# 第 9 章　绘　　图

在应用程序中，允许对位图和图标添加颜色和风格。由于所有的 Windows界面看起来基本上都相似，所以要使应用程序的外观看起来与他人的不同的唯一途径就是使用标记 (logo)和飞溅屏幕(splash screen)。在创建自己的控件和在 CAD应用程序中显示图形时，绘图显然也是很重要的。

可以用本章中的一些实例给应用程序赋予一些非同寻常的特征，本章实例包括：

实例31　使用非散射(nondiffused)颜色，在本例中将看到如何用非散射的颜色绘图。在第 1 章中可以找到散射颜色的“悲伤经历”(sad story)，但结论是对于图形应用程序，必须用非散射颜色来绘图。

实例32　伸展位图，在本例中将伸展(stretch)一个现有的位图来填充某一特定区域。

实例33　抓取屏幕，在这里将看到从屏幕的某个区域抓取的图像来装载一个位图。

实例34　输出一个DIB位图文件，本例将输出一个位图到一台设备无关位图 (DIB)文件。

## 9.1　实例31：使用非散射颜色

1. 目标

用非散射的颜色绘图，希望用户设置并保存他们自己的非散射的颜色选择方案，如图 9-1 所示。

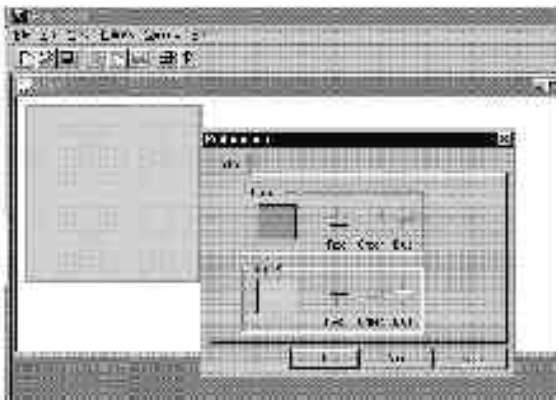

图9-1　非散射颜色选项

2. 策略

为了在自己的应用程序中使用非散射颜色绘图，需要创建、保存并使用自己的调色板。为了做到这一点，本例将用 MFC 的CPalette类。为了提示用户进行颜色选择，将根据用户的选择创建一个新的属性页，该属性页对于每一种基本颜色 (红色、绿色和蓝色)都有个滑动条，这样可以允许用户通过拖动滑动条到所选择颜色密度来挑选颜色。本例将用 AnimatePalette()成员函数来动态改变视窗中被选择的颜色。最后使用GetProfileBinary()函数和WriteProfileBinary ()函数保存用户的颜色选择。

3. 步骤

1) 创建一个应用程序调色板

在CMainFrame类中定义一个颜色数组，并且在类的构造函数中将它们设置为一些缺省颜色:

```
COLORREF m_rgbColors[NUM_COLORS];
:  :  :
CMainFrame::CMainFrame()
```

```
{
    m_rgbColors[COLOR1_COLOR] = RGB( 200,20,150 );
    m_rgbColors[COLOR2_COLOR] = RGB( 0,200,100 );
}
```

接下来在CMainFrame中，添加一个用于从这些颜色中创建调色板的函数。这意味着需要创建一个 CPalette对象并使用 CPalette中的 CreatePalette()函数创建一个调色板。CPalette::CreatePalette()需要一个LOGPALETTE结构作为参数，该结构中包含了所有希望加入到调色板中的颜色，并使用以下代码初始化：

```
void CMainFrame::CreatePalette()
{
    LOGPALETTE *lp = ( LOGPALETTE * )new BYTE[sizeof( LOGPALETTE ) +
        ( NUM_COLORS * sizeof( PALETTEENTRY ) )];
    lp -> palVersion = 0x300;
    lp -> palNumEntries = NUM_COLORS;
    for ( int i = 0; i<NUM_COLORS; i++ )
    {
        lp -> palPalEntry[i].peRed = GetRValue( m_rgbColors[i] );
        lp -> palPalEntry[i].peGreen = GetGValue( m_rgbColors[i] );
        lp -> palPalEntry[i].peBlue = GetBValue( m_rgbColors[i] );
        // reserve for animation
        lp -> palPalEntry[i].peFlags = PC_RESERVED;

    }
    if ( m_pPalette ) delete m_pPalette;
    m_pPalette = new CPalette;
    m_pPalette -> CreatePalette( lp );
    delete []lp;
}
```

在系统注册表中加载任何程序选项后，从 CMainFrame的OnCreate()函数中调用这个新的CMainFrame::CreatePalette()函数，这些程序选项中很快就会包含颜色选项。

2) 用应用程序的调色板绘图

为使用自己的调色板绘图，必须首先将调色板选入自己设备环境中，然后再在这个调色板中实现颜色。实现颜色 (realizing color)仅仅意味着Windows试图将所有的应用程序调色板中的颜色都加入到可用的系统调色板 (有关这方面的详细知识请参阅第 1章):

```
// select palette into device context and realize colors
CPalette *pOPalette = pDC -> SelectPalette(
    ( ( CMainFrame* )AfxGetMainWnd() ) -> GetPalette(),FALSE );
pDC -> RealizePalette();
```

注意同时为CMainFrame添加一个称为GetPalette()的封装函数，这个函数允许其他的类来访问这个应用程序的调色板。

无论何时使用非散射的颜色绘图，都用 PALETTEINDEX()宏来定义颜色，代码如下：

```
CPen pen( PS_SOLID,3,PALETTEINDEX( COLOR1_COLOR ) );
CPen *pOPen = pDC -> SelectObject( &pen );
CBrush brush( PALETTEINDEX( COLOR2_COLOR ) );
CBrush *pOBrush = pDC -> SelectObject( &brush );
```

COLOR1_COLOR和COLOR2_COLOR两个值只是应用程序调色板的索引。

当用画笔和画刷绘制直线和图形时，可使用合适的非散射颜色。

在绘图完成之后，重新选择原来的画笔、画刷和调色板:

```
pDC -> SelectObject( pOPen );
pDC -> SelectObject( pOBrush );
pDC -> SelectPalette( pOPalette,FALSE );
```

3) 创建一个颜色选项属性页

用Developer Studio创建一个新的对话框模板。设置这个模板为具有细框架的子风格。这个模板的标题将与属性页标签上出现的标题一样。

用Dialog Editor添加如图9-1所示的一个图像控件和三个滑动条，也可以只用本书附带光盘上提供给该实例的模板。

用ClassWizard为从CPropertyPage中派生的模板添加一个新的属性页。

添加该属性页到应用程序并作为首选项：

```
void CMainFrame::OnOptionsPreferences()
{
   CPropertySheet sheet( _T( "Preferences" ),this );
   m_pColorPage = new CColorPage;

   sheet.AddPage( m_pColorPage );

   m_pColorPage -> m_rgbColors[COLOR1_COLOR] =
      m_rgbColors[COLOR1_COLOR];
   m_pColorPage -> m_rgbColors[COLOR2_COLOR] =
      m_rgbColors[COLOR2_COLOR];

   sheet.DoModal();

   delete m_pColorPage;
}
```

