© 2014 IJIRT | Volume 1 Issue 5 | ISSN : 2349-6002

Dialog Based

Simmy Agarwal, Rekha
Dronacharya College of Engineering

Abstract: **For many applications, a dialog provides a sufficient user interface. The dialog window immediately appears when the user starts the application. The user can minimize the dialog window, and as long as the dialog is not system modal, the user can freely switch to other applications. In this example, the dialog functions as a simple calculator, as shown in Figure 1. Class Wizard takes charge of defining the class data members and generating the DDX (Dialog Data Exchange) function calls, everything but the coding of the compute function. The application's resource script, mymfc 23A.rc, defines an icon as well as the dialog.**

I. LIST BOX

List boxes are the controls that present the user with a list of items displayed in a box. In this program; we will create a list box and will present the items to the user as a list. The list box will present the items to the user as a list. Since we can add as many items, the list box won't be able to display the items at once so it will display the scroll bar at the right . When the user clicks a list box item , we will display that item in a text box.

STEPS:-

- Create a dialog based program named "list".
- Drag and drop a list box , a edit box and a text box on the dialog editor.

Creating member variables for the list box

- First of all, to add items to the list box we have to give it a name to refer to it. So ,go to view ->class wizard_>member variable->select IDC_LIST1->add variable ->write m ->select category as control and variable type->CListBox->ok.
- Similarly add m_text to the edit box.

Initializing data in the list box

- To initialize data in the list box we write in the function OnInitDialog() so go to view ->class wizard ->double click on the function OnInitDialog().

```
Bool CListDlg :: OnInitDialog()
{
        CDialog :: OnInitDilog()
      m _list.AddString("Item 01");
        m _list.AddString("Item 02");
        m _list.AddString("Item 03");
        m _list.AddString("Item 04");
```

Handling list box double clicks

- Now when the user double clicks an item , we will display that item in the textbox as go to view-> classwizard -> messagemaps -> select IDC_LIST1 ->select the message LBN_DblClick(double click on it)->ok->edit code

```
Void CListDlg :: OnDblClickList1()

{

    m_list = GetText(m_list.GetCurSet(),m_text);

UpdateData (false);

}
```

We used the function GetCurlSet() to get the list box current selection. It returns the index value of that item to m_text which is over text box.

```
m_text = "600";

UpdateData (false);

}
```

II. CHECK BOX

- create a dialog ased program named "check".
- Place there check boxes and edit box on the dialog editor.

Aligning the controls

- To align the controls , hold the control key and click on the control , clicking top 1 at the end making it the reference control.

A reference control is the control according to which all other controls get aligned.

Go to layout -> align -> left

We can do other format

Connecting check boxes to the code

- Go to view -> class wizard -> member variable -> click IDC_EDIT! ->Add variable ->write m_text.
- Go to view ->class wizard ->message maps -> select ID IDC_CHECK!_>BN_CLICKED (double click)-> Edit code

```
Void  CheckDlg :: OnCheck1()
{
     m_text = “check box1 clicked”;
  UpdateData(false);
```

- Repeat step 5 for checkbox 2 and checkbox3 .

III. Summary

- Open visual c++ and click the new item in the file menu, opeing the new dialog box.
- Now select the mfc app wizard (exe) entry in the new dialog box.
- Give the name of the new program
- Click ok to start vc++.
- Click the option mark in dialog based, click finish.
- Here app wizard indicates your class is created.
- Click the resource tab ,open the dialog folder, and click the entry for our program main window.this opens the dialog editor.
- Add a new button with a caption.
- Open class wizard to connect click me button to our code.
- Use class wizard to connect a button to our code
- Use class wizard to connect a button to an evebt handler.
- Add a member variable , m_edit to edit box.
- Add required code to OnButton1();