Custom edit control win32

Asked 13 years, 2 months ago Modified 4 years, 9 months ago Viewed 7k times

8

I finally managed to get my syntax highlighting done using richedit and iczelion's tutorials. Now that i find it, it certainly is not fast enough. I am thinking of taking this one step ahead: a custom edit control. But i do not know how to go about it. Could you guys tell me how to go about it? Give me some info to start on? Maybe even some tutorial or suggest some book?

Now I'm not asking for you guys to spell it out for me, just something to start on. I will be using C++/ASM/Win32 API for this. I'm sure many of you have already made custom edit controls befores, so may be you could even share your experience.

Thanks,

Devjeet

c++ winapi edit

Share Edit Follow Flag

asked Nov 10, 2011 at 1:42

devjeetroy
1,945 6 29 43

Visual Studio Samples has a WordPad and SuperPad sample – parapura rajkumar Nov 10, 2011 at 2:31

Thanks, but actually im not looking for MFC tutorials. – devjeetroy Nov 10, 2011 at 2:51

4 A proper custom edit control is decidedly nontrivial, once you take non-English languages into account. I suggest that you reconsider. – Raymond Chen Nov 10, 2011 at 3:59

Thanks for your response. I would like to tell you that this is just kind of a side project for me and i took it up for educational purposes(kinda like learning assmebly, not to say that assembly has lost its usefullness). So I will not consider non-english languages to begin with. So keep that in mind. But yeah, even I'm concerned about the size. By big, how many lines do you mean? I haven't worked on a project that's more than 700 lines, so to speak – devjeetroy Nov 10, 2011 at 4:19

It could also be your code that is parsing the text on the fly in the edit control, which I assume is then outputting the RTF to the control. For example, if you are parsing the entire text file on every key stroke as the user edits the source, then output all of the RTF to the control, it will be slow for larger files. – franji1 Nov 10, 2011 at 4:48

1 Answer

Sorted by: Highest score (default)

12

I spent one day on coding my own custom edit control - it's working well, so I would like to share my experience here, maybe for someone this code might be helpful... Because custom draw an common edit control is impossible (see here), you must write your own edit control. The general steps are:

```
// global vars
int select; // current selection position
int cursor; // current cursor position
HWND parent; // parent window
wchar_t buf[MAXINPUTBUF]; // edit buffer
WNDPROC oldproc; // old window procedure

// create custom control window
hWnd = CreateWindowW(L"static", NULL, WS_CHILD | WS_TABSTOP | SS_LEFT |
SS_NOTIFY, 0, 0, 0, 0, parent, NULL, (HINSTANCE)GetWindowLongPtr(parent,
GWL_HINSTANCE), NULL);

// todo: use SetProp() to store all global vars
oldproc = (WNDPROC)SetWindowLongPtrW(hWnd, GWL_WNDPROC, (LONG_PTR)InputWndProc);

SetWindowPos(hWnd, HWND_TOP, x, y, cx, cy, 0);
```