在此省略如何将应用程序的当前颜色传递到该页，以及滑动条是如何改变这些值的细节，请参考本实例结尾的程序清单 —— 主框架类和程序清单 —— 颜色属性页类，在此介绍了滑动条如何创建一个需要加入系统调色板的新 RGB颜色值的内容。

保存系统调色板中的新RGB颜色，可以通过使用如下所示的CPalette的AnimatePallette()函数来完成:

```
// select our palette into a device context
CDC *pDC = GetDC();
CMainFrame *pFrame = ( CMainFrame * )AfxGetMainWnd();
CPalette *pPalette = pFrame -> GetPalette();
CPalette *pOPalette = pDC -> SelectPalette( pPalette,FALSE );

// add new color to palette using animation
PALETTEENTRY  pentry;
pentry.peRed = GetRValue( m_rgbColors[i] );
pentry.peGreen = GetGValue( m_rgbColors[i] );
pentry.peBlue = GetBValue( m_rgbColors[i] );
```

```
pentry.peFlags = PC_RESERVED;
pPalette -> AnimatePalette( i,1,&pentry );

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

注意，即使AnimatePalette()函数中的某一个参数不使用一个设备环境，和它一起使用的调色板必须自己被选入设备环境(MUST ITSELF BE SELECTED INTO A DEVICE CONTEXT)来使该函数工作。换句话说，必须将 PPalette选入到pDC中，以使得AnimatePalette()可以正常工作。

4) 保存并恢复颜色选项

为了保存应用程序的颜色以备下次使用，无论当前使用什么函数，都需要添加下面几行来保存用户的颜色选项：

```
UINT size = sizeof( m_rgbColors );
AfxGetApp() -> WriteProfileBinary( "Settings","Colors",
   ( BYTE* )m_rgbColors, size);
```

为了恢复用户的颜色选项，无论当前使用什么函数，都需要添加下面几行来恢复用户的颜色选项：

```
BYTE *p;
UINT size;
if ( AfxGetApp() -> GetProfileBinary ( "Settings","Colors", &p,&size ) )

{
   memcpy( m_rgbColors,p,size );
   delete []p;
}
```

4. 注意

■ MFC应用程序的颜色缺省为散射颜色，这种颜色是由三种基本颜色组合形成的。但是对于某些应用程序，特别对于一个 CAD应用程序，这是不可接受的，因为散射颜色容易模糊。散射颜色和非散射颜色的区别，请参阅第 1章。

■ 用户可能会注意到在运行光盘中的示例时， AnimatePalette()不仅改变颜色属性页中矩形框的颜色，而且改变视的颜色。因为这是直接对系统调色板进行修改，而应用程序则是从系统调色板中获取其各种颜色。在用户视中的每一个像素都指向系统调色板中的一种颜色——如果用户改变了系统调色板中的颜色，那么也改变了像素的颜色。

■ 用户可能会注意到当另一个应用程序激活时，在应用程序中的非散射的颜色将会发生畸变。这是因为另一个应用程序已经接管了系统调色板并且将用户应用程序中的颜色丢弃。可以通过处理在CMainFrame中的两个消息来消除畸变。这两个消息是：WM_UPDATEPALETTE和WM_ONQUERYNEWPALETTE。处理这个功能的推荐方法是使用新的系统调色板并用UpdateColor ()函数来刷新用户应用程序的颜色。然而， UpdateColor()函数较慢并且容易产生颜色错误，因而许多图形应用程序在处于后台非激活状态时都将忽略颜色所发生的变化，或者只是通知自己的应用程序通过调用 WM_QUERYNEWPALETTE中的RedrawWindow()消息处理函数重新绘制自己，并且使用新的系统调色板。该函数将导致主窗口和其他的子窗口，特别是视，重新绘制它们。

5. 使用光盘时注意

运行随书附带光盘上的工程时，将会注意到视由两个颜色方块所填充。单击 Options和Preferences菜单命令打开一个首选项属性表单，然后拖动任何一个滑动条，注意到颜色方块逐渐变成另外一种非散射的颜色。如果用户退出并重新进入应用程序，将会注意到应用程序恢复上一次退出之前保存的颜色。

6. 程序清单 —— 主框架类

```
// MainFrm.h : interface of the CMainFrame class
//
/////////////////////////////////////////////////////////////////////////////

#if !defined( AFX_MAINFRM_H__CA9038EA_BODF_11D1_A18C_DCB3C85EBD34__INCLUDED_ )
#define AFX_MAINFRM_H__CA9038EA_BODF_11D1_A18C_DCB3C85EBD34__INCLUDED_

#if _MSC_VER >= 1000
#pragma once
#endif    // _MSC_VER >= 1000

#include "colorpage.h"

class CMainFrame : public CMDIFrameWnd
{
   DECLARE_DYNAMIC(CMainFrame)
public:
   CMainFrame();

// Attributes
public:

// Operations
public:
   void LoadOptions();
   void SaveOptions();

   void SetColorRef( int id,COLORREF rgb ){m_rgbColors[id] = rgb;};
   CPalette *GetPalette(){return m_pPalette;};

// Overrides
   // ClassWizard generated virtual function overrides
   // {{AFX_VIRTUAL( CMainFrame )
   virtual BOOL PreCreateWindow( CREATESTRUCT& cs );
   // }}AFX_VIRTUAL

// Implementation
public:
   virtual ~CMainFrame();
#ifdef _DEBUG
   virtual void AssertValid() const;
   virtual void Dump( CDumpContext& dc ) const;
#endif
```

```
protected:    // control bar embedded members
    CStatusBar    m_wndStatusBar;
    CToolBar      m_wndToolBar;

// Generated message map functions
protected:
    // {{AFX_MSG( CMainFrame )
    afx_msg int OnCreate( LPCREATESTRUCT lpCreateStruct );
    afx_msg void OnClose();
    afx_msg void OnOptionsPreferences();
    // }}AFX_MSG
    DECLARE_MESSAGE_MAP()
private:
    COLORREF m_rgbColors[NUM_COLORS];
    CColorPage *m_pColorPage;
    CPalette *m_pPalette;

    void CreatePalette();

};

/////////////////////////////////////////////////////////////////////////////

// {{AFX_INSERT_LOCATION}}
// Microsoft Developer Studio will insert additional declarations immediately
//     before the previous line.

