**THIẾT BỊ CHUỘT TRONG LẬP TRÌNH C TRÊN WINDOWS**

Trong một môi trường giao tiếp đồ họa, việc sử dụng thiết bị định vị chuột là hết sức cần thiết. Nhờ thiết bị chuột ta có thể di chuyển đến một điểm bất kì trên khung màn hình và thực hiện thao tác kích các nút chuột. Đối với một số ứng dụng đồ họa không quá phức tạp thì công cụ để thực hiện chủ yếu là chuột, riêng đối với các ứng dụng đòi hỏi sự phức tạp khi vẽ thì người ta dùng bút vẽ.

Với Windows thì thiết bị chuột là một thành phần quan trọng, nếu chúng ta bỏ thiết bị chuột thì vẫn khởi động bình thường và các chương trình vẫn có thể chạy được. Tuy nhiên khi đó ta sẽ lúng túng nhiều khi xử lý các ứng dụng trực quan tương tác với người dùng theo tọa độ hay định vị.

Cũng tương tự như bàn phím, thiết bị chuột cũng được dùng để nhập dữ liệu từ người dùng vào ứng dụng nhưng dữ liệu đây không phải là văn bản như khi nhập từ bàn phím mà là các thao tác vẽ hay xử lý các đối tượng đồ họa.

**Cơ bản về thiết bị chuột trong Windows**

Về cơ bản Windows hỗ trợ các loại thiết bị chuột có một nút, hai và ba nút, ngoài ra Windows còn có thể dùng thiết bị khác như joystick hay bút vẽ để bắt chước thiết bị chuột. Các ứng dụng trong Windows thường né việc dùng các nút thứ hai và nút thứ ba, để có thể sử dụng tốt trên các thiết bị chuột chỉ có một nút. Thông thường nút thứ hai của thiết bị chuột phải được dùng cho chức năng gọi thực đơn ngữ cảnh (*context menu*).

Theo nguyên tắc, ta có thể kiểm tra xem có tồn tại thiết bị chuột hay không bằng cách dùng hàm **GetSystemMetrics**.

fMouse = **GetSystemMetrics**( **SM\_MOUSEPRESENT**);

Giá trị trả về fMouse là **TRUE**(1) nếu có thiết bị chuột được cài đặt, và ngược lại bằng **FALSE**(0) nếu thiết bị chuột không được cài đặt vào máy.

Ta cũng có thể kiểm tra số nút của thiết bị chuột bằng cách dùng hàm trên theo câu lệnh sau:

nButton = **GetSystemMetrics**( **SM\_CMOUSEBUTTONS**);

Giá trị trả về là 0 nếu không cài đặt thiết bị chuột, tuy nhiên dưới Windows 98 giá trị trả về là 2 nếu chưa cài đặt thiết bị chuột.

Khi người dùng di chuyển thiết bị chuột thì Windows cũng di chuyển một ảnh bitmap nhỏ trên màn hình, ảnh này được gọi là con trỏ chuột (*mouse cursor*). Con trỏ chuột này có một điểm gọi là điểm nóng (*hot spot*) , điểm nóng này nằm trong hình bitmap, và do người tạo con trỏ chuột quyết định. Khi chúng ta tham chiếu tới một vị trí của con trỏ chuột thì chính là chúng ta đang dùng tọa độ của điểm nóng này.

Windows hỗ trợ một số các con trỏ chuột chuẩn, mà người lập trình chỉ cần gọi ra và dùng chứ không cần phải tạo các tập tin **\*.cur**. Ở trạng thái bình thường thì một ứng dụng trong Windows thường dùng con trỏ chuột **IDC\_ARROW** được định nghĩa trong**WINUSER.H**, điểm nóng chính là điểm trên cùng của mũi tên. Ngoài ra con trỏ chuột còn cho biết trạng thái của ứng dụng, ví dụ một ứng dụng đang xử lý cần một khoảng thời gian thì con trỏ chuột xuất hiện hình đồng hồ cát (**IDC\_WAIT**). Khi đó hầu như các chức năng giao tiếp với người dùng điều phải hoãn đến khi xử lý hoàn thành.

