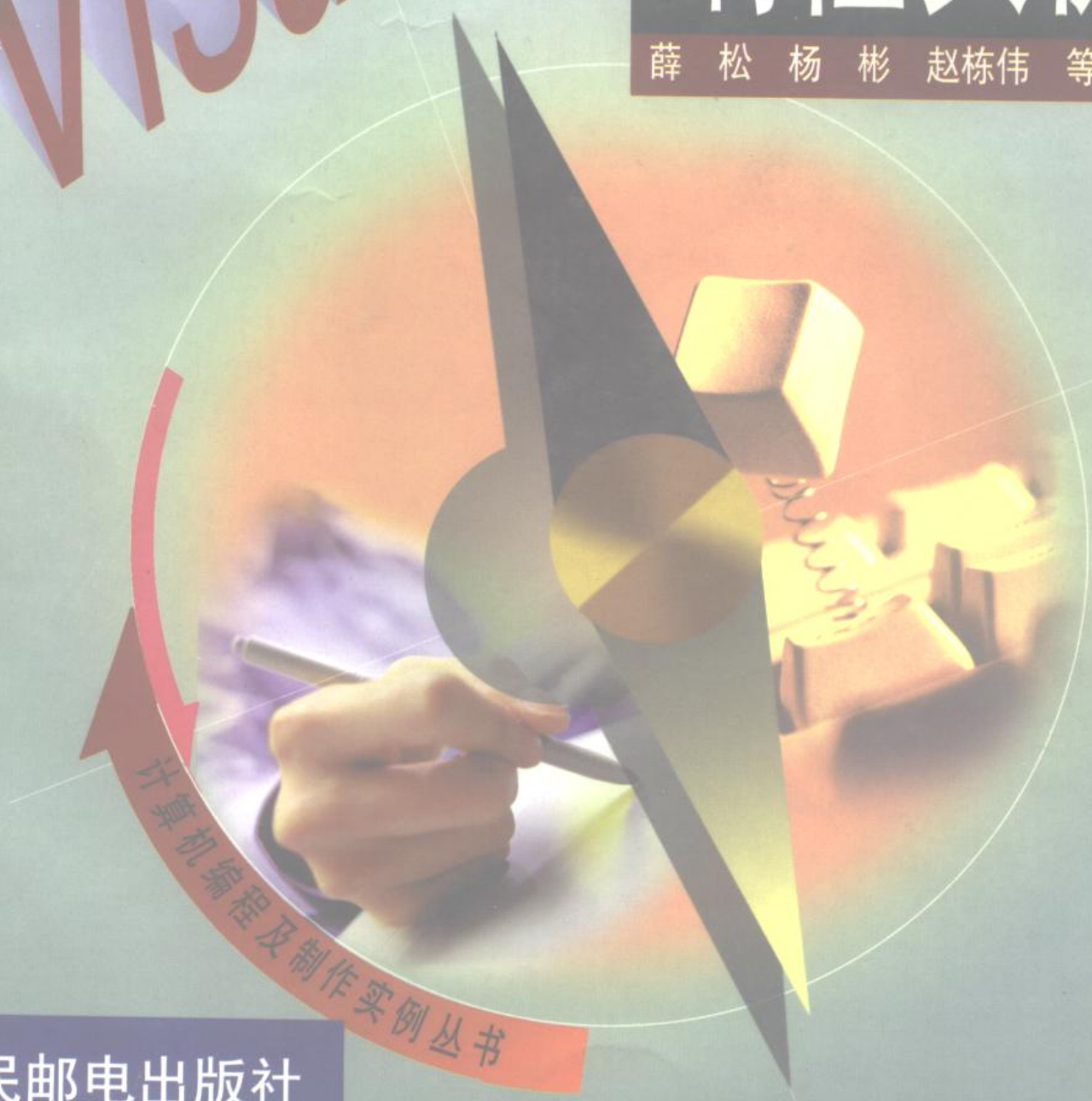
Visual C++ 6.0 编程实例

薛 松 杨 彬 赵栋伟 等编著



计算机编程及制作实例丛书

人民邮电出版社

计算机编程及制作实例丛书

Visual C++ 6.0 编程实例

薛 松 杨 彬 赵栋伟 等编著

人 民 邮 电 出 版 社

内 容 提 要

Visual C++ 6.0 是一个基于 Windows 98 操作平台的强大的编程工具，它为计算机编程人员提供了编程的新天地。本书以实例的方式详细地介绍了 Visual C++ 6.0 的基础知识、编程方法以及相关技巧。

本书所选的 29 个实例按照由浅入深的原则，对 Visual C++ 6.0 在编程实际应用方面的基本使用方法和一些高级编程技术进行了详尽的讲解和分析，可帮助编程人员解决许多工作中的实际问题。通过阅读本书，能使已有一定经验的计算机程序员进一步提高编程水平。