#endif
    // !defined( AFX_MAINFRM_H__CA9038EA_BODF_11D1_A18C_DCB3C85EBD34__INCLUDED_ )
// MainFrm.cpp : implementation of the CMainFrame class
//

#include "stdafx.h"
#include "Wzd.h"
#include "WzdProject.h"

#include "MainFrm.h"

#include <afxpriv.h>

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

/////////////////////////////////////////////////////////////////////////////
// CMainFrame

IMPLEMENT_DYNAMIC( CMainFrame, CMDIFrameWnd )
```

```
BEGIN_MESSAGE_MAP( CMainFrame, CMDIFrameWnd )
    // {{AFX_MSG_MAP( CMainFrame )
    ON_WM_CREATE()
    ON_WM_CLOSE()
    ON_COMMAND( ID_OPTIONS_PREFERENCES, OnOptionsPreferences )
    // }}AFX_MSG_MAP
END_MESSAGE_MAP()

static UINT indicators[] =
{
    ID_SEPARATOR,         // status line indicator
    ID_INDICATOR_CAPS,
    ID_INDICATOR_NUM,
    ID_INDICATOR_SCRL,
};

////////////////////////////////////////////////////////////////////////////
// CMainFrame construction/destruction

CMainFrame::CMainFrame()
{
    m_rgbColors[COLOR1_COLOR] = RGB( 200,20,150 );
    m_rgbColors[COLOR2_COLOR] = RGB( 0,200,100 );
    m_pPalette = NULL;
}

CMainFrame::~CMainFrame()
{
    m_pPalette -> DeleteObject();
    delete m_pPalette;
    m_pPalette = NULL;
}

int CMainFrame::OnCreate( LPCREATESTRUCT lpCreateStruct )
{
    if ( CMDIFrameWnd::OnCreate( lpCreateStruct ) = -1 )
        return -1;

    LoadOptions();
    CreatePalette();

    if ( !m_wndToolBar.Create( this ) ||
        !m_wndToolBar.LoadToolBar( IDR_MAINFRAME ) )
    {
        TRACE0( "Failed to create toolbar\n" );
        return -1;    // fail to create
    }

    if ( !m_wndStatusBar.Create( this ) ||
        !m_wndStatusBar.SetIndicators( indicators,
            sizeof( indicators )/sizeof( UINT ) ) )
```

```
    {
        TRACE0( "Failed to create status bar\n" );
        return -1;    // fail to create
    }

    // TODO: Remove this if you don't want tool tips or a resizeable toolbar
    m_wndToolBar.SetBarStyle( m_wndToolBar.GetBarStyle() |
        CBRS_TOOLTIPS | CBRS_FLYBY | CBRS_SIZE_DYNAMIC );

    // TODO: Delete these three lines if you don't want the toolbar to
    //     be dockable
    m_wndToolBar.EnableDocking( CBRS_ALIGN_ANY );
    EnableDocking( CBRS_ALIGN_ANY );
    DockControlBar( &m_wndToolBar );

    return 0;
}

BOOL CMainFrame::PreCreateWindow( CREATESTRUCT& cs )
{
    // TODO: Modify the Window class or styles here by modifying
    //     the CREATESTRUCT cs

    return CMDIFrameWnd::PreCreateWindow( cs );
}

/////////////////////////////////////////////////////////////////////////////
// CMainFrame diagnostics

#ifdef _DEBUG
void CMainFrame::AssertValid() const
{
    CMDIFrameWnd::AssertValid();
}

void CMainFrame::Dump( CDumpContext& dc ) const
{
    CMDIFrameWnd::Dump( dc );
}

#endif    //_DEBUG

/////////////////////////////////////////////////////////////////////////////
// CMainFrame message handlers

void CMainFrame::OnClose()
{
    SaveOptions();

    CMDIFrameWnd::OnClose();
}
```

```
void CMainFrame::LoadOptions()
{
   BYTE *p;
   UINT size;
   if ( AfxGetApp() -> GetProfileBinary( SETTINGS_KEY,COLORS_KEY,&p,&size ) )
   {
   memcpy( m_rgbColors,p,size );
   delete []p;
   }
}

void CMainFrame::SaveOptions()
{
   UINT size = sizeof( m_rgbColors );
   AfxGetApp() ->
      WriteProfileBinary( SETTINGS_KEY,COLORS_KEY,( BYTE* )m_rgbColors,size );
}

void CMainFrame::OnOptionsPreferences()
{
   CPropertySheet sheet( _T( "Preferences" ),this );
   m_pColorPage = new CColorPage;

   sheet.AddPage( m_pColorPage );

   m_pColorPage -> m_rgbColors[COLOR1_COLOR] = m_rgbColors[COLOR1_COLOR];
   m_pColorPage -> m_rgbColors[COLOR2_COLOR] = m_rgbColors[COLOR2_COLOR];

   sheet.DoModal();

   delete m_pColorPage;
}

void CMainFrame::CreatePalette()
{
   LOGPALETTE *lp = ( LOGPALETTE * )calloc( 1, sizeof( LOGPALETTE ) +
      ( NUM_COLORS * sizeof( PALETTEENTRY ) ) );
   lp -> palVersion = 0x300;
   lp -> palNumEntries = NUM_COLORS;
   for ( int i = 0; i < NUM_COLORS; i++ )
   {
      lp -> palPalEntry[i].peRed = GetRValue( m_rgbColors[i] );
      lp -> palPalEntry[i].peGreen = GetGValue( m_rgbColors[i] );
      lp -> palPalEntry[i].peBlue = GetBValue( m_rgbColors[i] );
      // reserve for animation
      lp -> palPalEntry[i].peFlags = PC_RESERVED;

   }
   if ( m_pPalette ) delete m_pPalette;
   m_pPalette = new CPalette;
   m_pPalette -> CreatePalette( lp );
   free( lp );
}
```

7. 程序清单 —— 颜色属性页类

```
#if !defined COLORPAGE_H
#define COLORPAGE_H

// ColorPage.h : header file
//

#include "WzdProject.h"

/////////////////////////////////////////////////////////////////////////////
// CColorPage dialog

class CColorPage : public CPropertyPage
{
    DECLARE_DYNCREATE( CColorPage )

// Construction
public:
    CColorPage();
    ~CColorPage();

// Dialog Data
    // {{AFX_DATA( CColorPage )
    enum { IDD = IDD_COLOR_PAGE };
    CStatic m_ctrlColor1Display;
    CStatic m_ctrlColor2Display;
    CSliderCtrl m_ctrlBlueSlider1;
    CSliderCtrl m_ctrlGrnSlider1;
    CSliderCtrl m_ctrlRedSlider1;
    CSliderCtrl m_ctrlBlueSlider2;
    CSliderCtrl m_ctrlGrnSlider2;
    CSliderCtrl m_ctrlRedSlider2;
    // }}AFX_DATA

    COLORREF m_rgbColors[NUM_COLORS];

// Overrides
    // ClassWizard generate virtual function overrides
    // {{AFX_VIRTUAL( CColorPage )
protected:
    virtual void DoDataExchange(CDataExchange* pDX);    // DDX/DDV support
    // }}AFX_VIRTUAL

// Implementation
protected:
    // Generated message map functions
    // {{AFX_MSG( CColorPage )
    virtual BOOL OnInitDialog();
    afx_msg void OnVScroll( UINT nSBCode, UINT nPos, CScrollBar* pScrollBar );
    afx_msg void OnPaint();
```

```
    //}}AFX_MSG
    DECLARE_MESSAGE_MAP()
private:
    void DrawColorRects();

};

#endif
// ColorPage.cpp : implementation file
//

#include "stdafx.h"
#include "wzd.h"
#include "ColorPage.h"
#include "WzdProject.h"
#include "MainFrm.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

/////////////////////////////////////////////////////////////////////////////
// CColorPage property page

IMPLEMENT_DYNCREATE( CColorPage, CPropertyPage )

CColorPage::CColorPage() : CPropertyPage( CColorPage::IDD )
{
    // {{AFX_DATA_INIT( CColorPage )
    // }}AFX_DATA_INIT
}