How to display keyboard input is described here. My window procedure looks as follows

```
LRESULT CALLBACK InputWndProc(HWND hWnd, UINT msg, WPARAM wParam, LPARAM lParam)
{

    switch (msg)
    {
    case WM_LBUTTONDOWN:
        //SetFocus(hWnd);
        PostMessageW(GetParent(hWnd), WM_NEXTDLGCTL, (WPARAM)hWnd, TRUE);
        break;

    case WM_KILLFOCUS:
        HideCaret(hWnd);
        DestroyCaret();
        break;

    case WM_SETFOCUS:
        {
            RECT r;
            GetClientRect(hWnd, &r);
            // Create a solid black caret.
            CreateCaret(hWnd, (HBITMAP) NULL, 2, r.bottom-r.top);

            ShowCaret(hWnd);
            InputWndRedraw(hWnd);

        }

        return FALSE;

    case WM_GETDLGCODE:
        return DLGC_WANTALLKEYS | DLGC_WANTARROWS;

    case WM_KEYDOWN:
    {
        switch (wParam)
        {
        case 'V':
```

```
                if (0x8000 & GetKeyState(VK_CONTROL))
                {
                    HANDLE h;
                    wchar_t *cb;
                    int len,slen;

                    InputWndDelete(hWnd);

                    OpenClipboard(NULL);
                    h = GetClipboardData(CF_UNICODETEXT);

                    cb = (wchar_t*)GlobalLock(h);

                    if (cb)
                    {
                        memcpy(buf+(cursor+len)*sizeof(wchar_t),
buf+cursor*sizeof(wchar_t), (slen-cursor)*sizeof(wchar_t));
                        memcpy(buf+cursor*sizeof(wchar_t), cb, len*sizeof(wchar_t));
                    }

                    GlobalUnlock(h);
                    CloseClipboard();
                    InputWndRedraw(hWnd);
                }
                break;

            case VK_RIGHT:

                    if (cursor-1 >= MAXINPUTBUF || cursor >= (int)wcslen(buf))
                        break;

                    cursor++;

                    if (!(GetKeyState(VK_SHIFT) & 0x8000))
                        select = cursor;

                    InputWndRedraw(hWnd);
                    break;

                case VK_TAB:
                    PostMessageW(GetParent(hWnd), WM_NEXTDLGCTL,
GetKeyState(VK_SHIFT) & 0x8000, FALSE);
                    break;

                case VK_LEFT:
                    if (cursor <= 0)
                        break;

                    cursor--;

                    if (!(GetKeyState(VK_SHIFT) & 0x8000))
                        select = cursor;

                    InputWndRedraw(hWnd);
                    break;

                case VK_HOME:
                    cursor = 0;

                    if (!(GetKeyState(VK_SHIFT) & 0x8000))
                        select = cursor;

                    InputWndRedraw(hWnd);
                    break;

                case VK_END:
```

```
                    cursor = wcslen(buf);

                    if (!(GetKeyState(VK_SHIFT) & 0x8000))
                        select = cursor;

                    InputWndRedraw(hWnd);
                    break;

                case VK_DELETE:
                    if (cursor >= (int)wcslen(buf))
                    {
                        InputWndDelete(hWnd);
                        InputWndRedraw(hWnd);
                        break;
                    }

                    if (select == cursor)
                        select ++;

                    InputWndDelete(hWnd);
                    InputWndRedraw(hWnd);
                break;

                case VK_BACK:

                    if (cursor <= 0)
                    {
                        InputWndDelete(hWnd);
                        InputWndRedraw(hWnd);
                        break;
                    }


                    if (select == cursor)
                        cursor --;


                    InputWndDelete(hWnd);
                    InputWndRedraw(hWnd);
                }

            }
            break;

        case WM_CHAR:
            if (wParam < VK_SPACE)
            break;


            InputWndDelete(hWnd);

            if (wcslen(buf)+1 < MAXINPUTBUF)
            {
                wmemmove(buf+(cursor+1)*sizeof(wchar_t), buf+cursor*sizeof(wchar_t),
    wcslen(s->buf)-cursor);
                buf[cursor] = wParam;
                cursor++;
                select = cursor;
            }

            InputWndRedraw(hWnd);

            break;

        case WM_ERASEBKGND:
            // no flickering
```

```
            return TRUE;

        case WM_PAINT:
            {
                HDC dc;
                PAINTSTRUCT paint;

                dc = BeginPaint(hWnd, &paint);
                InputWndDraw(hWnd, dc);
                EndPaint(hWnd, &paint);

            }
            return TRUE;

        }

        return CallWindowProcW(oldproc, hWnd, msg, wParam, lParam);
    }
```

Delete current selected text (from *select* to *cursor*).

```
    void InputWndDelete(HWND hWnd)
    {
        int len;

        len = wcslen(buf);

        if (select > cursor)
        {
            memcpy(buf+cursor*sizeof(wchar_t), buf+select*sizeof(wchar_t), (len -
    select)*sizeof(wchar_t));
            ZeroMemory(buf+(len-select+cursor)*sizeof(wchar_t), (MAXINPUTBUF-
    len+select-cursor)*sizeof(wchar_t));
            select = cursor;
        }
        else if (select < cursor)
        {
            memcpy(buf+select*sizeof(wchar_t), buf+cursor*sizeof(wchar_t), (len -
    cursor)*sizeof(wchar_t));
            ZeroMemory(buf+(len-cursor+select)*sizeof(wchar_t), (MAXINPUTBUF-
    len+cursor-select)*sizeof(wchar_t));
            cursor = select;
        }
        else
        {
            select = cursor;
        }
    }
```

Draw window on window DC

```
    void InputWndRedraw(HWND hWnd)
    {
        HDC hdc;

        HideCaret(hWnd);

        hdc = GetDC(hWnd);
        InputWndDraw(hWnd, hdc);
        ReleaseDC(hWnd, hdc);

        ShowCaret(hWnd);
```

```
}
```

Draw input buffer (buf*) on device context. Syntax highlighting and other formatting features goes here...

```
void InputWndDraw(HWND hWnd, HDC hdc)
{
    RECT r,cr;

    GetClientRect(hWnd, &cr);

    // draw selected rectangle FillRect()...

    CopyRect(&r,&cr);
    DrawTextW(hdc, buf, -1, &r, DT_LEFT | DT_TOP);


    if (cursor)
        DrawTextW(hdc, buf, cursor, &r, DT_LEFT | DT_TOP | DT_CALCRECT);
    else
        r.right = cr.left;

    if (GetFocus() == hWnd)
    {
        if (r.right > cr.right)
            SetCaretPos(cr.right, cr.top);
        else
            SetCaretPos(r.right, cr.top);
    }
}
```

Share Edit Follow Flag

edited Apr 9, 2019 at 18:08

answered Jan 18, 2015 at 15:19

vadim_hr

549 5 12

Great post! There are several approaches to customizing controls according to Microsoft. You are using traditional subclassing and that page also provides a new approach by SetWindowSubclass. – Steven Liang Apr 8, 2019 at 6:45