软件开发文集

第八辑

《软件开发文集》编委会 编

- 用 MFC 编写 Windows 95 程序(Ⅲ)——鼠标输入处理
- Open GL 三维图形
- Visual C++ 4.1 以 Active X 的 Internet 扩展为特色
- 客户机-服务器的高级数据存取对象

科学出版社

软件开发文集

第八辑

《软件开发文集》编委会　编

科学出版社

1996

内 容 简 介

本书是为软件开发人员和计算机用户编写的《软件开发文集》系列书的第八辑。本书向读者提供了开发技术规范、软件开发管理、开发平台、应用设计策略方面的技术资料。本辑的重点文章是：用 MFC 编写 Windows 95(Ⅱ)——鼠标输入处理；Open GL 三维图形；Visual C++ 4.1 以 Active X 的 Internet 扩展为特色；客户机-服务器的高级数据存取对象等。

本书适用于软件开发人员和计算机用户学习和参考。

图书在版编目(CIP)数据

软件开发文集　第八辑/《软件开发文集》编委会编.
北京：科学出版社，1996.12
ISBN 7-03-005618-3

Ⅰ.软…　Ⅱ.软…　Ⅲ.软件开发-文集　Ⅳ.TP311.52-53

中国版本图书馆 CIP 数据核字(96)第 19644 号

科学出版社出版
北京东黄城根北街 16 号
邮政编码：100717
北京管庄永胜印刷厂印刷
新华书店北京发行所发行　各地新华书店经售
*
1996 年 12 月第　一　版　　开本：787×1092　1/16
1996 年 12 月第一次印刷　　印张：6
印数：1—3500　　字数：130 000

定价：9.00 元

编者的话

近年来，我国的计算机产业发展迅速，PC 机销量已居世界第六位。同时，我国有一批人数众多、技术水平相当不错的软件开发人员，他们分布在全国各地、各个行业、各个领域。然而，由于计算机产业，特别是软件产业充满了变化，极富动态性，加之，我国幅员辽阔，通信技术又落后于较发达的国家，从而，软件开发人员面临着这样的问题：了解最新技术动态不及时、不方便，获取最新技术信息比较困难，所得到的信息往往不够全面、完整和深入，相互之间缺乏交流，因此，重复开发、重复别人走过的弯路的情况屡见不鲜。另一方面，全世界积累的资料浩如烟海，价格亦非常昂贵，鉴于国内经济承受能力有限，许多技术信息不能及时获得。有鉴于此，我们组织有关人员翻译、编写了以技术专集为特色的《软件开发文集》系列书，该套书将陆续出版、发行，及时向用户提供最新技术资料，力图为解决上述问题做一点尝试。

《软件开发文集》系列书的宗旨是为软件开发人员开辟一个相互交流经验和体会的园地，提供一条了解新技术、新工具、新产品的渠道，设立一个探讨软件开发理论和开发技术的论坛，帮助软件开发人员更快、更直接地获得有关软件开发的信息，提高软件开发人员的技术水平和工作效率，充分发挥软件的作用，提高软件的使用价值，促进我国软件产业与世界同步发展。

《软件开发文集》是面向软件开发人员的中、高档次的技术专集，所收集的文章向软件开发人员提供了最新科技动态，以及开发技术规范、开发经验和技巧、软件开发管理、应用设计策略、开发平台方面的内容等，同时，根据用户在软件使用和开发过程中遇到的难点、疑点给予一定的解答和帮助。《软件开发文集》资料主要来源于微软公司的“Microsoft System Journal”，“Developer News”，“Microsoft Development Library”，“TechNet” 等，并收入国内软件开发人员撰写的有关文章。欢迎国内广大软件开发人员为《软件开发文集》撰稿，介绍自己的经验、心得、体会。特别要注重文章内容适合我国国情。

目前，在国内面向计算机最终用户的图书、杂志丰富多彩，但面向软件开发人员的图书、杂志却寥寥无几。我们期望《软件开发文集》能够弥补这方面的空白，并且不辜负广大用户的期望，把握好学术方向，确定好信息服务和技术支持的层次和深度，把《软件开发文集》办成深受广大用户喜爱的丛书。

