求是科技

实效编程百例

Visual C++ 6.0 实效编程百例

★求是科技　肖宏伟　编著

★学习编程技巧

★积累编程经验

★剖析功能模块

★突出应用实效

源代码光盘
CD-ROM

人民邮电出版社

实效编程系列

Visual C++ 6.0 实效编程百例

★求是科技　肖宏伟　编著

人民邮电出版社

图书在版编目（CIP）数据

Visual C++ 6.0 实效编程百例 / 求是科技，肖宏伟编著.
—北京：人民邮电出版社，2002.7
ISBN 7-115-10422-0

Ⅰ. V... Ⅱ. ①求... ②肖... Ⅲ. C 语言—程序设计 Ⅳ. TP312

中国版本图书馆 CIP 数据核字（2002）第 049511 号

实效编程百例

Visual C++ 6.0 实效编程百例

◆ 编　　著　求是科技　肖宏伟
责任编辑　张立科

◆ 人民邮电出版社出版发行　北京市崇文区夕照寺街 14 号
邮编　100061　电子函件　315@ptpress.com.cn
网址　http://www. ptpress.com.cn
读者热线　010-67180876
北京汉魂图文设计有限公司制作
北京鸿佳印刷厂印刷
新华书店总店北京发行所经销

◆ 开本：787×1092　1/16
印张：22
字数：534 千字　　2002 年 7 月第 1 版
印数：1-6 000 册　　2002 年 7 月北京第 1 次印刷

ISBN 7-115-10422-0/TP・2962

定价：35.00 元（附光盘）

精确掌握编程语言的语法概念并不意味着可以完成功能强大的应用程序。相反，这才是“万里长征”的第一步。编程水平的提高需要在实际应用中不断地积累点滴经验。本书通过 110 个新颖别致、风格各异的应用实例详细讲解了如何利用 Visual C++的强大功能以及 MFC、API 函数、第三方控件开发应用程序。希望能帮助读者提高 Visual C++的编程水平。

本书实例的分类按照完成功能来划分，尽量将相关的内容归纳在一章中。通过每章的例子讲解有关界面外观设计、多媒体控制与图像处理、时间控制、操作系统、程序控制、磁盘文件、数据库、网络与通信、鼠标和键盘、数学算法与程序发布等内容，以使读者对开发计算机应用的各个方面有所了解，本书是广大 Visual C++开发和使用人员积累编程经验、拓展视野的得力助手。本书的一部分实例突出了实用性，模仿较常见的优秀软件的相关功能，另一部分实例侧重帮助读者理解重点、难懂概念，用最简单的代码说明最关键的问题。

每个实例的讲解分为 3 个步骤：

- 实例目的——讲解本例的功能所在，指出本例要到达的目的和效果，让读者做到心中有数。
- 实现方法——讲解技术原理/设计思路，给出技术原理的合理解释、规范的算法和流程描述，便于读者阅读代码、学习程序设计方法。
- 程序代码——给出具体的实现过程，包括界面设计、编写代码和注释，读者可参照实现。

由于时间、水平限制，书中缺点和不足之处在所难免，敬请读者批评指正。

编者

第 1 章　界面外观

- 带图标的菜单
- 显示倾斜文字
- 文字的颜色渐变
- 设置并叠加透明图片
- 颜色渐变进度条
- 透明窗体

实例 1　带图标的菜单

实例目的

在菜单项旁边增加一个图标可以使该菜单项的意义更明确，而且更能体现专业水准，如 Visual Studio 的菜单，本例介绍如何制作带图标的菜单。

编译并运行本例程序，如图 1-1 所示。文件菜单的【新建】、【打开】、【保存】和【打印】等菜单项旁边都有一个意义非常明确的图标。

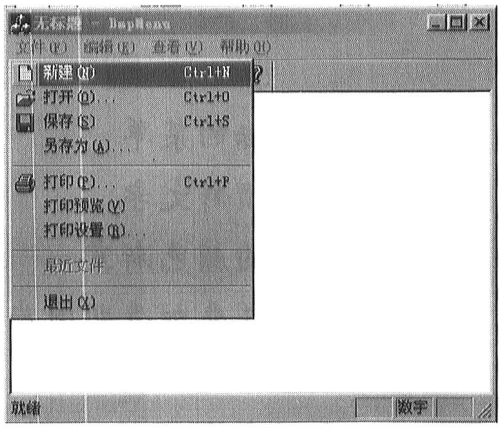

图 1-1　图标菜单

实现方法

本例提供了一个功能比较完善的菜单类 BCMenu，通过该类可以实现图标菜单的效果。先生成一个 BCMenu 类对象，然后用 LoadMenu 函数将菜单资源与菜单对象联系起来，接着通过 ModifyODMenuA 函数修改某个菜单项的图标，最后将其设置为 CMainFrame 的菜单。

