**第十课 绘图**

创建一个单文档的应用程序Graphic。

**1 画图**

第一步：

在Menu资源中增加一个“绘图”下拉菜单项，并增加“点”“直线”“矩形”“椭圆”四个菜单项，ID分别为IDM\_DOT, IDM\_LINE, IDM\_RECTANGLE, IDM\_ELLIPSE。并在view类中为这四个菜单项创建响应函数。

第二步：

在view类中增加UINT的类型的private成员变量m\_nDrawType，用以表示画的是什么，在点的响应函数中将m\_nDrawType设置为1，线响应函数中设置为2，矩形中设置为3，椭圆中为4.

第三步：

View类中增加CPoint类型的private成员变量m\_prOrigin，并添加OnLButtonDown和OnLButtonUp两个响应函数。在OnLbttonDown中设置起始点m\_ptOrigin=point。

第四步：

在OnLButtonUp响应函数中画图：

CClientDC dc(this);

switch(m\_nDrawType)

{

case 1:

dc.SetPixel(point,RGB(255,0,0));

break;

case 2:

dc.MoveTo(m\_ptOrigin);

dc.LineTo (point);

break;

case 3:

dc.Rectangle (CRect(m\_ptOrigin,point ));

break;

case 4:

dc.Ellipse (CRect(m\_ptOrigin,point ));

break;

}

第五步：

设置自己的画笔画刷。

在CClientDC dc(this);语句后添加如下代码：

CPen pen(PS\_SOLID,1,RGB(255,0,0));

dc.SelectObject (&pen);

CBrush \*pBrush=CBrush::FromHandle ((HBRUSH)GetStockObject(NULL\_BRUSH));

//选择透明的画刷，并将句柄转换为画刷指针

dc.SelectObject (pBrush);

**2添加一个设置对话框（线型和线宽）**

在对话框资源中添加一个对话框资源，其ID设置为IDD\_DLG\_SETTING。对话框字体设置为宋体，注意宋体前面没有@，有@代表字是倒过来的，大小选10.

在对话框中添加一个静态文本框，命名为“线宽”，添加一个编辑框（Edit Control），其ID为IDC\_LINE\_WIDTH，右击对话框，为对话框添加一个类，类名为CSettingDialog，给编辑框添加一个public的关联变量，变量类型为UINT，名为m\_nLineWidth。

在菜单资源中添加一个菜单项，名为“设置”，ID为IDM\_SETTING，并在view类中为其创建响应函数，view类中增加UITN的m\_nLineWidth的成员变量，用于保存线宽。

在“设置”菜单项的响应函数CGraphicView::OnSetting()中添加如下代码：

CSettingDialog dlg;

dlg.m\_nLineWidth =m\_nLineWidth;

dlg.DoModal(); //显示对话框

if(IDOK==dlg.DoModal ()) //如果按ok

{

m\_nLineWidth=dlg.m\_nLineWidth ;

}

在LBttonUp响应函数中讲CPen设置函数改为：

CPen pen(PS\_SOLID,m\_nLineWidth,RGB(255,0,0));

**设置线型**

在对话框中再添加一个组框（Group Box），命名为线性，在组框中添加三个单选框（Radio button），分别命名为“实线”“虚线”“点线”，并将“实线”单选框属性设置为Group，为实线点选框添加public的关联变量，变量类型为int，名为m\_nLineStyle，view类中增加int的m\_ nLineStyle成员变量，用于保存线型。。

在“设置”菜单项的响应函数CGraphicView::OnSetting()中相应位置再添加代码：

dlg.m\_nLineStyle =m\_nLineStyle;

m\_nLineStyle=dlg.m\_nLineStyle ;

在LBttonUp响应函数中讲CPen设置函数改为：

CPen pen(m\_nLineStyle,m\_nLineWidth,RGB(255,0,0)); // m\_nLineStyle为0时画实线，1虚线，2点线

**这里不用调用Updatedata**

对话框关联变量，需要用Updatedata()函数才可以更新数据， ok按钮集成了更新数据操作，初始化对话框也自动更新数据，所以这里不用调用。

**3颜色对话框**

在菜单中增加一个菜单项，名为颜色，ID为IDM\_COLOR，在view类中为其创建消息响应函数OnColor。

在view类中添加私有的COLORREF类型的m\_clr成员变量，用于保存颜色对话框所选择的颜色，在view的构造函数中将其初始化为红色m\_clr=RGB(255,0,0)。

在响应函数OnColor中添加如下代码：

CColorDialog dlg;

dlg.m\_cc.Flags|=CC\_RGBINIT; //设置颜色对话框初始颜色

dlg.m\_cc.rgbResult=m\_clr;

if(IDOK==dlg.DoModal ())

