用 Visual C++ 5.0 在对话框中实现“演职员表”

徐大宏 （湖南师范大学计算机部 410006）

摘要：本文阐述了在 VC5 中利用位图显示技术实现类似于电影电视中的“演职员表”这一有趣的功能。

关键词：Windows 3.2 Visual C++ 5.0 消息处理函数 位图

1. 引言

在 Windows 3.2 中有一个极其有趣的功能，给我们编程人员提供了一个参考模式及设计思想。在我们自己所设计的程序中，可借鉴利用此技术，一方面可以增加程序的趣味性；另一方面也能反映出一点著作权意识；再者还可以向软件的用户展现一下作者的信息。

在中文 Windows 3.2 中，按如下操作便可看到“演职员表”这一功能的具体表现：

(1)选用“程序管理器”窗口的“帮助”菜单中的“关于”项，则会出现“关于...”的对话框。

(2)同时按下 Ctrl 和 Shift 键，然后在标有“Microsoft Windows”的图标处双击鼠标左按钮，此时不会有什么“奇迹”出现。然后单击“确定”，以关闭对话框。

(3)第二次选用“关于”菜单项，同样还是按下 ctrl 和 Shift 键在标有“Microsoft Windows”的图标处双击鼠标左按钮，此时原图标处将会出现一飘动的 Windows 旗帜，并出现一些文字。然后仍用鼠标单击“确定”，以关闭对话框。

(4)再次选用“关于”菜单项，打开“关于”对话框，同样还是按下 Ctrl 和 Shift 键在标有“Microsoft Windows”的图标处双击鼠标左按钮，则在对话框中会出现类似于“演职员表”的向上滚动的字幕及动态变换的图标，向用户展现出 Windows 的所有工作人员的信息。

但上述功能在 Windows 95 / NT 中则不复存在。

2. “演职员表”技术的实现方法

下面，笔者在 Visual C++ 5.0 环境中，简述这一技术的实现方法。

(1)多次使用 Ctrl 和 Shift 键，以及双击鼠标左按钮时，可利用一全局变量记录该组合键所使用过的次数，也就是“关于”对话框的出现顺序(第 1 次，或第 2 次等)，然后依据不同的顺序而作出不同的反应动作(即写出不同程序代码段)

(2)鼠标双击时的相关处理。我们知道在 Visual C++ 的视类中处理鼠标双击消息(WM-LBUTTOND-BLCLK)时，要求窗口类具有 CS-DBLCLK 风格。且我们可在 View 类、Frame 类或 Dialog 类的 PreCreateWindow 函数中修改窗口类的风格。但 VC 的 AppWizard 在为我们生成对话框类时，已经默认包含有 CS-DBLCLK 这一风格。

同时当 VC 处理双击消息时，系统传给处理函数两个参数，分别为 UINT nFlags 及 CPoint poing；nFlags 表明单击鼠标时有何键按下，其值可为 MB-CONTROL 、MB-SHIFT、MB-LBUTTON、MB-MBUTTON、MB-RBUT-TON，这些值分别表示按下的是 Ctrl 或 Shift 键，或鼠标左中右按钮。point 表明鼠标的坐标位置。依据这些数据可确定是否应显示动态图标或滚动字幕。

(3)动态图标。可利用 DrawIcon 函数反复显示若干个不同状态的图标而形成动态效果。

(4)滚动字幕，可使用 TextOut 函数实现，但此种方法显示时，每次滚动或平移跳跃都只能以一行字或一个字为间距，不能形成比较平滑的左右平移或上下滚动，且灵活性不大。笔者在使用时，采用将左右平移或上下滚动的字串作成位图，然后使用位图的显示特技，便能实现非常平滑的左右平移或上下滚动效果。

