第11章 绘　图

位图和图标允许给应用程序添加色彩和风格。因为所有的 Windows界面在本质上都是相同的，实际上，商标和启动窗口是用来区别不同应用程序外观的唯一方法。显然，绘图对于创建自己的控件，以及在 CAD应用程序中显示图形也是重要的。可以用本章中的例子给他们的应用程序增添一些独特的风格。

例53　绘制图形　讨论一些MFC绘图工具。

例54　绘制文本　讨论怎样绘制文本。

例55　从任意位置装入一个图标并绘制　讨论从磁盘装入一个图标的方法。

例56　从任意位置装入一个位图和绘制一个位图　讨论从磁盘装入一个位图的方法。

例57　从文件中创建一个位图　采用位图装入过程控制位图，以包括创建一个位图调色板以及根据需要替换位图颜色的能力。

例58　创建一个自绘位图　用我们可得的MFC绘图工具在内存中创建一个位图。

11.1　例53　绘制图形

目标

绘制一个如图11-1中看到的图形。

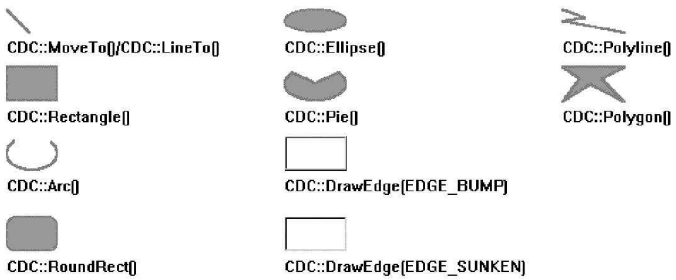

图11-1　用MFC可以绘制这些图形

策略

应用MFC的CDC类的不同绘图工具。

步骤

在Windows应用程序中绘图用一个设备环境完成，该设备环境定义用户在哪里绘图、用什么工具绘图以及采用什么绘图模式；设备环境取消了重复的参数调用，因而有助于简化Windows绘图工具。参见第4章有关该主题的详细内容。

1. 创建一个设备环境

1) 如果处理一个WM_PAINT消息或其他类似的消息，则可以提供一个设备环境，如果没有提供，则必须自己创建一个；如果要绘制一个屏幕，可以用下面的代码创建一个设备环境，这里的pWnd是CWnd类的实例的指针。该类的实例应该拥有需要绘制的窗口。

```
CDC*pDC=pWnd→ GetDC( );
```

2) 如果创建一个自己的设备环境，用完后必须销毁它；否则，会发生另一种内存泄漏，称为资源泄漏。销毁一个设备环境，用：

```
pWnd→ ReleaseDC (pDC);
```

注意 调用这些函数的类应控制需要绘制的窗口；如果没有这样的窗口类实例存在，可以使用AfxGetMainWnd( )→GetDC( )或::GetDC (NULL)，它将返回一个用桌面窗口特征初始化的设备环境。后面的函数返回一个设备环境句柄，在此处使用该返回的设备环境句柄，必须把它封装到CDC类的实例中。

设备环境具有一些预定义的绘图特征，其中之一是绘制线条的宽度和颜色，这一特征实际上包含在该设备环境指向的对象中。该对象叫做画笔(Pen)，它默认为绘制一个像素宽的黑色线条；如果需要别的特征，则需要创建自己的画笔对象。

2. 创建一个画笔

1) 根据画线所需的特征创建CPen类的一个实例。

```
CPen pen(
   PS_SOLID,              // solid line also
                          // PS_DASH,PS_DOT,PS_DASHDOT,
                          // PS_DASHDOTDOT, PS_INSIDEFRAME and
                          // PS_NULL
   2,                      // width in pixels
   RGB(128,128,128));     // color
```

2) 让设备环境指向该新画笔对象，但还要保存一个旧画笔的指针，以便以后能恢复它。

```
Cpen *pPen=pDC->SelectObject(&pen)；//save old pen
```

另一个在设备环境中预定义的特征是填充色(用来绘制封闭图形内部的颜色)，它与一个绘制封闭图形的函数一起使用；默认的颜色是白色，但通过告知设备环境使用一个新的画刷对象，可以改变填充颜色。

3. 创建一个画刷

1) 用需要的颜色创建CBrush类的一个实例。

```
CBrush(RGB(128,128,128)；//clolor
```

2) 让设备环境指向该新画刷对象，但还要保存旧对象的指针，以便以后能恢复它。

```
CBrush *pBrush=pDC->SelectObject(&brush)；//save old brush
```

