Popup Help Example

by Light at the End of the Tunnel Software

About Us

Howdy! **Light at the End of the Tunnel** is devoted(?) to producing examples of Visual Basic code for applications that you may have seen and always wondered "How'd they do that?"

In an effort to distribute useful (and needed) information, i've produced a few applications (source included) that we hope will enlighten the unenlightened, aid the depraved, right the wronged, and promote goodwill among insects. Hopefully, we"ve provided enough documentation about the program that you will be able to understand what's going on and use the information. However, if after a reasonable about of time you still don't get it, feel free to e-mail me with your questions. And, if you find it useful or learn anything from it, drop me an e-mail and let me know. Your response could spur the next production. And, if you **really** like it, and reside in Seattle, and want to hire a PC programmer who's looking to relocate...

Enjoy.

tim koffley
(miami, fl)
[CIS: 70334,16]
[AOL: Tim Servo]

About the Program

This program demonstrates **one** way to simulate those neat little popup help messages for the toolbars in all the latest version of Microsoft programs. If you don't know what i'm talking about i'll briefly describe the concept.

In all of the latest MS products, if you pause the mouse over a toolbar button for a longer period of time than you should if you knew what you wanted, a little message pops up with a text explanation of the button you're hovering over. It's great because with all the ways you can now customize toolbars you can quickly forget what picture stands for what function.

Anyway, i thought it was cool and wanted it for my own so i set about figuring out a way to do it with VB and **without** third-party controls. What resulted is this demo.

Basically, i figured the easiest way in VB to simulate a little window popping up was to just use a "mini" form that has no borders, buttons, etc., and is just a holder for the help message box. That way any window accounting is done by Windows; you don't have to create a window from scratch with Windows API calls.

This approach seemed to be fine except for one thing; whenever you display a form with the VB Show command, VB also tries to activate the form thus causing some unwanted side-effects. The way around it is to use a Windows API call to show the form with a flag to not activate it. Once it's displayed we can still use the VB Hide method to hide it.

The only other "trick" to getting it to work is synchronizing mouse motions to the help popup system. After some experimenting, it boiled down to using two timer controls. One to countdown while you're hovering over a button and one to track mouse movement once the help system has been activated. If you want to change the help delay, just play with the Interval property on Timer1. The thing that sets the countdown timer off is a MouseMove event for any of the toolbar buttons.

In words, here's the algorithm:

1) Mouse moves over button, start/reset countdown (Timer1)
2) Timer1 goes off. Are we still over the button that triggered the timer?
   If so, then activate help system and start tracking mouse movements with Timer2.
3) If we keep the mouse in the toolbar area then help remains active. Otherwise,
   cancel help system and Timer2.

There's only one minor instance i know of where the system doesn't behave like the MS toolbar help: if you sit on a button (which triggers the countdown) then move off the button into the toolbar area/background then back onto the button before the timer expires then the help system still fires. Well i say....SO WHAT!

Hi-Keeba!

References

Visual Basic Programmer's Guide to the Windows API: Daniel Appleman, Ziff-Davis Press
If you are doing anything with Windows API in VB you should own this book. If you aren't doing anything with API in VB, you're missing out on a lot of power.