程序代码

1. 建一个单文档应用程序 BmpMenu。
2. 将 BCMenu.cpp 和 BCMenu.h 加到工程中。
3. 在 CMainFrame 的头文件中的定义一个 BMenu 的对象（别忘了包含头文件 BCMenu.h）：

```
BCMenu m_menu;
```

4. 在 CMainFrame 中加一个生成图标菜单的函数 NewMenu，其定义如下：

```
HMENU CMainFrame::NewMenu()
{
        m_menu.LoadMenu(IDR_MAINFRAME);
        //将图标加到菜单中
        m_menu.ModifyODMenuA(NULL,ID_FILE_NEW,IDB_FILE_NEW);
        m_menu.ModifyODMenuA(NULL,ID_FILE_OPEN,IDB_FILE_OPEN);
        m_menu.ModifyODMenuA(NULL,ID_FILE_SAVE,IDB_FILE_SAVE);
        m_menu.ModifyODMenuA(NULL,ID_FILE_PRINT,IDB_FILE_PRINT);
        //返回图标菜单的句柄
        return m_menu.Detach();
}
```

5. 在 CBmpMenuApp 的 InitInstance 函数中加入如下代码，设置图标菜单为主窗口的菜单：

```
BOOL CBmpMenuApp::InitInstance()
{
        //其他初始化代码
        //……
        // Dispatch commands specified on the command line
        if (!ProcessShellCommand(cmdInfo))
                return FALSE;

        //将默认的菜单清除
        CMenu* pMenu = m_pMainWnd->GetMenu();
        if (pMenu)pMenu->DestroyMenu();
        //设置菜单为新的图标菜单
        HMENU hMenu = ((CMainFrame*) m_pMainWnd)->NewMenu();
        pMenu = CMenu::FromHandle( hMenu );
        m_pMainWnd->SetMenu(pMenu);
        ((CMainFrame*)m_pMainWnd)->m_hMenuDefault = hMenu;

        // The one and only window has been initialized, so show and update it.
        m_pMainWnd->ShowWindow(SW_SHOW);
        m_pMainWnd->UpdateWindow();

        return TRUE;
}
```

6. 在 CMainFrame 中加入 WM_MEASUREITEM，WM_MENUCHAR 和 WM_INITMENUPOPUP 3 个消息的响应函数，其代码如下：

```
void CMainFrame::OnMeasureItem(int nIDCtl, LPMEASUREITEMSTRUCT lpMeasureItemStruct)
{
        // TODO: Add your message handler code here and/or call default
        BOOL setflag=FALSE;
        if(lpMeasureItemStruct->CtlType==ODT_MENU){
                if(IsMenu((HMENU)lpMeasureItemStruct->itemID)){
                        CMenu* cmenu =
                                CMenu::FromHandle((HMENU)lpMeasureItemStruct->itemID);

                        if(m_menu.IsMenu(cmenu)){
                                m_menu.MeasureItem(lpMeasureItemStruct);
                                setflag=TRUE;
                        }
                }
        }

    if(!setflag)
        CFrameWnd::OnMeasureItem(nIDCtl, lpMeasureItemStruct);
}

LRESULT CMainFrame::OnMenuChar(UINT nChar, UINT nFlags, CMenu* pMenu)
{
        // TODO: Add your message handler code here and/or call default
        LRESULT lresult;
        if(m_menu.IsMenu(pMenu))
                lresult=BCMenu::FindKeyboardShortcut(nChar, nFlags, pMenu);
        else
                lresult=CFrameWnd::OnMenuChar(nChar, nFlags, pMenu);
        return(lresult);
}

void CMainFrame::OnInitMenuPopup(CMenu* pPopupMenu, UINT nIndex, BOOL bSysMenu)
{
        CFrameWnd::OnInitMenuPopup(pPopupMenu, nIndex, bSysMenu);
```

```
        if(!bSysMenu){
            if(m_menu.IsMenu(pPopupMenu))
                BCMenu::UpdateMenu(pPopupMenu);
        }
    }
```

