How do I create a topmost window that is never covered by other topmost windows?

devblogs.microsoft.com/oldnewthing/20110310-00

March 10, 2011

Raymond Chen

We already know that you can't create a window that is always on top, even in the presence of other windows marked always-on-top. An application of the *What if two programs did this?* rule demonstrates that it's not possible, because whatever trick you use to be on-top-of-always-on-top, another program can use the same trick, and now you have two on-top-of-always-on-top windows, and what happens? A customer who failed to understand this principle asked for a way to establish their window as "super-awesome topmost". They even discovered the answer to the "and what happens?" rhetorical question posed above.

> We have overridden the OnLostFocus and OnPaint methods to re-assert the TopLevel and TopMost window properties, as well as calling BringToFront and Activate. The result is that our application and other applications end up fighting back and forth because both applications are applying similar logic. We tried installing a global hook and swallowing paint and focus events for all applications aside from our own (thereby preventing the other applications from having the opportunity to take TopMost ahead of us), but we found that this causes the other applications to crash. We're thinking of setting a timer and re-asserting TopMost when the timer fires. Is there a better way?

This is like saying, "Sometimes I'm in a hurry, and I want to make sure I am the next person to get served at the deli counter. To do this, I find whoever has the lowest number, knock them unconscious, and steal their ticket. But sometimes somebody else comes in who's also in a hurry. That person knocks *me* unconscious and steals *my* ticket. My plan is to set my watch alarm to wake me up periodically, and each time it wakes me up, I find the person with the lowest number, knock them unconscious, and steal their ticket. Is there a better way?" The better way is to *stop knocking each other unconscious and stealing each other's tickets*. The customer (via the customer liaison) provided context for their question.

> This is not a general-purpose application. This application will be run on dedicated machines which operate giant monitors in retail stores. There are already other applications running on the computer which rotate through advertisements and other display information.
>
> The customer is writing another application which will also run on the machine. Most of the time, the application does nothing, but every so often, their application needs to come to the front and display its message, regardless of whatever the other applications are displaying. (For example, there might be a limited-time in-store promotion that needs to appear on top of the regular advertisements.)
>
> Unfortunately, all of these different programs were written by different vendors, and there is no coordination among them for who gets control of the screen. We were hoping that there was some way we could mark our window as "super topmost" so that when it came into conflict with another application running on the machine, it would win and the other application would lose.
>
> I'm thinking of recommending that the vendors all come up with some way of coordinating access to the screen so they can negotiate among themselves and not get into focus fights. (Easier said than done, since all the different applications running on the machine come from different vendors…)

Since there is no coordination among the various applications, you're basically stuck playing a game of walls and ladders, hoping that your ladder is taller than everybody else's wall. The customer has pretty much found the tallest ladder which the window manager provides. There is no "super topmost" flag. Sure, you can try moving to another level of the system, like say creating a custom desktop, but all that does is give you a taller ladder. And then one of the other applications is going to say, "I need to display a store-wide page (manager to the deli please, manager to the deli), overriding all other messages, even if it's a limited-time in-store promotion." And they'll try something nastier, like enumerating all the windows in the system and calling `ShowWindow(SW_HIDE)`. And then another application will say, "I need to display an important store-wide security announcement (Will the owner of a white Honda Civic, license plate 037-MSN, please return to your vehicle), overriding all other messages, even if it's a store-wide page." And it'll try something nastier, like setting their program as the screen saver, disabling the mouse and keyboard devices, and then invoking the screen saver on the secure desktop. And then another application will say, "I need to display a critical store-wide announcement (Fire in the automotive department. Everybody evacuate the building immediately), overriding all other messages, even if it's an important store-wide security announcement." And it'll try something nastier, like enumerating all the processes on the system, attaching to each one with debug privilege, and suspending all the threads. Stop the madness. The only sane way out is to have the programs coöperate to determine who is in control of the screen at any particular time.

In response to my hypothetical game of walls and ladders, one of my colleagues wrote, “Note to self: Do not get into a walls-and-ladders contest with Raymond.”

Raymond Chen

Follow