Trong lớp cửa sổ ta có trường định nghĩa con trỏ chuột cho ứng dụng như sau

wndclass.hCursor = **LoadCursor**( NULL, IDC\_ARROR);

Với thiết bị chuột ta có thể có các hành động như sau:

* *Kích chuột*: nhấn và thả một nút chuột.
* *Kích đúp chuột*: nhấn và thả chuột nhanh (nhấn 2 lần nhanh).
* *Kéo*: di chuyển chuột trong khi vẫn nắm giữ một nút.

**Xử lý các thông điệp từ thiết bị chuột**

Các thông điệp được tạo từ chuột rất khác với thông điệp của bàn phím, một thủ tục cửa sổ sẽ nhận thông điệp chuột bất cứ khi nào thiết bị chuột di chuyển qua cửa sổ hay kích vào trong cửa sổ, thậm chí cả trong trường hợp cửa sổ không được kích hoạt hay không nhận được sự quan tâm. Windows định nghĩa 21 thông điệp được phát sinh từ thiết bị chuột. Trong số đó có đến 11 thông điệp không liên quan đến vùng làm việc (*client area*). Những thông điệp không phải vùng làm việc (*nonclient-area messages*) thường được các ứng dụng bỏ qua không xử lý.

**Thông điệp chuột trong vùng làm việc**

Khi con trỏ chuột di chuyển vào vùng làm việc của một cửa sổ, thủ tục cửa sổ của nó sẽ nhận được thông điệp WM\_MOUSEMOVE. Bảng sau mô tả các thao tác làm việc với thiết bị chuột và các thông điệp phát sinh từ nó.

|  |  |  |  |
| --- | --- | --- | --- |
| Mô tả thông điệp kích thiết bị chuột | | | |
| Nút | Nhấn | Thả | Nhấn đúp |
| Trái | WM\_LBUTTONDOWN | WM\_LBUTTONUP | WM\_LBUTTONDBLCLK |
| Giữa | WM\_MBUTTONDOWN | WM\_MBUTTONUP | WM\_MBUTTONDBLCLK |
| Phải | WM\_RBUTTONDOWN | WM\_MBUTTONUP | WM\_RBUTTONDBLCLK |

Thủ tục cửa sổ của ứng dụng sẽ nhận được thông điệp của nút chuột giữa nếu máy tính cài thiết bị chuột có 3 nút. Tương tự như vậy với thông điệp thiết bị chuột phải, chúng ta cần có thiết bị chuột dùng 2 nút. Để nhận được thông điệp kích đúp thiết bị chuột thủ tục cửa sổ phải khai báo nhận thông điệp này.

Wndclass.style = CS\_HREDRAW|CS\_VREDRAW|CS\_DBLCLKS;

Trong thông điệp phát sinh từ thiết bị chuột thì tham số lParam sẽ chứa vị trí của thiết bị chuột, 16 byte thấp sẽ chứa giá trị tọa độ x, còn 16 byte cao sẽ chứa giá trị tọa độ của y. Để lấy ra hai giá trị này ta có thể dùng macro là LOWORD và HIWORD.

Giá trị wParam sẽ cho biết trạng thái của nút nhấn, phím **Shift**, và phím **Ctrl**. Chúng ta có thể kiểm tra các trạng thái này bằng cách dùng bit mặt nạ được định nghĩa trước trong **WINUSER.H**. Các mặt nạ này được bắt đầu bằng tiền tố **MK\_xxx** (**Mouse Key**)

|  |  |
| --- | --- |
| Mô tả trạng thái của chuột và phím nhấn | |
| MK\_LBUTTON | Nút chuột trái nhấn |
| MK\_MBUTTON | Nút chuột giữa nhấn |
| MK\_RBUTTON | Nút chuột phải nhấn |
| MK\_SHIFT | Phím **Shift**được nhấn |
| MK\_CONTROL | Phím **Ctrl**được nhấn |

Ví dụ khi nhận được thông điệp WM\_LBUTTONDOWN chúng ta muốn kiểm tra xem phím **Ctrl** có được nhấn hay không bằng cách so giá trị wParam với mặt nạ MK\_CONTROL.

...