CColorPage::~CColorPage()
{
}

void CColorPage::DoDataExchange( CDataExchange* pDX )
{
    CPropertyPage::DoDataExchange( pDX );
    // {{AFX_DATA_MAP( CColorPage )
    DDX_Control( pDX, IDC_COLOR1_DISPLAY, m_ctrlColor1Display );
    DDX_Control( pDX, IDC_COLOR2_DISPLAY, m_ctrlColor2Display );
    DDX_Control( pDX, IDC_BLUE_SLIDER1, m_ctrlBlueSlider1 );
    DDX_Control( pDX, IDC_GRN_SLIDER1, m_ctrlGrnSlider1 );
    DDX_Control( pDX, IDC_RED_SLIDER1, m_ctrlRedSlider1 );
    DDX_Control( pDX, IDC_BLUE_SLIDER2, m_ctrlBlueSlider2 );
    DDX_Control( pDX, IDC_GRN_SLIDER2, m_ctrlGrnSlider2 );
    DDX_Control( pDX, IDC_RED_SLIDER2, m_ctrlRedSlider2 );
    // }}AFX_DATA_MAP
}
```

```
BEGIN_MESSAGE_MAP( CColorPage, CPropertyPage )
    // {{AFX_MSG_MAP( CColorPage )
    ON_WM_VSCROLL()
    ON_WM_PAINT()
    // }}AFX_MSG_MAP
END_MESSAGE_MAP()

////////////////////////////////////////////////////////////////////////////
// CColorPage message handlers

BOOL CColorPage::OnInitDialog()
{
    CPropertyPage::OnInitDialog();

    // setup slider bars
    m_ctrlBlueSlider1.SetTicFreq ( 15 );
    m_ctrlBlueSlider1.SetRange( 0, 255, TRUE );
    m_ctrlBlueSlider1.SetPos ( 255-GetBValue( m_rgbColors[COLOR1_COLOR] ) );
    m_ctrlGrnSlider1.SetTicFreq ( 15 );
    m_ctrlGrnSlider1.SetRange ( 0, 255, TRUE );
    m_ctrlGrnSlider1.SetPos ( 255-GetGValue( m_rgbColors[COLOR1_COLOR] ) );
    m_ctrlRedSlider1.SetTicFreq ( 15 );
    m_ctrlRedSlider1.SetRange ( 0, 255, TRUE );
    m_ctrlRedSlider1.SetPos ( 255-GetRValue( m_rgbColors[COLOR1_COLOR] ) );
    m_ctrlBlueSlider2.SetTicFreq ( 15 );
    m_ctrlBlueSlider2.SetRange ( 0, 255, TRUE );
    m_ctrlBlueSlider2.SetPos ( 255-GetBValue( m_rgbColors[COLOR2_COLOR] ) );
    m_ctrlGrnSlider2.SetTicFreq ( 15 );
    m_ctrlGrnSlider2.SetRange ( 0, 255, TRUE );
    m_ctrlGrnSlider2.SetPos ( 255-GetGValue( m_rgbColors[COLOR2_COLOR] ) );
    m_ctrlRedSlider2.SetTicFreq ( 15 );
    m_ctrlRedSlider2.SetRange ( 0, 255, TRUE );
    m_ctrlRedSlider2.SetPos ( 255-GetRValue( m_rgbColors[COLOR2_COLOR] ) );


    return TRUE;        // return TRUE unless you set the focus to a control
                        //     EXCEPTION: OCX Property Pages should return FALSE
}

void CColorPage::OnPaint()
{
    CPaintDC dc( this );        // device context for painting

    DrawColorRects();
}

void CColorPage::OnVScroll( UINT nSBCode, UINT nPos, CScrollBar* pScrollBar )
{
    int i;
    int color = 0;
    UINT id = pScrollBar -> GetDlgCtrlID();
```

```
switch ( nSBCode )
{
   case SB_TOP:
      color = 255;
      break;
   case SB_BOTTOM:
      color = 0;
      break;
   case SB_LINEDOWN:
   case SB_LINEUP:
   case SB_PAGEDOWN:
   case SB_PAGEUP:
      switch ( id )
      {
         case IDC_RED_SLIDER1:
            color = m_ctrlRedSlider1.GetPos();
            break;
         case IDC_BLUE_SLIDER1:
            color = m_ctrlBlueSlider1.GetPos();
            break;
         case IDC_GRN_SLIDER1:
            color = m_ctrlGrnSlider1.GetPos();
            break;
         case IDC_RED_SLIDER2:
            color = m_ctrlRedSlider2.GetPos();
            break;
         case IDC_BLUE_SLIDER2:
            color = m_ctrlBlueSlider2.GetPos();
            break;
         case IDC_GRN_SLIDER2:
            color = m_ctrlGrnSlider2.GetPos();
            break;
      }
      break;
   case SB_THUMBPOSITION:
   case SB_THUMBTRACK:
      color = nPos;
      break;
   case SB_ENDSCROLL:
      break;
}

if ( nSBCode! = SB_ENDSCROLL )
{
   color = 255-color;

   switch ( id )
   {
      case IDC_RED_SLIDER1:
      case IDC_RED_SLIDER2:
         i = COLOR1_COLOR;
```

```
            if ( id = IDC_RED_SLIDER2 ) i = COLOR2_COLOR;
               m_rgbColors[i] = RGB( color,GetGValue( m_rgbColors[i] ),
                  GetBValue( m_rgbColors[i] ) );
            break;
         case IDC_BLUE_SLIDER1:
         case IDC_BLUE_SLIDER2:
            i = COLOR1_COLOR;
            if ( id = IDC_BLUE_SLIDER2 ) i = COLOR2_COLOR;
               m_rgbColors[i] = RGB( GetRValue( m_rgbColors[i] ),
                  GetGValue( m_rgbColors[i] ),color );
            break;
         case IDC_GRN_SLIDER1:
         case IDC_GRN_SLIDER2:
            i = COLOR1_COLOR;
            if ( id == IDC_GRN_SLIDER2 ) i = COLOR2_COLOR;
               m_rgbColors[i] = RGB(GetRValue( m_rgbColors[i] ),color,
                  GetBValue( m_rgbColors[i] ) );
            break;
      }

      // select palette into a device context
      // (NOTE: you MUST select a palette into a device context for
      //     AnimatePalette() to work!)
      CDC *pDC = GetDC();
      CMainFrame *pFrame = ( CMainFrame * )AfxGetMainWnd();
      CPalette *pPalette = pFrame -> GetPalette();
      CPalette *pOPalette = pDC -> SelectPalette( pPalette,FALSE );

      // add new color to palette using animation
      PALETTEENTRY  pentry;
      pentry.peRed = GetRValue( m_rgbColors[i] );
      pentry.peGreen = GetGValue( m_rgbColors[i] );
      pentry.peBlue = GetBValue( m_rgbColors[i] );
      pentry.peFlags = PC_RESERVED;
      pPalette -> AnimatePalette( i,1,&pentry );

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

      // change color in CMainFrame
      pFrame -> SetColorRef( i,m_rgbColors[i] );
   }
   CPropertyPage::OnVScroll( nSBCode, nPos, pScrollBar );
}

void CColorPage::DrawColorRects()
{
   // create a solid brush using our palette
   CDC *pDC = GetDC();
   CMainFrame *pFrame = ( CMainFrame * )AfxGetMainWnd();
   CPalette *pPalette = pFrame -> GetPalette();
```

