GetSystemMetrics with the SM\_SWAPBUTTON parameter, this is usually unnecessary. The button triggered by the index finger is considered the left button, even if it's physically on the right side of the mouse.

Setting Mouse Parameters

You can set other mouse parameters, such as the double-click speed, using the SystemParametersInfo function. This function allows you to set or obtain various mouse-related settings from within your Windows application.

Fun facts:

* The mouse cursor is a small bitmapped picture that moves on the display as the user moves the mouse.
* The hot spot is the single-pixel point on the cursor that indicates the precise location on the display.
* Windows supports several predefined mouse cursors, such as IDC\_ARROW, IDC\_CROSS, and IDC\_WAIT.
* Programmers can also design their own custom cursors.
* The default cursor for a particular window is specified when defining the window class structure.
* Common mouse actions include clicking, double-clicking, and dragging.
* On a three-button mouse, the buttons are called the left button, the middle button, and the right button.
* On a two-button mouse, there is only a left button and a right button.
* The single button on a one-button mouse is a left button.
* The plural of "mouse" is a matter of debate, with both "mice" and "mouses" being considered acceptable.
* The Microsoft Manual of Style for Technical Publications recommends avoiding the plural "mice" and using "mouse devices" instead.

Overview

Client-area mouse messages are notifications sent by Windows to a window's procedure when mouse events occur within the window's client area. These messages provide information about the mouse's position, button state, and modifier keys.

Mouse Messages vs. Keyboard Messages

Unlike keyboard messages, which are only sent to the window that has the input focus, mouse messages are sent to any window that the mouse cursor passes over or clicks within, regardless of whether the window is active or has the input focus. This allows windows to respond to mouse interactions even when they are not in the foreground.

Types of Mouse Messages

Windows defines 21 mouse messages, but only 10 of them relate to the client area. These messages can be categorized into three types:

* Mouse movement: The WM\_MOUSEMOVE message is sent when the mouse cursor moves within the client area.
* Button press/release: When a mouse button is pressed or released within the client area, the window procedure receives one of the messages shown in the table in the text.
* Double-click: Double-click messages are sent only if the window class has been defined to receive them.

Extracting Mouse Position

The value of lParam in the client-area mouse messages contains the position of the mouse cursor. The low word is the x-coordinate, and the high word is the y-coordinate, both relative to the upper-left corner of the client area. These values can be extracted using the LOWORD and HIWORD macros.

Extracting Mouse Button State and Modifier Keys

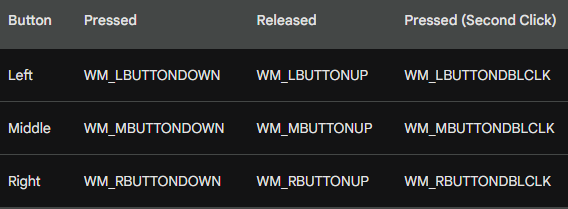
The value of wParam in the client-area mouse messages indicates the state of the mouse buttons and the Shift and Ctrl keys. These states can be tested using the bit masks defined in the WINUSER.H header file, which have the prefix "MK" for "mouse key".

WM\_LBUTTONDOWN Message and Active Window

Clicking the left mouse button in the client area of an inactive window causes Windows to make the clicked window active and then send the WM\_LBUTTONDOWN message to the window procedure. This allows the window to respond to the click even if it was not previously active.

WM\_LBUTTONDOWN and WM\_LBUTTONUP Messages

A window procedure may receive a WM\_LBUTTONDOWN message without a corresponding WM\_LBUTTONUP message, or vice versa. This can happen if the mouse button is pressed or released outside the window's client area.



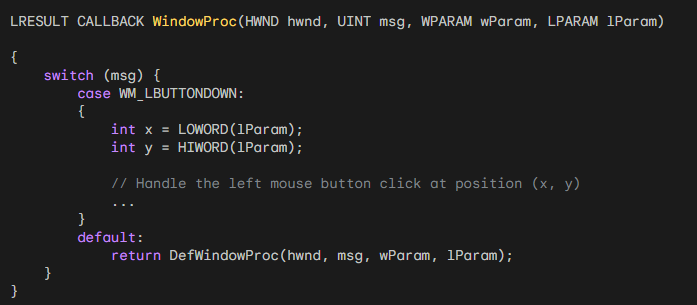
Mouse Capture

A window procedure can capture the mouse and continue to receive mouse messages even when the mouse is outside the window's client area. This is useful for operations that require continuous mouse tracking, such as drawing or dragging.

System Modal Message Boxes and Dialog Boxes

When a system modal message box or dialog box is on the display, no other program can receive mouse messages. These modal boxes prevent switching to another window while they are active.

Here's an example of how to handle the WM\_LBUTTONDOWN message in C code using WinAPI:



This code snippet defines a window procedure function called WindowProc that handles the WM\_LBUTTONDOWN message. When the left mouse button is pressed within the window's client area, the function extracts the mouse coordinates (x, y) from the lParam parameter and performs the corresponding action.

Connect folder in Chapter 7 has the code. Here’s the video for it’s working…