3. 运用实例

本文件重点介绍如何形成平滑的左右平移或上下滚动效果，但不涉及对 Ctrl 和 Shift 的识别处理。

使用 VC 中 AppWizard 生成一单文档应用程序，其他步骤都可使用系统默认值。

(1)直接利用向导产生的“关于”对话框，但需对其作些适当修改，以适应本程序实际的需要，如变换大小及去掉其默认图标等。

(2)在程序的资源中增加一位图，务请注意位图的大小。如若实现上下滚动效果，则将位图的宽度限制成与对话框中空白(待显示位图)处基本一致，高度可任意或视待显示字串及图像而定；如若实现左右平移效果，则将

位图的高度限制成与对话框中空白(待显示位图)处基本一致,宽度则任意或视待显示字串及图像而定。并在位图中加入待显示的字串及图像。

(3)使用 MFC ClassWizard 在"关于"对话框 CAboutDlg 类中增加响应 WM-CREATE、WM-TIMER、WM-PAINT 消息处理函数,及 OK 按钮的 BN-CLICKED 事件响应函数,各函数其代码如下,其中粗体部分为笔者所增加。

```
void CAboutDlg::OnPaint()
{
    CPaintDC dc(this); // device context for painting
    // 在对话框中产生白色背景块
    CBrush Brush;
    Brush.CreateSolidBrush(RGB(255,255,255));
    CBrush * pOldBrush=dc.SelectObject(&Brush);
    CRect rect;
    GetClientRect(&rect);
    dc.PatBlt(rect.left+5,
      rect.top+5,
      (int)rect.Width()*4/7,
      rect.Height()-10,
      PATCOPY);
    dc.SelectObject(pOldBrush);
    // 在背景块周围产生蓝色边框线
    CPen Pen;
    CPen * pOldPen;
    Pen.CreatePen(PS-SOLID,1,RGB(0,0,255));
    pOldPen=dc.SelectObject(&Pen);
    dc.MoveTo(rect.left+5,rect.top+5);
    dc.LineTo((int)rect.Width()*4/7+5,rect.top+
5);
    dc.LineTo((int)rect.Width()*4/7+5,rect.Height
()-5);
    dc.LineTo(rect.left+5,rect.Height()-5);
    dc.LineTo(rect.left+5,rect.top+5);
}

void CAboutDlg::OnTimer(UINT nIDEvent)
{
    // 产生对话框的显示设备情境对象
    CClientDC dc(this);
    CRect rect;
    // 得到对话框客户区大小
    GetClientRect(&rect);
    // 在显示设备情境对象中显示位图
    dc.BitBlt(
      rect.left+6, // 目标设备情境处的 x 坐标
      rect.top+8, // 目标设备情境处的 y 坐标
      (int)rect.Width()*4/7-2, // 待显示在目标设
备上的宽度
      rect.Height()-13, // 待显示在目标设备上的高
度
      &m-MemDC, // 源设备情境对象
      0, // 源设备情境中的起始 x 坐标
      yPos, // 源设备情境中的起始 y 坐标
      SRCCOPY); // 位图显示光栅操作模式
    // 变换源位图中的显示起始位置
    yPos++;
    if ( yPos >= m-bm.bmHeight-rect.Height()-13
)
      yPos=0;
    dc.DeleteDC();
    CDialog::OnTimer(nIDEvent);
}
void CAboutDlg::OnOK()
{
    // 取消定时器事件
    KillTimer(1);
    // 删除位图实例
    ::DeleteObject(&m-selfBmp);
    CDialog::OnOK();
}
int CAboutDlg::OnCreate(LPCREATESTRUCT lpCreat-
eStruct)
{
    if (CDialog::OnCreate(lpCreateStruct)==-1)
      return -1;
    // 设置定时器,每 40 毫秒触发一次
    SetTimer(1,40,NULL);
    return 0;
}
```

