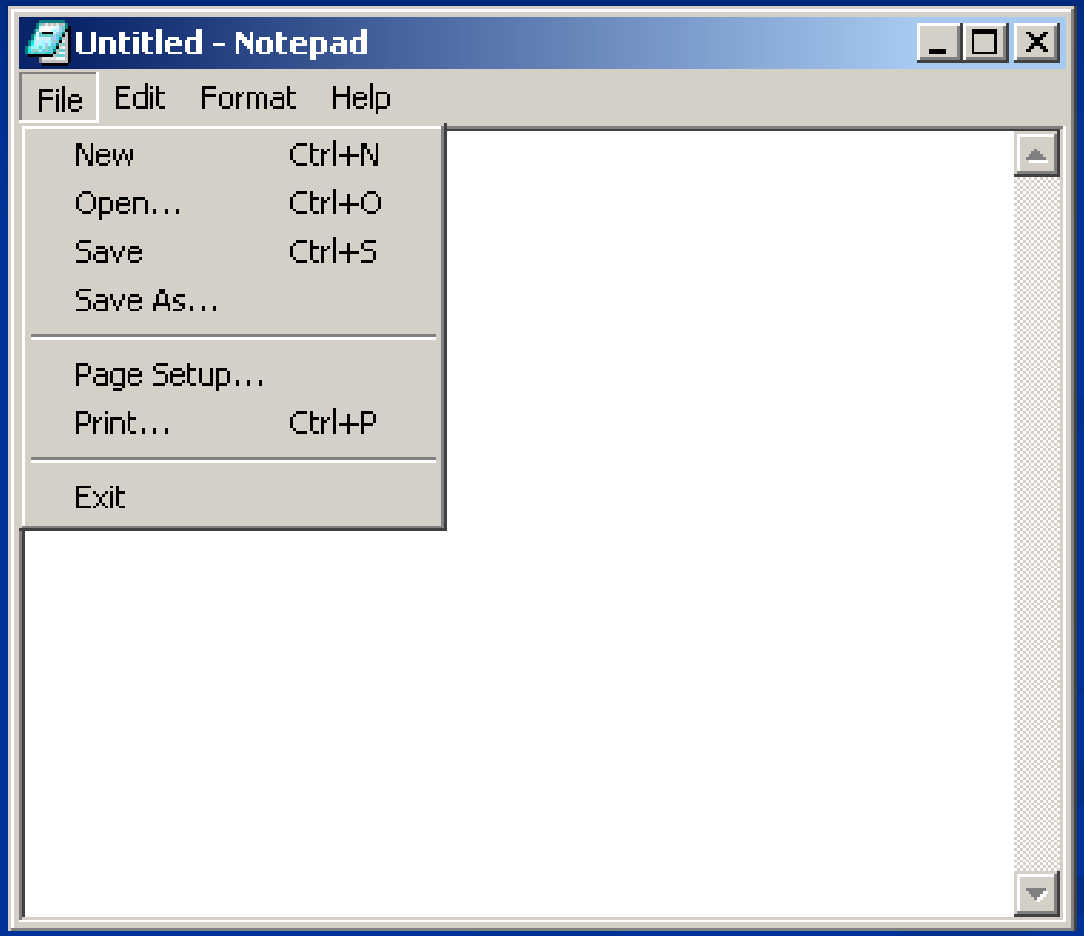
Lecture

Review of Last Lecture

Review of Today's Lecture

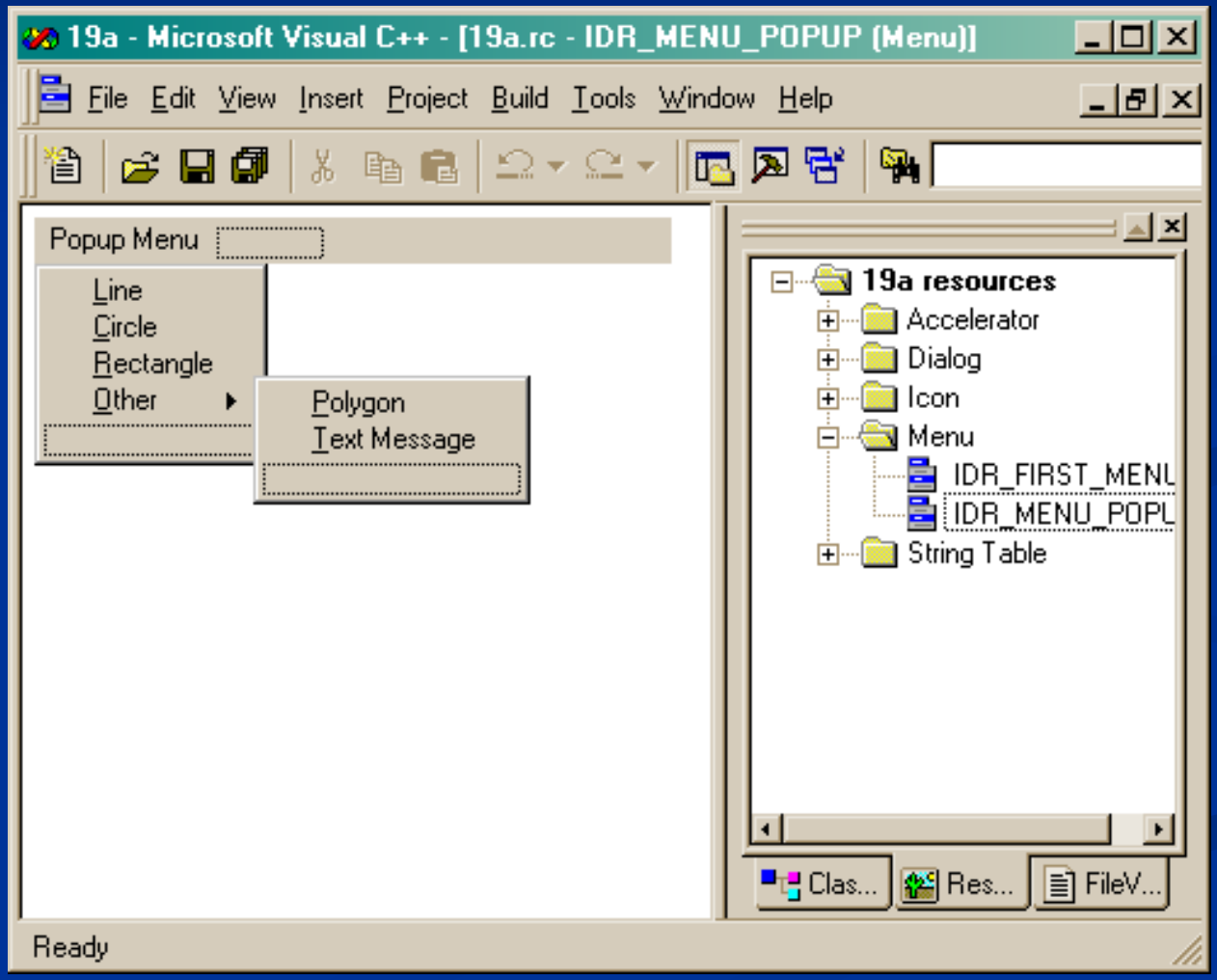

Untitled - Notepad
File
Edit
Format
Help
New Ctrl+N
Open... Ctrl+O
Save Ctrl+S
Save As...
Page Setup...
Print... Ctrl+P
Exit

GetSubMenu()

Retrieves a handle to the drop-down menu or submenu activated by the specified menu item

03

Popup menu in the resource editor (Visual Studio 6.0)



Floating popup menus

```
IDR_MENU_POPUP MENU DISCARDABLE
BEGIN
    POPUP "Popup Menu"
    BEGIN
        MENUITEM "&Line",                   ID_POPUPMENU_LINE
        MENUITEM "&Circle",
       ID_POPUPMENU_CIRCLE
       MENUITEM "&Rectangle",     ID_POPUPMENU_RECTANGLE
        POPUP "&Other"
        BEGIN
           MENUITEM "&Polygon",    ID_OTHER_POLYGON
           MENUITEM "&Text Message", ID_OTHER_TEXTMESSAGE
        END
    END
END
```

