Child Window Controls

More notes at:

http://www.cs.binghamton.edu/~reckert/360/class5a.htm

Child Window Controls

Windows created by a parent window

- An app uses them in conjunction with parent
- Normally used for simple I/O tasks
- Properties, appearance, behavior determined by predefined class definitions
  - But behavior can be customized
  - Easy to set them up as common Windows objects
    - buttons, scroll bars, etc.
- Can also define custom Child Window Controls

- Allow user to display/select/input info in standard ways
- Windows Environment does most of work in:
  - painting/updating a Control's screen area
  - determining what user is doing
- Often used as input devices for parent window
- Are the "working components" of Dialog Boxes
- Windows OS contains each control's ***WinProc***
  - so messages to controls are processed in predefined way
- Parent window communicates with controls by sending/receiving messages

Six “Classic” Control Types

- Go back to first versions of Windows

Type	Window Class	MFC Class
Static Text	**“STATIC”**	**CStatic**
Button	**“BUTTON”**	**CButton**
Edit Control	**“EDIT”**	**CEdit**
List Box	**“LISTBOX”**	**CListBox**
Combo Box	**“COMBOBOX”**	**CComboBox**
Scroll Bar	**“SCROLLBAR”**	**CScrollBar**

- All are windows

Creating Controls--Win32 API

- ***CreateWindow ()***
  - ***F***or any kind of window, including a control
  - Typically called in response to WM_CREATE or WM_SIZE
- Parameters:
  - 1. Predefined control window class names:
    - "STATIC", "BUTTON", "EDIT", "LISTBOX", "COMBOBOX", "SCROLLBAR", others
  - 2. Name of the window
    - BUTTON, EDIT, STATIC classes:
      - text in center of control
    - COMBOBOX, LISTBOX, SCROLLBAR classes:
      - ignored (use "")

- 3. Window style

  WS_, SS_, BS_, ES_, LBS_, CBS_, SBS_ (see CreateWindow help)
  - Several styles can be combined with the bitwise or operator (|)
  - All controls should include WS_CHILD style
- Parameters 4-7:
  - X,Y position (Relative to the upper left corner of parent window client area)
  - Width & Height
- 8. Handle to the parent window

- 9. Handle to "menu"
  - Controls don't have menus
  - So hMenu parameter used to hold control's integer ID
  - ID value passed with WM_COMMAND message generated when user interacts with the control
  - Allows program to identify which control was activated
- 10. Handle to instance of program creating control
  - ***GetWindowLong()*** usually used to get this value
  - Could declare a global HINSTANCE variable and use value that comes in from WinMain(…)
- 11. Pointer to window creation data
  - Normally NULL

Example (Win32 API)

- In response to WM_CREATE in Main Window's WndProc():

```
HWND  hMyButton;
HINSTANCE  hInstance;
hInstance = (HINSTANCE) GetWindowLong (hWnd,
    GWL_HINSTANCE);
hMyButton = CreateWindow ("BUTTON", "Push Me",
    WS_CHILD | BS_PUSHBUTTON, 10, 10, 130, 60, hWnd,
    (HMENU)ID_MYBUTTON, hInstance, NULL);
ShowWindow (hMyButton, SW_SHOWNORMAL);
```

- Could be done in response to WM_SIZE message
  - Then width & height of client area can be gotten from LOWORD & HIWORD of lParam

Messages from Controls

- Most work as follows:
  - User interacts with the control
  - WM_COMMAND message sent to parent window
  - LOWORD(wParam) = Control ID
  - lParam = control's window handle
  - HIWORD(wParam) = notification code
    - identifies what the user action was

Win32 API Control Message Handlers

- Put Control message handlers in same switch/case statement with menu handlers (WM_COMMAND)
- Done just as for menu handlers

Sending Messages to Controls, Win32 API

- ***SendMessage( )***--sends message to a window's WinProc( )
- Doesn't return until message has been processed
- Parameters:
  - Handle of destination window
  - ID of message to send
  - wParam and lParam values containing message data, if any
    - Need to be type cast

Example, Win32 API

- **Send a message to hMyControl**

  SendMessage (hMyControl, WM_SETTEXT, 0, (LPARAM) "Hello") ;
  - Here message is WM_SETTEXT
  - When received, control's ***WndProc()*** changes control's window name (text string displayed)
  - For this message wParam must be 0;
- There are many messages that can be sent to a control
- Depend on type of control
  - See online help

Static Controls

- Lots of styles, see online help on “Static Control Styles”. Some examples:
  - SS_BITMAP, SS_CENTER, SS_GRAYFRAME, SS_ICON, SS_SIMPLE, SS_WHITEFRAME, etc.
- Change text with WM_SETTEXT message
  - May need to format values with wsprintf( )
- Retrieve text with WM_GETTEXT message or GetWindowText()
  - Can convert string to values using sscanf( )
- Static Controls do not send messages
- Program examples: static, static_mfc

Button Controls

- Some Styles: BS_PUSHBUTTON, BS_RADIOBUTTON, BS_CHECKBOX, BS_OWNERDRAW, BS_GROUPBOX, etc.
- Button notification codes:
  - BN_CLICK (also BN_DOUBLECLICK)
- Some messages you can send to buttons:
  - BM_SETCHECK, BM_GETCHECK, BM_SETSTATE, BM_GETSTATE, etc.
- Program examples: button

Child Window Controls: List Boxes, Edit Controls

List Box Controls

- Lots of styles: see on-line help on LBS_
  - LBS_STANDARD very common
    - can send messages to parent
- Program communicates with list box by sending it messages; some common List Box messages:
  - LB_RESETCONTENTS, LB_ADDSTRING, LB_GETCURSEL, LB_GETTEXT, LB_DELETESTRING
- Some List Box Notification codes:
  - LBN_SELCHANGE, LBN_DBLCLK
- Combo boxes much like list boxes (CBS_, CB_, CBN_)
- Program examples: listbox, combo

EDIT CONTROLS

- **For viewing and editing text**
- Current location kept track of with a "carat"
  - A small vertical line
- Backspace, Delete, arrow keys, highlighting work as expected
- Scrolling possible (use WS_HSCROLL, WS_VSCROLL styles)
- No ability to format text with different fonts, sizes, character styles, etc.
  - Use Rich Edit Control for this

Edit Control Styles

- Some common styles
  - ES_LEFT, ES_CENTER, ES_RIGHT, ES_MULTILINE, ES_AUTOVSCROLL, ES_PASSWORD
    - See Online Help on "Edit Styles"

Edit Control Text

- Text in an edit control stored as one long character string
- Carriage return <CR> is stored as ASCII code (0x0D,0x0A)
- <CR> inserted automatically if a line doesn't fit and wraps
- NULL character inserted only at end of last line of text

Edit Control Messages

- User interacts with edit control,
  - WM_CONTROL message to parent
  - LOWORD(wParam) = Control ID
  - lParam = control's window handle
  - HIWORD(wParam) = EN_** notification code
    - identifies what the user action was
    - e.g., EN_CHANGE
    - See Online Help EN_***

Sending Messages to an Edit Box

- As with other controls use SendMessage()
- Some important messages
  - WM_GETTEXT, WM_SETTEXT
  - Multiline edit boxes:
    - EM_GETLINECOUNT(multiline edit boxes)
      - Returns number of lines in the control
      - EM_GETLINE: Copy a line to a buffer
      - EM_LINEINDEX: Get a line's character index
      - Number of characters from the beginning of edit control to start of specified line
  - EM_LINELENGTH to get length of line
- See Edit1 example program