由于时间仓促，《软件开发文集》难免存在这样或那样的问题，敬请广大的用户及读者提出宝贵意见和建议，以利改进，为我国软件产业的发展作出应有的贡献。

《软件开发文集》编委会

编委会名单

顾　问

杜家滨　冯玉琳　秦人华　林资山

主　编

郑茂松

副主编

（按姓氏笔画为序）

刘晓融　李　浩　廖恒毅

编　委

（按姓氏笔画为序）

王淑兰　巴建芬　华京梅　查良钿

目　录

1

用MFC编写Windows 95程序(Ⅱ)

——鼠标输入处理

如果现实生活真像电影、电视中所讲的那样，传统的输入设备早就该让位于诸如语音识别、视网膜扫描器之类的具有未来主义色彩的装置了。可惜现实远非如此，所以现今市场上绝大多数应用软件所接收的输入仍主要来自鼠标和键盘。Windows 95 能够自动处理这些输入。例如：将菜单内的鼠标单击转换成命令消息，通知应用程序“一个菜单项被选中”。

与 Windows 95 的其他输入一样，鼠标和键盘输入以消息的形式出现。当一个鼠标键被按下或释放、当鼠标被移动、当键盘上一个键被按下或释放时，键盘或鼠标驱动器就在一个特殊的系统输入队列中放入一个事件报告。这个输入队列称为原始输入队列。由操作系统生成的分离执行线程(thread)将报告从原始输入队列中取走，合成鼠标或键盘消息，并将此消息放入相应的应用程序的消息队列中。鼠标和键盘消息就可以和其他消息同样的方式被处理或分配。

上述输入模型与 Windows 3.1 不同 ，Windows 3.1 将鼠标和键盘消息存储在一个单一的输入队列中，该队列为所有正在运行的应用程序共享。Windows 95 下的 16 位应用程序仍共享一个输入队列，但 32 位应用程序则分别具有私有的输入队列。这样可以避免由于一个应用程序不能快速响应鼠标，键盘消息而影响到其他应用程序运行。一个应用程序还可以拥有多个消息队列。在一个多线程的(multithread)基于 Win 32 的应用程序中，每条接收-发送消息的线索都接受一个私有输入队列。这种为每一个应用程序(或应用程序的每条线索)分配一个私有消息队列作法的好处似乎并不明显，但如果你曾经遇到过这种情况：一个运行起来很糟糕的 16 位应用程序，因为处理消息太慢而使系统其他部分也只能缓慢“爬行”。那么你就会感到这么作的优点。幸运的是，操作系统处理鼠标和键盘消息的方式对应用程序来说是透明的。在一个消息循环中，当::Get Message( )函数被调用后，鼠标和键盘消息被从队列中抽出并分配给相应的应用程序，而在上述过程中，应用程序并不用关心消息是以何种方式在哪里存储的。

学习读解 Windows 下鼠标和键盘输入，主要学习的内容是各种鼠标和键盘事件消息。本文将讨论鼠标消息，以后的文章中再讨论键盘输入的处理。本文中举了两个程序作为例子，一个是 tic-tac-toe 游戏，用以说明程序是如何响应鼠标单击或双击的；一个是绘画程序，说明如何进行鼠标捕获，并处理非用户区鼠标消息。

一、得到用户区鼠标消息

Windows 95 使用 20 余种消息类型来报告鼠标事件。这些消息可以分为两大类：用户区鼠标消息，报告发生在窗口用户区的鼠标事件；非用户区鼠标消息，这些消息属于窗口非用户区中发生的鼠标事件。所谓事件，可以指按下或释放鼠标按钮，双击鼠标按钮，或鼠标的移动。你可以忽略在你的窗口非用户区发生的事件，让 Windows 去处理它们。

Windows 使用图表 1 中所列的消息来报告发生在窗口用户区的鼠标消息。当然，应用程序所能收到的消息依赖于所安装的鼠标，若鼠标只有两个按钮，应用程序自然不会收到 WM_MBUTTONDOWN 消息。同样，若鼠标只有一个按钮，也就不会收到 WM_RBUTTON 消息。大多数运行 Windows 的 PC 机装有双按钮鼠标，所以可以认为鼠标右按钮是存在的。如果你想确认这一点(或希望知道是否有三个按钮)，可以按以下方式调用 ::GetSystemMetrics( )：

```
::GetSystemMetrics(SM_CMOUSEBUTTONS);
```