本书适合所有计算机用户和从事编程工作的技术人员阅读参考，也可供学习计算机知识的其它读者使用。

计算机编程及制作实例丛书

Visual C + + 6.0 编程实例

◆ 编　　著　薛　松　杨　彬　赵栋伟　等
责任编辑　王晓明

◆ 人民邮电出版社出版发行　北京市崇文区夕照寺街 14 号
北京顺义振华印刷厂印刷
新华书店总店北京发行所经销

◆ 开本：787 × 1092　1/16
印张：38.25
字数：965 千字　　1999 年 8 月第 1 版
印数：1 – 6 000 册　　1999 年 8 月北京第 1 次印刷

ISBN 7-115-07968-4/TP·1214

定价：56.00 元

丛书前言

随着我国计算机应用的普及与发展，各种不同用途的应用软件不断地被开发出来，并迅速地为人们掌握和使用。目前，除了专业软件开发人员外，许多一般的计算机用户也已经开始自己开发一些应用程序。计算机的应用开发工作包括很多方面，例如：图像处理开发、动画制作、专项应用程序开发、通用应用程序开发、游戏软件的制作等等。开发这些项目需要使用各种各样的开发工具，以便能够大大地缩短开发周期并减少开发费用。因此，很多从事计算机应用开发工作的人员迫切需要了解和掌握各种计算机编程和制作的方法。为适应广大读者的这种需求，我们特地组织出版了这套《计算机编程及制作实例丛书》，以各种实例的方式，向读者介绍计算机编程和制作方面的知识。

这套丛书主要是针对应用程序开发者而编写的，力求使读者能够轻松地学习开发的方法并熟练地掌握使用相关的开发工具。对于需要编制应用程序或利用计算机进行开发的用户，这套丛书不仅是一个开发指南，而且还给出了其它图书很少涉及到的开发技巧。通过本丛书中介绍的一个个具体的例子，用户便可以比较容易地掌握各种开发软件的功能和使用技巧。

由于这套丛书是以实例的方式由浅入深地讲述编程和开发内容，所以不仅适用于那些刚刚开始从事编程或开发制作的计算机使用者，而且对已经有较多开发经验的计算机用户也同样有较大的帮助。易学易懂且实用性强是本丛书的特点，书中每一个例子里的程序或方法，读者都可以直接引用，相信会使广大读者受益匪浅。

为把这套丛书编得更好，我们真诚地希望广大读者提出宝贵的意见和建议。

前　言

本书是一本有关 Microsoft Visual C++ 6.0 应用程序开发的图书，书中通过一系列的实际的例子向读者介绍如何利用 Microsoft Visual C++ 6.0 的强大功能来开发各种应用程序。

许多程序员过去都使用过 Borland C/C++ 的集成开发环境(Integrate Development Environment，IDE)，它最早是在 Turbo Pascal 上实现的。实际上 IDE 仅仅是让编辑器和编译器一同工作，这样软件开发人员就可以在编辑器里写下源代码，然后随时通过菜单上的菜单项或者快捷键启动编译器进行编译。当编译器发现源代码的错误时，会指出错误的位置和原因，以便于开发人员进行修改。IDE 的出现给软件开发人员带来了极大的方便。

Microsoft 公司在 1987 年推出了基于 IDE 的 C 语言集成开发环境 Quick C，并将其封装在 Microsoft 公司的 Big C 5.0 软件包里。Quick C 给使用 C 语言的编程引进了一些很有用的东西，但由于 DOS 操作系统对内存的限制，软件开发人员使用 Quick C 的集成开发环境后，剩给应用程序可以使用的空间就很有限了。因此有的时候，软件开发人员不得不离开 Quick C 的集成开发环境来调试和运行程序。这样一来就使 Quick C 的集成开发环境脱离了它的初衷，另一方面让人感到软件开发更加神秘和复杂了。

Windows 的出现改变了这种情况， Microsoft Windows 3.1 把计算机引进了正式的 IDE 时代。Microsoft 公司不但在界面上将 Microsoft C/C++ 7.0 升级到了一个新天地，而且把更多的力量放在了编译器内部。在 Microsoft C/C++ 7.0 里，Microsoft 公司引进了 MFC(Microsoft Foundation Class)库 1.0 版。当时的 MFC 还很不完善，也并不好用，但如果没有它，C++就不会像今天这样流行了。Microsoft C/C++ 7.0 给人的印象最深的恐怕就是编译速度特别慢，当然这与当时的计算机的速度很慢也有较大的关系。随着 Microsoft 公司软件版本的升级，Microsoft C/C++ 8.0 很快发布了。Microsoft C/C++ 8.0 是一个真正的基于 Windows 的集成开发环境，Microsoft C/C++ 8.0 也就是人们熟知的 Visual C++ 1.0 版，在这个版本中 Microsoft 公司对 MFC 做了大量的完善工作。然后就是 Visual C++ 1.5 版，国内使用的比较多的是它的一个更新一些的版本 Visual C++ 1.51。Visual C++ 1.51 是一个很好用的集成开发环境，也就是从 Visual C++ 1.5 之后 Microsoft 公司基本放弃了对 16 位编程的支持，而把全部力量放在 32 位的编程上。MFC 也发展得很快，它确实给软件开发带来了好处。在经历了比较短的 Visual C++ 2.0 版本后，为了和 MFC 的版本同步，Visual C++没有经历 Visual C++ 3.0，直接到了 Visual C++ 4.0，从而结束了 Visual C++和 MFC 版本不统一的小小混乱。然而，计算机技术的发展总是很快，没有多久 Visual C++和 MFC 又使用了不同的版本号。