注意 上面两步代表了创建CPen和CBrush类实例的最基本的方法，有关其他方法参见第4章和MFC文档。

4. 用CDC类成员函数绘制图形

1) 用该设备环境画一条直线，用：

```
pDC->MoveTo(5,5);
pDC->LineTo(25,25);
```

一条线的起始和结束坐标分成两个函数调用，以便绘制多条相邻的线条时，具有最小量的参数进出栈。这里的数字，以及本例中的其余数字，都用逻辑单位；当绘制屏幕时，逻辑单位等于屏幕像素。

注意 本例和本章中的其他例子使用实例调用参数；当然，也可以使用自己的函数。

2) 绘制一个矩形，用：

```
pDC->Rectangle(CRect(5,55,50,85));
```

3) 绘制弧，用：

```
pDC->Arc(CRect(5,115,50,145),      // area encompassing the arc
   CPoint(5,115),                  // starting point of the arc
   CPoint(50,115));                // end point
```

4) 绘制圆角矩形，用：

```
pDC->RoundRect(CRect(5,185,50,215),
   CPoint(15,15));   // distance from corner to draw arc
```

5) 绘制椭圆或圆，用：

```
pDC->Ellipse(CRect(250,5,305,25));
```

6) 绘制饼图，用：

```
pDC->Pie(CRect(250,55,305,85),       // area encompassing chart
CPoint(250,55),                      // starting point
CPoint(305,55));                     // end point
```

7) 绘制一个外观与控件、弹出式和重叠窗口等同的窗口框架，用：

```
pDC->DrawEdge(CRect(250,115,305,145),
   EDGE_BUMP,   // also EDGE_ETCHED,EDGE_RAISED,EDGE_SUNKEN
   BF_RECT);    // also BF_LEFT,BF_BOTTOM,BF_RIGHT,BF_TOP
```

8) 绘制一系列相临的线，用：

```
POINT pt[8];
pt[0].x=495;
pt[0].y=5;
pt[1].x=510;
pt[1].y=10;
pt[2].x=515;
pt[2].y=12;
pt[3].x=495;
pt[3].y=15;
pt[4].x=550;
pt[4].y=25;
pDC->Polyline(pt, 5);
```

9) 绘制一个具有多条相临边的封闭图形，用：

```
pt[0].x=495;
pt[0].y=55;
pt[1].x=550;
pt[1].y=55;
pt[2].x=530;
pt[2].y=65;
pt[3].x=550;
pt[3].y=85;
```

```
pt[4].x=520;
pt[4].y=70;
pt[5].x=495;
pt[5].y=85;
pt[6].x=510;
pt[6].y=65;
pt[7].x=495;
pt[7].y=55;
pDC->Polygon(pt, 8);
```

说明

- 一个设备环境只是一段内存，初始化计划进行绘制的设备的所有特征。使用一个设备环境可以提高性能，因为不用传输 10或15个参数到CDC:LineTo( )，而只需传输两个。线条的起始位置、颜色、厚度、绘制的合法位置、绘制的设备类型已经在环境中定义。
- 有4种类型的设备环境，除了屏幕环境以外，还有一个内存设备环境，一个打印机设备环境和一个信息的设备环境。信息设备环境比其他三种类型更简洁，并只用来获取一个设备的信息而不是用来绘图。在本章的后面部分将使用一个内存设备环境，在内存中创建一个位图图像。一个打印机设备环境用来满足需要，虽然，它可能在创建的时候与你期望的需要不同。
- 绘制图形远不止本例所介绍的内容，参见第 4章和MFC文档，以获取更详细的内容。

CD说明

在CD上执行该工程时，可以看到视图充满了一些可以用 MFC和Windows API绘制的基本图形。

11.2 例54 绘制文本

目标

在视图中绘制文本(见图11-2)。

This is drawn text.