返回值是鼠标按钮数目，若未安装鼠标，则返回 0 值。

表 1　用户区鼠标消息

消息	处理函数	起因事件
WM_LBUTTONDOWN	OnLButtonDown	按下鼠标左键
WM_LBUTTONUP	OnLButtonUp	释放鼠标左键
WM_LBUTTONDBLCLK	OnLButtonDblClk	双击鼠标左键
WM_MBUTTONDOWN	OnMButtonDown	按下鼠标中键
WM_MBUTTONUP	OnMButtonUp	释放鼠标中键
WM_MBUTTONDBLCLK	OnMButtonDblClk	双击鼠标中键
WM_RBUTTONDOWN	OnRButtonDown	按下鼠标右键
WM_RBUTTONUP	OnRButtonUp	释放鼠标右键
WM_RBUTTONDBLCLK	OnRButtonDblClk	双击鼠标右键
WM_MOUSEMOVE	OnMouseMove	在窗口用户区上移动鼠标

表 1 中，WM_XBUTTONDOWN 和 WM_XBUTTONUP 消息报告鼠标按钮被按下或释放。一个 WM_LBUTTONDOWN 消息后面一般会跟着一个 WM_LBUTTONUP 消息，但不可认为必定如此。鼠标消息总是被分配到光标当前所在窗口，所以如果用户在一个窗口的用户区按下了鼠标左按钮，而在释放按钮之前移出了这个窗口，那么该窗口就会只收到一个 WM_LBUTTONDOWN 消息，而收不到 WM_LBUTTONUP 消息。在这两种消息的配对并不重要时，许多应用程序仅响应按钮按下的消息，忽略按钮释放消息。当两种消息的配对是必要的时候，程序可以在收到按钮按下消息后，捕获鼠标，并在按钮释放消息到来后释放它。当鼠标被捕获时，所有鼠标消息，即使它发生在程序窗口之外，也会被发送给实行了捕获操作的窗口。这样就保证了无论鼠标按钮在何处被释放，应用程序

口都会收到一个按钮释放消息。以后将简短讨论鼠标捕获问题。

当一个鼠标按钮被双击后(双击间隔由控制面板设定或用::SetDoubleClickTime( )设定),第二个按钮按下消息被一个 WM_XBUTTONDBLCLK 消息代替。应该注意的是,这种情况只有在窗口风格(style)包含有 CS_DBLCLKS 时才会发生。MFC 登记的缺省窗口类具有 CS_DBLCLKS 风格,所以只有当你用 AfxRegisterWndClass( )函数登记你自己的窗口类时才需要考虑窗口风格问题。对于一个 CS_DBLCLKS 风格的窗口,在窗口用户区两次快速单击鼠标左按钮会产生以下消息序列:

```
WM_LBUTTONDOWN
WM_LBUTTONUP
WM_LBUTTONDBLCLK
WM_LBUTTONUP
```

如果窗口未被登记为可处理双击,则上述操作产生如下消息序列:

```
WM_LBUTTONDOWN
WM_LBUTTONUP
WM_LBUTTONDOWN
WM_LBUTTONUP
```

对于这些消息的处理,由你自己决定。也许你希望在同一对象(或窗口同一区域)内双击。单击鼠标会产生两个无关的任务。然而由于一个双击消息总是紧跟在一个单击消息之后,所以对它们的响应不容易彻底分离开。一般的解决方式是:第一次单击选中一个对象,第二个单击对该对象进行某一种操作。例如:当你双击 Windows 95 Explorer 右边的文件夹(folder)时,第一次单击选择文件夹,第二次单击打开它并显示其内容。

WM_MOUSEMOVE 消息表示光标移动到一个窗口的用户区内。在鼠标移动过程中,光标所在窗口会收到大量报告光标位置的 WM_MOUSEMOVE 消息,许多程序只是简单地忽略这些消息,而一个绘图程序则使用这些消息在鼠标移动时不断地更新(显示)光标的 x,y 坐标。当按住鼠标左键移动鼠标时,Windows 95 的 shell 使用 WM_MOUSEMOVE 消息来拖拉一个选定窗口。不管如何使用这些消息都应确保能够尽快地处理它们,以免鼠标的移动显得不规则,不平稳。为了不使应用程序被大量的 WM_MOUSEMOVE 消息"淹没",Windows 95 并不是每当光标移动一个像素就发送一个消息,除非硬件速度跟得上。如果鼠标在屏幕上快速移动,一个应用程序也许只会收到几个 WM_MOUSEMOVE 消息,这些消息报告鼠标移动轨迹上的几个点。只有当鼠标在屏幕上缓慢移动时,才会每移动一个像素就发送一个 WM_MOUSEMOVE 消息。

