定制 Visual C++ 应用程序工具条技术

杨少波　（中国科学院计算技术研究所　100080）

摘要：本文通过实例着重介绍在 Visual C++ 5.0 版应用程序中定制工具条的技术实现，以编制出命令交互丰富的 Windows 应用程序。

关键词：Visual C++　应用程序工具条　定制

1. 简介

微软 Visual C++ 5.0 版 Develop Studio 是一个功能强大的 Windows 应用程序开发平台，用户利用其中的 MFC 类库及 AppWizard、ClassWizard、资源编辑器等工具，可以快速地生成标准 Windows GUI 用户界面的应用程序；但为丰富应用程序的命令交互，可能需要增加或定制应用程序工具条。本文通过实例论述这方面的技术实现，并给出功能实现的关键部分程序代码。

2. 新增工具条

Visual C++ 的 AppWizard 工具正常情况下只为应用程序生成一个标准工具条，用户在某些应用场合下，需扩充或改变工具条以丰富应用程序的命令交互。图 1 为本文实例，其中新增了两组工具条，主要实现过程如下。

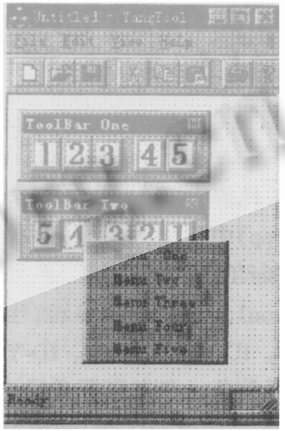

图 1　新增工具条

(1)内嵌工具条成员数据变量。利用资源编辑器设计出工具条按钮位图;再在主窗框类的定义文件(MainFram.h)中增加一个内嵌的工具条按钮成员数据变量。

```
class CMainFrame : public CFrameWnd
{
public: CToolBar m-YangToolBar1; //内嵌工具条成员数据变量
    CToolBar m-YangToolBar2;
    // ClassWizard generated virtual function overrides
    //{{AFX-VIRTUAL(CMainFrame)
    public:
    virtual BOOL PreTranslateMessage(MSG * pMsg);
    //}}AFX-VIRTUAL
    ... //其他成员定义
};
```

(2)定义出工具条按钮 ID 数组。在主窗框类的实现文件(MainFrm.cpp)中定义出工具条按钮 ID 数组 static UINT YangToolBarIDs[] ={

ID-YangButton1, ID-YangButton2, ID-YangButton3, ID-SEPARATOR, ID-YangButton4, ID-YangButton5,};

(3)创建出工具条。在主窗框类的 OnCreate()成员函数中,加入创建工具条的代码,并调用工具条 CToolBar::SetButtons()函数将各个子控制按钮 ID 与工具条位图 ID 相关联。

```
int CMainFrame:: OnCreate ( LPCREATESTRUCT lpCreateStruct)
{ if (CFrameWnd::OnCreate(lpCreateStruct) == -1)
    return -1;
  if (! m-YangToolBar1. Create ( this, WS-CHILD | WS-VISIBLE | CBRS-TOP|
    CBRS-TOOLTIPS|CBRS-SIZE-DYNAMIC)
    || ! m-YangToolBar1. LoadToolBar ( IDR-YangToolBar))
  {
    TRACE0("Failed to Create Toolbar One \ n");
  return -1; // 不能创建工具条一
  }
  m-YangToolBar1.SetButtons(YangToolBarIDs,
      sizeof(YangToolBarIDs)/sizeof(UINT));
  m-YangToolBar1. SetWindowText (-T ( " ToolBar One"));
  //使工具条具一有漂浮特性
  m-YangToolBar1. EnableDocking ( CBRS-ALIGN-ANY);
  EnableDocking(CBRS-ALIGN-ANY); //漂浮工具条
  DockControlBar ( &m-YangToolBar1, AFX-IDW-DOCKBAR-TOP);
  if (! m-YangToolBar2. Create ( this, WS-CHILD | WS-VISIBLE | CBRS-TOP|
    CBRS-TOOLTIPS|CBRS-SIZE-DYNAMIC)
    || ! m-YangToolBar2. LoadToolBar ( IDR-YangToolBar))
  {
    TRACE0("Failed to Create Toolbar Two \ n");
    return -1; // 不能创建工具条二
  }
  //动态更改工具条按钮布局
  m-YangToolBar2. SetButtons ( NULL, sizeof ( YangToolBarIDs)/sizeof(UINT));
  m-YangToolBar2. SetButtonInfo(0, ID-YangButton5, TBBS-BUTTON,4);
  m-YangToolBar2. SetButtonInfo(1, ID-YangButton4, TBBS-CHECKBOX,3);
  m-YangToolBar2.SetButtonInfo(2,ID-SEPARATOR, TBBS-SEPARATOR,7);
  m-YangToolBar2. SetButtonInfo(3, ID-YangButton3, TBBS-BUTTON,2);
  m-YangToolBar2. SetButtonInfo(4, ID-YangButton2, TBBS-BUTTON,1);
  m-YangToolBar2. SetButtonInfo(5, ID-YangButton1, TBBS-BUTTON,0);
  //设置工具条标题文字
  m-YangToolBar2. SetWindowText (-T ( " ToolBar Two"));
  //使工具条二具有漂浮特性
  m-YangToolBar2. EnableDocking ( CBRS-ALIGN-ANY);
  EnableDocking(CBRS-ALIGN-ANY);
  DockControlBar ( &m-YangToolBar2, AFX-IDW-DOCKBAR-BOTTOM);
  ... //其他程序代码
  return 0;
  }
```