Internet 的流行同样也影响了 Visual C++和 MFC。在 Visual C++ 4.0 的版本里，引进了对 Internet 编程的支持并设计了新的类库。Visual C++ 5.0 又增加了一些新类，对产品的界面也做了很多改动。对于 Microsoft 公司最擅长的在线帮助，Visual C++ 5.0 的帮助确实做得相当好，随着软件项目越来越大，Visual C++5.0 提供了对同一个开发组内进行类和其它代码共享的支持，同时合并了 Active Template Library，并显著地改善了编译器对代码优化的能力。

1998 年，Microsoft 公司推出了 Visual C++ 6.0，由于 Microsoft 公司对于 Visual C++ 5.0

的界面还是满意的，因此在 Visual C++ 6.0 的界面上基本没有什么变化。Visual C++ 6.0 对 Internet 的编程提供了更好的支持，专门提供了几个和 Internet 相关的类，软件开发人员甚至不用写一行代码就可以利用它做一个很不错的 Internet 浏览器软件。Visual C++ 6.0 把帮助专门抽了出来，MSDN 库作为一个单独的应用程序来使用。Visual C++ 6.0 的推出，为广大软件开发人员提供了更加方便好用的开发工具，但是，许多软件开发人员给每个函数都提供一个小例子的希望，由于种种原因，却仍然没有被 Microsoft 公司所采纳。

在计算机技术发展突飞猛进的今天，在我们生活的方方面面，计算机几乎无处不在。在这种情况下，社会各行各业需要更多的软件开发人员和更多的应用软件。本书的作者根据当前我国计算机的普及应用情况和许多读者的实际需要，把自己多年来对软件开发的理解、经验和技巧融入本书的一个个实际的例子中，供广大读者和软件开发人员学习和参考。

由于这是一本介绍 Visual C++ 6.0 应用实例的书，所以书中没有从最基本的一般知识讲起，而是假定读者已经学习过 C 和 C++编程语言，并进行过一些 Windows 编程，对 MFC 有一定的基本认识。当然，本书的读者也不一定必须是编程方面的行家，但如果用户理解一些基本概念，如指针、类和消息等，就会发现，文本和范例代码更容易读懂。我们真诚地希望这本书的出版能为广大读者和软件开发人员带来一些学习和工作中的方便。

本书主要由薛松、杨彬和赵栋伟编写，参加本书编写的还有：王红、李志媛、陈旭、刘彬、松蕊、王雪梅、李志新、肖红妹、李晓娟、赵家根、金威娣、赵晓波、徐臻青、曹铮、张勇、周晓兰、韩平、牛玲、顾克勤、刘宁辉、刘勇等，他们都为本书的编写做出了很大的贡献。

由于计算机技术的发展十分迅速，加上我们自身水平有限，所以书中错误在所难免，诚挚地希望广大读者批评指正。我们的电子邮件地址是 xuesongs@public.east.cn.net、yangbin_b_c@usa.net 和 zhaodw@163.net。

作 者

1999 年 3 月

目　录

实例 1　创建自绘式按钮类

主要内容

本例提要

下压式按钮是 Wndows 程序中使用最为广泛的控件之一，标准的下压按钮是一个矩形按钮上加一个标签。在早期的 Windows 程序中所有的按钮都是这样标准的下压按钮，但是后来人们渐渐地希望按钮上不止是显示简单的文本，更希望能显示图形。为此，Microsoft 在 MFC 中为用户提供了 CbitmapButton 类，它可以利用四幅位图来表示按钮的释放、按下、得到焦点和被禁止这四种状态，另外，还可以直接利用 Cbutton 类，它可以使用 SetIcon、SetCursor、SetBitmap 等几个函数来在按钮上显示图像。但这些标准控件还有其缺点，例如不能改变按钮的形状、不能设置标签的文本（文本只能作为图像存储在位图、光标等资源中）等等。在下面的例子中，我们将介绍一些特殊效果的按钮控件，包括立体效果圆按钮、Flat 风格按钮、包含位图和文字的下按钮等。这些按钮可以使用户的程序界面显得更加丰富多彩。