在 MFC 中,WM_LBUTTONDOWN 消息由类中 OnLButtonDown 函数处理,WM_LBUTTONUP 消息由 OnLButtonUP 函数处理,依此类推,各个处理用户区鼠标消息的函数原型如下:

```
void OnMsgName(UNIT nFlags,CPoint point);
```

函数中 nFlags 包含一系列标识位用来说明鼠标按钮状态及 Shift 和 Ctrl 键当时的状态。Point 说明当时光标的位置。在所有情况下,光标位置都以用户坐标给出,该坐标以窗口用户区的左上角为原点,坐标总是以像素为度量单位。例如:一个 WM_

LBUTTONDOWN 消息中，Point.x 等于 32，Point.y 等于 64，这就意味着鼠标左键在用户区左上角右边第 32 个像素，下边第 64 个像素被按下。

你可以利用表 2 中位屏蔽(bit mask)，由 nFlags 参数可以看出某一特定按钮或键的状态是上或下。

只有当鼠标左键被按下时，表达式 nFlags&MK_LBUTTON 值为 TRUE，而如果同时 ctrl 键也被按下，则 nFlags&MK_CONTROL 也为 TRUE。有些程序在 shift 键或 ctrl 键与鼠标按钮同时按下时会产生不同的响应。例如：通过检查 WM_MOUSEMOVE 消息中 nFlags 参数的 MK_CONTROL 位可以判断鼠标移动的同时 Ctrl 键是否被按下，如果是的，一个画图程序就会限制用户只能画出直线。

表 2　检查 nFlags 参数的标识位

屏蔽位	含义
MK_LBUTTON	按下鼠标左键
MK_MBUTTON	按下鼠标中键
MK_RBUTTON	按下鼠标右键
MK_CONTROL	按下 Ctrl 键
MK_SHIFT	按下 Shift 键

如果一个窗口包含有子窗口，例如，按钮或滚动条，那么，当鼠标按钮在子窗口内被击打(或光标在子窗口中移动)时，父窗口不会接到鼠标消息。原因是：按照前述的鼠标消息发送给光标所在窗口的规则，鼠标消息被直接发送给子窗口了。在我前一篇文章的例子程序中，应用程序本身不处理任何鼠标消息却可以响应在滚动条上的鼠标单击，原因就在于此。在一个窗口的垂直滚动条(Scroll bar)上单击鼠标，Windows 会向滚动条的窗口过程发送一个鼠标消息，随后滚动条向其父窗口发送 WM_VSCROLL 消息。处理鼠标消息的是滚动条的窗口过程，该过程深藏于 USER 中。对于以上所述应该注意：一个用户区鼠标消息表示一个鼠标事件发生在你的窗口的用户区，而不是在别的什么地方。

二、Tic Tac 应用程序

为了证明处理鼠标消息是件很简单的事情，我们来建立一个接收鼠标输入的应用程序。如图 1、图 2。Tic Tac 是一个 tic-tac-toe 游戏(一种类似于五子棋的游戏)的程序，它响应 3 种用户区鼠标事件：单击左键、单击右键、双击左键。在一个空白方格中单击鼠标左键，程序会在该空格中放置一个 X，在一空白方格中单击右键，程序会在该方格中放置一个○(程序通过保证×、○必须交替放置来防止作敝)。在分隔各方格的黑色粗实线上双击鼠标左键，程序会清除棋盘并开始新一盘游戏。每次放置×或○后，程序会检查是否有人胜利或是平局，平局指当 9 个方格都已填满且没有一方在水平或垂直或对角线上将自己的三个棋子连成一线。除了提供一个处理鼠标的样例，Tic Tac 还介绍了几个易用的新的 MFC 函数，如：CWnd::MessageBox( )，该函数显示一个消息框窗口；CRect::PtInRect( )，该函数可以很快地告诉你一个给定的点是否位于 CRect 所代表的矩形之内。