{

m\_clr=dlg.m\_cc.rgbResult;

}

在LBttonUp响应函数中讲CPen设置函数改为：

CPen pen(m\_nLineStyle,m\_nLineWidth, m\_clr);

画点的函数改为：

dc.SetPixel(point,m\_clr);

**4字体对话框**

在菜单中增加一个菜单项，名为字体，ID为IDM\_FONT，在view类中为其创建消息响应函数OnFont。

在view类中添加私有的CFont类型的m\_font成员变量，用于保存字体对话框所选择的字体，增加私有的CString类型的m\_strFontName成员变量，用于保存字体的名称，在窗口中输出这个名称。

在响应函数OnFont中添加如下代码：

CFontDialog dlg;

if(IDOK==dlg.DoModal ()) //如果按ok键

{

if(m\_font.m\_hObject ) //如果已经绑定了一种字体资源

{

m\_font.DeleteObject ();

}

m\_font.CreateFontIndirectW (dlg.m\_cf.lpLogFont ); //将字体对话框中选择的字体类型绑定到m\_font

m\_strFontName=dlg.m\_cf .lpLogFont ->lfFaceName ; //保存选择的字体名称

Invalidate(); //强制窗口重绘在重绘函数中用选择的字体输出字体名

}

在重绘函数OnDraw中添加如下代码：

CFont \*pOldFont=pDC->SelectObject (&m\_font);

pDC->TextOutW (0,0,m\_strFontName);

pDC->SelectObject (pOldFont);

**5在设置对话框中增加示例功能**

在设置对话框中增加一个组框，ID为IDC\_SAMPLE，名称为“示例”。然后为对话框上的编辑框在CSettingDialog类中增加EN\_CHANGE事件处理程序，这样编辑框一旦输入数据就会相应，同样对三个单选框也增加BN\_CLICKED事件处理程序。在四个消息响应函数中增加代码：Invalidate()；这样四个响应函数都将响应交由窗口重绘函数OnPaint处理。

在CSetingDialog类中增加public的COLORREF类型的成员变量m\_clr。在view类的OnSetting函数中设置dlg.m\_clr =m\_clr，这样示例中绘制的线就以颜色对话框中选择的颜色来绘制。

在对话框类的重绘函数OnPaint中添加如下代码：

UpdateData();

CPen pen(m\_nLineStyle,m\_nLineWidth,m\_clr);

dc.SelectObject (&pen);

CRect rect;

GetDlgItem(IDC\_SAMPLE)->GetWindowRect(&rect); //获取示例栏在桌面的坐标位置

ScreenToClient(&rect); //将相对于桌面的坐标位置转换为相对于对话框的

dc.MoveTo (rect.left+20,rect.top+rect.Height ()/2);

dc.LineTo (rect.right -20,rect.top+rect.Height ()/2);

**6.改变对话框的背景色和控件文字颜色**

每个控件被绘制时都发出WM\_CTlColor消息。在CSettingDialog类中添加WM\_CTlColor消息的消息响应函数。

HBRUSH CSettingDialog::OnCtlColor(CDC\* pDC, CWnd\* pWnd, UINT nCtlColor)

对话框中每个控件被绘制都要调用一次上述响应函数，其中pDC是当前控件的DC，pWnd是当前控件的指针，nCtlColor是当前控件的类型。如果只想对某一个控件操作，可在响应函数中调用pWnd->GetDlgCtrlID ()判断当前控件的ID是不是等于想操作控件的ID。

在CSettingDialog类中增加一个私有的画刷CBrush m\_brush;并初始化为蓝色：

m\_brush.CreateSolidBrush (RGB(0,0,255));

将设置线型的组框的ID改为IDC\_LINE\_STYLE，在OnCtlColor响应函数中，添加如下代码：

if(pWnd->GetDlgCtrlID ()==IDC\_LINE\_STYLE) //如果当前控件是线型组框

{

pDC->SetTextColor (RGB(255,0,0)); //设置组框字体为红色

pDC->SetBkMode(TRANSPARENT); //字体背景色为透明

return m\_brush; //组框画刷为蓝色

}

if(pWnd->GetDlgCtrlID ()==IDC\_LINE\_WIDTH) //如果当前控件是输入线宽的文本框

{

pDC->SetTextColor (RGB(255,0,0));

//pDC->SetBkMode(TRANSPARENT);

pDC->SetBkColor (RGB(0,0,255));

return m\_brush;

