برنامه نویسی بازی های کامپیوتری (قسمت هفتم)

بهترین اندازه دید : 1024 * 768

نویسنده : حسن مهدی اصل (مهدی 2190)
mehdi_2190@yahoo.com

WWW.Persian-Designers.COM

صفحه کلید :

در زمانهای گذشته دسترسی به صفحه کلید بسیار مشکل بود . باید یک مدیر وقفه نوشته ، یک جدول وضعیت ایجاد نموده و کارهای فراوان انجام می دادید . اما در این زمان از Direct Input برای دسترسی به صفحه کلید ، ماوس و دسته بازی و هر وسیله ورودی دیگر استفاده می کنید . اما همچنان لازم است تا استفاده از کتابخانه Win 32 را برای دسترسی به صفحه کلید و ماوس بیاموزید . حداقل به آنها نیاز دارید تا واکنش مناسب به تبادلات GUI را نشان داده و یا برنامه های نمایشی جالبی ایجاد کنید .

صفحه کلید متشکل از تعدادی کلید ، یک میکرو کنترولر و مدار الکترونیک پشتیبان است . هنگامی که کلید یا کلید هایی بر روی صفحه کلید را فشار می دهید ، جریانی از پاکتهای اطلاعاتی به ویندوز ارسال شده تا کلیدهایی فشار داده شده را معرفی نمایند . ویندوز این جریان را پردازش نموده و پیامهای رویداد صفحه کلید را ارسال می کند . خوشبختانه در ویندوز به چند روش می توان به پیامهای صفحه کلید دسترسی داشت .

- با پیام WM_CHAR
- با پیامهای WM_KEYUP و WM_KEYDOWN
- با فراخوانی تابع GetAsynckystate()

شیوه کار هر یک از این روشها اندکی متفاوت است . پیامهای WM_CHAR و WM_KEYDOWN توسط ویندوز تولید می شوند هرگاه که کلیدی فشار داده شود یا رویدادی اتفاق بیافتد . البته تفاوتهایی بین اطلاعات ارسال شده در هر پیام وجود دارد . هنگامی که کلیدی مانند A را روی صفحه کلید فشار می دهید ، دو قطعه اطلاعات تولید می شود :

- کد اسکن
- کد اسکی

کد اسکن ، یک کد منحصر به فرد منتسب به هر یک از کلیدهای صفحه کلید است که ربطی به کد اسکی ندارد . اغلب فقط لازم است بدانید که مثلا کلید A فشار داده شده است و لزومی ندارد تا فشار کلید shift را ردگیری نمایید . معمولا از صفحه کلید فقط به عنوان کلید های لحظه ای استفاده می کنید . اینکار از طریق کدهای اسکن انجام می گیرد . پیام WM_KEYDOWN مسئول تولید کدهای اسکن به هنگام فشار دادن کلید ها است .

کد اسکی ، متفاوت است . به طور مثال اگر کلید A روی صفحه کلید را فشرده باشید و کلید shift یا Caps Lock را فشار ندهید ، یک کارکتر a مشاهده می فرمائید . اگر کلیدهای Shift+A را فشار دهید کارکتر A را مشاهده می فرمائید . پیام WM_CHAR مسئول ارسال این گونه پیام ها است .

می توانید از هر تکنیک مورد نظر استفاده نمایید . مثلا اگر بخواهید یک برنامه واژه پرداز بنویسید ، حتملا باید از پیام WM_CHAR استفاده کنید ، چون وضعیت کارکترها و کدهای اسکی اهمیت بیشتری دارد . اما اگر بخواهید یک بازی کامپیوتری بنویسید که در آن کلید F معرف عملیات آتش است و کلید shift معرف سپر باشد ، دیگر کد اسکی کلیدها اهمیتی ندارد . فقط باید بدانید که آیا یک کلید فشرده شده یا نه .

روش نهایی خواندن صفحه کلید ، استفاده از تابع Win 32 به نام GetAsynckystate() است .
تابع GetAsynckstate() آخرین وضعیت صفحه کلید را در یک جدول وضعیت ردگیری می کند . مانند آرایه ای از سوئیچ های بولی . این روش را پیشنهاد می کنم چون نیازی به نوشتن مدیر رویداد ندارد .
اکنون جزئیات هر روش را به ترتیب بررسی می کنیم . کار را با پیام WM_CHAR شروع می کنیم . پارامترهای پیام WM_CHAR به شرح زیر است :

Wparam – در برگیرنده کد اسکی کلید فشار داده شده .
Lparam - در برگیرنده یک بردار وضعیت کد بندی شده با بیت که سایر کلیدهای کنترلی که ممکن است فشار داده شوند را توصیف می کند . کد بندی بیت را در جدول زیر می توانید مشادهد کنید .