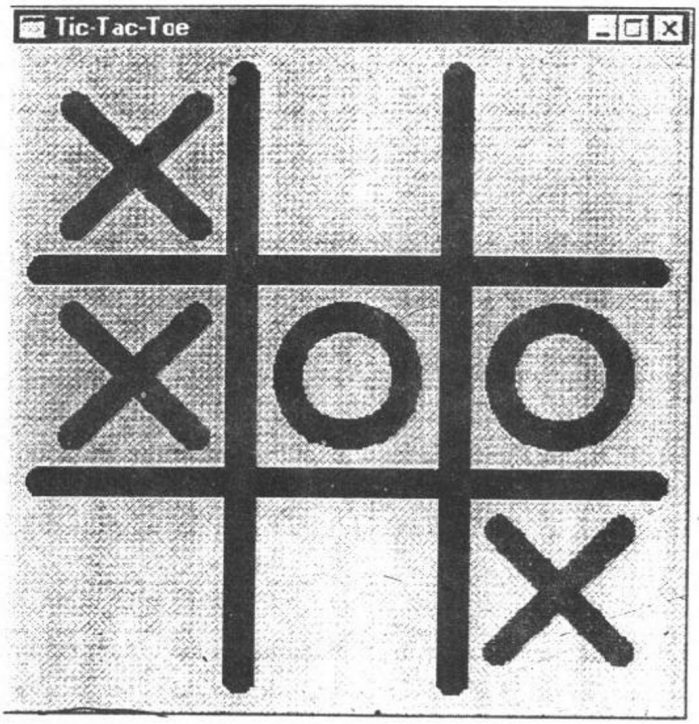

图 1　TicTac

TICTAC. H

```
class CMyApp:public CWinApp
{
public:
BOOL InitInstance ( );
};
class CMainWindow:public CWnd
{
private:
    CRect m_rect[9];          // Rectangles denoting grid squares
    int m_nNextChar;          // Next character(Ex for X or OH for O)
    int m_nGameGrid[9];       // The game grid(0=empty square)

    int GetRectID (CPoint);
    void DrawBoard (CDC * );
    void DrawX (CDC * ,int);
    void DrawO (CDC * ,int);
    void ResetGame ( );
    void CheckForGameOver ( );
    int IsWinner ( );
    BOOL IsDraw ( );

public:
    CMainWindow ( );
protected:
```

```
    void PostNcDestroy ( );

        afx_msg void OnPaint ( );
        afx_msg void OnLButtonDown (UINT,CPoint);
        afx_msg void OnLButtonDblClk (UINT,CPoint);
        afx_msg void OnRButtonDown (UINT,CPoint);

        DECLARE_MESSAGE_MAP ( )
};

#define EX 1
#define OH 2
```

