

Re: Iconized application takes looooong to wake up

# Re: Iconized application takes looooong to wake up

---

*Source:* <http://coding.derkeiler.com/Archive/Tcl/comp.lang.tcl/2007-06/msg01162.html>

---

- *From:* Melissa Schrupf <[m\\_schrumpf\\_at\\_yahoo\\_com\\_NOT@xxxxxxxxxxxxxx](mailto:m_schrumpf_at_yahoo_com_NOT@xxxxxxxxxxxxxx)>
  - *Date:* Tue, 26 Jun 2007 19:44:12 -0400
- 

Donal K. Fellows wrote:

Melissa Schrupf wrote:

Anecdotally, I find application restore times to be a more common affliction of Windows.

I suspect an unfortunate interaction between the scheduler and the way Tk manages resources (which is non-standard by comparison with most Win applications). But I don't understand the details well enough to be able to point to anything useful.

Donal.

Okay, semi-objective data taken.

Not having the original code, nor, I believe, being familiar with the extension that the OP uses, I figure that the thing most likely to drastically balloon memory usage into the hundreds of MB is stored display information.

Thus, I set about creating large display items.

My first attempt was simply:

```
set im [image create photo]
button .b -image $im
pack .b

set dr {}
for {set i 0} {$i<0x100} {incr i} {
for {set j 0} {$j<8} {incr j} {
lappend dr [format "%02x%02x%02x" $i $j $j]
}
}
```

Re: Iconized application takes looooong to wake up

Re: Iconized application takes loooooong to wake up

```
set dat {}  
for {set i 0} {$i<0x800} {incr i} { lappend dat $dr }
```

\$im put \$dat

Memory usage 101,396 K / VM size 131,996 K.

Minimized and restored quickly, regardless of whether it was let to sit for long periods of time or not.

Incidentally, on a side note, trying to run TWO instances of Wish84.exe (8.4.13, in this case) failed miserably. The test system has 1GB of core, and was using only a few hundred of it at the time.

Upon running a second instance:

Fatal Error in Wish

Fail to create pixmap with Tk\_GetPixmap in ImgPhotoInstanceSetSize

Weird. Why would two separate instances of Wish.exe share resources in any way?

Anyway, moving on, my next attempt was multiple widgets.

```
for {set k 0} {$k<0x40} {incr k} {  
  frame .f$k  
  for {set i 0} {$i<0x40} {incr i} {  
    for {set j 0} {$j<8} {incr j} {  
      set col [format "%02x%02x%02x" $i $j $j]  
      frame .f${k}.f${i}_$j -bg $col -height 1 -width 1  
      pack .f${k}.f${i}_$j -side left -expand false -fill none  
    }  
  }  
  pack .f$k -side top -expand false -fill none  
}
```

Mem usage 101,396 K / VM size 131,996 K.

Now, aside from the fact that Windows choked, and would only produce a fraction of the widgets before refusing to draw any more (a long-standing bug), it did take *\*forever\** to render the window: initially, upon a fast minimize and restore, and upon a delayed minimize and restore. No difference, per se, between the cases, just poor performance.

Now, since the image trial wouldn't run more than one instance at a time, and the second trial never completed rendering, I needed something better with which to test:

```
canvas .c -bg #660099
```

Re: Iconized application takes loooooong to wake up

## Re: Iconized application takes looooong to wake up

```
pack .c -expand true -fill both
for {set i 0; set ii [expr {$i+10}]} {$i<0x1000} \
{incr i 10; incr ii 10} {
for {set j 0; set jj [expr {$j+10}]} {$j<0x1000} \
{incr j 5; incr jj 5} {
.c create oval $i $j $ii $jj -outline blue -fill red
}
}
```

"Mem Usage" is 191,036 K, "VM Size" is 188,812 K  
When minimized, Mem Usage drops to 1,180 K.

But, no matter how long I allowed the process to sit, restore of a maximized window took approximately 2 seconds, whether I restored immediately, or after letting it sit for an hour.

At this point, I should mention the target system details:

Windows 2000 Tcl/Tk 8.4.13  
Physical Memory: 1046520  
Available: 685500  
System Cache: 172420

That's before I started testing with my new "big canvas" app.

Run one instance, and Available Physical Memory drops to 503784 K.

I worked with canvas windows nearly maximized, if that matters at all.

Start multiple other instances of application, one at a time, until Available Physical Memory drops to near 0 (ensures process is suitably paged out).

Incidentally, this doesn't take very long, and paging out is pretty responsive.

Now, restore original application. Wham. System takes a major hit. The window frame displays, but the contents are paging back in. Slowly. A few KB at a time, according to Task Manager. In an attempt to speed things up, I kill all other Wish instances.

Here's where it gets interesting.

Even after killing all of the other instances, it still took about a minute *after* the other processes were killed for the original instance to get paged back in. Thereafter, the few other applications on the machine (Non-Tcl/Tk) took forever to get paged back in when activated. Explorer windows would take 10 seconds or more to open.

Re: Iconized application takes loooooong to wake up

Golden. But how could I stop at simple recreating the issue? I had to go for more.

That's when I really screwed the pooch. I ran instances of the "big canvas" Tk app until my pagefile was nearly full. About the time my test machine became unresponsive for 50 minutes, I issued a command to kill all but one instance.

About ten minutes later, they died, and Task Manager display started getting updated again.

I clicked to restore the single remaining instance of the Tk app. SIX MINUTES after Task Manager showed Available Physical Memory back at 600 MB, the application window redrew.

That's what I call high-quality memory management.

My guess is the OP has something similar to my less extreme case, where all physical memory is used, and the Tk app is slooooooowly paged back in. I'd look for a memory pig on that system other than (in addition to) the rather large Tk app.

Finding some method of keeping the Tk app in active memory is just going to swap out the other process(es) and make them fight over it.

If I could, I'd run screaming from Windows.

Hope this helps.

--

MKS

.