(4)在 CAboutDlg 类中增加如下私有成员变量,并在类构造函数中作相应处理。

```
BITMAP m-bm;
int yPos;
```

```
    CDC m-MemDC;
    CBitmap m-selfBmp;
CAboutDlg::CAboutDlg() : CDialog(CAboutDlg::IDD)
{
    //从资源中装载位图 IDB-BMPSELF 为自定义位图 ID 号
    m-selfBmp.LoadBitmap(IDB-BMPSELF);
    //取得位图大小等信息
    m-selfBmp.GetObject(sizeof(m-bm),&m-bm);
    //创建内存设备情境,以装载位图
    m-MemDC.CreateCompatibleDC(NULL);
    m-MemDC.SelectObject(&m-selfBmp);
    yPos=0;
}
```

"演职员表"这一技术实现的关键是位图的显示处理,也即对 BitBlt 函数的灵活运用。我们知道在显示设备情境对象中显示出来的位图大小可以小于待显示的内存设备情境对象中的位图大小,这样,我们便可通过定时器函数来变化待显示的位图大小及起始位置。本程序仅实现了由下至上的平滑滚动效果,每次上滚动 1 象素,我们还可以实现由上至下的平滑滚动效果,或自左至右,自右至左的水平平移效果,只要适当的变化 BitBlt 的几个参数及位图的大小即可。

在上述代码中,本程序中所使用位图应存在于资源中,这样位图数据均包含于可执行文件中,有两点不足之处:一是造成可执行文件变大;二是位图的变换很不灵活,每改变一次位图数据,则必须重新编译连接一次源程序。可对本程序代码作如下变化,以便作到,程序运行时动态地从外部文件中装载位图。当位图变化后,可执行文件无需重新编译连接。

·将上述代码中的类成员变量 CBitmap m-selBmp 改为 HBITMAP * m-pselfBmp;

·对 CaboutDlg 类构造函数作如下修改:

```
CAboutDlg::CAboutDlg() : CDialog(CAboutDlg::IDD)
{
    //从文件中装载位图
    m-pselfBmp=(HBITMAP *)::LoadImage(
      AfxGetInstanceHandle(), //取得应用实例句柄
      "selfbmp.bmp", //外部位图文件名
      IMAGE-BITMAP, //格式为位图
      0,0, //宽度高度
      LR-LOADFROMFILE); //从文件中装载
    //取得位图大小等信息
    ::GetObject(m-pselfBmp,sizeof(m-bm),&m-bm);
    //创建内存设备情境,以装载位图
    m-MemDC.CreateCompatibleDC(NULL);
    m-MemDC.SelectObject(m-pselfBmp);
    yPos=0;
}
```

其他代码不要作任何修改。

4. 结语

本程序代码在 Visual C++ 5.0 环境下调试通过,且可运行于 Windows 95 及 NT 中。

当然本程序还可以作如下一些改进:可将上述代码形成一个类,以实现代码重用;可将其编写成 ActiveX 控件;不使用定时器事件,改用多线程技术;位图的显示可改用函数 StretchBlt,以便能根据显示设备情境中的大小而自动伸缩位图等。

笔者旨在阐述这一技术实现方法,以期抛砖引玉,同行高手们不妨在借鉴的基础上作出更好更充分的改进运用。

附注:本文程序在支持中文的 VC 环境中调试编译,对话框中可支持汉字显示,AppWizar 可生成汉化菜单。如若朋友们使用 VC 时还不支持这一功能的话,按如下操作即可:

(1)首先将 VC5 光盘中的 DevStudio \ SharedIDE \ bin \ ide \ Appwzchs.dll 这一动态链接库文件拷贝至自己硬盘中 VC5 的安装目录中的 DevStudio \ SharedIDE \ bin \ ide 下,该库为简体中文的支持文件。

(2)在设置修改对话框资源属性时,将 Language 的值设为 Chinese (P.R.C.)。

参 考 文 献

[1] Visual C++ 4.0 从入门到精通 [美] Michael J. Young 邱仲潘等译

[2] Visual C++5 开发人员指南 [美] Bennett, D. 徐军等译

(来稿时间:1999 年 4 月)