3. 动态更改工具条按钮

可根据应用程序执行时的需要，动态地更改工具条命令按钮以定制工具条。

(1)改变按钮的布局。在调用 CToolBar::SetButtons()函数时如将第一个参数置为 NULL，再调用成员函数 SetButtonInfo()可对每个子控制按钮进行重新布局，以调整显示位置和筛选命令按钮(图 1 中新增的第二个工具条为第一个工具条的反向布置，程序代码见前文(3)节)。

(2)创建出 Check - Box 选项按钮。在调用 CToolBar::SetButtonInfo()关联按钮 ID 时，如指定 TBBS-CHECKBOX 风格(见前文(3)节)，则可创建出选项工具按钮；也可在 ON-UPDATE-COMMAND-UI 消息响应函数中，调用 CCmdUI::SetCheck()成员函数，可将命令型按钮转化为选项型按钮。

```
void CMainFrame::OnUpdateYangButton4(CCmdUI * pCmdUI)
{ pCmdUI->SetCheck(1);}
```

(3)显示或隐藏工具条。调用 CFrameWnd::ShowControlBar(&m-YangToolBar1, TRUE, FALSE)可以显示或隐藏工具条(第二个参数置为 FALSE，CFrameWnd::ShowControlBar(&m-YangToolBar1, TRUE, FALSE))。

(4)改变工具条按钮位图外现。调用 CToolBar::LoadBitmap(新工具条位图 ID)可用新的工具条位图代替现有的工具条位图；也可调用另一个成员函数 CToolBar::LoadToolBar(新工具条 ID)来动态更改工具条外观。

```
m-YangToolBar1.LoadToolBar(IDR-YangToolBar);
//新位图
```

4. 使工具条具有漂浮特性

当工具条具有漂浮性能时，用户可拖放它到指定位置处；但必须在主窗框类中加入满足如下三个条件的程序代码，否则工具条不被漂浮。

首先调用 CToolBar::EnableDocking()以允许主窗框漂浮工具条；其次调用 CFrameWnd::EnableDock()允许工具条自身能被漂浮；最后调用 CFrameWnd::DockControlBar()将工具条置于主窗框的某一位置处(本例新增的第一个工具条在初始状态时停伯在主窗框的顶部，第二个在底部，程序代码见前文(3)节)。

```
... //允许主窗框漂浮工具条
m-YangToolBar1.EnableDocking(CBRS-ALIGN-ANY);
EnableDocking(CBRS-ALIGN-ANY); //漂浮工具条
DockControlBar(&m-YangToolBar1, AFX-IDW-DOCKBAR-TOP);
... //定位到主窗框某一位置处
```

5. 工具条响应鼠标右键消息弹出浮动菜单

在 Windows 95 的 Win32 应用程序中，用户通常单击右键可以弹出浮动式的属性菜单，本文实现了给工具条增加右键弹出属性菜单功能，其主要实现原理是在主窗框类中覆盖 PreTranslateMessage()函数，拦截右键单击 WM-RBUTTONDOWN 消息，弹出浮动式的属性菜单(见图 1 所示)。

```
BOOL CMainFrame::PreTranslateMessage(MSG * pMsg)
{   if (pMsg->message == WM-RBUTTON-DOWN) //拦截鼠标右键消息
    {   CWnd * pWnd = CWnd::FromHandlePermanent(pMsg->hwnd);
        CControlBar * pBar = DYNAMIC-DOWNCAST(CControlBar, pWnd);
        if (pBar != NULL)
        { CMenu Menu;
          CPoint pt; //得到鼠标右键单击的位置
          pt.x = LOWORD(pMsg->lParam);
          pt.y = HIWORD(pMsg->lParam);
          pBar->ClientToScreen(&pt);
          if (Menu.LoadMenu(IDR-YangMenu)) //加载菜单资源
          { CMenu * pSubMenu = Menu.GetSubMenu(0);
            if (pSubMenu! =NULL)
            { pSubMenu->TrackPopupMenu(TPM-LEFT-ALIGN | TPM-RIGHTBUTTON,
                  pt.x, pt.y, this);
              //显示出属性弹出菜单
              }
          }
        }
    return CFrameWnd::PreTranslateMessage(pMsg);
}
```

参考文献

[1] 何晓刚 译，Microsoft Visual C++ 4.0 教程，1997，北京：科学出版社

(来稿时间：1998 年 5 月)