**if**(wParam & **MK\_CONTROL**)

{

/\* Có giữ phím control \*/

}

else

{

/\* Không giữ phím control \*/

}

Như chúng ta đã biết khi di chuyển thiết bị chuột qua vùng làm việc thì thông điệp WM\_MOUSEMOVE sẽ được gởi đến cho thủ tục cửa sổ đó. Nhưng Windows không phát sinh thông điệp này cho từng pixel trên màn hình mà tuỳ thuộc vào thông số phần cứng của thiết bị chuột được cài đặt và tốc độ làm việc của nó.

Khi chúng ta kích nút chuột trái vào vùng làm việc của một cửa sổ không kích hoạt (*inactive window*) thì Windows sẽ kích hoạt cửa sổ này tức là cửa sổ vừa được kích sau đó truyền thông điệp **WM\_LBUTTONDOWN** vào thủ tục **WndProc**của cửa sổ. Khi một cửa sổ nhận được thông điệp **WM\_XXXDOWN** thì không nhất thiết phải nhận được thông điệp **WM\_XXXUP** hay ngược lại. Điều này được giải thích như sau, khi người dùng kích trái vào một cửa sổ và giữ luôn nút chuột vừa kích rồi kéo thiết bị chuột đến một vùng thuộc phạm vi của cửa sổ khác mới thả. Khi đó cửa sổ đầu tiên sẽ nhận được thông điệp nhấn chuột **WM\_LBUTTONDOWN** và cửa sổ thứ hai sẽ nhận được thông điệp nhả thiết bị chuột **WM\_LBUTTONUP**. Tuy nhiên tình huống trên sẽ không xuất hiện với hai trường hợp ngoại lệ sau :

* Thủ tục **WndProc** của một cửa sổ đang thực hiện việc bắt giữ thiết bị chuột (*mouse capture*), đối với tình trạng này thì cửa sổ tiếp tục nhận được thông điệp chuột cho dù con trỏ chuột có di chuyển ra ngoài vùng làm việc của cửa sổ. Kiểu này thường xuất hiện trong các ứng dụng vẽ hay thao tác đối tượng đồ họa, ví dụ khi ta vẽ một đường thẳng dài và kéo ra ngoài vùng làm việc của cửa sổ, thì khi đó cửa sổ vẽ sẽ bắt giữ toạ độ của thiết bị chuột để tạo đường thẳng và có thể cho thanh cuộn cuộn theo.
* Nếu xuất hiện hộp thông tin trạng thái (*model*) của hệ thống, thì không có chương trình nào khác nhận được thông điệp của thiết bị chuột. Hộp thoại trạng thái hệ thống và hộp thoại trạng thái của ứng dụng ngăn cản việc chuyển qua cửa sổ khác trong một ứng khi nó chưa giải quyết xong hay vẫn còn trạng thái kích hoạt (*active*).

**Thông điệp của thiết bị chuột ngoài vùng làm việc**

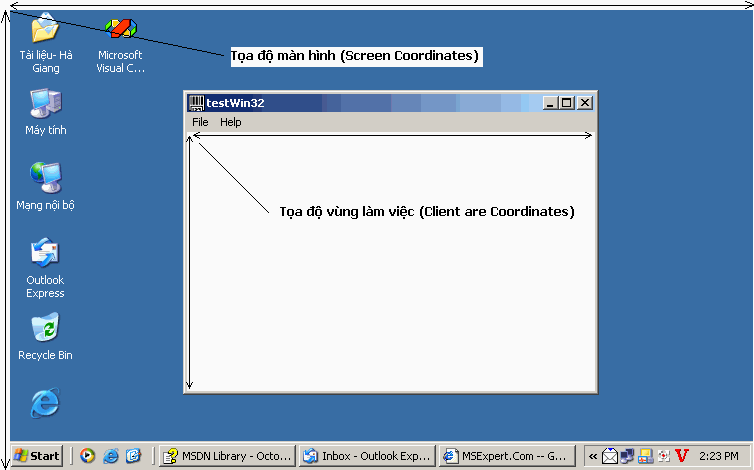
Với các thông điệp của thiết bị chuột vừa tìm hiểu trong phần trước đều được phát sinh khi thiết bị chuột nằm trong vùng làm việc của cửa sổ. Khi di chuyển con trỏ chuột ra khỏi vùng làm việc của cửa sổ nhưng vẫn ở trong phạm vi của cửa sổ thì khi đó các thông điệp của thiết bị chuột sẽ được phát sinh dạng thông điệp của thiết bị chuột ngoài vùng làm việc (*nonclient-area*). Ngoài vùng làm việc của một cửa sổ là cửa sổ thanh tiêu đề, thực đơn, và thanh cuộn của cửa sổ.