图11-2 绘制文本的两个例子

策略

用CDC类的CDC::TextOut( )和CDC::DrawText( )成员函数绘制文本，有关其他文本绘制函数参见11.2.4节“ 说明 ”部分。

步骤

如果不存在设备环境，则用前面例子中介绍的技术创建一个。一种常见的方法如下：

```
CDC *pDC=pWnd->GetDC();    // where pWnd is a pointer to an
                           // MFC window class
```

1. 使用TextOut( )

1) 绘制一个文本串，用：

```
CString str("This is drawn text");
pDC->TextOut(
x,y,                    // device location (ex: screen pixels)
str,                      // text as either a string pointer or
                          // a CString variable
str.getLength());         // text length
```

x和y变量定义文本位置的左上角，如果要 x和y指示别的位置，如文本的中心位置，可以用CDC::SetTextAlign( )改变x和y的含义。

2) 使x和y代表文本中央位置，在调用 TextOut( )之前调用下面的函数：

```
pDC->SetTextAlign(TA_CENTER); //(also TA_RIGHT)
```

也可以用下面的函数，改变y对齐方式：

```
pDC->SetTextAlign(TA_BOTTOM); //(also TA_BASELINE)
```

3) 用不同的标准字形绘制文本，在绘制前使用下面的函数调用 (参见MFC文档有关其他库存字体)：

```
pDC->SelectStockObject(ANSI_VAR_FONT);
```

4) 创建绘制文本的字体，可以用：

```
CFont font;
font.CreateFont(
   -22,                 // point size
   0, 0, 0,
   FW_NORMAL,           // weight, also FW_BOLD
   0,                   // if 1=italic
   0,                   // if 1=underline
   0,                   // if 1=strike through
   0, 0, 0, 0, 0,
   "Courier");          // typeface
   CFont *pFont = (CFont *)pDC->SelectObject(&font);
```

或者，如果只想选取一种基于点数和字形的字体，用 CreatePointFont( )。

CreateFont( )不是真正地创建一种字体，相反它扫描系统当前安装的字体，寻找一种与用户调用参数中指定的标准最匹配的字体，并使用这种字体。

5) 改变窗口的默认字体，可以用：

```
pwnd->SetFont(pFont);            //the font
```

默认字体被自动地选进从那个窗口创建的任一设备环境中。

6) 改变文本的颜色，用：

```
pDC->SetTextColor(RGB(100,100,100));
```

7) 改变文本的背景色，用：

```
pDC->SetBkColor(RGB(200,200,200));
```

除非背景模式设为不透明，否则背景色被忽略。不透明意味着在绘制文本前，先绘制背景矩形；透明模式意味着文本被绘制在当前背景之上。

8) 打开不透明背景模式，用：

```
pDC->SetBkMode(OPAQUE);
```

9) 打开透明背景模式，用：

```
pDC->SetBkMode(TRANSPARENT);
```

CDC::TextOut( )是最简单的绘图函数，如果希望系统做其他工作，可以用CDC::DrawText( )。DrawText( )允许在一个指定的矩形区域内绘制多行文本，也允许在该矩形内指定文本对齐方式。

2. 使用DrawText( )

用DrawText( )绘制文本，用：

```
pDC->DrawText(
   str,              // a CString value
   rect,             // a bounding rectangle
   DT_CENTER);       // an alignment style-text will be centered
```

说明

其他文本绘制函数包括：

- ExtTextOut( )，该函数裁剪给定矩形外的绘制文本。
- TabbedTextOut( )，使用用户提供给该函数的跳格键位置表，扩大插入文本中的跳格距离。
- DrawState( )，用来绘制无效文本，该文本看起来被蚀刻一样。

CD说明

在CD上执行该工程时，可以看到视图中有两行绘制文本。

11.3 例55 从任意位置装入一个图标并绘制

目标

从资源文件或直接从一个图标文件中装入一个图标，在应用程序中绘制。

策略

用三种不同的方法装入一个图标。第一种方法，使用一个称为 LoadIcon( )的应用程序类的成员函数，它从应用程序的资源中装入一个图标；第二种方法，用 Window API函数LoadImage( )直接从一个磁盘文件中装入一个图标；第三种方法，用 Windows API函数ExtractIcon( )从另一个应用程序的可执行文件中抽出一个图标。

步骤

1. 从应用程序的资源中装入一个图标

装入一个在应用程序资源中定义的图标，用：

```
HICON hicon;
hicon=AfxGetApp()->LoadIcon(IDR_MAINFRAME);
```

2. 直接从一个.ico磁盘文件中装入一个图标

从一个.ico文件装入一个图标，使用下面的方法。本例从Wzd.ico装入一个图标。

```
hicon = (HICON)LoadImage(
   NULL,             // handle of the instance that contains
```

```
                                    // the image
    "Wzd.ico",                      // name or identifier of image
    IMAGE_ICON,                     // type of image-
                                // can also be IMAGE_CURSOR or IMAGE_ICON
    0,0,                            // desired width and height
    LR_LOADFROMFILE);               // load flags
```