```
    CPalette *pOPalette = pDC -> SelectPalette( pPalette,FALSE );
    CBrush brush1( PALETTEINDEX( COLOR1_COLOR ) );
    CBrush brush2( PALETTEINDEX( COLOR2_COLOR ) );

    // draw the rectangle
    CRect rect;
    m_ctrlColor1Display.GetWindowRect( &rect );
    ScreenToClient( &rect );
    rect.DeflateRect( 2,2,2,2 );
    CBrush *pOBrush = pDC -> SelectObject( &brush1 );
    pDC -> Rectangle( &rect );

    m_ctrlColor2Display.GetWindowRect( &rect );
    ScreenToClient( &rect );
    rect.DeflateRect( 2,2,2,2 );
    pDC -> SelectObject( &brush2 );
    pDC -> Rectangle( &rect );

    // unselect everything
    pDC -> SelectObject( pOBrush );
    pDC -> SelectPalette( pOPalette,FALSE );
    ReleaseDC( pDC );
}
```

## 9.2 实例32：伸展位图

1. 目标

伸展位图以匹配屏幕上的特定区域，如图 9-2所示。

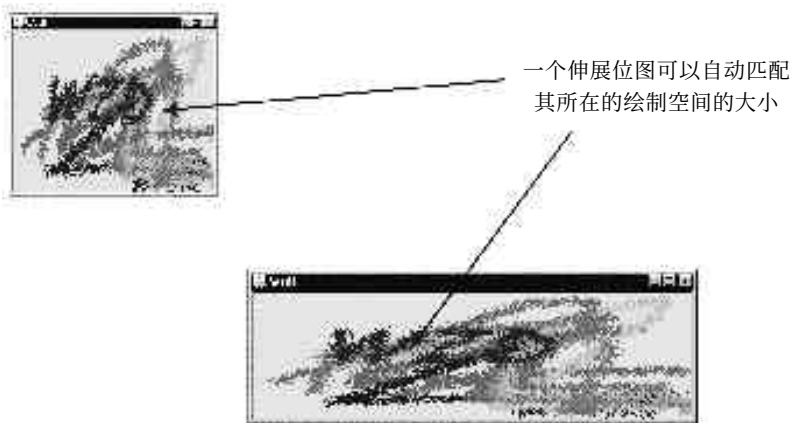


图9-2 伸展位图

2. 策略

使用CDC::SetStretchMode()函数设置伸展模式，并使用 CDC::StretchBlt()函数完成实际的伸展。本例将该功能封装到自己的位图类中。

3. 步骤

1) 创建新的位图类

用ClassWizard创建从Cbitmap派生的新位图类。为该新类添加一个新成员函数： Stretch ()。

2) 创建伸展位图对象

为Stretch()函数赋三个参数，前两个参数是即将创建的已伸展位图的宽度和高度，最后一个参数是下面将讨论的伸展模式:

```
CBitmap *CWzdBitmap::Stretch( int nWidth, int nHeight, int nMode )
{
```

创建一个设备环境。由于不必为该函数传递设备环境，因而可从桌面获得设备环境:

```
CDC dcTo, dcFrom, dcScreen;
dcScreen.Attach( ::GetDC( NULL ) );
```

从该屏幕设备环境创建两个内存设备环境，一个用于未伸展位图，另一个用于已伸展位图:

```
// create "from" device context and select the loaded bitmap into it
dcFrom.CreateCompatibleDC( &dcScreen );
dcFrom.SelectObject( this );

// create a "to" device context select a memory bitmap into it
dcTo.CreateCompatibleDC( &dcScreen );
```

在内存中创建一个空白位图，该空白位图具有与所要创建的位图相同的高度和宽度，同时，需要将该空白位图选入 To设备环境:

```
CBitmap *pBitmap = new CBitmap;
pBitmap -> CreateCompatibleBitmap( &dcScreen, nWidth, nHeight );
dcTo.SelectObject( pBitmap );
```

假设在Stretch()函数中已经使用 LoadBitmap()函数或者其他方式装载了一个位图对象。现在便可以获得该位图的尺寸大小，如下所示：

```
// get original bitmap size
BITMAP bmInfo;
GetObject( sizeof( bmInfo ),&bmInfo );
```

设置伸展模式，然后进行伸展。注意将从该类内部的位图伸展到刚才创建的空白位图对象：

```
// set the stretching mode
dcTo.SetStretchBltMode( nMode );

// stretch loaded bitmap into memory bitmap
dcTo.StretchBlt( 0, 0, nWidth, nHeight,
   &dcFrom, 0, 0, bmInfo.bmWidth, bmInfo.bmHeight, SRCCOPY );
```

Stretch()要做的最后工作是清除设备环境，并返回一个指针到这个被伸展的新位图类对象。完成以后，调用程序负责删除该位图对象:

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

// it's up to the caller to delete this new bitmap
```

```
    return pBitmap;

}
```

3) 使用新的位图类

为了使用新位图类，先装载一个位图到该类中：

```
CWzdBitmap m_bitmap;
:  :  :
m_bitmap.LoadBitmap( IDB_WZD_BITMAP );
```

然后创建该位图的伸展版本：

```
// get a bitmap stretched to size of client area of view
CRect rect;
GetClientRect( &rect );
CBitmap *pBitmap = m_bitmap.Stretch( rect.Width(), rect.Height(),
    COLORONCOLOR );
    // also HALFTONE - - slower but attempts to average colors
    // BLACKONWHITE - - monotone - - scacrifices white for black pixels
    // WHITEONBLACK - - monotone - - scacrifices black for white pixels

    // if you use HALFTONE, must also call the following next to
    //    realign the brush
::SetBrushOrgEx( pDC -> m_hDC, 0, 0, NULL );
```

接下来可以按通常方式在屏幕上绘制该伸展位图：

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

// draw bitmap
pDC -> BitBlt( 0, 0, rect.Width(), rect.Height(), &dcComp, 0, 0, SRCCOPY );

delete pBitmap;
```

4. 注意

■ CDC::Stretch()函数对于大位图工作得非常好，这样的位图有成百上千的像素需要处理。在较小的分辨率下，例如16×16像素大小的标准工具栏按钮，该函数必须采取某些折衷方式是很显然的。线条模糊、图像变形对专业级的外观显示都是不可接受的。在这样的分辨率下最好避免使用Stretch()函数，而应当对不同的分辨率采用不同的位图绘制函数。对于大小适中的位图，此时伸展模式就比较重要了。在使用HALFTONE模式伸展或者压缩一个位图的时候，StretchBlt()函数可以进行颜色平均。即如果每个像素所用的颜色可在多种颜色之间进行选择，则该函数使用平均颜色值。这样做可能较慢，但比起不能进行颜色平均的 COLORONCOLOR模式而言，其效果更容易使人接受。对于单色位图 (仅有黑白色)，可以为伸展模式选择BLACKONWHITE或者WHITEONBLACK。在该函数处于BLACKONWHITE的模式之下，或对新图上的像素点所用颜色需要在黑色或者白色之间进行选择时，总是选择黑色，而WHITEONBLACK模式则相反。

■ 最好伸展基准位图(reference bitmap)而不要持续伸展已经伸展过的位图，特别是在伸

展了几次的情况下更是如此。换句话说不要通过伸展位图 A来创建位图B，然后再通过伸展B来创建位图C。最好分别伸展A以得到B和C。这是因为每次使用StretchBlt()函数都会导致一些图像变形，如果每次都用同一位图进行伸展，则这些图像变形就会逐渐累积起来。

5. 使用光盘时注意

执行随书附带光盘上的工程的时候，你会注意到视由一个位图所填充，该位图的大小随用户缩小或者扩大视时而改变。

6. 程序清单 —— 位图类