Nói chung với các thông điệp của thiết bị chuột phát sinh từ ngoài vùng làm việc thì chúng ta không quan tâm lắm, thay vào đó ta giao phó cho hàm mặc định xử lý là **DefWindowProc** thực hiện. Điều này cũng giống như là thông điệp bàn phím hệ thống mà ta đã tìm hiểu trong các phần trước.

Cũng tương tự như thông điệp xuất phát từ vùng làm việc, các thông điệp ngoài vùng làm việc được định nghĩa với từ **NC** vào sau dấu "\_", ta có bảng mô tả các thông điệp phát sinh từ ngoài vùng làm việc như sau.

|  |  |  |  |
| --- | --- | --- | --- |
| Mô tả các thông điệp chuột ngoài vùng làm việc | | | |
| Nút | Nhấn | Thả | Nhấn đúp |
| Trái | WM\_NCLBUTTONDOWN | WM\_NCLBUTTONUP | WM\_NCLBUTTONDBLCLK |
| Giữa | WM\_NCMBUTTONDOWN | WM\_NCMBUTTONUP | WM\_NCMBUTTONDBLCLK |
| Phải | WM\_RBUTTONDOWN | WM\_NCRBUTTONDOWN | WM\_NCRBUTTONDBLCLK |

Các tham số lParam và wParam cũng hơi khác so với các thông điệp thiết bị chuột phát sinh trong vùng làm việc. Với tham số lParam của thông điệp phát sinh từ ngoài vùng làm việc sẽ chỉ ra vị trí ngoài vùng làm việc nơi mà thiết bị chuột di chuyển hay kéo tới. Vị trí này được định danh bởi các giá trị định nghĩa trong **WINUSER.H**được bắt đầu với **HT**(viết tắt cho *hit-test*).

Minh họa tọa độ màn hình và tọa độ vùng làm việc

Tham số lParam sẽ chứa tọa độ x ở 16 byte thấp và tọa độ y ở 16 byte cao. Tuy nhiên, đây là tọa độ màn hình, không phải là tọa độ vùng làm việc giống như thông điệp phát sinh từ vùng làm việc. Do đó chúng ta phải chuyển về tọa độ vùng làm việc để xử lý tiếp nếu cần.

Để chuyển từ tọa độ màn hình sang tọa độ làm việc hay ngược lại từ tọa độ làm việc sang tọa độ màn hình ta dùng hai hàm tương ứng được Windows cung cấp như sau :

**ScreenToClient**( hwnd, &pt );

**ClientToScreen**( hwnd, &pt );

pt là biết cấu trúc **POINT**, hai hàm trên sẽ nhận tham chiếu đến biến pt do đó sau khi gọi hàm ta sẽ được giá trị pt tương ứng ở tọa độ mới.

**Ví dụ minh hoạ thiết bị chuột**

Đoạn chương trình thực hiện chức năng vẽ tự do bằng các thao tác: Nhấn chuột trái để vẽ một đường thẳng từ vị trí con trỏ của bút vẽ đến vị trí chuột trái kích, ngoài ra ta cũng có thể vẽ bằng cách nhấn phím trái và giữ luôn sau đó ta kéo chuột đến vị trí bất kì rồi thả. Nếu ta nhấn chuột phải thì sẽ đổi màu của bút vẽ.

