第三课　键盘与鼠标

一、键盘

1. 系统键、普通键和字符键

1) 系统键：F10、Alt、Alt+其它键。

2) 普通键：按键时没有同时按住Alt键。

3) 字符键：对应可显示字符的普通键。

2. 按键消息

```
WM_SYSKEYDOWN - 系统键按下，可能连续出现
  WM_SYSKEYUP - 系统键弹起，不会连续出现
   WM_KEYDOWN - 普通键按下，可能连续出现
     WM_KEYUP - 普通键弹起，不会连续出现
        wParam - 虚键码
        lParam - 附加信息，如连续按键次数等
```

3. 字符消息

```
WM_CHAR - TranslateMessage函数针对字符键的WM_KEYDOWN消息翻译所得
 wParam - ASCII字符码
 lParam - 附加信息，如连续按键次数等
```

```
BOOL TranslateMessage (const MSG* lpMsg)
{
    if (lpMsg -> message != WM_KEYDOWN)
        return FALSE;

    if (根据lpMsg -> wParam虚键码判断所按不是字符键)
        return FALSE;

    WPARAM wParam = lpMsg -> wParam;
    LPARAM lParam = lpMsg -> lParam;

    ...

    if (CapsLock未打开)
        wParam += 'a' - 'A';

    PostMessage (lpMsg -> hwnd, WM_CHAR, wParam, lParam);

    return TRUE;
}
```

范例：WinKey

注意：每按一次字符键产生三个消息，
依次是WM_KEYDOWN、WM_CHAR和WM_KEYUP。

练习：WinCircle
通过上下左右键移动窗口中的红色圆形，
要求圆形不得超过窗口边界。

二、鼠标
---

1. 基本消息

```
WM_LBUTTONDOWN - 左键按下
  WM_LBUTTONUP - 左键弹起
WM_MBUTTONDOWN - 中键按下
  WM_MBUTTONUP - 中键弹起
WM_RBUTTONDOWN - 右键按下
  WM_RBUTTONUP - 右键弹起
  WM_MOUSEMOVE - 移动
        wParam - 其它按键状态，以下值的位或：
    MK_LBUTTON - 左键
    MK_MBUTTON - 中键
    MK_RBUTTON - 右键
    MK_CONTROL - Ctrl键
      MK_SHIFT - Shift键
        lParam - LOWORD鼠标水平坐标，HIWORD鼠标垂直坐标
```

注意：单击消息不会连续出现，按下消息和对应的弹起消息一定是成对出现的。

2. 双击消息

```
WM_LBUTTONDBLCLK - 左键双击
WM_MBUTTONDBLCLK - 中键双击
WM_RBUTTONDBLCLK - 右键双击
          wParam - 其它按键状态，以下值的位或：
      MK_LBUTTON - 左键
      MK_MBUTTON - 中键
      MK_RBUTTON - 右键
      MK_CONTROL - Ctrl键
        MK_SHIFT - Shift键
          lParam - LOWORD鼠标水平坐标，HIWORD鼠标垂直坐标
```

如果窗口类风格中没有CS_DBLCLKS，那么窗口就不会收到双击消息，如左键双击：
```
    WM_LBUTTONDOWN
    WM_LBUTTONUP
    WM_LBUTTONDOWN
    WM_LBUTTONUP
```

如果窗口类风格中有CS_DBLCLKS，那么窗口就可以收到双击消息，如左键双击：
```
    WM_LBUTTONDOWN
    WM_LBUTTONUP
    WM_LBUTTONDBLCLK*
    WM_LBUTTONUP
```

3. 滚轮消息

```
WM_MOUSEWHEEL - 鼠标滚轮转动，发送此消息
       wParam - HIWORD滚动量(120的整数倍，正负表示滚动方向)，
                LOWORD其它按键状态，以下值的位或：
   MK_LBUTTON - 左键
   MK_MBUTTON - 中键
   MK_RBUTTON - 右键
   MK_CONTROL - Ctrl键
     MK_SHIFT - Shift键
       lParam - LOWORD鼠标水平坐标，HIWORD鼠标垂直坐标
```

需要告知编译器当前操作系统的版本高于Windows NT 4.0，
以使WM_MOUSEWHEEL宏有效：为此可在
#include <windows.h>
之前加上
#define _WIN32_WINNT 0x0400。

范例：WinMouse

练习：WinRect
通过鼠标拖动窗口中的绿色方块。
通过滚轮使方块垂直移动，Ctrl+滚轮使之水平移动。
要求方块不得超过窗口边界。

三、定时器
---

1. 定时器消息

1) 可以在程序中设置定时器，当到达时间间隔时，
   定时器会向程序发送一个WM_TIMER消息。

2) 定时器的精度可以达到毫秒级，但是准确度很低。

2. 消息参数

```
wParam - 定时器ID
lParam - 不使用
```

3. 使用定时器

1) 创建定时器

```
UINT SetTimer (
    HWND      hWnd,        // 窗口句柄
    UINT      nIDEvent,    // 定时器ID
    UINT      uElapse,     // 时间间隔
    TIMERPROC lpTimerFunc // 定时器处理函数指针，
                           // 若为NULL则由窗口过程函数处理WM_TIMER消息，
                           // 否则不产生WM_TIMER消息，
                           // 直接调用定时器处理函数
```

```
);
```

成功返回新建定时器ID，失败返回0。

```
void CALLBACK TimerProc (
    HWND  hwnd,     // 窗口句柄
    UINT  uMsg,     // 消息ID，只取WM_TIMER
    UINT  idEvent, // 定时器ID
    DWORD dwTime    // 从系统启动到此刻的毫秒数
);
```

2) 处理WM_TIMER消息或者定义定时器处理函数

```
SetTimer (hWnd, IDT_MYTIMER, 1000, NULL);

case WM_TIMER:
    if (wParam == IDT_MYTIMER)
    {
        ...
    }
    break;

SetTimer (hWnd, IDT_MYTIMER, 1000, MyTimerProc);

void CALLBACK MyTimerProc (HWND hwnd, UINT uMsg, UINT idEvent, DWORD dwTime)
{
    if (idEvent == IDT_MYTIMER)
    {
        ...
    }
}
```

3) 关闭定时器

```
BOOL KillTimer (
    HWND hWnd,     // 窗口句柄
    UINT uIDEvent // 定时器ID
);
```

成功返回TRUE，失败返回FALSE。

如：KillTimer (hWnd, IDT_MYTIMER);

范例：WinTimer

练习：WinBall
变色反弹球，左键单击暂定/继续。
想想如何看到球的运动轨迹？