3. 从一个DLL或.exe文件中装入一个图标

从另一个应用程序的可执行文件中抽取一个图标，可以用下面的方法。本例中抽取在Wzd.exe中发现的第二个图标。

```
HINSTANCE hinst=AfxGetInstanceHandle();
hicon=ExtractIcon(hinst,"Debug\\wzd.exe",1);
```

要确定一个可执行文件或DLL文件有多少个图标，用a-1索引调用ExtractIcon( )，图标数量返回到hIcon中。

4. 绘制一个图标

用下面的方法可以把一个图标绘制到任何窗口。这里的（0,0）是图标的左上角坐标。

```
pDC->DrawIcon(0,0,hicon);
```

5. 销毁一个图标

必须手工销毁任何一个装入的或者用LoadImage( )装入或ExtractIcon( )抽取的图标，以避免资源内存泄漏。

```
DestroyIcon(hicon);
```

说明

- 在应用程序终止前，用LoadIcon( )装入的图标实际上一直保留着，以后装入同样的图标资源，只要返回当前驻留的图标对象的句柄。

CD说明

在CD上执行该工程时，可以看到一个图标被绘制在视图中。

11.4 例56 从任意位置装入一个位图和绘制一个位图

目标

从资源文件或任意位图文件中装入一个位图。

策略

首先，用CBitmap类装入一个在应用程序资源中定义的位图；然后，用Windows API函数LoadImage( )从一个.bmp文件中装入一个位图。

步骤

1. 添加位图到应用程序的资源中

有两种把位图装入到工程中的方法。第一种，用Developer Studio的位图编辑器创建一个

位图；第二种，用Developer Studio的Insert/Insert Resource菜单命令下的Import命令装入一个位图。注意分配给位图的ID。

2. 从应用程序的资源中装入位图

要把位图装入应用程序中进行绘制，用下面的方法，可以用自己的位图ID替代IDB_WZD。

```
CBitmap bitmap;
bitmap.LoadBitmap(IDB_WZD);
```

3. 从一个.bmp文件中装入位图

在运行时用LoadImage( )装入一个位图。

```
CBitmap bitmap;
HBITMAP hbitmap = (HBITMAP)::LoadImage(
   NULL,                    // handle of the instance that contains
                            // the image
   "Wzd2.bmp",              // name or identifier of image
   IMAGE_BITMAP,            // type of image-
                          // can also be IMAGE_CURSOR or IMAGE_ICON
   0,0,                     // desired width and height
   LR_LOADFROMFILE);   // load from file

// attach this bitmap object to our bitmap class
bitmap.Attach(hbitmap);
```

4. 绘制一个位图

绘制一个位图可以用下面的方法，注意用 BitBlt( )需要两个设备环境，而不是一个。

```
CDC dcComp;
dcComp.CreateCompatibleDC(pDC);
dcComp.SelectObject(&bitmap);

// get size of bitmap for BitBlt()
BITMAP bmInfo;
bitmap.GetObject(sizeof(bmInfo),&bmInfo);

// use BitBlt() to draw bitmap
pDC->BitBlt(0,0,bmInfo.bmWidth,bmInfo.bmHeight,
   &dcComp, 0,0,SRCCOPY);
```

说明

- 位图和图标间的主要的区别，在于可以用透明色创建一个图标。每次使用一个透明色的图标时，绘制图标处的背景将显示出来。可以用LoadImage( )以及LR_LOADFROM-FILE标志和LR_LOADTRANSPARENT标志的“或”操作假造这一效果，LoadImage( )用应用程序背景色替换位图中的一种颜色 (如按钮色)；LoadImage( )根据位图左上角像素颜色，选择要替换的颜色。
- 一些早期版本的Windows NT不允许使用带LR_LOADFROMFILE标志的LoadImage( )。如果系统没有升级，可以用下个例子中的方法装入一个位图，在那个例子中不是使用应用程序资源，而是使用 CFile类直接从磁盘装入位图文件，然而要确保从文件的 12字节文件头后开始读取文件。