技术概要

本例中包含多个控件例子，实现它们的基本技术则是自绘制控件。Windows 支持自绘制控件，缺省设置下 Windows 将绘制消息发送到控件的父窗口，从而在父窗口可以定制控件的可视化外观和行为。但这样有一个缺点是从 CWnd 类派生的类必须有控件自绘制行为的代码。这样，如果在两个不同的对话框中有两个类似的控件，就必须在这两个地方为它们各自编写自己的定制行为的代码，这使得代码的重用变得不可能。

为了解决这个问题，MFC 发展了支持自绘制控件的结构体系。MFC 为标准绘制消息提供了缺省的实现代码，它译解自绘制消息并将绘制参数发送给控件本身。由于绘制、消息监测是在控件类中实现的，而不是在控件的拥有者中实现的，所以叫做“自绘制”。

这种机制允许用户根据自己的意愿创建可重用的控件类。在创建中，用户只需简单地说明 Cbutton 的子类并处理 DrawItem 方法就可以实现自绘制按钮。这是一种面向对象的程序设计方法。由于传递给 DrawItem 方法的结构细节是相当复杂的，从而使得创建定制按钮也变得很复杂。下面的例子中提供了一些很有特色的按钮，通过这些实例，读者可以一步步地深入了解自绘制按钮的细节。

实例过程（一）——创建圆形下压按钮

创建测试项目

选择 File 菜单中的 New 命令，在弹出的 New 对话框中选择 Project 标签页，然后选择列

表中的 MFC AppWizard (exe)，在 Project name 编辑框中输入工程名 NewButtons。在第一步选择基于单文档选项，由于在程序中不需要文档数据，因此去掉文档－视结构支持检查框，如图 1－1示。

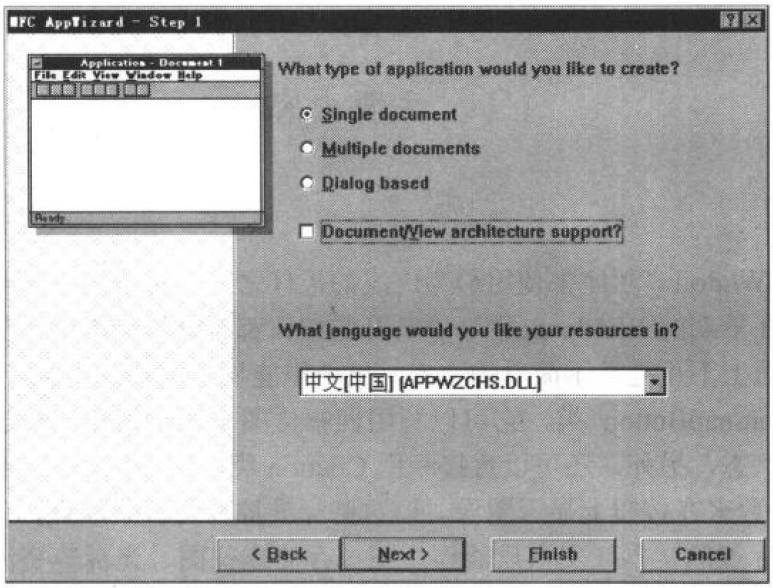


图 1－1　在 AppWizard 的第一步中去掉文档－视结构支持

按下 Finish 按钮，AppWizard 将自动创建如下的类：

- 应用类 CNewButtonsApp 在文件 NewButtons.h 和 NewButtons.cpp 中
- 框架类 CMainFrame 在文件 MainFrm.h 和 MainFrm.cpp 中
- 视类 CChildView 在文件 ChildView.h 和 ChildView.cpp 中

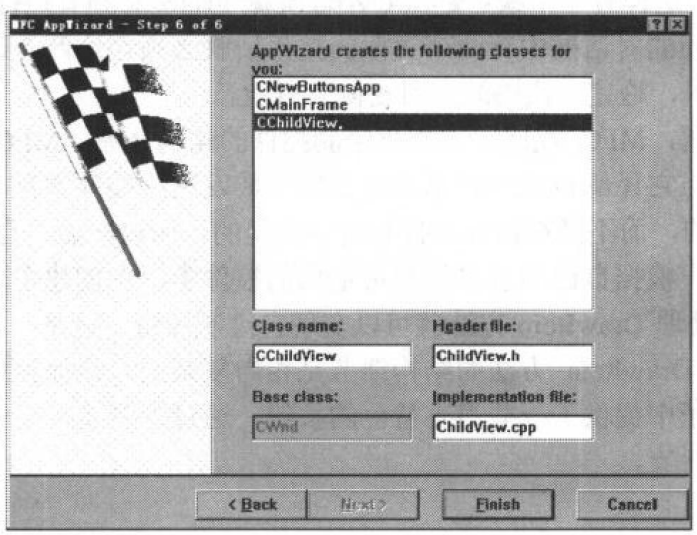


图 1－2　AppWizard 自动生成的类结构

其他特征包括：

- Initial toolbar in main frame
- Initial status bar in main frame

- 3D Controls
- Uses shared DLL implementation (MFC42.DLL)
- ActiveX Controls support enabled
- Localizable text in:　中文[中国]

