

Re: Difficulties in deriving a truly universal GUI Scripting Language

Source: <http://www.tech-archive.net/Archive/VC/microsoft.public.vc.mfc/2006-08/msg01993.html>

- *From:* Joseph M. Newcomer <newcomer@xxxxxxxxxxxxx>
 - *Date:* Tue, 15 Aug 2006 19:13:29 -0400
-

But note that the caret is a graphical artifact with no semantic content. What the caret means is whatever the control that is displaying it chooses to have it mean. I see possible risks here.

Now, the next issue is what is meant by "re-examining the screen". Do you specify in the scripting language "I want this caret before the 'i' in 'is'" and let it do whatever it has to do to achieve that (and I can think of a lot of ways to get the wrong result here) or does the scripting language have to contain a loop by which the scripter writes out what needs to be done (same problems of getting the wrong answer, but more work).

Automated "black box" testing harnesses were challenging even in the days of text-only scripting, where we had to look for responses from the compiler and such. An unexpected response could ruin a whole night's testing run because the script had no idea how to handle it. Ultimately it all depends on how much the script has to be "perfect" and how much it only matters that *something* happened.
joe

On Tue, 15 Aug 2006 16:12:08 -0500, "Peter Olcott" <olcott@xxxxxxx> wrote:

"Joseph M. Newcomer" <newcomer@xxxxxxxxxxxxx> wrote in message news:vgu3e2135n02nmv12m5776cmcbmqfg2jn6@xxxxxxxxxxx

Just another set of glitches I thought of, in rereading that article.
Consider an edit control with horizontal scrolling. You click into it, between a couple characters, expecting that's where the caret is going to end up. For example

[This is a t]est

where the est is off the end of the control. You want to change 'is' to 'was', so you click between the i and the s. You might find something like

Thi[s is a |test]

Re: Difficulties in deriving a truly universal GUI Scripting Language

where the | indicates the caret position. This is because horizontal edit controls tend of spontaneously scroll underneath you. I'd suggest the problem of recovering from this is going to be at least one problem in writing a script.

You would merely have to re-examine the screen display after every input action whenever you encounter a control with this behavior. You might also need to have a way to determine the current location of the caret, if the screen caret was capable of adjusting its precise position from where it was placed. There are API calls to determine the precise location of the mouse cursor and screen caret, aren't there?

Another is the fact that you might have to search for the phrase "This is a test", but it doesn't actually appear on the screen. But it might. Or not. For example, if the user has selected "large fonts", you will have a potentially different screen display than for the default fonts. Now, in the case of an edit control, GetWindowText will suffice, but consider a situation where a listbox or combobox has the text scrolled off the edge of the screen and is owner-draw. A human would see ambiguous strings and know to scroll, but how would a scripting language? What if the programmer actually failed to provide a horizontal scrollbar in the listbox?

The scripting language might not know to scroll, but the author of the script might know to tell the script to look for scroll-bars, and to determine the attributes of the screen display fonts. Automated testing could change things such as the screen font, and see if the correct results are consistently produced, reporting any errors in the end of its run.

joe

On Tue, 15 Aug 2006 11:28:21 -0500, "Peter Olcott" <olcott@xxxxxxx> wrote:

http://groups.google.com/group/microsoft.public.vc.mfc/browse_thread/thread/32f6ea916eb79

The above post has some really good insights into this

Re: Difficulties in deriving a truly universal GUI Scripting Language

problem. I want to delve a little deeper into this question, and see if any of the prior insights have changed. It looks like one of the most difficult aspects of this problem is providing a way so that the Mouse can always "see" where to click.

Joseph M. Newcomer [MVP]
email: newcomer@xxxxxxxxxxxxx
Web: <http://www.flounder.com>
MVP Tips: http://www.flounder.com/mvp_tips.htm

Joseph M. Newcomer [MVP]
email: newcomer@xxxxxxxxxxxxx
Web: <http://www.flounder.com>
MVP Tips: http://www.flounder.com/mvp_tips.htm

.