**LRESULT CALLBACK WndProc**(**HWND**hWnd, **UINT**message, **WPARAM**wParam, **LPARAM**lParam)

{

**HDC**hdc;

**static POINT**oldPoint;

**static int**iC;

**int**WIDTH\_PEN = 2;

**HPEN**oPen,pen;

**COLORREF**Col [ ] ={ **RGB**(0, 0, 0) ,

**RGB**(255 ,0 ,0),

**RGB**(0, 255, 0),

**RGB**(0, 0, 255),

**RGB**(255, 255, 0)};

**POINT** point;

**TCHAR** str [255];

**switch**( message ) *// Xử lý thông điệp*

{

**case**WM\_LBUTTONDOWN:

/\* Vẽ đường thẳng từ vị trí trước đó đến vị trí chuột hiện tại\*/

hdc = **GetDC**( hWnd );

pen = **CreatePen**( PS\_SOLID,WIDTH\_PEN,Col [ iC] );

oPen = ( **HPEN**) **SelectObject**( hdc,pen );

point.x = **LOWORD**( lParam );

point.y = **HIWORD**( lParam );

**MoveToEx**( hdc, oldPoint.x, oldPoint.y, NULL );

**LineTo**( hdc, point.x, point.y );

oldPoint = point;

/\* Chọn lại bút vẽ trước đó và hủy bút vẽ vừa tạo\*/

**SelectObject**( hdc, oPen );

**DeleteObject**( pen );

**ReleaseDC**( hWnd, hdc );

break;

**case**WM\_RBUTTONDOWN:

/\* Chuyển index của bảng màu sang vị trí tiếp theo, nếu cuối bảng màu thì quay lại màu đầu tiên\*/

iC = ( iC+1 ) % ( **sizeof**( Col ) / **sizeof**( **COLORREF**) );

**break**;

**case**WM\_MOUSEMOVE:

/\* Xuất toạ độ chuột hiện thời lên thanh tiêu đề\*/

**sprintf**( str,"Toa do chuot x = %d, To do y = %d", **LOWORD**(lParam), **HIWORD**(lParam));

**SetWindowText**( hWnd, str );

/\* Kiểm tra xem có giữ phím chuột trái hay không\*/

**if**( wParam & MK\_LBUTTON )

{

hdc = **GetDC**( hWnd );

pen = **CreatePen**( PS\_SOLID,WIDTH\_PEN,Col [ iC ] );

oPen = ( **HPEN**) **SelectObject**( hdc, pen );

point.x = **LOWORD**( lParam );

point.y = **HIWORD**( lParam );

**MoveToEx**( hdc, oldPoint.x, oldPoint.y, NULL );

**LineTo**( hdc, point.x, point.y );

oldPoint = point;

**SelectObject**( hdc, oPen );

**DeleteObject**( pen );

**ReleaseDC**( hWnd, hdc );

}

**break**;

**case** WM\_DESTROY:

**PostQuitMessage**( 0 );

**break**;

**default**:

**return DefWindowProc**( hWnd, message, wParam, lParam );

}

**return**0;

}

# [C/C++] Click chuột trong Dev-C++ – Mouse event in Dev-C++

BY [ADMIN](http://tinhoc.ictu.vn/?author=1) · OCTOBER 3, 2013

Trong bài viết này mình sẽ hướng dẫn về một số hàm bắt sự kiện click chuột trong chế độ [đồ họa Dev-C++](http://nguyenvanquan7826.com/2013/10/03/cc-do-hoa-trong-dev-c/).

void getmouseclick( int kind, int& x, int& y );  
-> lấy tọa độ x,y ( theo pixel) mà ở đó có click  
bool ismouseclick( int kind );  
-> trả về true nếu xảy ra click  
int mousex( ); -> lấy tọa độ chuột theo trục x  
int mousey( ); -> lấy tọa độ chuột theo trục y

WM\_MOUSEMOVE : bắt sự kiện khi di chuyển chuột trong màn hình đồ họa  
WM\_LBUTTONDBLCLK : bắt sự kiện khi nhấn đúp chuột trái  
WM\_LBUTTONDOWN : bắt sự kiện khi nhấn chuột trái  
WM\_LBUTTONUP : bắt sự kiện khi thả sau khi nhấn chuột trái  
Tuơng tự với chuột giữa và chuột phải  
WM\_MBUTTONDBLCLK  
WM\_MBUTTONDOWN  
WM\_MBUTTONUP  
WM\_RBUTTONDBLCLK  
WM\_RBUTTONDOWN  
WM\_RBUTTONUP

Còn rất nhiều hàm khác bạn xem tại đây:<http://www.cs.colorado.edu/~main/bgi/doc/>