Dialog Boxes

More notes at:

http://www.cs.binghamton.edu/~reckert/360/class9.htm

Dialog Boxes

- **Popup child windows created by Windows**
- **Used for special-purpose input & output**
  - **Principal I/O mechanism in Windows**
- **Contain several child window controls**
- **Layout & what it does is are predefined (template--a resource)**
- **How it does is determined by a "Dialog box procedure"**
- **Destroyed immediately after use**

Steps in Using:

- 1. Set up the template in the resources (.rc file)
  - Specifies controls used, their style/layout
  - Can be prepared "visually" with Visual Studio dialog box editor
  - Or "manually" with a text editor

- 2. (Win32 API) Write the dialog box procedure
  - Code to carry out dialog box's tasks
  - Placed in .cpp file
  - Provides message-processing capability
  - Messages from controls handled inside this procedure
  - Messages can be sent to the dialog box
  - A callback function like main window procedure *WndProc()*
  - But not the same
    - Part of the callback is inside Windows
    - It interprets some keystrokes (tab)
    - It calls our procedure
- 2. (MFC) Instantiate a CDialog object

Types of Dialog Boxes

- **Modal**
- **Modeless**
- **System Modal**

Steps in Designing, Creating, Using a Modal Dialog Box: Win32 API

1. **Determine child window controls needed inside**
2. **Design dialog box template (easiest with dialog box editor)**
3. **Write message-processing function**
   - `Like a WndProc`
4. **Activate dialog box by calling *DialogBox()***
   - **Typically in response to menu item selection in *WndProc()***
5. **Resulting dialog box stays on screen until call to *EndDialog()***
   - **from inside dialog box function**

DialogBox(…)

- **Parameters:**
  - **1. App's instance handle**
  - **2. Dialog box ID name**
    - **Specified in dialog box template when .rc file created**
  - **3. Handle of dialog box's parent window**
  - **4. Address of *dialog box function* that will process its messages**
    - **A callback function**
- **Creates modal dialog box from app's dialog box template resources**
- **Displays dialog box & switches msg-processing to it**
- **Control returned when its msg-processing function terminates dialog box**
  - **By calling *EndDialog()* ;**

WM_INITDIALOG Message

- **Like ordinary an window's WM_CREATE message**
- **Processed before window (dialog box) is made visible**
- **Good place to put dialog box initialization code**

EndDialog(…)

- **Destroys dialog box**
- **Returns control to function (*WndProc()*) that called *DialogBox()***
- **Parameters:**
- **1. window handle passed to dialog box function (hDlg)**
- **2. integer value returned by *DialogBox()***
  - **Way of getting info from dialog box function to calling program**
    - **Could also use global variables to retrieve data from the Dialog Box**

User Interaction with Dialog Box Controls

- **WM_COMMAND message**
  - **LOWORD(wParam) contains control ID (as usual)**
  - **lParam, wParam contain message data (as usual)**

Exchanging Data with a Dialog Box

- **Exchanging data between dialog box function and app's *WndProc()***
- ***SendMessage()* could be used to send message to control inside, BUT:**
  - **Need to know control's handle**
  - **Not known since Windows creates the controls**
  - **IDs are known--specified in resource template**
- **Use *GetDlgItem ()* to get control's handle:**
  - ***hControl = GetDlgItem(hDlg, controlID);***
- **Then *SendMessage(hControl, Msg, wParam, lParam);***

- **Both functions can be combined using *SendDlgItemMessage():***
- ***SendDlgItemMessage(hDlg, controlID, Msg, wParam, lParam);***
  - **Sends Msg to control whose ID is controlID**