**第9课 定制外观**

新建单文档的应用程序Style。

**1修改窗口大小，窗口名**

**在CMainFrame::PreCreateWindow(CREATESTRUCT& cs)中对cs进行修改。**

修改大小：

cs.cx=300;

cs.cy=200;

更改窗口明称：

cs.style&=~FWS\_ADDTOTITLE; //或者cs.style= WS\_OVERLAPPEDWINDOW;

cs.lpszName ="fushenghao";

**窗口创建后改变外观，在CMainFrame::OnCreate函数中修改**

//SetWindowLong(m\_hWnd,GWL\_STYLE,WS\_OVERLAPPEDWINDOW); //设置窗口类型

SetWindowLong(m\_hWnd,GWL\_STYLE,GetWindowLong(m\_hWnd,GWL\_STYLE)&~WS\_MAXIMIZEBOX); //获取窗口类型后去掉最大化按钮

**2修改光标图标和背景**

窗口的类型和大小是在创建窗口的时候设定的，而图标光标是在设计窗口类的时候指定的，由MFC底层代码自动生成的。

**可以编写自己的窗口类然后注册**，在PreCreateWindow函数中：

WNDCLASS wndcls;

wndcls.cbClsExtra =0;

wndcls.cbWndExtra =0;

wndcls.hbrBackground =(HBRUSH)GetStockObject(BLACK\_BRUSH);

wndcls.hCursor =LoadCursor(NULL,IDC\_HELP); //光标为问号

wndcls.hIcon=LoadIcon(NULL,IDI\_ERROR); //图标设置为错误图标

wndcls.hInstance =AfxGetInstanceHandle(); //获取应用程序句柄

wndcls.lpfnWndProc =::DefWindowProcA; //默认回调函数

wndcls.lpszClassName ="fsh";

wndcls.lpszMenuName =NULL;

wndcls.style =CS\_HREDRAW|CS\_VREDRAW;

RegisterClass(&wndcls);

cs.lpszClass ="fsh";

发现这样只能改变图标，不能改变光标和背景，因为view覆盖在框架类之上，因此在view类的PreCreateWindow函数中设置：

cs.lpszClass ="fsh";

**可以使用AfxRegisterWndClass函数**

CMainFrame::PreCreateWindow中：

cs.lpszClass=AfxRegisterWndClass(CS\_HREDRAW | CS\_VREDRAW,0,0,

LoadIcon(NULL,IDI\_WARNING));

//第一个参数是窗口类型，第二个光标，第三个背景，第四个图标，图标改为感叹号，框架类中光标背景无意义设为0

CStyleView::PreCreateWindow中：

cs.lpszClass=AfxRegisterWndClass(CS\_HREDRAW | CS\_VREDRAW,

LoadCursor(NULL,IDC\_CROSS),(HBRUSH)GetStockObject(BLACK\_BRUSH),0);

//光标设为十字，背景为黑色

**窗口创建后修改图标光标背景**

在OnCreate（）中修改：

SetClassLong(m\_hWnd,GCL\_HICON,(LONG)LoadIcon(NULL,IDI\_ERROR));

修改光标图标可以在view类中增加WM\_CREATE响应函数OnCreate函数，并添加如下代码：

SetClassLong(m\_hWnd,GCL\_HBRBACKGROUND,(LONG)GetStockObject(BLACK\_BRUSH));

SetClassLong(m\_hWnd,GCL\_HCURSOR,(LONG)LoadCursor(NULL,IDC\_HELP));

**创建一个不断变化的图标。用定时器和SetClassLong完成**

a.准备三个图标文件，放在RES文件夹，Insert->Resource-三个图标，

b.在CMainFrame中增加图标句柄数组，m\_hIcons[3]

m\_hIcons[0]=LoadIcon(AfxGetInstanceHandle(),MAKEINTRESOURCE(IDI\_ICON1));

//之前的图标都是系统图标所以第一个参数都设置为NULL，现在是自己的图标所以第一个参数要设置为应用程序句柄，使用了三种方法获取应用程序句柄。MAKEINTRESOURCE是一个宏，它将整数转化为Win32的资源类型，简单的说它是一个类型转换#define MAKEINTRESOURCEA(i) (LPSTR)((DWORD)((WORD)(i)))

m\_hIcons[1]=LoadIcon(theApp.m\_hInstance,MAKEINTRESOURCE(IDI\_ICON2));