■ 有关BitBlt( )、位图和设备环境的详细内容参见第 4章。

CD说明

在CD上执行该工程时，有两个位图绘制在视图中。

11.5 例57 从文件中创建一个位图

目标

从一个.bmp文件中装入一个位图，并潜在地修改它。

策略

从应用程序资源中，把一个原始位图文件装入内存中。创建一个位图，并调用 Windows API的CreateDIBitmap( )。然而，在位图传送到CreateDIBitmap( )之前，还可以修改该位图的调色板，并把该功能封装到CBitmap派生类的成员函数中。

步骤

1. 创建一个新的位图类

用ClassWizard从CBitmap派生一个新类，用文本编辑器添加一个 LoadBitmapEx( )成员函数到该类中。

2. 从位图文件创建一个位图对象

1) 在LoadBitmapEx( )开始处获取一个设备环境。为了避免在 LoadBitmapEx( )的调用参数中要求一个设备环境，将从桌面获取一个设备环境。

```
CDC dcScreen;
dcScreen.Attach(::GetDC(NULL));
```

2) 然后，从应用程序的资源中把位图装入内存中。

```
HRSRC hRsrc = FindResource(AfxGetResourceHandle(),
MAKEINTRESOURCE(nID),RT_BITMAP);
HGLOBAL hglb = LoadResource(AfxGetResourceHandle(), hRsrc);
LPBITMAPINFOHEADER lpBitmap =
   (LPBITMAPINFOHEADER)LockResource(hglb);
```

现在，lpBitmap变量指向堆中包含位图资源的一段内存。顺便提一下，这是把任何应用程序资源装入内存的方法。不同的 RT_BITMAP代表计划装入的资源类型。

现在，需要创建传递到CreateDIBitmap( )的位图数据的指针。

3) 创建三个传递到CreateDIBitmap( )的位图结构的指针变量。

```
// get pointers into bitmap structures
// (header, color table and picture bits)
LPBITMAPINFO pBitmapInfo = (LPBITMAPINFO)lpBitmap;
LPBITMAPINFOHEADER pBitmapInfoHeader =
   (LPBITMAPINFOHEADER)lpBitmap;
// determine number of colors in bitmap's palette now because
// bitmap pixel is right after it
```

```
int nNumberOfColors=0;
if (lpBitmap->biClrUsed)
   nNumberOfColors = lpBitmap->biClrUsed;
else if (pBitmapInfoHeader->biBitCount <= 8)
   nNumberOfColors = (1<<pBitmapInfoHeader->biBitCount);
// else there IS no color table
LPBYTE pBitmapPictureData = (LPBYTE)lpBitmap+lpBitmap->biSize+
   (nNumberOfColors*sizeof(RGBQUAD));
```

注意 如果指定像素颜色的位数是24位，则位图文件没有颜色表，因为每个像素项本身包含RGB颜色的全部定义。

4) 因为现在知道位图文件头以后的大小，可以保存它为以后使用，这些数值在创建一个位图对象时是不需要的。

```
m_Width = lpBitmap->biWidth;
m_Height = lpBitmap->biHeight;
```

但是，不能只传递一个位图颜色表的指针给 CreateDIBitmap( )。首先，必须用位图的颜色表创建一个应用程序调色板；然后，把该应用程序调色板选到一个设备环境中，再把该设备环境传递给CreateDIBitmap( )。

5) 要创建一个应用程序调色板，把位图的颜色表填入一个逻辑调色板，然后用 CPalette类创建应用程序调色板。

```
// create a logical palette from the color table in this bitmap
   if (nNumOfColors)
   {
      LOGPALETTE *pLogPal = (LOGPALETTE *) new BYTE[
         sizeof(LOGPALETTE) + (nNumberOfColors *
         sizeof(PALETTEENTRY))];
      pLogPal->palVersion    = 0x300;
      pLogPal->palNumEntries = nNumberOfColors;

      for (int i = 0;  i < nNumberOfColors;  i++)
      {
      // if flag set, replace grey color with window's
      // background color
         if (bTransparent &&
            pBitmapInfo->bmiColors[i].rgbRed==192 &&
            pBitmapInfo->bmiColors[i].rgbGreen==192 &&
            pBitmapInfo->bmiColors[i].rgbBlue==192)
         {
            pBitmapInfo->bmiColors[i].rgbRed=
               GetRValue(::GetSysColor(COLOR_BTNFACE));
            pBitmapInfo->bmiColors[i].rgbGreen=
               GetGValue(::GetSysColor(COLOR_BTNFACE));
            pBitmapInfo->bmiColors[i].rgbBlue=
               GetBValue(::GetSysColor(COLOR_BTNFACE));
         }
         pLogPal->palPalEntry[i].peRed   =
            pBitmapInfo->bmiColors[i].rgbRed;
         pLogPal->palPalEntry[i].peGreen =
```

```
            pBitmapInfo->bmiColors[i].rgbGreen;
        pLogPal->palPalEntry[i].peBlue  =
            pBitmapInfo->bmiColors[i].rgbBlue;
        pLogPal->palPalEntry[i].peFlags = 0;
    }
    m_pPalette=new CPalette;
    m_pPalette->CreatePalette(pLogPal);
    delete []pLogPal;
    dcScreen.SelectPalette(m_pPalette,TRUE);
    dcScreen.RealizePalette();
}
```