```
#ifndef WZDBITMAP_H
#define WZDBITMAP_H

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

// Constructors
    CWzdBitmap();

    CBitmap *Stretch( int nWidth, int nHeight, int nMode );

// Implementation
public:
    virtual ~CWzdBitmap();

// Attributes
// 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()
{
}

CWzdBitmap::~CWzdBitmap()
{
}

CBitmap *CWzdBitmap::Stretch( int nWidth, int nHeight, int nMode )
{
    CDC dcTo,dcFrom,dcScreen;
```

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

    // create "from" device context and select the loaded bitmap into it
    dcFrom.CreateCompatibleDC( &dcScreen );
    dcFrom.SelectObject( this );

    // create a "to" device context select a memory bitmap into it
    dcTo.CreateCompatibleDC( &dcScreen );
    CBitmap *pBitmap = new CBitmap;
    pBitmap -> CreateCompatibleBitmap( &dcScreen, nWidth, nHeight );
    dcTo.SelectObject( pBitmap );

    // get original bitmap size
    BITMAP bmInfo;
    GetObject( sizeof( bmInfo ),&bmInfo );

    // set the stretching mode
    dcTo.SetStretchBltMode( nMode );

    // stretch loaded bitmap into memory bitmap
    dcTo.StretchBlt( 0, 0, nWidth, nHeight,
       &dcFrom, 0, 0, bmInfo.bmWidth, bmInfo.bmHeight, SRCCOPY );

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

    // it's up to the caller to delete this new bitmap
    return pBitmap;

}
```

## 9.3 实例33：抓取屏幕

1. 目标

将屏幕上的某块区域抓进位图。

2. 策略

也许读者已经知道可以用 BitBlt()函数在屏幕上绘制位图。但是否了解还可以将该过程反过来，即在屏幕上的某个区域抓进位图。在没有当前系统调色板的情况下抓取的内容可能什么也没有，因为位图不过是指向颜色表的指针数组，所以还需要抓取当前的系统调色板。由于该功能与位图密切相关，这里将它封装进一个自己的位图类中。

3. 步骤

1) 抓取屏幕

用ClassWizard创建派生自Cbitmap的新类。

用Text Editor为该类添加新函数Capture()，该函数以一个RECT结构作为其调用参数。该RECT结构将包含所抓取屏幕区域的屏幕坐标。在该函数中首先删除保存在该类中的上一次的调色板和位图对象：

```
void CWzdBitmap::Capture( CRect &rect )
```

```
{
    // cleanup old captures
    if (m_pPalette)
    {
        DeleteObject();
        delete m_pPalette;
    }
```

接下来，由于调用程序提供了该位图最终所具备的坐标尺寸，可以将其作为属性值保存：

```
// save width and height
m_nWidth = rect.Width();
m_nHeight = rect.Height();
```

为该屏幕创建一个设备环境。一般情况下该设备环境设置为写入，现在则将其设置为读取:

```
CDC dcScreen;
dcScreen.CreateDC( "DISPLAY", NULL, NULL, NULL );
```

创建一个空白的位图对象并将其选入一个内存设备环境：

```
CDC dcMem;
dcMem.CreateCompatibleDC( &dcScreen );
CreateCompatibleBitmap( &dcScreen, m_nWidth, m_nHeight );
dcMem.SelectObject( this );
```

现在用BitBlt ()函数拷贝屏幕内容到空白位图：

```
dcMem.BitBlt( 0, 0, m_nWidth, m_nHeight, &dcScreen, rect.left, rect.top,
    SRCCOPY );
```

2) 抓取系统调色板

正如以上提到的，位图在没有当前系统调色板的情况下是毫无意义的。需要通过该调色板来再次绘制该位图并获得相同的颜色。这里使用 CPalette的CreatePalette()成员函数来创建该调色板，其唯一的调用参数是一个 LOGPALETTLE结构，需要在该结构中填充当前系统调色板。

创建一个空的足够大的LOGPALETTE结构以包含屏幕的颜色：

```
int nColors = ( 1 << ( dcScreen.GetDeviceCaps( BITSPIXEL ) *
     dcScreen.GetDeviceCaps( PLANES ) ) );
LOGPALETTE *pLogPal = ( LOGPALETTE * )new BYTE[
    sizeof(LOGPALETTE) + ( nColors * sizeof( PALETTEENTRY ) )];
```

用颜色方案和颜色号初始化该结构的头部：

```
pLogPal -> palVersion    = 0x300;
pLogPal -> palNumEntries = nColors;
```

用::GetSystemPaletteEntries()捕获当前系统颜色：

```
::GetSystemPaletteEntries( dcScreen.m_hDC, 0, nColors,
    ( LPPALETTEENTRY )( pLogPal -> palPalEntry ) );
```

创建调色板并完成设备环境清除工作：

```
m_pPalette = new CPalette;
```

```
m_pPalette -> CreatePalette( pLogPal );

// clean up
delete []pLogPal;
dcMem.DeleteDC();
dcScreen.DeleteDC();
```

了解该新类可以参考本实例结尾的程序清单 —— 位图类。

3) 使用新的位图类

在本例中将抓取整个桌面的图像放入位图：

```
CWnd *pWnd = GetDesktopWindow();
pWnd -> GetWindowRect( &rect );

CWzdBitmap bitmap;
bitmap.Capture( rect );
```

如下显示该位图：

```
// select bitmap palette
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_nWidth, m_bitmap.m_nHeight,
   &dcComp, 0,0,SRCCOPY );

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

4. 注意

在实例18中曾使用本实例打印应用程序的视图。首先抓取视图，然后将其转变为设备无关位图(DIB)，最后将其拷贝到打印机。

5. 使用光盘时注意

执行随书附带的光盘上的工程，单击 Test/Wzd菜单项，则当前的桌面内容将在视中显示。

6. 程序清单 —— 位图类

```
#ifndef WZDBITMAP_H
#define WZDBITMAP_H

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

// Constructors
```

```
    CWzdBitmap();

    void Capture( CRect &rect );
    CPalette *GetPalette(){return m_pPalette;};

// Implementation
public:
    virtual ~CWzdBitmap();

// Attributes
    int m_nWidth;
    int m_nHeight;

// Operations
private:
    CPalette *m_pPalette;
};
#endif
// WzdBitmap.cpp : implementation of the CWzdBitmap class
//

#include "stdafx.h"
#include "WzdBtmap.h"

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

IMPLEMENT_DYNAMIC( CWzdBitmap, CBitmap )


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

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

void CWzdBitmap::Capture( CRect &rect )
{
    // cleanup old captures
    if ( m_pPalette )
    {
        DeleteObject();
        delete m_pPalette;
    }
```

```
    // save width and height
    m_nWidth = rect.Width();
    m_nHeight = rect.Height();

    ////////////////////////////////////////
    // copy screen image into a bitmap object
    ////////////////////////////////////////

    // create a device context that accesses the whole screen
    CDC dcScreen;
    dcScreen.CreateDC( "DISPLAY", NULL, NULL, NULL );

    // create an empty bitmap in memory
    CDC dcMem;
    dcMem.CreateCompatibleDC( &dcScreen );
    CreateCompatibleBitmap( &dcScreen, m_nWidth, m_nHeight );
    dcMem.SelectObject( this );

    // copy screen into empty bitmap
    dcMem.BitBlt(0,0,m_nWidth,m_nHeight,&dcScreen,rect.left,rect.top,SRCCOPY);

    // this bitmap is worthless without the current system palette, so...