Only one drop-down starting with `POPUP "Popup Menu"`

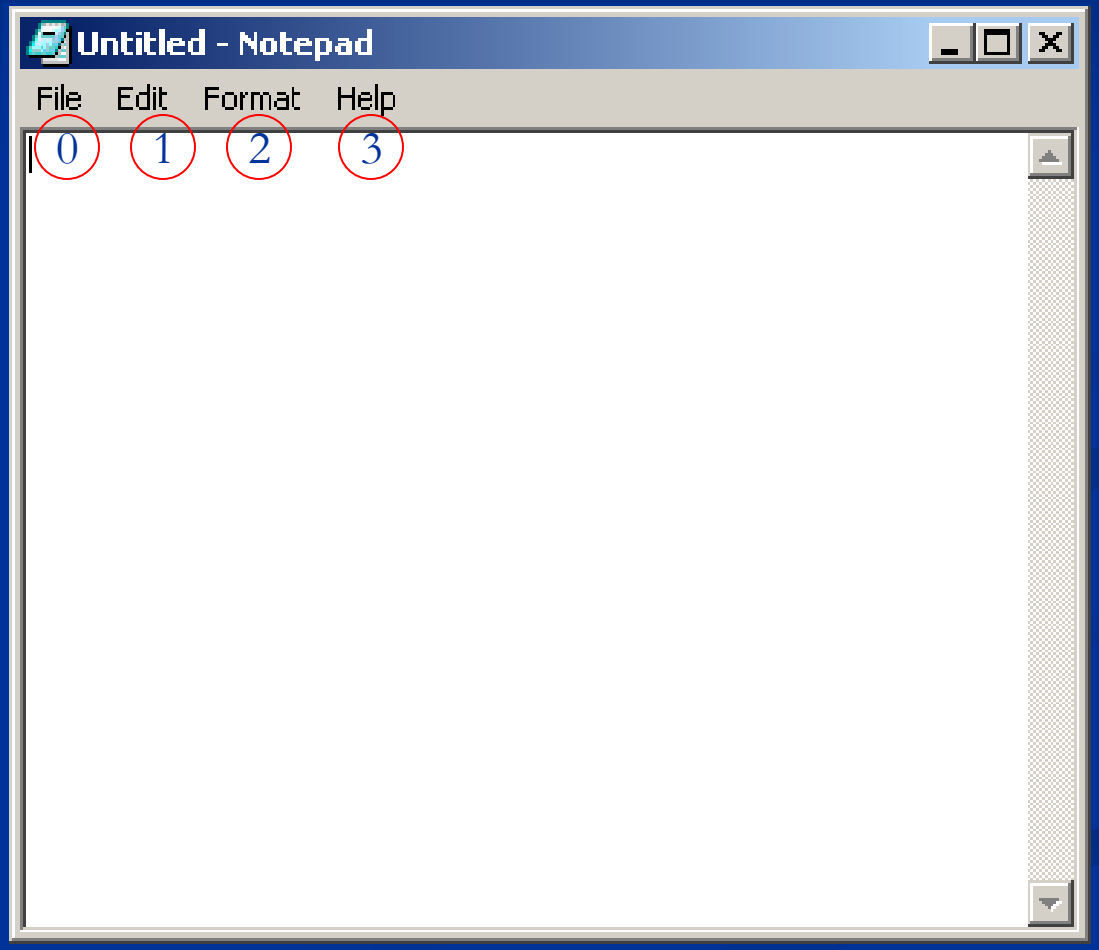
Floating popup menus

```
HMENU hPopupMenu;

Bar hPopupMenu = LoadMenu(hInstance,
  MAKEINTRESOURCE(IDR_MENU_POPUP));
Bar hPopupMenu = GetSubMenu(hPopupMenu,
  0);
```

GetSubMenu()

```
HMENU GetSubMenu(
  HMENU hMenu,  // handle to menu
  int nPos      // menu item position
);
```