当创建一个逻辑调色板时，还有机会修改位图中的颜色。特别指出，这里发现的任何灰颜色，都可用按钮表面的颜色替换，因此当位图绘制在一个按钮表面背景色上时，位图看起来是透明的。

6) 现在可以把3个指向位图结构的指针与实现的调色板一起传递到 ::CreateDIBitmap( )以创建一个位图对象。

```
HBITMAP bitmap = ::CreateDIBitmap(dcScreen.m_hDC,
    pBitmapInfoHeader,CBM_INIT, pBitmapPictureData,
    pBitmapInfo, DIB_RGB_COLORS);
```

7) 创建一个位图对象以后，可以把该位图对象捆绑到位图类对象。

```
Attach(bitmap);
```

8) 有关该位图类的完备清单参见本节的“ 清单 —— 位图类 ”。

3. 使用新的位图类

要绘制该位图，首先把它的调色板选到设备环境中，然后用 BitBlt( )或StretchBlt( )把它绘制到屏幕上。

```
// load bitmap with our new function
m_bitmap.LoadBitmapEx(IDB_WZD,TRUE);

// select our palette into the device context
CPalette *pOldPal =
    pDC->SelectPalette(m_bitmap.GetPalette(),FALSE);
pDC->RealizePalette();

// get device context to select bitmap into
CDC dcComp;
dcComp.CreateCompatibleDC(pDC);
dcComp.SelectObject(&m_bitmap);

// draw bitmap
pDC->BitBlt(0,0,m_bitmap.m_Width,m_bitmap.m_Height, &dcComp,
    0,0,SRCCOPY);

// reselect old palette
pDC->SelectPalette(pOldPal,FALSE);
```

说明

■ 位图中的每个点只是指向颜色表的一个指针，一个位图有它自己的颜色表。然而，当

一个位图被绘制到屏幕上时，它必须与系统上的其他位图和其他颜色表进行竞争，这可能导致每次只能显示有限数量的单色。在需要的时候，这些颜色表转换为调色板，并被选入系统调色板。::CreateDIBitmap( )把一个位图内的点指针指向该新调色板。有关该主题的详细讨论参见第4章。

- 从磁盘而不是通过应用程序资源直接装入该文件时，确保文件指针跳过包含文件头的12个字节，然后用 CFile类把文件读到 CShareMem文件中，该文件被传递到 ::CreateDIBitmap( )。