    ////////////////////////////////////////////
    // save system palette in this bitmap's palette
    ////////////////////////////////////////////

    // create an empty logical palette that’s big enough to hold all the colors
    int nColors = ( 1 << ( dcScreen.GetDeviceCaps( BITSPIXEL ) *
        dcScreen.GetDeviceCaps( PLANES ) ) );
    LOGPALETTE *pLogPal = ( LOGPALETTE * )new BYTE[
        sizeof( LOGPALETTE ) + ( nColors * sizeof( PALETTEENTRY ) )];

    // initialize this empty palette’s header
    pLogPal -> palVersion    = 0x300;
    pLogPal -> palNumEntries = nColors;

    // load this empty palette with the system palette's colors
    ::GetSystemPaletteEntries( dcScreen.m_hDC, 0, nColors,
            ( LPPALETTEENTRY )( pLogPal -> palPalEntry ) );

    // create the palette with this logical palette
    m_pPalette = new CPalette;
    m_pPalette -> CreatePalette( pLogPal );

    // clean up
    delete []pLogPal;
    dcMem.DeleteDC();
    dcScreen.DeleteDC();
}
```

## 9.4 实例34：输出DIB位图文件

1. 目标

输出位图到设备无关位图(DIB)文件。

2. 策略

所有文件类型的位图都是设备无关的，也就是说它们包含其本身的颜色表。当位图被装载到一个应用程序使用时，分解该位图为头、颜色表和图像数据 (指向颜色表的指针)。颜色表被放到一个调色板中。该调色板在绘制位图之前必须选入一个设备环境，而头则转变为一个bitmap头。

但是在本例中，将该过程颠倒过来，从其 bitmap头、调色板和图像数据来创建一个 DIB位图。::GetDIBits()API则完成的最后组装工作，然后将该对象与任何其他二进制文件一样写入磁盘。这里将所有的这些功能全部封装到新的位图类中。

3. 步骤

1) 创建DIB对象

用ClassWizard从CBitmap创建新类。

为该新类添加CreateDIB()函数。该函数将从当前装载到 (如LoadBitmap)该位图类的位图创建一个DIB。首先创建一个空的DIB头，可以用该位图头中的信息填充它：

```
HANDLE CWzdBitmap::CreateDIB( int *pbmData )
{
   BITMAPINFOHEADER bi;
   memset( &bi, 0, sizeof( bi ) );
   bi.biSize = sizeof( BITMAPINFOHEADER );
   bi.biPlanes = 1;
   bi.biCompression = BI_RGB;
```

从BITMAP头获取并保存该位图的尺寸：

```
   BITMAP bm;
   GetObject(sizeof( bm ),( LPSTR )&bm );
   bi.biWidth = bm.bmWidth;
   bi.biHeight = bm.bmHeight;
```

获取每个像素所需要的位数。这就是每个图像数据指针的大小。该数据指针越大，在颜色表中被定义的颜色也越多，但同时位图也越大：

```
   int bits = bm.bmPlanes * bm.bmBitsPixel;
   if ( bits <= 1 )
      bi.biBitCount = 1;
   else if ( bits <= 4 )
      bi.biBitCount = 4;
   else if ( bits <= 8 )
      bi.biBitCount = 8;
   else
      bi.biBitCount = 24;
```

这里用GetDIBts()函数来填充颜色表和图像数据。但是首先要创建足够大的内存区域来装载该位图，因此必须确定颜色表和图像数据要多大。

将每一种颜色定义 (4字节)所指定大小与各种可能颜色的数目相乘，以此计算颜色表的大

小：

```
int biColorSize = 0;
if ( bi.biBitCount! = 24 ) biColorSize = ( 1 << bi.biBitCount );
biColorSize* = sizeof( RGBQUAD );
```

将每一个颜色指针的大小与位图尺寸 (大小为宽度乘以高度)相乘，以此计算图像数据的大小，::GetDIBits()将根据DWORD基准来输出每一行，因此必须确保计算的图像数据大小必须正确:

```
bi.biSizeImage = ( DWORD )bm.bmWidth * bi.biBitCount;
   // bits per row
bi.biSizeImage = ( ( ( bi.biSizeImage ) + 31 ) / 32 ) * 4;
   // DWORD aligned
bi.biSizeImage* = bm.bmHeight;            // bytes required for whole bitmap
```

这一步分配大小足以装载该位图头、颜色表和图像数据的全局内存区域，并将到目前为止所创建的位图头复制到其中：

```
HANDLE hDIB = ::GlobalAlloc( GHND, bi.biSize + biColorSize +
   bi.biSizeImage );
LPBITMAPINFOHEADER lpbi =
   ( LPBITMAPINFOHEADER )::GlobalLock( hDIB );
*lpbi = bi;
```

在使用::GetDIBits()之前的最后一步是用选入的位图调色板创建设备环境：

```
CDC dc;
dc.Attach( ::GetDC( NULL ) );
CPalette *pPal = dc.SelectPalette( m_pPalette,FALSE );
dc.RealizePalette();
```

现在创建了::GetDIBits()所需要的所有部分，接下来让该函数完成其工作，如下所示：

```
::GetDIBits( dc.m_hDC, ( HBITMAP )m_hObject, 0,
    ( UINT )bi.biHeight, ( LPSTR )lpbi + ( WORD )lpbi ->
    biSize + biColorSize, ( LPBITMAPINFO )lpbi, DIB_RGB_COLORS );
```

清除设备环境并返回DIB的句柄：

```
::GlobalUnlock( hDIB );
dc.SelectPalette( pPal,FALSE );
dc.RealizePalette();

// return handle to the DIB
return hDIB;
}
```

2) 将DIB对象存入磁盘

现在为该类添加另一个函数，该函数将文件名作为参数，本例将用该函数来保存位图文件。基本上是在CreateDIB()函数中复制所创建的DIB到磁盘文件中，但是首先还要创建一个头：位图文件头。

首先为该类添加新函数，它使用CreateDIB()函数来得到当前位图对象的DIB句柄：

```
void CWzdBitmap::SaveBitmapEx( CString sFile )
{
   // create a DIB bitmap
```

