链滴

Win32 汇编学习 (7)：鼠标输入消息

作者：Akkuman
原文链接：https://ld246.com/article/1518197054735
来源网站：链滴

这次我们将学习如何在我们的窗口过程函数中处理鼠标按键消息。例子演示了如何等待鼠标左键按下息，我们将在按下的位置显示一个字符串。

<!--more-->

理论：

和处理键盘输入一样，WINDOWS将捕捉鼠标动作并把它们发送到相关窗口。这些活动包括左、右键下、移动、双击、滚轮消息WM_WHEEL等。WINDOWS并不像处理键盘输入那样把所有的鼠标消息导向有输入焦点的窗口，**任何鼠标经过的窗口都将接收到鼠标消息，无论有否输入焦点**。另外，窗口会接收到鼠标在非客户区移动的消息（WM_NCMOVE），但大多数的情况下我们都会将其忽略掉。鼠标在某窗口客户区移动时，该窗口将接收到WM_MOUSEMOVE消息。一个窗口若想处理WM_LBUTONDBCLK或 WM_RBUTTONDBCLK，那么它的窗口类必须有CS_DBLCLKS风格，否则它就会接受一堆的按键起落（WM_XBUTTONDOWN或WM_XBUTTONUP)的消息。 对于所有的消息，**窗口过函数传入的参数lParam包含了鼠标的位置，其中低位为x坐标，高位为y坐标**，这些坐标值都是相对窗口客户区的左上角的值，wParam中则包含了鼠标按钮的状态。

例子：

```
.386
.model flat,stdcall
option casemap:none
WinMain proto :DWORD,:DWORD,:DWORD,:DWORD

include \masm32\include\windows.inc
include \masm32\include\user32.inc
include \masm32\include\kernel32.inc
include \masm32\include\gdi32.inc
includelib \masm32\lib\user32.lib
includelib \masm32\lib\kernel32.lib
includelib \masm32\lib\gdi32.lib

.data
ClassName db "SimpleWinClass",0
AppName  db "Our First Window",0
MouseClick db 0

.data?
hInstance HINSTANCE ?
CommandLine LPSTR ?
hitpoint POINT <>

.code
start:
    invoke GetModuleHandle, NULL
    mov    hInstance,eax
    invoke GetCommandLine
    mov CommandLine,eax
    invoke WinMain, hInstance,NULL,CommandLine, SW_SHOWDEFAULT
    invoke ExitProcess,eax

WinMain proc hInst:HINSTANCE,hPrevInst:HINSTANCE,CmdLine:LPSTR,CmdShow:DWORD
```

```
    LOCAL wc:WNDCLASSEX
    LOCAL msg:MSG
    LOCAL hwnd:HWND
    mov   wc.cbSize,SIZEOF WNDCLASSEX
    mov   wc.style, CS_HREDRAW or CS_VREDRAW
    mov   wc.lpfnWndProc, OFFSET WndProc
    mov   wc.cbClsExtra,NULL
    mov   wc.cbWndExtra,NULL
    push  hInst
    pop   wc.hInstance
    mov   wc.hbrBackground,COLOR_WINDOW+1
    mov   wc.lpszMenuName,NULL
    mov   wc.lpszClassName,OFFSET ClassName
    invoke LoadIcon,NULL,IDI_APPLICATION
    mov   wc.hIcon,eax
    mov   wc.hIconSm,eax
    invoke LoadCursor,NULL,IDC_ARROW
    mov   wc.hCursor,eax
    invoke RegisterClassEx, addr wc
    invoke CreateWindowEx,NULL,ADDR ClassName,ADDR AppName,\
          WS_OVERLAPPEDWINDOW,CW_USEDEFAULT,\
          CW_USEDEFAULT,CW_USEDEFAULT,CW_USEDEFAULT,NULL,NULL,\
          hInst,NULL
    mov   hwnd,eax
    invoke ShowWindow, hwnd,SW_SHOWNORMAL
    invoke UpdateWindow, hwnd
    .WHILE TRUE
              invoke GetMessage, ADDR msg,NULL,0,0
              .BREAK .IF (!eax)
              invoke DispatchMessage, ADDR msg
    .ENDW
    mov     eax,msg.wParam
    ret
WinMain endp

WndProc proc hWnd:HWND, uMsg:UINT, wParam:WPARAM, lParam:LPARAM
    LOCAL hdc:HDC
    LOCAL ps:PAINTSTRUCT

    .IF uMsg==WM_DESTROY
       invoke PostQuitMessage,NULL
    .ELSEIF uMsg==WM_LBUTTONDOWN
       mov eax,lParam
       and eax,0FFFFh
       mov hitpoint.x,eax
       mov eax,lParam
       shr eax,16
       mov hitpoint.y,eax
       mov MouseClick,TRUE
       invoke InvalidateRect,hWnd,NULL,TRUE
    .ELSEIF uMsg==WM_PAINT
       invoke BeginPaint,hWnd, ADDR ps
       mov    hdc,eax
       .IF MouseClick
```

```
            invoke lstrlen,ADDR AppName
            invoke TextOut,hdc,hitpoint.x,hitpoint.y,ADDR AppName,eax
        .ENDIF
        invoke EndPaint,hWnd, ADDR ps
    .ELSE
        invoke DefWindowProc,hWnd,uMsg,wParam,lParam
        ret
    .ENDIF
    xor    eax,eax
    ret
WndProc endp
end start
```

分析：

```
.ELSEIF uMsg==WM_LBUTTONDOWN
    mov eax,lParam
    and eax,0FFFFh
    mov hitpoint.x,eax
    mov eax,lParam
    shr eax,16
    mov hitpoint.y,eax
    mov MouseClick,TRUE
    invoke InvalidateRect,hWnd,NULL,TRUE
```

窗口过程处理了WM_LBUTTONDOWN消息，当接收到该消息时，lParam中包含了相对于窗口客户左上角的坐标，我们把它保存下来，放到一个结构体变量（POINT）中，该结构体变量的定义如下：

```
POINT STRUCT
    x   dd ?
    y   dd ?
POINT ENDS
```

然后我们把标志量MouseClick设为TRUE，这表明至少有一次在客户区的左键按下消息。

```
        mov eax,lParam
        and eax,0FFFFh
        mov hitpoint.x,eax
```

由于lParam是一个32位长的数，其中高、低16位分别包括了y、x坐标所以我们做一些小处理，以便存它们。

```
        shr eax,16
        mov hitpoint.y,eax
```

保存完坐标后我们设标志MouseClick为TRUE，这是在处理WM_PAINT时用来判断是否有鼠标左键下消息。然后我们调用InvalidateRect函数迫使WINDOWS重新绘制客户区。

```
        .IF MouseClick
            invoke lstrlen,ADDR AppName
            invoke TextOut,hdc,hitpoint.x,hitpoint.y,ADDR AppName,eax
        .ENDIF
```

绘制客户区的代码首先检测MouseClick标志位，再决定是否重绘。因为我们在首次显示窗口时还没

左键按下的消息，所以我们在初始时把该标志设为FALSE，告诉WINDOWS不要重绘客户区，当有左按下的消息时，它会在鼠标按下的位置绘制字符串。注意在调用TextOut函数时，其关于字符串长度参数是调用lstrlen函数来计算的。