CD说明

在CD上执行该工程时，可以看到视图被一个位图所填充，该位图使用在位图中定义的原始颜色，显示出非扩散颜色。

清单——位图类

```
#ifndef WZDBITMAP_H
#define WZDBITMAP_H

class CWzdBitmap : public CBitmap
{
public:
   DECLARE_DYNAMIC(CWzdBitmap)

// Constructors
   CWzdBitmap();

   void LoadBitmapEx(UINT nID, BOOL bIconBkgrd);
   CPalette *GetPalette(){return m_pPalette;};

// Implementation
public:
   virtual ~CWzdBitmap();

// Attributes
   int    m_Width;
   int    m_Height;
// Operations

private:
   CPalette *m_pPalette;
};
#endif

// WzdBitmap.cpp : implementation of the CWzdBitmap class
//

#include "stdafx.h"
#include "WzdBitmap.h"
```

```
//////////////////////////////////////////////////////////////////////////
// CWzdBitmap

IMPLEMENT_DYNAMIC(CWzdBitmap, CBitmap)


CWzdBitmap::CWzdBitmap()
{
    m_pPalette=NULL;
}

CWzdBitmap::~CWzdBitmap()
{
    if (m_pPalette)
    {
        delete m_pPalette;
    }
}

void CWzdBitmap::LoadBitmapEx(UINT nID, BOOL bTransparent)
{
// can only load once
    ASSERT(!m_pPalette);

    CDC dcScreen;
    dcScreen.Attach(::GetDC(NULL));

// find and lock bitmap resource
    HRSRC hRsrc = FindResource(AfxGetResourceHandle(),
        MAKEINTRESOURCE(nID),RT_BITMAP);
    HGLOBAL hglb = LoadResource(AfxGetResourceHandle(), hRsrc);
    LPBITMAPINFOHEADER lpBitmap = (LPBITMAPINFOHEADER)LockResource(hglb);

// get pointers into bitmap structures (header, color table and picture bits)
    LPBITMAPINFO pBitmapInfo = (LPBITMAPINFO)lpBitmap;
    LPBITMAPINFOHEADER pBitmapInfoHeader = (LPBITMAPINFOHEADER)lpBitmap;
    int nNumberOfColors=0;
    if (lpBitmap->biClrUsed)
        nNumberOfColors = lpBitmap->biClrUsed;
    else if (pBitmapInfoHeader->biBitCount <= 8)
        nNumberOfColors = (1<<pBitmapInfoHeader->biBitCount);
    LPBYTE pBitmapPictureData = (LPBYTE)lpBitmap+lpBitmap->biSize+
        (nNumberOfColors*sizeof(RGBQUAD));

    // get width and height
    m_Width = lpBitmap->biWidth;
    m_Height = lpBitmap->biHeight;

// create a logical palette from the color table in this bitmap
    if (nNumberOfColors)
```

```
    {
        LOGPALETTE *pLogPal    = (LOGPALETTE *) new BYTE[sizeof(LOGPALETTE) +
            (nNumberOfColors * sizeof(PALETTEENTRY))];
        pLogPal->palVersion    = 0x300;
        pLogPal->palNumEntries = nNumberOfColors;

        for (int i = 0;  i < nNumberOfColors;  i++)
        {
// if flag set, replace grey color with window's background color
            if (bTransparent &&
                pBitmapInfo->bmiColors[i].rgbRed==192 &&
                pBitmapInfo->bmiColors[i].rgbGreen==192 &&
                pBitmapInfo->bmiColors[i].rgbBlue==192)
            {
                pBitmapInfo->bmiColors[i].rgbRed=
                    GetRValue(::GetSysColor(COLOR_BTNFACE));
                pBitmapInfo->bmiColors[i].rgbGreen=
                    GetGValue(::GetSysColor(COLOR_BTNFACE));
                pBitmapInfo->bmiColors[i].rgbBlue=
                    GetBValue(::GetSysColor(COLOR_BTNFACE));
            }


            pLogPal->palPalEntry[i].peRed   = pBitmapInfo->bmiColors[i].rgbRed;
            pLogPal->palPalEntry[i].peGreen =
                pBitmapInfo->bmiColors[i].rgbGreen;
            pLogPal->palPalEntry[i].peBlue  = pBitmapInfo->bmiColors[i].rgbBlue;
            pLogPal->palPalEntry[i].peFlags = 0;
        }
        m_pPalette=new CPalette;
        m_pPalette->CreatePalette(pLogPal);
        delete []pLogPal;
        dcScreen.SelectPalette(m_pPalette,TRUE);
        dcScreen.RealizePalette();
    }

// create device dependant bitmap
    HBITMAP bitmap = ::CreateDIBitmap(dcScreen.m_hDC, pBitmapInfoHeader,
        CBM_INIT, pBitmapPictureData, pBitmapInfo, DIB_RGB_COLORS);

// attach this new bitmap object to our CBitmap class
    Attach(bitmap);

// release dc
    ::ReleaseDC(NULL, dcScreen.Detach());
}
```

11.6 例58 创建一个自绘位图

目标

在内存中创建一个空位图，然后用 MFC绘图工具绘制。

策略

创建一个内存设备环境并绘制它，所有 Windows绘图函数包含一个设备环境，用以跟踪绘制设备的特征；一个内存设备环境允许直接绘制内存。本例中，将这些功能封装到一个从CBitmap派生的类。