کد بندی بیت برای بردار وضعیت کلید :

بیت ها	توصیف آن
0-15	شامل شمارش تکرار است . یعنی تعداد دفعات تکرار ضربات کلید در نتیجه نگهداشتن کلید توسط کاربر
16-23	شامل کد اسکن است . این مقدار بستگی به کارخانه اصلی سازنده OEM دارد .
24	پرچم کلید بولی توسعه یافته . اگر این عدد 1 باشد ، کلید یک کلید توسعه یافته مانند کلیدهای Alt و Ctrl سمت راست است .
29	بولی : نشان می دهد که آیا کلید Alt فشار داده شده است .
30	بولی : نشان دهنده وضعیت قبلی کلید است . بدون استفاده است .
31	بولی نشان دهنده وضعیت انتقال کلید است . اگر مقدار آن 1 باشد کلید رها شده است ، در غیر این صورت کلید فشار داده شده است .

به منظور پردازش پیام WM_CHAR باید یک دستگیره پیام مانند زیر بنویسید :

```
Case WM_CHAR:
{
Int ascii_code = wparam;
Int key_state  = lparam ;

//...
}break;
```

البته می توانید اطلاعات وضعیت های مختلف را آزمایش کنید . به طور مثال شیوه آزمایش فشار دادن کلید Alt به صورت زیر است :

```
#define ALT_STATE_BIT  0*20000000
If (key_state & ALT_STATE_BIT )
{
// انجام بده فلان کار را
}
```

به عنوان مثالی از پردازش پیام WM_CHAR یک کد آزمایشی همراه جزوه موجود می باشد که کارکتر و بردار وضعیت را به هنگام فشار کلید در شکل مبنای شانزده چاپ می کند . نام فایل Demo7-1.cppمی باشد و کد مربوط به پردازش و نمایش اطلاعات WM_CHAR به صورت زیر است :

```
Case WM_CHAR :
{
Char ascii_code = wparam ;
Unsigned int key_state  = lparam ;
Hdc = GetDC(hwnd);
SetTextColor(hdc , RGB(0,255,0));
SetBKColor (hdc , RGB(0,0,0));
SetBKMode ( hdc , OPAQUE );
Sprintf(buffer , "WM_CHAR : Charatcer = %c  " , ascii_code);
TextOut( hdc , 0 , 0 , buffer , strlen(buffer));
Sprintf (buffer , "key state = 0x%x  " , key_state);
TextOut( hdc , 0 , 16 , buffer , strlen(buffer));
ReleaseDC(hwnd , hdc );
}break;
```

پیام WM_KEYDOWN برای رویداد صفحه کلید شبیه پیام WM_CHAR است ، جز اینکه اطلاعات آن پردازش نمی شود . اطلاعات ارسال شده در یک پیام WM_KEYDOWN کد اسکن مجازی که کلید است نه کد اسکی آن . کدهای اسکن مجازی مشابه کدهای اسکن استاندارد هستند که توسط صفحه کلید تولید می شوند اما کدهای اسکن مجازی برای همه نوع صفحه کلید یکسان است . مثلا ممکن است که اسکن برای یک کلید خاص در صفحه 67 باشد ولی بر روی صفحه کلید دیگری همان کد 69 باشد .

راه حل ویندوز این بود که کدهای اسکن واقعی توسط یک جدول جستجو به کدهای مجازی تبدیل شوند . ما هم مانند برنامه نویسان از کدهای اسکن مجازی استفاده نموده و اجازه می دهیم تا ویندوز عملیات ترجمه را انجام دهد . جزئیات پیام WM_KEYDOWN به شرح زیر است :

پیام WM_KEYDOWN
Wparam – در برگیرنده کد مجازی مربوط به کلید فشار داده شده است . در جدول زیر لیستی از کلیدهای رایج را مشاهده خواهید کرد .
Lparam - در برگیرنده یک بردار وضعیت کد بندی شده بیتی است که توصیف کننده سایر کلیدهای کنترلی خاص است .

کدهای کلید مجازی :

سمبل	مقدار هگزا دسیمال	توصیف کلید
VK_BACK	08	Back space key
VK_TAB	09	Tab key
VK_RETURN	0D	Enter key
VK_SHIFT	10	Shift key
VK_CONTROL	11	Ctrl key
VK_PAUSE	13	Pause key
VK_ESCAPE	1B	Esc key
VK_SPASE	20	Spacebar
VK_PRIOR	21	Page up key
VK_NEXT	22	Page down key
VK_END	23	End key
VK_HOME	24	Home key
VK_LEFT	25	Left arrow key
VK_UP	26	Up arrow key
VK_RIGHT	27	Right arrow key
VK_INSERT	2D	Ins key
VK_DELETE	2E	Del key
VK_HELP	2F	Help key
No VK_Code	30-39	0-9 keys
No VK_Code	41-5A	A-Z keys
VK-F1 >> VK_F12	70-7B	F1-F12 keys