如图 1－2所示，由于去掉了文档－视结构支持，因此在项目中将不再出现文档类了，对于一些简单的项目来讲，VC6 添加的这个选项实在是有助于简化程序的结构。

☛ 添加新 Button 类与结构定义

1. 添加新类 XMemDC

使用 ClassView 或 ClassWizard 向工程中添加一个新类 XCirButton，在 New Class 对话框中输入类信息（如图 1－3所示）：

类名：XMemDC

基类：CDC

文件：MemDC.h 和 MemDC.cpp

注意，要在 Class Type 下拉列表框中选择 generic Class。

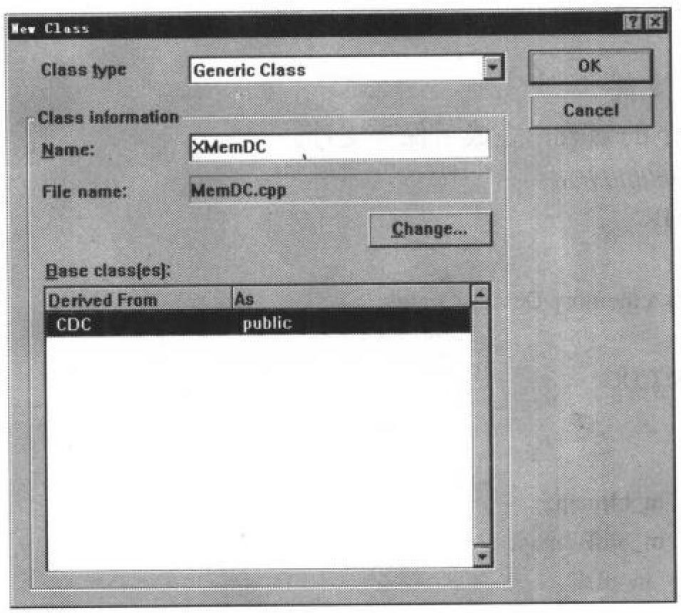


图 1－3　利用 ClassWizard 添加类 XMemDC

2. 添加新类 XCirButton

使用 ClassView 或 ClassWizard 向工程中添加一个新类 XCirButton，在 New Class 对话框中输入类信息：

类名：XCirButton

基类：CButton

文件：CirButton.h 和 CirButton.cpp

按下 OK 按钮，ClassWizard 会自动生成必需的类框架结构，如图 1－4所示。

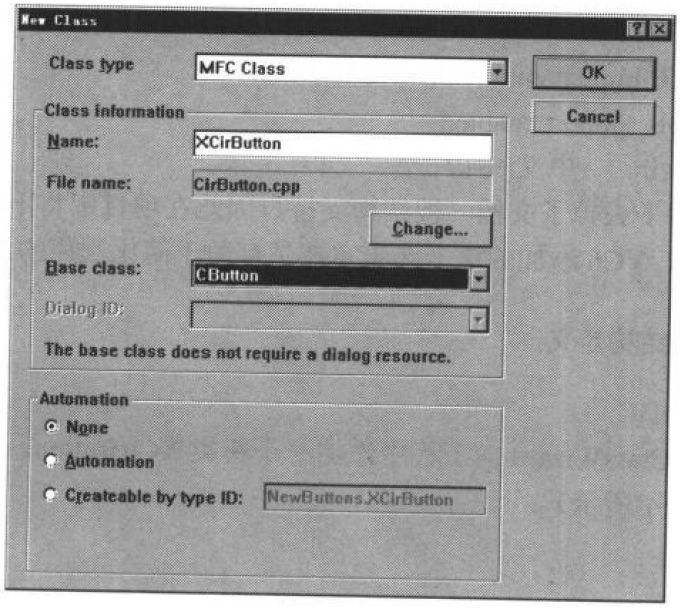


图 1－4　利用 ClassWizard 向项目中添加类 XCirButton

☛ 添加和修改按钮类代码

1．修改 MemDC.h 文件

对 MemDC.h 作如下的修改，完成后保存文件。

```
//////////////////////////////////////////////
// XMemDC - memory DC
//
// This class implements a memory Device Context
//
class XMemDC : public CDC
{
private:
    CBitmap*    m_bitmap;
    CBitmap*    m_oldBitmap;
    CDC*        m_pDC;
    CRect       m_rcBounds;
public:
    //类构造函数
    XMemDC(CDC* pDC, const CRect& rcBounds) : CDC()
    {
        CreateCompatibleDC(pDC);
        m_bitmap = new CBitmap;
        m_bitmap->CreateCompatibleBitmap(pDC, rcBounds.Width(), rcBounds.Height());
        m_oldBitmap = SelectObject(m_bitmap);
        m_pDC = pDC;
        m_rcBounds = rcBounds;
    }
```

```
    //析构函数
    ~XMemDC()
    {
            m_pDC->BitBlt(m_rcBounds.left, m_rcBounds.top,
                          m_rcBounds.Width(), m_rcBounds.Height(),
                          this,
                          m_rcBounds.left, m_rcBounds.top, SRCCOPY);
            SelectObject(m_oldBitmap);

            if (m_bitmap != NULL)
                    delete m_bitmap;
    }

    XMemDC* operator->()
      {
            return this;
    }
};
```

2. 修改 MemDC.cpp 文件

修改文件 MemDC.cpp，删除 ClassWizard 添加的 XMemDC()和~MemDC()实行方法，只保留如下的代码：

```
#include "stdafx.h"