步骤

1. 创建一个新位图类

1) 用ClassWizard创建一个从CBitmap派生的新类；

2) 用文本编辑器创建该类的一个 CreateBitmapEx( )成员函数；

2. 从内存中创建一个自绘位图

虽然我们绘制到内存，但需要用最后显示的设备的特征初始化设备环境，首先需要获得一个真实设备的设备环境，本例中采用桌面。与其他设备一样，桌面也是良好的设备。

1) 在CreateBitmapEx( )的开始处，把桌面设备环境封装到 CDC类的一个实例中。

```
CDC dcScreen;
dcScreen.Attach(::GetDC(NULL));
```

2) 创建内存设备环境，并用桌面的特征对它进行初始化。

```
CDC dcMem;
dcMem.CreateCompatibleDC(&dcScreen);
```

3) 创建一个空位图对象，并告知内存设备环境这是它要绘制的地方。

```
CreateCompatibleBitmap(&dcScreen, size.cx, size.cy);
dcMem.SelectObject(this);
```

4) 像绘制到屏幕一样，绘制到内存设备环境。本例中，把应用程序的图标绘制到矩形上的一个圆内。

```
CBrush bluebrush,greenbrush;
bluebrush.CreateSolidBrush(RGB(0,0,255));
greenbrush.CreateSolidBrush(RGB(0,255,0));
dcMem.FillRect(CRect(0,0,size.cx,size.cy), &bluebrush);
dcMem.SelectObject(&greenbrush);
dcMem.Ellipse(0,0, size.cx, size.cy);
HICON hicon=AfxGetApp()->LoadIcon(IDR_MAINFRAME);
dcMem.DrawIcon((size.cx-32)/2,(size.cy-32)/2,hicon);
```

5) 使用完后，释放屏幕和内存设备环境。

```
dcMem.DeleteDC();
::ReleaseDC(NULL, dcScreen.Detach());
```

有关该位图类的详细清单，参见本节的“ 清单 —— 派生的位图类 ”。

说明

- 在内存中创建一个自绘位图是一个只要绘制图形一次就可以重复使用的方法。也可以用元文件或路径名来完成相同的任务，参见第 4章有关元文件和路径的描述。
- 如果应用程序使用自身的调色板，必须把该调色板选进内存设备环境中，然后开始绘制。

CD说明

在CD上执行该工程时，在视图中可以看到一个位图，它是用标准的 MFC图标和一个绘制的圆和矩形创建的。

清单 —— 派生的位图类

```
#ifndef WZDBITMAP_H
#define WZDBITMAP_H

class CWzdBitmap : public CBitmap
{
public:
    DECLARE_DYNAMIC(CWzdBitmap)

// Constructors
    CWzdBitmap();

    void CreateBitmapEx(CSize size);

// Implementation
public:
    virtual ~CWzdBitmap();

// Attributes
    int    m_Width;
    int m_Height;

// Operations

};
#endif

// WzdBitmap.cpp : implementation of the CWzdBitmap class
//

#include "stdafx.h"
#include "WzdBitmap.h"
#include "resource.h"

////////////////////////////////////////////////////////////////////////////
// CWzdBitmap

IMPLEMENT_DYNAMIC(CWzdBitmap, CBitmap)


CWzdBitmap::CWzdBitmap()
{
    m_Width=0;
    m_Height=0;
}
```

```
CWzdBitmap::~CWzdBitmap()
{
}

void CWzdBitmap::CreateBitmapEx(CSize size)
{
    CDC dcMem;
    CDC dcScreen;
    dcScreen.Attach(::GetDC(NULL));

    // create our bitmap in memory
    dcMem.CreateCompatibleDC(&dcScreen);
    CreateCompatibleBitmap(&dcScreen, size.cx, size.cy);
    dcMem.SelectObject(this);

    // do our drawing
    CBrush bluebrush,greenbrush;
    bluebrush.CreateSolidBrush(RGB(0,0,255));
    greenbrush.CreateSolidBrush(RGB(0,255,0));
    dcMem.FillRect(CRect(0,0,size.cx,size.cy), &bluebrush);
    dcMem.SelectObject(&greenbrush);
    dcMem.Ellipse(0,0, size.cx, size.cy);
    HICON hicon=AfxGetApp()->LoadIcon(IDR_MAINFRAME);
    dcMem.DrawIcon((size.cx-32)/2,(size.cy-32)/2,hicon);

    // delete and release device contexts
    dcMem.DeleteDC();
    ::ReleaseDC(NULL, dcScreen.Detach());

    m_Width=size.cx;
    m_Height=size.cy;
```