

Re: How would you do it

Source: <http://www.tech-archive.net/Archive/VC/microsoft.public.vc.mfc/2004-12/1741.html>

From: Joseph M. Newcomer (*newcomer_at_flounder.com*)

Date: 12/26/04

Date: Sun, 26 Dec 2004 17:35:09 -0500

It takes perhaps 20 minutes to create a trivial CWnd-derived typein control. My inclination in this case would be to create as many as needed, keeping a display list of these around. As long as you only want a single font and size in the window, it is real easy (adding the font handler takes another fifteen minutes)

```
class CMyAnnotation : public CWnd {
public:
    void Create(CPoint pt);
    void Draw();
protected:
    CString text;
    CFont font; // optional
    BOOL drawBorder; // TRUE to draw edit/drag border
}

typedef CList<CMyAnnotations *, CMyAnnotations *> Annotations;
```

Create the window large enough to hold a border drawing. When you create the window, it is at point pt, of height $h = tm.tmHeight + tm.tmInternalLeading + \text{borderheight}$, and width $w = 2 * \text{bordheight} + \text{empty}$. Then, as it gets each WM_CHAR message, it gets the character width of the character, adds that to the width of the window, plops the character down at the current x,y position, appends it to the string text, does an InvalidateRect on the parent of the current annotation window, for the rectangle which is the current editing window, followed an UpdateWindow (this forces the bitmap to redraw). Then Invalidate() the current window. The Draw() method will redraw the text, and, if in input mode, the border. When <enter> is hit to end the input (simple assumption: one line), clear the border flag and invalidate the parent window for the input rectangle of the text window, then invalidate the text window. This redraws the annotation without the border.

If you click the mouse down, the WM_LBUTTONDOWN event enters the border mode, sets the border-draw flag, and does an Invalidate() of the annotaiton window. Now it appears with its border. A subsequent WM_LBUTTONDOWN does nothing, but a drag with the button down will chngae the x,y coordinates of the window (hint: figure out dx,dy of the mouse from the nominal 0,0 of the text, which is NOT the 0,0 of the windw, since we allow for the border width, and then subtract this dx,dy from the current mouse position during the drag. Don't forget to invalidate the parent window at the former location before drawing in the new location.

microsoft.public.vc.mfc: Re: How would you do it

Alternatively, you can use XOR/XORNOT or something like that to draw the border, so each time you draw the border twice it disappears.

This would make a good 2-hour lab in the introductory MFC course I used to teach.

Ability to set the caret and edit the text is a more serious exercise, as is text highlighting. That's good for a couple hours alone. If you want to be lazy, right-click to a menu with edit and pop up a dialog with the text; upon completion, take the edited text from the dialog and replace it. Not as slick, but five to ten minutes' work.

joe

On Wed, 22 Dec 2004 09:14:54 -0700, "Jonathan Wood" <jwood@softcircuits.com> wrote:

>*First of all, I would not use CStatic control's. There's no reason for all
>those windows.*
>
>*Simply create a data structure for each item. For starters, each item will
>have a RECT to indicate it's location. If it contains text, then it should
>also contain a string. Then when you need to paint your window, just loop
>through and draw each item.*
>
>*Painting to a bitmap would be similar task: Just draw to a bitmap instead of
>the screen DC.*
>
>*I did this in a very simple program. It can be download from
><ftp://ftp.softcircuits.com/rodent/easydtp.zip> (sorry, no source code).*

Joseph M. Newcomer [MVP]
email: newcomer@flounder.com
Web: <http://www.flounder.com>
MVP Tips: http://www.flounder.com/mvp_tips.htm