Submenu numbering



WM_RBUTTONDOWN message

lParam: *client-area* coordinates of mouse cursor

low-word: x-coordinate

high-word: y-coordinate

Structures to represent a point

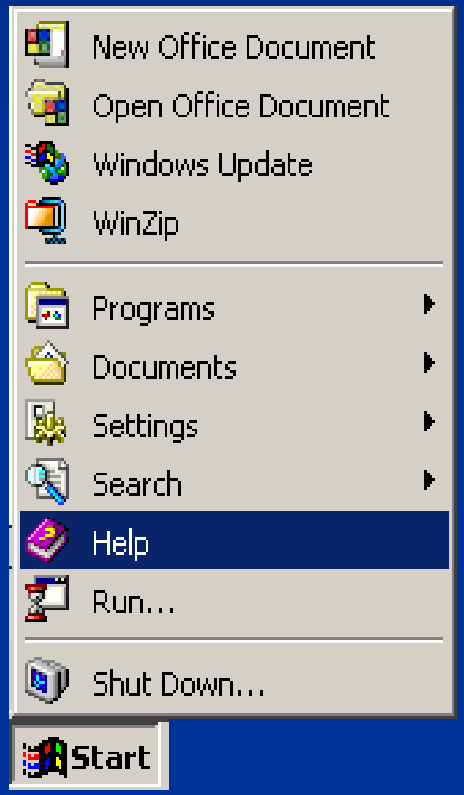
```
typedef  struct tagPOINT {
  LONG x;
  LONG y;
} POINT;

typedef  struct tagPOINTS {
  SHORT x;
  SHORT y;
} POINTS;
```

Declared in Windef.h

Floating popup menus: Main Window Procedure

```
POINTS pts; POINT pt;
… … … …
   case WM_RBUTTONDOWN:
Bar   pts = MAKEPOINTS(lParam);
BarS  pt.x = pts.x;
BarE        pt.y = pts.y;
Bar         ClientToScreen(hWnd, &pt);
Bar  result = TrackPopupMenu(hPopupMenu,
   TPM_LEFTALIGN | TPM_TOPALIGN |
   TPM_RETURNCMD | TPM_LEFTBUTTON,
            pt.x, pt.y, 0, hWnd, 0);
```

Mouse Tracking



Floating popup menus: Main Window Procedure

```
POINTS pts; POINT pt;
… … … …
   case WM_RBUTTONDOWN:
      pts = MAKEPOINTS(lParam);
      pt.x = pts.x;
      pt.y = pts.y;
      ClientToScreen(hWnd, &pt);
    result = TrackPopupMenu(hPopupMenu,
      TPM_LEFTALIGN |
      TPM_TOPALIGN |
      TPM_RETURNCMD |
      TPM_LEFTBUTTON,
      pt.x, pt.y, 0, hWnd, 0);
```

Floating popup menus: Main Window Procedure

```
result = TrackPopupMenu(hPopupMenu,
   TPM_LEFTALIGN |       TPM_TOPALIGN |
   TPM_RETURNCMD | TPM_LEFTBUTTON,
   pt.x, pt.y, 0, hWnd, 0);

switch(result)
   {
       case ID_POPUPMENU_CIRCLE:
   MessageBox(hWnd, "Circle", "Popup clicked",
   MB_OK);
              break;

   case ID_POPUPMENU_RECTANGLE:
   MessageBox(hWnd, "Rectangle", "Popup clicked",
   MB_OK);
       break;

   }
```

Manipulating Menus at runtime

HMENU CreateMenu(VOID);

creates an empty menu

BOOL AppendMenu(… … …);

appends a new item to the end of the specified menu bar, drop-down menu, submenu, or shortcut menu.

BOOL InsertMenu (… … …);

inserts a new menu item into a menu, moving other items down the menu.

Manipulating Menus at runtime

inserts a new menu item at the specified position in a menu.

```
BOOL InsertMenuItem(
  HMENU hMenu,            // handle to menu
  UINT uItem,             // identifier or position
  BOOL fByPosition,       // meaning of uItem
  LPCMENUITEMINFO lpmii   // menu item information
);
```

Manipulating Menus at runtime

BOOL ModifyMenu(… … …);