TICTAC. CPP

```
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
// TicTac demonstrates mouse handling in an MFC program.
//
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

#include <afxwin.h>
#include "tictac.h"

CMyApp myApp;

////////////////////////////////////
//CMyApp member functions

BOOL CMyApp::InitInstance
{
    m_pMainWnd=new CMainWindow;
    m_pMainWnd->newWindow(CWnd Window);
    m_pMainWnd->UpdateWindow ( );
    return TRUE;
////////////////////////////////////
//CMainWindow message map and member functions

BEGIN_MESSAGE_MAP (CMainWindow,CWnd)
    ON_WM_PAINT ( )
    ON_WM_LBUTTONDOWN ( )
    ON_WM_LBUTTONDBLCLK ( )
    ON_WM_RBUTTONDOWN ( )
END_MESSAGE_MAP ( )
```

```
CMainWindow::CMainWindow ( )
{
    int nCoords[9][4]={
          16,   16,112,112,
         128,   16,224,112,
         240,   16,336,112,
          16,  128,112,224,
         128,  128,224,224,
         240,  128,336,224,
          16,  240,112,336,
         128,  240,224,336,
         240,  240,336,336,
    };
    m_nNextChar=EX;
    for(int i=0;i<9;i++){
        m_nGameGrid[i]=0;
        m_rect[i].SetRect (nCoords[i][0],nCoords[i][1],
                           nCoords[i][2],nCoords[i][3]);
}
CString myClass=AfxRegisterWndClass (CS_DBLCLKS,
                    myApp.LoadStandardCursor (IDC_ARROW),
                    (HBRUSH)::GetStockObject (LTGRAY_BRUSH),
                    myApp.LoadStandardIcon (IDI_APPLICATION));

    CreateEx (WS_EX_DLGMODALFRAME,myClass,"Tic-Tac-Toe",
        WS_OVERLAPPED | WS_SYSMENU | WS_CAPTION | WS_MINIMIZEBOX,
        CW_USEDEFAULT,CW_USEDEFAULT,CW_USEDEFAULT,
CW_USEDEFAULT,
        NULL,NULL,NULL);

    CRect rect (0,0,352,352);
    CalcWindowRect (&rect);

    SetWindowPos (NULL,0,0,rect.Width ( ),rect.Height ( ),
        SWP_NOZORDER | SWP_NOMOVE(SWP_NOREDRAW);

void CMainWindow::PostNcDestroy( )
{
    delete this;
```

```
}
void CMainWindow::OnPaint
{
    CPaintDC dc (this);
    DrawBoard(&dc);
}
void CMainWindow::OnLButtonDown(UINT nFlags,CPoint point)
{
    if(m_nNextChar !=EX)
      return;
    int nPos=GetRectID(point);
    if(nPos=-1)
      return;
    if(m_nGameGrid[nPos] !=0)
      return;

    m_nNextChar=OH;
    m_nGameGrid[nPos]=EX;

    CClientDC dc (this);
    DrawX (&dc,nPos);

    CheckForGameOver ( );
}

void CMainWindow::OnRButtonDown (UINT nFlags,CPoint point)
{
    if(m_nNextChar !=OH)
      return;

    int nPos=GetRectID (point);
    if(nPos=-1)
      return;

    if(m_nGameGrid[nPos] !=0)
      return;

    m_nNextChar=EX;
    m_nGameGrid[nPos]=OH;
```

```
CClientDC dc (this);
    DrawO (&dc,nPos);

    CheckForGameOver ( );
}
void CMainWindow::OnLButtonDblClk (UINT nFlags,CPoint point)
{
    CClientDC dc (this);
    if (dc.GetPixel (point)=RGB (0,0,0))
       ResetGame ( );
}

int. CMainWindow::GetRectID (CPoint point)
{
    for (int i=0;i<9;i++){
        if(m_rect[i]. PtInRect (point))
          return i;
    }
    return -1;
}

void CMainWindow::DrawBoard (CDC * pDC)
    Open_pen (PS_SOLID,16,RGB(0,0,0));
    CPen * pOldPen=pDC->SelectObject(&pen);
    pDC->MoveTo (120,16);
    pDC->LineTo (120,336);

    pDC->MoveTo (232,16);
    pDC->LineTo (232,336);

    pDC->MoveTo (16,120);
    pDC->LineTo (336,120);

    pDC->MoveTo (16,232);
    pDC->LineTo (336,232);

    for (int i=0; i<9; i++){
        if(m_nGameGrid[i]==EX)
          DrawX (pDC,i);
        else if (m_nGameGrid[i]==OH)
          DrawO (pDC,i);
```

```
    }
        pDC->SelectObject (pOldPen);
    }
    void CMainWindow::DrawX (CDC * pDC,int nPos)
    {
        CPen pen (PS_SOLID,16,RGB (255,0,0));
        CPen * pOldPen=pDC->SelectObject (&pen);
        pDC->MoveTo (m_rect[nPos].left+16,m_rect[nPos].top+16);
        pDC->LineTo (m_rect[nPos].right-16,m_rect[nPos].bottom-16);

        pDC->MoveTo (m_rect[nPos].left+16,m_rect[nPos].bottom-16);
        pDC->LineTo (m_rect[nPos].right-16,m_rect[nPos].top+16);

        pDC->SelectObject (pOldPen);

    void CMainWindow::DrawO (CDC * pDC,int nPos)
    {

        OPen pen (PS_SOLID,16,RGB (0,0,255));
        OPen * pOldPen=pDC->SelectObject (&pen);
        pDC->SelectStockObject (NULL_BRUSH);

        pDC->Ellipse (m_rect[nPos].left+16,m_rect[nPos].top+16,
            m_rect[nPos].right-16,m_rect[nPos].bottom-16);

        pDC->SelectObject (pOldPen);
    }

    void CMainWindow::CheckForGameOver ( )
    {
        int nWinner;

        if (nWinner=IsWinner ( )){

            CString string=(nWinner==EX)?"The x's win!":"The O's win!";
            MessageBox(string,"Game Over",MB_ICONEXCLAMATION | MB_OK);
            ResetGame ( );
        }
        else if (IsDraw ( )){
            MessageBox ("It's a draw!","Game Over";
                        MB_ICONEXCLAMATION | MB_OK);
```