Over-documenting TTM_RELAYEVENT and why it results in a one-second periodic timer running as long as the tooltip is visible

devblogs.microsoft.com/oldnewthing/20200110-00

January 10, 2020

Raymond Chen

If you create a Windows classic Win32 tooltip control, you get to specify whether you want the control to subclass the window with which is associated, or whether you promise to forward mouse messages with the `TTM_RELAYEVENT` message. The tooltip control uses these mouse messages to help decide when the tooltip should be displayed and hidden.

If you agree to forward the messages yourself, then you may find that the tooltip control runs a one-second periodic timer for as long as the tooltip is visible. What's that for?

The timer is there so the tooltip can detect when the mouse has left the associated window. When that happens, the tooltip hides itself because the mouse is no longer on the tooltip target.

Why does it need a timer to do this?

Rewind to 1994. The tooltip control is being developed, and the `TTM_RELAYEVENT` message gives the tooltip control insight into what the mouse is doing in the associated window. It uses this information to detect that the mouse has dwelled inside the tooltip target for the required amount of time, which causes the tooltip to appear. It also uses this information to detect that the mouse has moved to another part of the associated window that is not part of the tooltip target, at which point it can remove the tooltip.

But there's another case that isn't covered by this: The tooltip needs to know when the mouse has left the tooltip target due to the mouse leaving the associated window entirely.

Since mouse messages are delivered to the window under the mouse cursor,[1] moving the mouse out of the associated window entirely means that the associated window has nothing to forward to the control via the `TTM_RELAYEVENT` message The only way for the tooltip to know that the mouse has left the window entirely is for it to run a timer and poll the mouse position.

That's how things were in Windows 95.

The documentation for the `TTM_RELAYEVENT` message says

> A tooltip control processes only the following messages passed to it by the **TTM_RELAY-EVENT** message:
>
> - WM_LBUTTONDOWN
> - WM_LBUTTONUP
> - WM_MBUTTONDOWN
> - WM_MBUTTONUP
> - WM_MOUSEMOVE
> - WM_RBUTTONDOWN
> - WM_RBUTTONUP
>
> All other messages are ignored.

Move forward to 1998. The `TrackMouseEvent` function was added. Among other things, this allows a window to be notified when the mouse leaves the window outright. Great, the tooltip control can take advantage of this so that it doesn't need to poll the mouse to find out whether it left the window. It can just wait for the `WM_MOUSELEAVE` message.

Except that it can't.

Because the `TTM_RELAYEVENT` message already had documentation that said "All other messages are ignored." Programs were written based on the fact that only the messages given in the documentation need to be forwarded to the tooltip control. If they got any other message, they "optimized" their code by not bothering to forward it to the tooltip control.

This meant that the tooltip control would never get the `WM_MOUSELEAVE` message, since the documentation told people that the tooltip control ignored the message.

So despite the availability of an efficient and battery-friendly way of detecting whether the mouse has left a window, the tooltip control cannot use it because the documentation revealed too much information, and people came to rely on that extra information.

If the documentation had merely said, "Forward all messages between `WM_MOUSEFIRST` and `WM_MOUSELAST` to the tooltip control," without enumerating which mouse messages the tooltip control actually cares about, then it would have been possible to use the efficient version, because everybody would be forwarding all mouse messages, which includes the new `WM_MOUSELEAVE` message.

So things are bad because we wrote too much documentation. The documentation described the implementation rather than the contract.

All is not lost, however.

If you set the `TTF_SUBCLASS` flag when you create a tooltip target, then you are telling the tooltip control to subclass the window in order to grab the mouse messages. In this case, you don't need to (and shouldn't) use the `TTM_RELAYEVENT` message. And if the tooltip control is subclassing the window, it can see *all* the messages, and that includes the `WM_MOUSELEAVE` message.

So use the `TTF_SUBCLASS` flag when you create your tooltip targets. Your tooltip will respond more promptly to the user moving out of the window, and you won't burn up the user's battery.

[1] Assuming that mouse capture is not in effect. Tooltips do not capture the mouse, because that would prevent the user from using the mouse to do normal mouse things.

Raymond Chen

Follow