changes an existing menu item

Superseded by

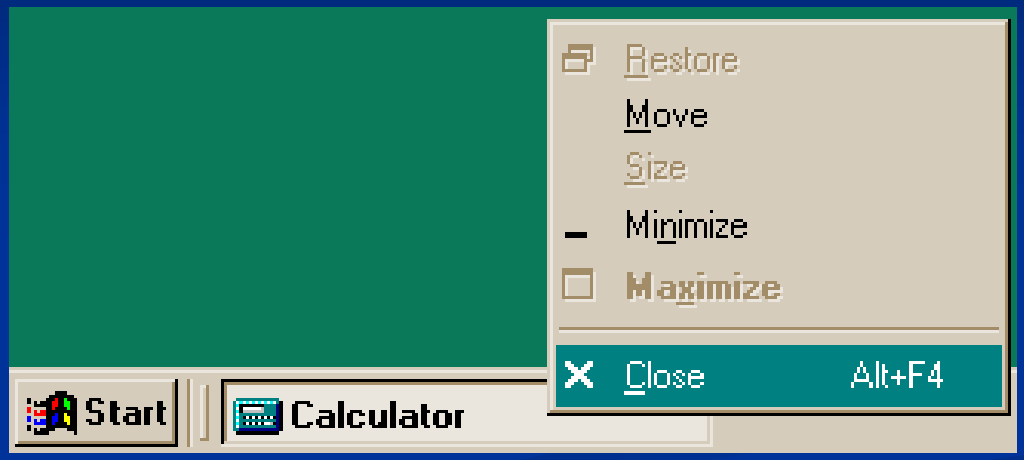
BOOL SetMenuItemInfo(… … …);