实例 2　显示倾斜文字

实例目的

本例使用 CFont 类和 LOGFONT 结构创建字体，达到显示倾斜文字的效果。编译运行本例程序，如图 1-2 所示，旋转的字符串是“oooo... I am rotating!”。

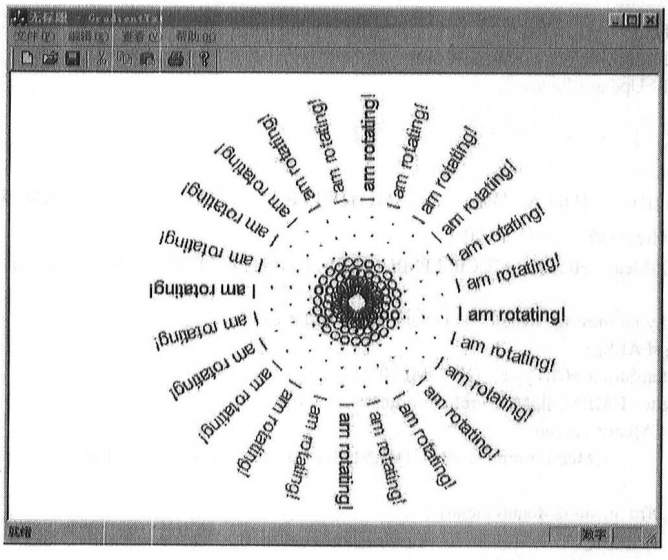

图 1-2　倾斜文字

实现方法

CFont 从 CGdiObject 派生而来，通过 CreateFont、CreateFontIndirect、CreatePointFont 和 CreatePointFontIndirect 等函数创建。如同其他 GDI 对象一样，使用前将用 SelectObject()函数其选进设备场景中就可以了。LOGFONT 结构包含了所要创建字体的详细信息，其中 ifEscapement 成员指定了文本行和 x 轴的角度，角度的单位是十分之一度而不是度，例如，ifEscapement 为 450 表示字体旋转 45°。为确保所有的字体沿坐标系统的同一方向旋转，一定要设置 ifEscapement 成员的 CLIP_LH_ANGLES 位，否则有些字体可能反向旋转，本例每间隔 15°画一个串。

程序代码

1. 通过 AppWizard 生成一个单文档程序 GradientTxt。
2. 在视的 OnDraw()函数中添加如下代码：

```
void CGradientTxtView::OnDraw(CDC* pDC)
{
```

```
        CGradientTxtDoc* pDoc = GetDocument();
        ASSERT_VALID(pDoc);
        //得到客户区的大小
        CRect rcClient;
        GetClientRect (rcClient);
        //创建输出字符串.
        CString str (_T ("oooo. . . I am rotating!"));
        //输出透明红色字体
        pDC->SetBkMode (TRANSPARENT);
        pDC->SetTextColor (RGB (255,0,0));
        CFont font;
        LOGFONT stFont; //字体定义结构
        //设置字体格式
        memset(&stFont, 0, sizeof(LOGFONT));
        stFont.lfHeight=MulDiv(14, -pDC->GetDeviceCaps(LOGPIXELSY), 72);
        stFont.lfWeight=FW_NORMAL;
        stFont.lfClipPrecision=CLIP_LH_ANGLES;
        strcpy (stFont.lfFaceName, "Arial");
        //每隔 15°输出字符串
        for (int nAngle=0; nAngle<3600; nAngle+=150)
        {
            //设定新的旋转角度
            stFont.lfEscapement=nAngle;
            //创建字体并选进设备场景
            font.CreateFontIndirect(&stFont);
            CFont* pOldFont=pDC ->SelectObject(&font);
            //输出字体
            pDC->TextOut(rcClient.left + rcClient.Width()/2,rcClient.top + rcClient.Height()/2,str);
            //原来字体
            pDC->SelectObject(pOldFont);
            font.DeleteObject();
        }
}
```