```
int bmData;
HANDLE hDIB = CreateDIB( &bmData );
```

接下来锁定句柄以获取指向该DIB的内存指针：

```
LPBYTE lpBitmap = ( LPBYTE )::GlobalLock( hDIB );
int bmSize = ::GlobalSize( hDIB );
```

**注意** 句柄实际上是指向指针的指针。这样就可以使得它最后所指向的对象在内存中移动。为什么要移动对象呢？这样做的话，只要内存有了空洞(hole)就可以压缩内存——空洞是由于其他应用程序释放了内存而产生的。因为句柄不直接指向内存，而是实际上指向一个内存指针表，所有系统要做的就是更新该指针表。但是在锁定句柄的情况下，可以通知系统停止移动对象并返回指向该对象的真实指针，这样就可以像访问其他内存对象一样访问自己的对象了。如果是为了继续压缩未分配的内存，则在任何时候都不应该锁定内存。

这一步创建一个空白位图文件头并用该位图的有关细节填充:

```
BITMAPFILEHEADER bmfh;
bmfh.bfType = 'MB';    // (actually 'BM' for bitmap)
bmfh.bfSize = sizeof(BITMAPFILEHEADER)+bmSize;
bmfh.bfReserved1 = 0;
bmfh.bfReserved2 = 0;
bmfh.bfOffBits = bmData;
```

首先用CFile创建一个二进制文件，然后写入DIB的头:

```
CFile file;
file.Open( sFile, CFile::modeCreate|CFile::modeWrite );
file.Write( &bmfh,sizeof( BITMAPFILEHEADER ) );
file.Write( lpBitmap,bmSize );
file.Close();
```

进行清除：

```
::GlobalUnlock( hDIB );
::GlobalFree( hDIB );
}
```

3) 使用新的位图函数

使用抓取屏幕或者绘制该位图方式中的一种创建位图后，可以将其存入磁盘：

```
bitmap.SaveBitmapEx(
   "dib.bmp"    // file name and path to save to
   );
```

4. 注意

本例使用了实例18中的CreateDIB()函数，它在打印位图前将位图转变成DIB。实际上在打印机接受之前需要将任何位图都转变成DIB。理论上只需要在打印机的设备环境中实现位图的调色板，然后调用BitBlt()函数即可。但是BitBlt()看起来并不阻止屏幕调色板到打印机调色板的转换。

5. 使用光盘时注意

运行随书附带光盘上的工程时，单击Test/Wzd菜单项。当前桌面内容将被抓取并随后写

入文件dib.bmp内。

6. 程序清单 —— 位图类

```
#ifndef WZDBITMAP_H
#define WZDBITMAP_H

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

// Constructors
    CWzdBitmap();

    void SaveBitmapEx( CString sFile );
    HANDLE CreateDIB( int *pbmData = NULL );

// Implementation
public:
    virtual ~CWzdBitmap();

// Attributes
    int m_nWidth;
    int m_nHeight;

// Operations
private:
    CPalette *m_pPalette;
};
#endif
// WzdBitmap.cpp : implementation of the CWzdBitmap class
//

#include "stdafx.h"
#include "WzdBtmap.h"

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

IMPLEMENT_DYNAMIC( CWzdBitmap, CBitmap )


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

CWzdBitmap::~CWzdBitmap()
{
    if ( m_pPalette )
    {
```

```
        delete m_pPalette;
    }
}


void CWzdBitmap::SaveBitmapEx( CString sFile )
{
    // create a DIB bitmap
    int bmData;
    HANDLE hDIB = CreateDIB( &bmData );

    // get a memory pointer to it
    LPBYTE lpBitmap = ( LPBYTE )::GlobalLock( hDIB );
    int bmSize = ::GlobalSize( hDIB );

    // create file
    CFile file;
    file.Open( sFile, CFile::modeCreate|CFile::modeWrite );

    // write the bitmap header
    BITMAPFILEHEADER bmfh;
    bmfh.bfType = 'MB' ;    // (actually 'BM' for bitmap)
    bmfh.bfSize = sizeof( BITMAPFILEHEADER ) + bmSize;
    bmfh.bfReserved1 = 0;
    bmfh.bfReserved2 = 0;
    bmfh.bfOffBits = bmData;
    file.Write( &bmfh,sizeof( BITMAPFILEHEADER ) );

    // write the bitmap body
    file.Write( lpBitmap,bmSize );

    // cleanup
    file.Close();
    ::GlobalUnlock( hDIB );
    ::GlobalFree( hDIB );
}


HANDLE CWzdBitmap::CreateDIB(int *pbmData)
{
    ////////////////////////////////////////////
    // create DIB header from our BITMAP header
    ////////////////////////////////////////////

    BITMAPINFOHEADER bi;
    memset( &bi, 0, sizeof( bi ) );
    bi.biSize = sizeof( BITMAPINFOHEADER );
    bi.biPlanes = 1;
    bi.biCompression = BI_RGB;

    // get and store dimensions of bitmap
```

```
BITMAP bm;
GetObject( sizeof( bm ),( LPSTR )&bm );
bi.biWidth = bm.bmWidth;
bi.biHeight = bm.bmHeight;

// get number of bits required per pixel
int bits = bm.bmPlanes * bm.bmBitsPixel;
if ( bits <= 1 )
   bi.biBitCount = 1;
else if ( bits <= 4 )
   bi.biBitCount = 4;
else if ( bits <= 8 )
   bi.biBitCount = 8;
else
   bi.biBitCount = 24;


// calculate color table size
int biColorSize = 0;
if ( bi.biBitCount! = 24 ) biColorSize = ( 1 << bi.biBitCount );
biColorSize* = sizeof( RGBQUAD );

// calculate picture data size
bi.biSizeImage = ( DWORD )bm.bmWidth * bi.biBitCount;     // bits per row
bi.biSizeImage = ( ( ( bi.biSizeImage ) + 31 ) / 32 ) * 4;
   // DWORD aligned
bi.biSizeImage* = bm.bmHeight;    // bytes required for whole bitmap

// return size to caler in case they want to save to file
if ( pbmData )
*pbmData = bi.biSize + biColorSize;

//////////////////////////////////////////
// get DIB color table and picture data
//////////////////////////////////////////

// allocate a hunk of memory to hold header, color table and picture data
HANDLE hDIB = ::GlobalAlloc( GHND, bi.biSize + biColorSize +
   bi.biSizeImage );

// get a memory pointer to this hunk by locking it
LPBITMAPINFOHEADER lpbi = ( LPBITMAPINFOHEADER )::GlobalLock( hDIB );

// copy our header structure into hunk
*lpbi = bi;

// get a device context and select our bitmap's palette into it
CDC dc;
dc.Attach( ::GetDC( NULL ) );
CPalette *pPal = dc.SelectPalette( m_pPalette,FALSE );
dc.RealizePalette();
```

```
    // load our memory hunk with the color table and picture data
    ::GetDIBits( dc.m_hDC, ( HBITMAP )m_hObject, 0, ( UINT )bi.biHeight,
        ( LPSTR )lpbi + ( WORD )lpbi -> biSize + biColorSize,
        ( LPBITMAPINFO )lpbi, DIB_RGB_COLORS );

    // clean up
    ::GlobalUnlock( hDIB );
    dc.SelectPalette( pPal,FALSE );
    dc.RealizePalette();

    // return handle to the DIB
    return hDIB;
}
```