یک پیام WM_KEYUP نیز وجود دارد . در این پیام نیز wparam شامل کد مجازی کلید و lparam شامل بردار وضعیت کلید است . پیام WM_KEYUP زمانی ارسال می شود که کلید رها شده باشد .
به طور مثال اگر پیام WM_KEYDOWN برای کنترل استفاده می کنید ، به کد زیر توجه فرمایید :

```
Case WM_KEYDOWN :
{
Int virtual_code  =(int)wparam;
Int key_state      =(int)lparam;
Switch(virtual_code)
{
Case VK_RIGHT : {} break;
Case VK_LEFT :   {} break;
Case VK_UP :      {} break;
Case VK_DOWN :{} break;
// و کیس های بیشتر را بنا بر احتیاج خود قرار بدهید
Default : break;
}
Return (0);
}break;
```

به عنوان یک تجره خوب سعی کنید تا سورس کد قبلی در برنامه Demo7-1.cpp را طوری تغییر دهید تا به جای پیام WM_CHAR از پیام WM_KEYDOWN پشتیبانی نماید .

آخرین روش خواندن صفحه کلید ، فراخوانی یکی از تابعهای وضعیت صفحه کلید است :

Getkeyboardstate() ، Getkeystate() و یا GetAsynckeystate() .

تابع GetAsynckeystate() را پیشنهاد می کنم ، چون برای یک کلید انفرادی عمل می کند . اگر به عملکرد سایر تابعها علاقه دارید ، بهتر است به win32SDK سری بزنید تا اطلاعات کاملی دریافت کنید .
الگوی عمومی تابع به صورت زیر می باشد :

```
SHORT GetAsynckeystate(int virtual_key);
```

کد مجازی کلید مورد نظر را به این تابع ارسال می کنید و اگر بیت با ارزش مقدار برگشتی معادل 1 باشد کلید فشار داده شده است . برخی ماکروها را برای سهولت این عملیات در پایین می توانید مشاهده نمایید :

```
#define KEYDOWN(vk_code) ((GetAsynkeystate(vk_code) & 0*8000 ? 1 : 0 )
#define KEYUP(vk_code)       ((GetAsynkeystate(vk_code) & 0*8000 ? 0 : 1 )
```

مزیت استفاده از این تابع آن است که تابع با حلقه رویداد زوج نمی شود . یعنی هر جایی می توانید فشار کلیدها را آزمایش کنید . به طور مثال مشغول نوشتن یک بازی هستید و می خواهید کلیدهای جهت دار ، فاصله خالی و شاید کلید Ctrl را ردگیری کنید . لزومی ندارد با پیامهای WM_CHAR یا WM_KEYDOWN کار کنید ، کافی است کدی به صورت زیر بنویسید :

```
If (KEYDOWN(VK_DOWN))
{
// کارکتر به سمت پائین حرکت کند
}
If (KEYDOWN(VK_SPACE))
{
//کارکتر عمل آتش را انجام داد
}
```

ممکن است بخواهید زمان رها شدن یک کلید را ردگیری کنید تا چیزی را خاموش کنید یا از حرکت بیاندازید:

```
If (KEYUP(VK_ENTER))
{
 // موتور خاموش شود
}
```

خوب دوستان وقت یک تمرین واقعی رسیده تا دانسته هایمان را در قالب یک تمرین مرور کنیم . برای دسترسی به کد کامل این تمرین می توانید از فایل Demo7-2.cpp همراه جزوه استفاده کنید .

اگر کد اصلی را به طور کامل بررسی کنید متوجه می شوید که در مدیر پیام پنجره از پیامهای WM_CHAR و WM_KEYDOWN اثری نیست . هر چه تعداد پیامهای پردازش شده در ()WinMain کمتر باشد بهتر است . این قسمت همه پردازش بازی را بر عهده دارد (()WinMain) . دقت کنید که تاخیر زمانی یا همزمان سازی وجود ندارد . بنابراین سرعت اجرا افزایش خواهد یافت . در جزوات بعدی انشا ا... در مورد زمان بندی و شیوه قفل نمودن پردازها به یک نرخ فریم مشخص و سایر موضوعات وابسته بحث خواهد شد . ولی پیش از آن در جزوه بعدی به ورودی ماوس خواهیم پرداخت .
دوستان به امید خدا این جزوه هم به پایان رسید . فقط تمرین فراموشتون نشه .
موفق و پیروز باشید

حسن مهدی اصل
Mehdi_2190@yahoo.com
www.persian-designers.com