changes an existing menu item

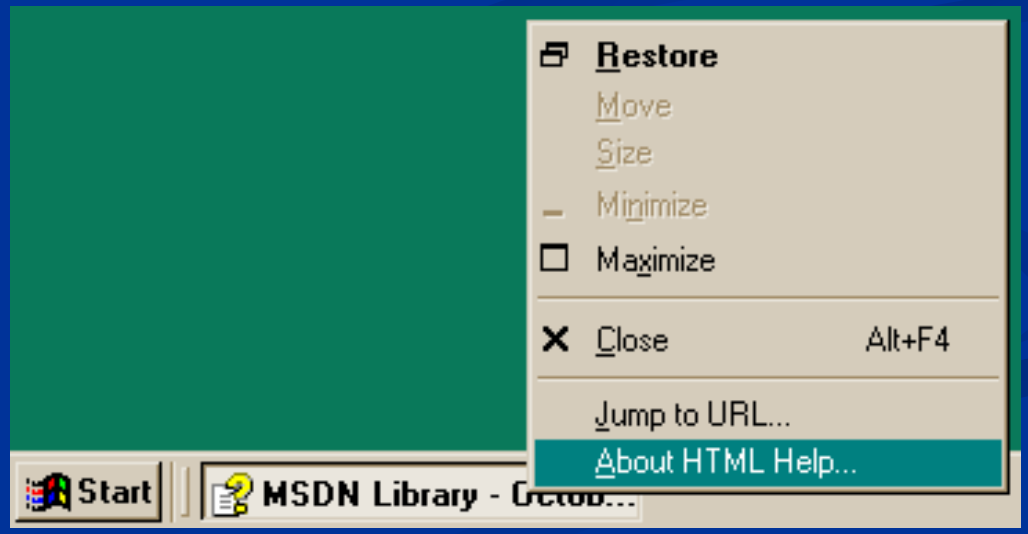
modified *system/window menu*

Right-click at the taskbar buttons

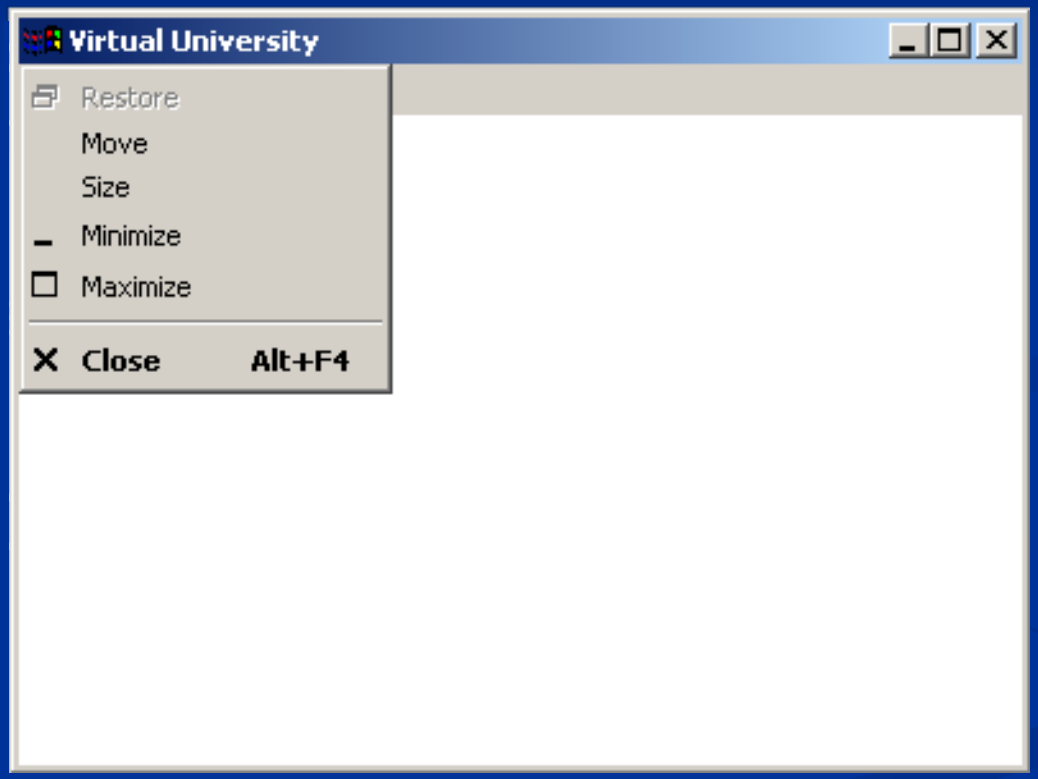
Most applications



MSDN library



The system menu or window menu



The system menu or window menu

```
HMENU GetSystemMenu(
  HWND hWnd,     // handle to window
  BOOL bRevert   // reset option TRUE:reset
);
```

The system menu or window menu

The window menu initially contains items with various identifier values, such as

Identifier	Menu item
`SC_MOVE`	`Move`
`SC_SIZE`	`Size`
`SC_CLOSE`	`Close`

Time

- Local Time

- UTC (Universal Coordinated Time)
  historically GMT (Greenwich Mean Time)

UTC (Universal Coordinated Time)

Geographically independent time

Time information in Windows

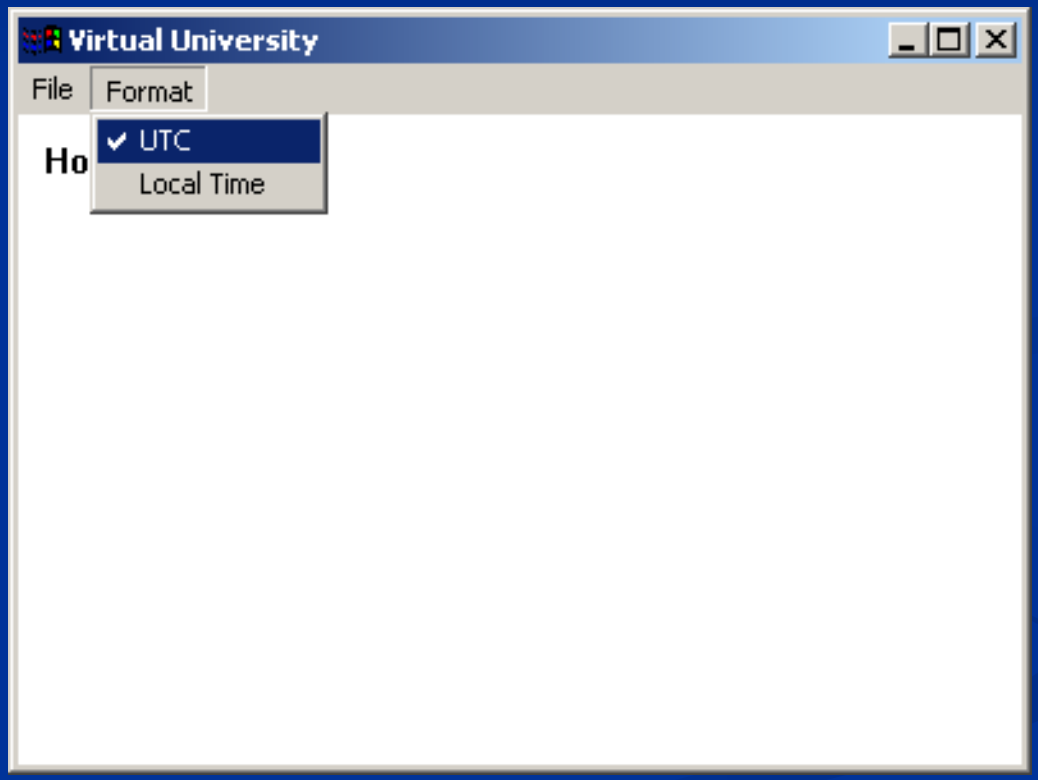
```
VOID GetSystemTime(
  LPSYSTEMTIME lpSystemTime    // system time
);
```

retrieves the system time in UTC format.

```
VOID GetLocalTime(
  LPSYSTEMTIME lpSystemTime    // system time
);
```

retrieves the current local date and time.

