Why doesn't my MessageBox wrap at the right location?

devblogs.microsoft.com/oldnewthing/20110624-00

June 24, 2011

Raymond Chen

A customer reported that the `MessageBox` function was wrapping text “at the wrong location.”

> Our program displays a message with the `MessageBox` function, and we use the ‘\n’ character to force line breaks at positions we choose. But we’ve found that starting in Windows Vista, the line breaks we placed are not working. The `MessageBox` function is inserting its own line breaks, which interferes with our custom text layout. It used to be that the width of the message box would expand to fit the longest line in the message.

The `MessageBox` function is one of those “leave the driving to us” type of functions. You give it a string to display, select which buttons you want, and sit back and relax while the `MessageBox` function does the work. The trade-off for the simplicity is that you also lose control over the experience. The `MessageBox` function decides where to place the dialog and how big to make it, which in turn determines where the line breaks go. The algorithm used by `MessageBox` to determine the size of the message box has changed many times over the lifetime of Windows. In Windows 3.1, it was the width of the longest line, or 5/8 of the width of the screen, whichever was smaller. Windows 95 chose the smallest of the following which resulted in a dialog box that fit inside the working area:

- the width of the longest line,
- 5/8 of the width of the working area,
- 3/4 of the width of the working area,
- 7/8 of the width of the working area,

Notice that even in Windows XP, the dialog box was not guaranteed to be the width of the longest line. If the longest line was more than 5/8 of the width of the working area, the `MessageBox` is probably going to insert its own line breaks beyond ones you inserted explicitly. Windows Vista changed the algorithm again, in recognition of two things. First, monitors are now much larger than they were in 1995. And second, the consequence of these larger monitors is that the 7/8 rule resulted in message boxes that were unreadable because they were 10 inches wide and half an inch tall. The old algorithm did not age well, but then again, it was written back in the days when the really cool kids had 1024×768 screens.

Nowadays, even the kids from the wrong side of the tracks have screens that are regularly 1400 or even 1600 pixels wide. The new algorithm merely adds another option to the table of choices:

- the width of the longest line,
- 278 DLU,
- 5/8 of the width of the working area,
- 3/4 of the width of the working area,
- 7/8 of the width of the working area,

Note that the details of the `MessageBox` line breaking algorithm are provided for historical purposes only. Do not write code which relies on them, because who knows, they may change again in the next version of Windows.

The algorithm was changed at the request of the visual design team; after all, it's the job of the visual design team to make these sorts of visual design decisions. If you don't want some black-turtleneck-wearing designer in Redmond to decide where your line breaks go, then don't use `MessageBox`. And you don't have to go and convert every single message box to its own dialog template; just write a function called `MessageBoxWithLineBreaksExactlyWhereIPutThem` function that takes the same parameters as `MessageBox` but lays out the text exactly the way you want it. (The `DT_CALCRECT` flag will be useful here.)

Raymond Chen

Follow