实例 3　文字的颜色渐变

实例目的

本例使用设备场景 CDC，并且结合定时器显示渐变颜色的文字。编译并运行本例程序，如图 1-3 所示，渐变的文字是“Hello... I am Shading!”。

图 1-3　文字渐变

实现方法

先安装一个定时器，通过 CDC 类的 SetTextColor 函数在 OnTimer()函数中改变字符串的颜色，并且注意尽量使颜色变化的梯度小。

程序代码

1. AppWizard 生成一个单文档程序 TextShadePrj。
2. 在 CTextShadePrjView 的头文件中添加 3 个变量，保存文字颜色的 RGB 值。

```
protected:
        int m_nBlue;
        int m_nGreen;
        int m_nRed;
```

3. 初始化变量。

```
CTextShadePrjView::CTextShadePrjView()
{
        m_nRed = m_nGreen = m_nBlue = 0;
}
```

4. 通过向导添加 OnInitialUpdate()函数，开启定时器。

```
void CTextShadePrjView::OnInitialUpdate()
{
    CView::OnInitialUpdate();
//开启定时器
    SetTimer(0,1000,NULL);
}
```

5. 通过向导添加 OnTimer()函数，更改 RGB 变量值。

```
void CTextShadePrjView::OnTimer(UINT nIDEvent)
{
    CView::OnTimer(nIDEvent);

    //改变文字的 rgb 值，每次渐变 50（自定）
    if(m_nRed<=205)
        m_nRed += 50;
    else if(m_nRed == 255)
        m_nRed = 0;
    else
        m_nRed = 255;

    if(m_nGreen<=205)
        m_nGreen += 50;
    else if(m_nRed == 255)
        m_nGreen = 0;
    else
        m_nGreen = 255;

    if(m_nBlue<=205)
        m_nBlue += 50;
    else if(m_nBlue == 255)
        m_nBlue = 0;
    else
        m_nBlue = 255;

    //重画文字
    Invalidate();
}
```

6. 通过向导添加 OnDestroy()函数，关调定时器。

```
void CTextShadePrjView::OnDestroy()
{
```

```
        //关闭定时器
        KillTimer(0);

        CView::OnDestroy();
}
```

7. 在 OnDraw 函数中设置文字颜色，并输出文字。

```
void CTextShadePrjView::OnDraw(CDC* pDC)
{
        CTextShadePrjDoc* pDoc = GetDocument();
        ASSERT_VALID(pDoc);

        //创建输出字符串.
        CString str (_T ("Hello. . . I am Shading!"));
        //设置字体颜色
        pDC->SetTextColor (RGB (m_nRed,m_nGreen,m_nBlue));

        //输出字体
        pDC->TextOut(100,100,str);
}
```

实例 4　设置并叠加透明图片

实例目的

本例介绍如何绘制透明位图，编译并运行本例程序，如图 1-4 所示。上层的位图一共有 8 个透明部分，透过它们可以清楚地观察到下层的图片。

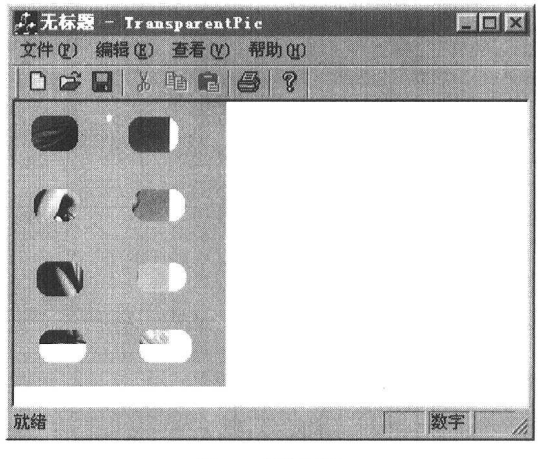

图 1-4　透明图片

实现方法

绘制“透明”位图的关键是创建一个“掩码”位图(mask bitmap)，“掩码”位图是一个单色位图，它是位图中图像的一个单色剪影。在 Windows 编程中，绘图都要用到设备描述表，我们需创建两个内存设备描述表：位图设备描述表(image DC)和“掩码”位图设备描述表(mask DC)。位图设备描述表用来装入位图，而“掩码”位图设备描述表用来装入“掩码”位图。在“掩码”位图设备描述表中制作“掩码”位图的方式是先创建一个单色的 Bitmap，装入 mask DC，然后，以“SRCCOPY”的方式将装有

