

Windows Programming: Using Edit Control

Seong Jong Choi
chois@mmlab.net

[Multimedia Lab.](#)

Dept. of Electrical and Computer Eng.
University of Seoul
Seoul, Korea

What is a Control?

- *A control* is a child window.
 - A window that has the `WS_CHILD` style. A child window always appears within the client area of its parent window.
- perform simple input and output (I/O) tasks.
- often used within dialog boxes, but can be used in other windows.
- The following are the standard Windows controls:
 - Buttons
 - Combo Boxes
 - Edit Controls
 - List Boxes
 - Rich Edit Controls
 - Scroll Bars
 - Static Controls
 - Text Object Model
 - Windowless Rich Edit Controls

How to Create an Edit Control?

```
hOutWnd = CreateWindow (
    "EDIT",                // Window Class Name (Predefined)
    NULL,
    WS_BORDER | WS_CHILD | WS_VISIBLE | //Window Styles
    WS_VSCROLL | ES_LEFT | ES_MULTILINE | ES_AUTOVSCROLL,
    0,0,0,0,              //Position and Size
    hWnd,                 // handle to parent or owner window
    (HMENU)ID_OUTBOX,    //menu handle or child identifier
    (HINSTANCE) GetWindowLong(hWnd, GWL_HINSTANCE),
    // handle to application instance
    0);
```

When to Create the Control?

- After the creation of the main window.
- Before normal operation.
- Therefore, after WM_CREATE message

```
LRESULT CALLBACK WndProc(HWND hWnd, UINT message, WPARAM wParam, LPARAM lParam)
{
    ...
    switch (message)
    {
        case WM_CREATE:
            // Create Edit control for typing to be sent
            hOutWnd = CreateWindow("EDIT",
                NULL,
                WS_BORDER | WS_CHILD | WS_VISIBLE|WS_VSCROLL| ES_LEFT |
                ES_MULTILINE | ES_AUTOVSCROLL,
                0,0,0,0,
                hWnd,
                (HMENU) ID_OUTBOX,
                (HINSTANCE) GetWindowLong(hWnd, GWL_HINSTANCE),
                0);
            break;
        case WM_
```

We Need Two Edit Controls.

- The one for user input.
- The other is for echo.
 - Later, you'll use this control for displaying received characters from the peer.
 - Set window style `ES_READONLY`

How to Position and Size the Controls

- Split the main window horizontally.
- Use `MoveWindow()` in `WM_SIZE` message handler.
- *lParam* specifies the new dimension of the client area (main window)
 - The low-order word of *lParam* specifies the new width .
 - The high-order word of *lParam* specifies the new height.

```
case WM_SIZE :
    // Size OutBox Edit Control
    MoveWindow(hOutWnd,
        1,1, // Upper Left Corner
        LOWORD(lParam) - 2, // Width of Parent Window
        (HIWORD(lParam) / 2) - 2, // Half the height of Parent
        TRUE); // repaint

    // Size Inbox Edit Control
    MoveWindow(hInWnd,
        1, (HIWORD(lParam) / 2) + 1, // Half Way down right side
        LOWORD(lParam) - 2, // Width of Parent Window
        (HIWORD(lParam) / 2) - 2, // Half the height of Parent
        TRUE); // repaint
    break;
```

How Do We Know the User Input?

- Edit controls send `WM_COMMAND` messages to the parent window procedure. The meanings of the `wParam` and `lParam` variables are:
 - `LOWORD(wParam)` Child window ID
 - `HIWORD(wParam)` Notification code
 - `lParam` Child window handle
- The notification codes are:
 - `EN_SETFOCUS` Edit control has gained the input focus.
 - `EN_KILLFOCUS` Edit control has lost the input focus.
 - `EN_CHANGE` Edit control's contents will change.
 - `EN_UPDATE` Edit control's contents have changed.
 - `EN_ERRSPACE` Edit control has run out of space.
 - `EN_MAXTEXT` Edit control has run out of space on insertion.
 - `EN_HSCROLL` Edit control's horizontal scroll bar has been clicked.
 - `EN_VSCROLL` Edit control's vertical scroll bar has been clicked.
- We will use `EN_CHANGE` notification for detecting user input.