//此处需要用到theAPP对象，故要在文件中声明extern CStyleApp theApp;

m\_hIcons[2]=LoadIcon(AfxGetApp()->m\_hInstance,MAKEINTRESOURCE(IDI\_ICON3));

然后将其初始化

c.然后在定时器中实现

**3.工具栏的编程**

**创建一个工具图标，作为快捷方式**

在资源-》Toolbar-》IDR\_MAINFRAME中绘制一个新的工具栏图标，并将ID号改为IDM\_TEST,在Menu-》IDR\_MAINFRAME中帮助菜单下添加一个TEST菜单项，其ID号也设置为IDM\_TEST，并为其在框架类中创建消息响应函数，添加响应代码MessageBox("test");

这样运行后菜单中的TEST菜单项和工具栏中的项有相同的作用。

**工具栏中添加分隔符以及删除工具栏中一个图标**

添加分隔符只需在工具栏资源中将图标拖动一下即可，删除一个图标只需将该图标从工具栏资源中拖出即可。

**4.创建一个新的工具栏的方法**

第一步：

在资源中创建一个新的工具栏资源，ID号为IDR\_TOOLBAR1。

第二步：

MainFrm.h中增加protected的CToolBar类型的成员变量：CToolBar m\_newToolBar;

第三步：调用CreateEx及LoadToolBar函数

CMainFrame::OnCreate函数中添加如下代码：

if (!m\_newToolBar.CreateEx(this, TBSTYLE\_FLAT, WS\_CHILD | WS\_VISIBLE | CBRS\_RIGHT | CBRS\_GRIPPER | CBRS\_TOOLTIPS | CBRS\_FLYBY | CBRS\_SIZE\_DYNAMIC) ||

!m\_newToolBar.LoadToolBar(IDR\_TOOLBAR1))

// 创建工具栏并加载工具栏并是工具栏停靠在右边

{

TRACE0("未能创建工具栏");

return -1; // 未能创建

}

m\_newToolBar.EnableDocking (CBRS\_ALIGN\_ANY);

// 使工具栏可以停靠在客户端任意一个位置

//EnableDocking(CBRS\_ALIGN\_ANY); //让主框架窗口可以支持任意位置停靠，由于之前调用过，所以这里不用再调用了

DockControlBar(&m\_newToolBar); // 使工具栏可以停靠在主框架窗口上

**点击“新的工具栏”菜单时，隐藏工具栏**

第一种方法：

在菜单资源上增加一个菜单项“新的工具栏”，ID为IDM\_VIEW\_NEWTOOL，并在框架程序中创建响应函数OnViewNewtool()，在该函数中：

if(m\_newToolBar.IsWindowVisible())

{

m\_newToolBar.ShowWindow(SW\_HIDE);

}

else

{

m\_newToolBar.ShowWindow(SW\_SHOW);

}

RecalcLayout(); //工具栏隐藏后让工具条也消失

DockControlBar(&m\_newToolBar); //让工具栏被拖动后，隐藏显示功能不出错

第二种方法：

在OnViewNewtool()函数中添加如下代码：

ShowControlBar(&m\_newToolBar,!m\_newToolBar.IsWindowVisible(),FALSE);

//第二个参数为ture时显示工具栏，false时隐藏工具栏，第三个参数为false让显示的时候立即显示没有延时

**给“新的工具栏”菜单项加复选标记**

给该菜单项增加UPDATE COMMAND UI响应函数CMainFrame:: OnUpdateViewNewtool (CCmdUI \*pCmdUI)，并添加代码：

pCmdUI->SetCheck(m\_newToolBar.IsWindowVisible()); //为真复选上，即在该菜单项前打钩

**5.状态栏编程**

第一步：

在字符串表资源中添加两个资源：

IDS\_TIMER 时钟

IDS\_PROGRESS 进度栏

第二步：

在MainFrm.cpp文件中指示器数组indicators中添加这两个资源ID号.

static UINT indicators[] =

{

ID\_SEPARATOR, // 状态行指示器

IDS\_TIMER, // 新添加的时间指示器

IDS\_PROGRESS, // 新添加的进度栏指示器

ID\_INDICATOR\_CAPS,

ID\_INDICATOR\_NUM,

ID\_INDICATOR\_SCRL,

};