#include "MemDC.h"
```

3. 为 XCirButton 添加成员函数 DrawItem

用 ClassWizard 为 XButton 添加虚函数 DrawItem，如图 1－5所示。

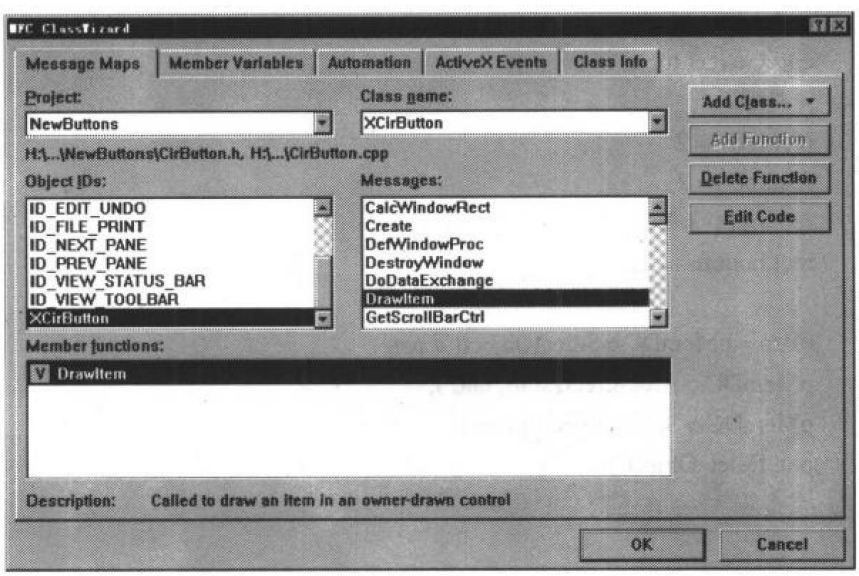


图 1－5　为 XCirButton 添加 DrawItem 方法

Windows 在更新按钮显示时要调用 DrawItem 方法，它调用函数完成绘制边界、填充客户区、绘制文字等工作。按照下面的方法修改 DrawItem 的代码：

```
void XCirButton::DrawItem(LPDRAWITEMSTRUCT lpDrawItemStruct)
{
    // TODO: Add your code to draw the specified item
    CRect Rect = lpDrawItemStruct->rcItem;
    CDC *pDC = CDC::FromHandle(lpDrawItemStruct->hDC);
    CRgn rgn;
    CRect rect;
    rect = Rect;
    UINT state = lpDrawItemStruct->itemState;

    XMemDC  *pMemDC = new XMemDC( pDC, Rect);
    CBrush hbr;
    hbr.CreateSolidBrush(GetSysColor(COLOR_BTNFACE));
    //pDC->FillRect( &Rect, &hbr);
    pMemDC->FillRect( &Rect, &hbr);
    hbr.DeleteObject();
    if( !(state & ODS_DISABLED) )
    {
        if (state & ODS_SELECTED)
        {
            CPen pen( PS_SOLID, 2, GetSysColor( COLOR_3DHILIGHT ) );
            CPen *pPen;
            CPoint start, end;

            start.x = rect.left;
            start.y = rect.bottom;
            end.x = rect.right;
            end.y = rect.top;

            rect.left += 2;
            rect.top += 2;
            rect.right -= 2;
            rect.bottom -= 2;

            pPen = pMemDC->SelectObject( &pen );
            pMemDC->Arc( &rect, start, end );
            pMemDC->SelectObject( pPen );
            pen.DeleteObject();
        }
    }

    rgn.CreateEllipticRgn( rect.left, rect.top, rect.right, rect.bottom );
```

```
        pMemDC->SelectClipRgn( &rgn );
        GradientFill( pMemDC, &Rect );

        SetRound();

        CString title;
        GetWindowText( title );
        if( title.GetLength() != 0 )
        {
            DrawButtonText( pMemDC, &Rect, title );
            title.ReleaseBuffer();
        }

        delete pMemDC;
    }
```