How Do We Know the User Input?

```
case WM_COMMAND:
    wmId = LOWORD(wParam);
    wmEvent = HIWORD(wParam);
    // Parse the menu selections:
    switch (wmId)
    {
        case IDM_ABOUT:
            DialogBox(hInst, (LPCTSTR)IDD_ABOUTBOX, hWnd, (DLGPROC)About);
            break;
        case IDM_EXIT:
            DestroyWindow(hWnd);
            break;
        //Parse Edit control's notification
        case ID_OUTBOX:
            switch (HIWORD(wParam)) {
                case EN_CHANGE:
                    // Text has changed! Put OutWnd's text in InWnd
                    char str[128];
                    GetWindowText(hOutWnd, str, 128);
                    SetWindowText(hInWnd, str);
            }
            ...
    }
}
```

Echoing to the Other Control

- `GetWindowText(hOutWnd, str, 128);`
 - copies the text of the specified window's title bar (if it has one) into a buffer.
 - If the specified window is a control, the text of the control is copied.
- `SetWindowText(hInWnd, str);`
 - changes the text of the specified window's title bar (if it has one).
 - If the specified window is a control, the text of the control is changed.

An Example: Echoing Edit Controls 1/3

```
LRESULT CALLBACK WndProc(HWND hWnd, UINT message, WPARAM wParam, LPARAM lParam)
{
    int wmId, wmEvent;
    PAINTSTRUCT ps;
    HDC hdc;
    TCHAR szHello[MAX_LOADSTRING];
    LoadString(hInst, IDS_HELLO, szHello, MAX_LOADSTRING);

    switch (message)
    {
        case WM_CREATE:
            // Create Edit control for typing to be sent
            hOutWnd = CreateWindow("EDIT",
                NULL,
                WS_BORDER | WS_CHILD | WS_VISIBLE | WS_VSCROLL | ES_LEFT |
                ES_MULTILINE | ES_AUTOVSCROLL,
                0,0,0,
                hWnd,
                (HMENU) ID_OUTBOX,
                (HINSTANCE) GetWindowLong(hWnd, GWL_HINSTANCE),
                0);

            // Create Edit control for receiving
            hInWnd = CreateWindow("EDIT",
                NULL,
                WS_BORDER | WS_CHILD | WS_VISIBLE | WS_VSCROLL | ES_LEFT |
                ES_MULTILINE | ES_AUTOVSCROLL | ES_READONLY,
                0,0,0,
                hWnd,
                (HMENU) ID_INBOX,
                (HINSTANCE) GetWindowLong(hWnd, GWL_HINSTANCE),
                0);

            break;
    }
}
```

An Example: Echoing Edit Controls 2/3

```
case WM_SETFOCUS :
    SetFocus (hOutWnd);
break;

case WM_SIZE :
    // Size OutBox Edit Control
    MoveWindow(hOutWnd,
        1, // Upper Left Corner
        LOWORD(Param) - 2, // Width of Parent Window
        (HIWORD(Param) / 2) - 2, // Half the height of Parent
        TRUE); // repaint

    // Size InBox Edit Control
    MoveWindow(hInWnd,
        1, (HIWORD(Param) / 2) + 1, // Half Way down right side
        LOWORD(Param) - 2, // Width of Parent Window
        (HIWORD(Param) / 2) - 2, // Half the height of Parent
        TRUE); // repaint
break;

case WM_COMMAND:
    wParam = LOWORD(wParam);
    lParam = HIWORD(wParam);
    // Parse the menu selections:
    switch (wParam)
    {
        case IDM_ABOUT:
            DialogBox(hInst, (LPCSTR)IDD_ABOUTBOX, hWnd, (DLGPROC)About);
            break;
        case IDM_EXIT:
            DestroyWindow(hWnd);
            break;
        //Parse edit control's notification
        case ID_OUTBOX:
            switch (HIWORD(wParam)) {
                case EN_CHANGE:
                    // Text has changed! Put OutWnd's text in InWnd
                    char str[128];
                    GetWindowText(hOutWnd, str, 128);
                    SetWindowText(hInWnd, str);
            }
        default:
            return DefWindowProc(hWnd, message, wParam, lParam);
    }
break;
```

2006-10-25

Edit Control

11

An Example: Echoing Edit Controls 3/3

```
case WM_PAINT:
    hdc = BeginPaint(hWnd, &ps);
    // TODO: Add any drawing code here...
    RECT rt;
    GetClientRect(hWnd, &rt);
    DrawText(hdc, szHello, strlen(szHello), &rt, DT_CENTER);
    EndPaint(hWnd, &ps);
    break;

case WM_DESTROY:
    PostQuitMessage(0);
    break;

default:
    return DefWindowProc(hWnd, message, wParam, lParam);
}
return 0;
}
```

2006-10-25

Edit Control

12