# Why does the version 6 animation control not use a background thread?

devblogs.microsoft.com/oldnewthing/20060316-12

March 16, 2006

Raymond Chen

Many people have noticed that the animation bar control in version 6 of the common controls no longer uses a background thread to draw the animation. Instead, it acts as if the `ACS_TIMER` style is always set, even if the caller didn't pass it. Why is that? The first reason is that the background thread didn't actually help any. In order to draw transparent animations, the painting loop needs to query the animation control's parent to obtain the background color, and that query entails sending a message to the main UI thread. Consequently, the main UI thread must remain responsive to messages in order for the animation to render properly. If the main UI thread stopped responding to messages, the `WM_CTLCOLORSTATIC` message sent to obtain the background color would hang, and painting would wedge. Therefore, the UI thread had to remain responsive to messages. And if that's the case, then there's no need for the background thread after all. The UI thread can just run a timer and draw the frames in response to the timer message. The second reason for getting rid of the background thread is that it actually made things worse. Another thread means more context switches and more memory pressure, since the additional thread needs a stack and other supporting data structures. Admittedly, this is a comparatively weak reason. The third reason is that using a background thread for painting simply encouraged bad code. Using a background thread for painting meant that the UI thread could stop responding to messages for long periods of time and usually get away with it because the query for the background color comes early in the animation cycle when the main UI thread most likely has not yet gotten into the part of the long-running procedure that stops responding to messages. As a result, the background thread encouraged programs to stop pumping messages on UI threads because "it seems to work fine". The result is programs that fail to maintain a responsive UI, resulting in periodic mysterious hangs in the window manager when another program tries to broadcast a message and gets wedged up behind the unresponsive window. This leads to increased frustration for the end user and a general feeling that "Windows sucks".

By making the suckage obvious, programmers will be more likely to notice that they are doing something bad and do something to address it. Masking the problem with a background animation thread merely allows the problem to persist.

Raymond Chen

**Follow**