4. 为 XCirButton 类添加 GradientFill 函数

GradientFill 用于为按钮着色，它通过填充一系列的椭圆来体现一个球形体的立体效果。每个椭圆的颜色深度逐渐增加，而椭圆逐渐下移，当所有的椭圆画完后，就出现了一个椭球体。

```
void XCirButton::GradientFill(CDC *pDC, CRect *pRect)
{
    CBrush* pBrush[64];

    for (int i=0; i<64; i++)
    {
        pBrush[i] = new CBrush (RGB (0, 0, 255 - (i * 4)));
    }

    int nWidth = pRect->Width ();
    int nHeight = pRect->Height ();
    CRect rect;

    for (i=0; i<nHeight; i++)
    {
        rect.SetRect (0, i, nWidth, i + 1);
        //TRACE("SetRect: left=%d top=%d right=%d bottom=%d\n",
        //      rect.left, rect.top, rect.right, rect.bottom);
        pDC->FillRect (&rect, pBrush[(i * 63) / nHeight]);
    }

    for (i=0; i<64; i++)
        delete pBrush[i];
}
```

5. 为 XCirButton 类添加 SetRound 函数

函数 SetRound 用于设置按钮控件的区域。

```
void XCirButton::SetRound(void)
{
    HRGN rgn;
    CRect wrect;
    GetClientRect(&wrect);

     SetWindowRgn( NULL, FALSE );
    rgn = CreateEllipticRgn( wrect.left, wrect.top,
                             wrect.right, wrect.bottom );

    SetWindowRgn(rgn, TRUE);
}
```

6. 为 XCirButton 类添加 DrawButtonText 方法

函数 DrawButtonText 用于绘制按钮的标题。为了简单起见，在这里固定使用白色作为窗口标题的颜色，用户也可以修改代码设置自己想使用的颜色。

```
void XCirButton::DrawButtonText(CDC *pDC, CRect *pRect, CString &text)
{
    CFont font;
    int nHeight = -16;

    font.CreateFont (nHeight, 0, 0, 0, FW_BOLD,
        TRUE, 0, 0, DEFAULT_CHARSET, OUT_CHARACTER_PRECIS,
        CLIP_CHARACTER_PRECIS, DEFAULT_QUALITY, DEFAULT_PITCH |
        FF_DONTCARE, "Times New Roman");

    pDC->SetBkMode(TRANSPARENT);
    pDC->SetTextColor(RGB (255, 255, 255));

    CFont* pOldFont = pDC->SelectObject (&font);
    pDC->DrawText (text, -1, pRect, DT_SINGLELINE | DT_CENTER |
                                    DT_VCENTER);

    pDC->SelectObject (pOldFont);

}
```

7. 修改 CirButton.cpp 文件，添加头文件

修改文件 CirButton.cpp，添加下面的头文件包含代码：

```
#include "MemDC.h"
```

☛ 创建测试代码

1.　添加新菜单项

在 ResourceView 中打开菜单 IDR_MAINFRAME，向其中添加菜单条“测试”，在下面加入一个菜单项 IDM_TEST_CIRBUTTON，如图 1－6所示。