第三步：

至此可以在窗口右下角显示五个状态条，时钟、进度栏、大小写等，为了让时钟栏显示时间在OnCreate函数中添加如下代码：

CTime t=CTime::GetCurrentTime(); //获取当前时间

CString str=t.Format ("%H:%M:%S"); //以时：分：秒格式显示

CClientDC dc(this);

CSize sz=dc.GetTextExtent (str); //获得str显示所需要的大小

int index=0;

index=m\_wndStatusBar.CommandToIndex(IDS\_TIMER); //获得时钟栏在状态指示器数组中的位置，其实就是1.

m\_wndStatusBar.SetPaneInfo (index,IDS\_TIMER,SBPS\_NORMAL,sz.cx);

//设置时钟栏的长度

m\_wndStatusBar.SetPaneText (index,str); //将时间在状态栏显示

第四步：

设置一个定时器，在定时器响应函数中，重新获取当前时间，并显示，用以更新显示的时间。

**6.进度栏**

1 在Mainfrm.h中增加protected的成员变量CProgressCtrl m\_progress

2 在OnCreate中添加如下代码：

CRect rect;

m\_wndStatusBar.GetItemRect(2,&rect); //获取状态栏中第二项即进度栏的位置

m\_progress.Create(WS\_CHILD | WS\_VISIBLE,// | PBS\_VERTICAL,

rect,&m\_wndStatusBar,123); //创建进度栏，第一个参数是属性，可以是水平的竖直的，第二个参数是位置，第三个参数是父窗口指针，这里设置为状态栏窗口，最后一个参数随便取的一个ID号。

m\_progress.SetPos(50); //设置进度条中当前进度，100为满格，50为半格

这时运行发现不正确，矩形大小获取不正确，这是因为在OnCreate中状态栏还没有创建完成，因此，获取失败，解决办法可以是自定义消息，当程序执行OnCreate消息时，将自定义的消息放入消息队列中，当Oncreate消息执行完成后，再执行自定义的消息响应函数。

3在MainFrm.h中#define UM\_PROGRESS WM\_USER+1 //自定义消息号要大于WM\_USER

在MainFrm.h中添加消息映射函数afx\_msg LRESULT OnProgress(WPARAM ,LPARAM);

在MainFrm.cpp中消息映射中添加ON\_MESSAGE ( UM\_PROGRESS, &CMainFrame::OnProgress )

在MainFrm.cpp中，消息响应函数：

LRESULT CMainFrame::OnProgress(WPARAM ,LPARAM)

{

CRect rect;

m\_wndStatusBar.GetItemRect(2,&rect);

m\_progress.Create(WS\_CHILD | WS\_VISIBLE | PBS\_SMOOTH,

rect,&m\_wndStatusBar,123);

m\_progress.SetPos(50);

return true;

}

4最后在OnCreate中调用 PostMessage(UM\_PROGRESS);//不能用SendMessage()因为这个函数发送的消息是立即响应的，即Oncreate响应还没结束就去响应自定义的消息

**解决重绘时进度栏改变的问题**

在OnPain()中重写代码

CRect rect;

m\_wndStatusBar.GetItemRect(2,&rect);

m\_progress.Create(WS\_CHILD | WS\_VISIBLE | PBS\_SMOOTH,

rect,&m\_wndStatusBar,123);

m\_progress.SetPos(50);

此时将OnCreate中PostMessage(UM\_PROGRESS);注释掉，因为创建窗口的过程就调用了OnPain()函数。为了让进度随时间改变在定时器消息处理函数中加入

m\_progress.StepIt();

**7.显示鼠标位置**

在View中增加OnMouseMove()处理函数

CString str;

str.Format("x=%d,y=%d",point.x,point.y);

//((CMainFrame\*)GetParent())->m\_wndStatusBar.SetWindowText(str);

//((CMainFrame\*)GetParent())->SetMessageText(str);

//((CMainFrame\*)GetParent())->GetMessageBar()->SetWindowText(str);

// ((CMainFrame\*)GetParent())->m\_wndStatusBar.SetPaneText (0,str);

GetParent()->GetDescendantWindow(AFX\_IDW\_STATUS\_BAR)->SetWindowText(str);

以上五种方法人选一种

**8.加入启动画面**

Project-Component and ->Visual C++ Components->SplashScreen->插入

在vs2010下暂时还没有找到插入组件操作。