The clock application



Clock Example: resource definition

```
IDR_FIRST_MENU MENU DISCARDABLE
BEGIN
    POPUP "&File"
     BEGIN
         MENUITEM "E&xit",          ID_FILE_EXIT
    END
    POPUP "F&ormat"
     BEGIN
         MENUITEM "&UTC",           ID_FORMAT_UTC
         MENUITEM "&Local Time",    ID_FORMAT_LOCALTIME
    END
END
```

Clock Example: Window Procedure

```
static SYSTEMTIME st;
 enum Format { UTC, LOCAL };
 static enum Format format;

   case WM_CREATE:
   SetTimer(hWnd, ID_TIMER, 1000, NULL);
   format=LOCAL;
   GetLocalTime(&st);
 hOurMenu = GetMenu(hWnd);
   CheckMenuItem(hOurMenu,
ID_FORMAT_LOCALTIME, MF_BYCOMMAND |
MF_CHECKED);
```

Clock Example: Window Procedure

```
case WM_COMMAND:
    switch(LOWORD(wParam))
    {
    case ID_FORMAT_UTC:
            if(format == UTC)
                    break;
            format = UTC;
            hOurMenu = GetMenu(hWnd);
    result = CheckMenuItem(hOurMenu, ID_FORMAT_UTC,
MF_BYCOMMAND | MF_CHECKED);
            result = CheckMenuItem(hOurMenu,
ID_FORMAT_LOCALTIME, MF_BYCOMMAND | MF_UNCHECKED);
            DrawMenuBar(hWnd);
    (format == UTC) ? GetSystemTime(&st) :
GetLocalTime(&st);
            GetClientRect(hWnd, &rect);
    InvalidateRect(hWnd, &rect, TRUE);
            break;
```

Clock Example: Window Procedure

```
case WM_PAINT:
     hDC = BeginPaint(hWnd, &ps);

     wsprintf(msg, "Hour: %2d:%02d:%02d",
st.wHour, st.wMinute, st.wSecond);

     TextOut(hDC, 10, 10, msg,
lstrlen(msg));

     EndPaint(hWnd, &ps);
     break;
```

Clock Example: Window Procedure

```
case WM_TIMER:
  if(wParam == ID_TIMER)
  {
    (format == UTC) ? GetSystemTime(&st) :
GetLocalTime(&st);
     GetClientRect(hWnd, &rect);
     InvalidateRect(hWnd, &rect, TRUE);
          break;
  }
  break;
```

WYSIWYG editors

What You See Is What You Get

Dialogs

Modal Dialogs disable their owner windows

- MessageBox() creates a modal dialog

Modeless Dialogs do not disable their owner windows

Modal Loop

Dialog Resource Definition Statement

Dialog Resource Template

```
IDD_DIALOG_ABOUT DIALOG DISCARDABLE  0, 0, 265, 124
STYLE DS_MODALFRAME | WS_POPUP | WS_CAPTION | WS_SYSMENU
    CAPTION "About"
    FONT 8, "MS Sans Serif"
BEGIN
    DEFPUSHBUTTON    "OK",IDOK,208,7,50,14
    PUSHBUTTON       "Cancel",IDCANCEL,208,24,50,14
    LTEXT            "Some copyright text", IDC_STATIC,
                       67, 27,107,47
    ICON             IDI_ICON_VU,IDC_STATIC,17,14,20,20
END
```

Creating a Modal Dialog

```
INT_PTR DialogBox(
  HINSTANCE hInstance,  // handle to module
  LPCTSTR lpTemplate,   // dialog box template
  HWND hWndParent,      // handle to owner window
  DLGPROC lpDialogFunc  // dialog box procedure
);
```