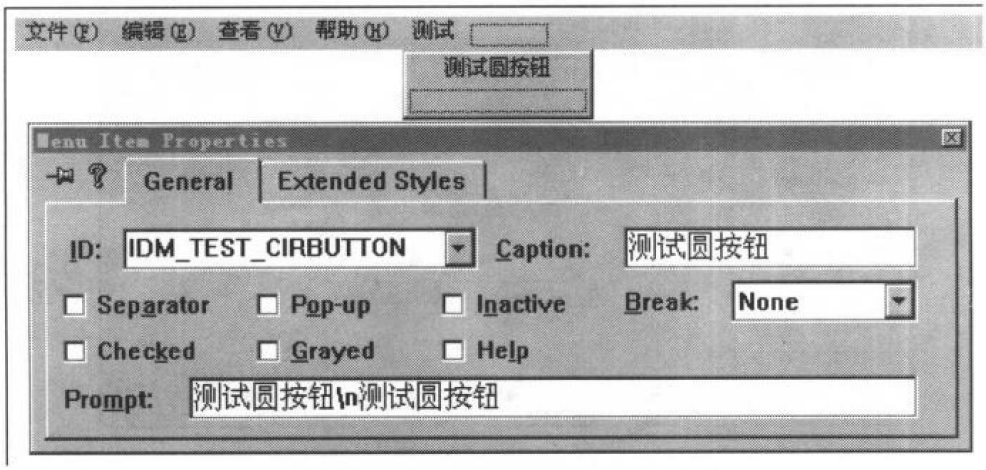


图 1－6　添加测试菜单项

2.　创建测试对话框

利用 ResourceView 加入对话框 IDD_CIRBUTTON_DLG，设置两个按钮的 Owner draw 属性，如图 1－7所示。

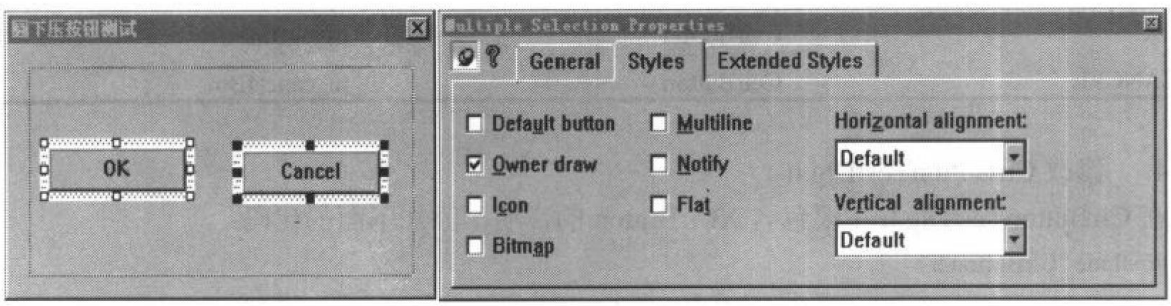


图 1－7　设置按钮的 Owner draw 属性

3.　创建一个新类 CCirButtonTestDlg

该对话框要和一个类联系起来，在 ResourceView 中双击对话框，ClassWizard 会弹出对话框询问是否为对话框创建新类，按下 OK 按钮，按下面的内容填写对话框：

类名：CCirButtonTestDlg

基类：CDialog

文件：CirButtonTestDlg.h, CirButtonTestDlg.cpp

对话框：IDD_CIRBUTTON_DLG

如图 1－8所示即为添加新类 CcirButtonTestDlg 时的对话框以及所添加的内容。

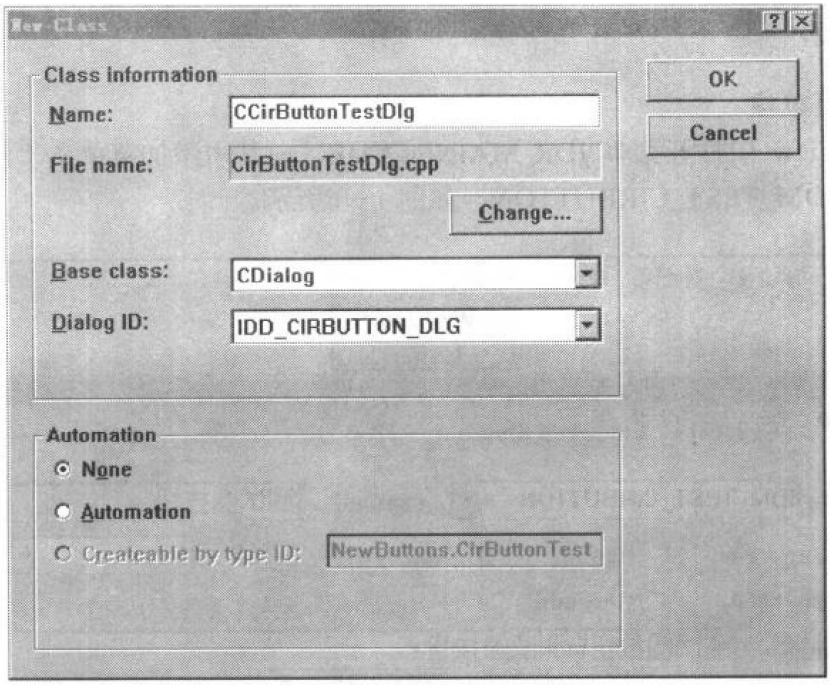


图 1－8　添加新类 CCirButtonTestDlg

利用 ClassWizard 加入成员变量，如表 1－1所示。

表 1－1　　向 CCirButtonTestDlg 中加入控件变量

控件 ID	类型	成员
IDOK	CcirButton	m_okBtn
IDCANCEL	CcirButton	m_cancelBtn

4.　修改 CirButtonTestDlg.h

在 CirButtonTestDlg.h 中要包含 XCirButton 的声明，添加下面的代码：

```
#include "CirButton.h"
```

5.　添加事件响应函数

利用 ClassWizard 添加 IDM_TEST_CIRBUTTON 的响应函数 OnTestCirbutton：

```
void CMainFrame::OnTestCirbutton()
{
    // TODO: Add your command handler code here
    CCirButtonTestDlg dlg;
    dlg.DoModal();
}
```

6.　修改 MainFrm.cpp

在 MainFrm.cpp 中添加 CCirButtonTestDlg 的头文件：

```
#include "CirButtonTestDlg.h"
```