Why doesn’t GetTextExtentPoint return the correct extent for strings containing tabs?

devblogs.microsoft.com/oldnewthing/20181012-00

October 12, 2018

Raymond Chen

A customer reported that the `GetTextExtentPoint` and `GetTextExtentPoint32` functions do not return the correct extents for strings that contain tabs. The documentation does say that they do not support carriage return and linefeed, but nothing about tabs.

The `TextOut` and `GetTextExtentPoint` functions do not interpret control characters. They take the string you pass, convert the code points to glyphs, string the glyphs together, and display or measure the result.

They don’t move the virtual carriage to the “left margin” when they encounter a U+000D CARRIAGE RETURN, or move it down by the “line height” when they encounter a U+000A LINE FEED, or forward to the next “tab stop”, when they encounter a U+0009 CHARACTER TABULATION, or to the left by “some distance” when they encounter a U+0008 BACKSPACE,[1] or clear the “screen” when they encounter a U+000C FORM FEED, or change the “typewriter ribbon color” when they encounter U+000E SHIFT IN and U+000F SHIFT OUT, or beep the speaker when they encounter a U+0007 BELL.

At best, you’ll get the graphics for the various control characters, like ␉ for the horizontal tab, but more likely you’ll get ugly black boxes.

If you want to render text with tabs, use `TabbedTextOut`. If you want to measure text with tabs, use `GetTabbedTextExtent`. The `DrawText` function can both render and measure, and it also supports carriage returns and line feeds.

Still no luck with backspace, changing the typewriter ribbon color, clearing the screen, or beeping the speaker, though. For those you’re on your own.

[1] What would that even mean if you backspaced beyond the start of the string? Does this mean you could have a string whose extent is negative?

Raymond Chen

Follow