}

**更改控件字体**

在对话框资源中增加一个静态文本框，名为“程序员”，ID为IDC\_TEXT。在对话框类中增加私有的CFont m\_font，并在对话框类的构造函数中初始化字体m\_font.CreatePointFont (200,"华文行楷")。

在OnCtlColor响应函数中，添加如下代码：

if(pWnd->GetDlgCtrlID ()==IDC\_TEXT)

{

pDC->SelectObject (&m\_font);

}

**改变OK按纽的字体和背景**

上述方法对于button不起作用。

改变字体颜色

1 创建一个新类，CTestBtn,基类为CButton

2在类中增加虚函数DrawItem()，添加代码。查看msdn：

UINT uStyle = DFCS\_BUTTONPUSH;

// This code only works with buttons.

ASSERT(lpDrawItemStruct->CtlType == ODT\_BUTTON);

// If drawing selected, add the pushed style to DrawFrameControl.

if (lpDrawItemStruct->itemState & ODS\_SELECTED)

uStyle |= DFCS\_PUSHED;

// Draw the button frame.

::DrawFrameControl(lpDrawItemStruct->hDC, &lpDrawItemStruct->rcItem,

DFC\_BUTTON, uStyle);

// Get the button's text.

CString strText;

GetWindowText(strText);

// Draw the button text using the text color red.

COLORREF crOldColor = ::SetTextColor(lpDrawItemStruct->hDC, RGB(255,0,0));

::DrawText(lpDrawItemStruct->hDC, strText, strText.GetLength(),

&lpDrawItemStruct->rcItem, DT\_SINGLELINE|DT\_VCENTER|DT\_CENTER);

::SetTextColor(lpDrawItemStruct->hDC, crOldColor);

3将OK按纽关联成员变量。变量类型为CTestBtn，注意将OK按纽的OwnerDraw特性选中。

改变button的颜色

方法一：利用CSXBtn类更改Button控件背景(为Button添加成员变量 CSXBtn m\_btnSX;)(孙鑫编写)。

方法二：

利用CBtnST类更改Button控件背景（网友编写）Graphic工程里  
1、为Button控件添加成员变量 CBtnST m\_btnST;  
2、重载对话框的OnInitDialog虚函数  
 BOOL CSetDlg::OnInitDialog()  
 {  
  CDialog::OnInitDialog();

  // TODO:  在此添加额外的初始化  
  m\_btnST.SetActiveBgColor(RGB(0,0,255));//设置活动时(鼠标移到其上时)的背景色  
  m\_btnST.SetActiveFgColor(RGB(0,255,0));

  m\_btnST.SetInactiveBgColor(RGB(255,255,0));//设置不活动时的背景色  
  m\_btnST.SetInactiveFgColor(RGB(255,0,255));  
  return TRUE;  // return TRUE unless you set the focus to a control  
 }

**7 在窗口中贴图**

4个步骤

1、创建位图

CBitmap bitmap;

bitmap.LoadBitmap(IDB\_BITMAP1);

2、创建兼容DC

CDC dcCompatible;

dcCompatible.CreateCompatibleDC(pDC);

3、将位图选到兼容DC中

dcCompatible.SelectObject(&bitmap);

4、将兼容DC中的位图贴到当前DC中。在WM\_EraseBkgnd()中调用，但不能再调用基类的擦除背景函数。也可以在OnDraw函数中完成，但效率低，图像会闪烁，因为它先擦除背景，慢。

pDC->BitBlt(rect.left,rect.top,rect.Width(),

rect.Height(),&dcCompatible,0,0,SRCCOPY);

在WM\_ERASEBKGND中实现(也可在OnDraw函数中实现)  
BOOL CGraphicView::OnEraseBkgnd(CDC\* pDC)   
{  
 CBitmap bitmap;  
 bitmap.LoadBitmap(IDB\_BITMAP1);

 CDC dcCompatible;  
 dcCompatible.CreateCompatibleDC(pDC);

 dcCompatible.SelectObject(&bitmap);

     CRect rect;  
 GetClientRect(&rect);  
 //pDC->BitBlt(0,0,rect.Width(),rect.Height(),&dcCompatible,0,0,SRCCOPY);//不能拉伸位图

 BITMAP bmp;  
 bitmap.GetBitmap(&bmp); //为了获取位图大小   
     pDC->StretchBlt(0,0,rect.Width(),rect.Height (),&dcCompatible,0,0,bmp.bmWidth,bmp.bmHeight,SRCCOPY);  
  //可以拉伸位图以适应窗口大小  
 //return CView::OnEraseBkgnd(pDC);//不能执行，否则位图背景会被擦除，无法显示  
 return TRUE;  
}