位图的位图设备描述表绘制(BitBlt)到 mask DC 上。这样，mask DC 的显示平面中的位图即是“掩码”位图。

所以绘制“透明”位图的实际操作步骤是：先将位图设备描述表以“SRCINVERT”的方式绘制(BitBlt)到显示设备描述表上，然后将“掩码”位图设备描述表以“SRCAND”的方式绘制(BitBlt)到显示设备描述表上，最后将位图设备描述表以“SRCINVERT”的方式绘制(BitBlt)到显示设备描述表上。这样除“透明色”外的其余位图部分(图像部分)就被绘制到窗口上了。

程序代码

1. 通过 AppWizard 生成一个单文档的应用程序 TransparentPic。
2. 导入（import）一个底层显示的位图文件，其 ID 为 IDB_BITMAP1。
3. 新建一个位图资源，其 ID 为 IDB_BITMAP2，适当调整其大小，并填充 8 个黑色椭圆部分，其余部分的颜色任意。
4. 给视图类增加一个函数 DrawTransparent，其代码如下：

```
void CTransparentPicView::DrawTransparent(CDC *pDC, int x, int y, COLORREF crColour)
{
    COLORREF crOldBack=pDC->SetBkColor(RGB(255,255,255));
    COLORREF crOldText=pDC->SetTextColor(RGB(0,0,0));
    CDC dcImage, dcMask;
    CBitmap bmp;
    bmp.LoadBitmap(IDB_BITMAP2);
    //IDB_BITMAP1 是待显示位图的资源 ID
    BITMAP bm;bmp.GetBitmap(&bm);
    int nWidth=bm.bmWidth,nHeight=bm.bmHeight;
    //为图像及 mask 各创建一个 DC
    dcImage.CreateCompatibleDC(pDC);
    dcMask.CreateCompatibleDC(pDC);
    //把图像装入 image DC
    CBitmap* pOldBitmapImage=dcImage.SelectObject(&bmp);
    //为“掩码”位图创建一个单色 bitmap
    CBitmap bitmapMask;
    bitmapMask.CreateBitmap(nWidth, nHeight, 1, 1, NULL); //把 mask 位图装入 mask DC
    CBitmap* pOldBitmapMask = dcMask.SelectObject(&bitmapMask); //用透明色创建“掩码”位图
    dcImage.SetBkColor(crColour);//crColor 是位图中的透明色
    dcMask.BitBlt(0, 0, nWidth, nHeight, &dcImage, 0, 0, SRCCOPY);//分 3 步进行实际的绘制
    pDC->BitBlt(x, y, nWidth, nHeight, &dcImage, 0, 0, SRCINVERT);
    pDC->BitBlt(x, y, nWidth, nHeight, &dcMask, 0, 0, SRCAND);
    pDC->BitBlt(x, y, nWidth, nHeight, &dcImage, 0, 0, SRCINVERT); //恢复原先设置
    dcImage.SelectObject(pOldBitmapImage);
    dcMask.SelectObject(pOldBitmapMask);
    pDC->SetBkColor(crOldBack);
    pDC->SetTextColor(crOldText);
}
```

5. 在视图类的 OnDraw 函数中增加显示位图的代码：

```
void CTransparentPicView::OnDraw(CDC* pDC)
{
    CTransparentPicDoc* pDoc = GetDocument();
    ASSERT_VALID(pDoc);
    //画底层位图
    CBitmap bmp;
    bmp.LoadBitmap(IDB_BITMAP1);
    BITMAP bm;
    bmp.GetBitmap(&bm);
    int nWidth=bm.bmWidth,nHeight=bm.bmHeight;
    CDC MemDC;
    MemDC.CreateCompatibleDC(pDC);
    CBitmap* pOldBmp = MemDC.SelectObject(&bmp);
```