Subject: Re: retain and graphics_level=2
Posted by davidf on Wed, 29 Oct 1997 08:00:00 GMT
View Forum Message <> Reply to Message

Stein Vidar Hagfors Haugan (s.v.h.haugan@astro.uio.no) writes
in response to one of my posts:

>> The fact is, object graphics windows do not have to have
>> a pixmap around to "buffer" the window's contents. The
>> object graphics themselves *are* the buffer! In fact you
>> might say that the whole point of object graphics is that
>> they are persistent and that the object "scene" can be
>> reproduced at any time (albeit slowly sometimes).
> ^^^^^^^^^^^^^^^^^^^^^^^^
> And this is a bit worrying. Let me say that I have very little
> experience (yet) with object graphics, but I don't understand
> why one is forced (implicitly) to use a slow method to
> refresh windows that have been overlaid by another window.
>
>> I have gotten into the habit of setting Retain=0 and
>> Expose=1 on any window I create for object graphics
>> output. On any expose event I simply call the Draw
>> method on the Window. Simple. Easy. And it works
>> every time.
>
> But how slow is it, really?

Like everything else, it depends. On my non-OpenGL-
accelerated machines I find it noticeably slower, but
not unbearably slow for most object graphics programs.
It obviously depends on how complicated the graphics
scene is to render. The thing about graphics objects
is that they are truly three-dimensional, and the
scene itself is in a 3D space, even if what you want
to render is on a 2D plane. There is an overhead in
carrying all that 3D information around. (This is the
essential problem, by the way, that makes working with
object graphics programs over an X-terminal essentially
impossible. There are, as yet, no standard X commands
for quickly rendering a 3D scene.)

In practice, I think people will be using object graphics
for those occasions when it obviously makes sense (e.g.,
when you want to rotate a 3D surface) and continuing to use
direct graphics for those occasions when it doesn't
(e.g. line plots, images, etc.). This, as it happens, is
what RSI currently recommends. Unfortunately, I have found
it reasonably difficult to combine object graphics and

direct graphics modules into one seamless application.

Most of the time the problems involve the use of color. The object graphics system uses a different color model than the direct graphics system and it is quite difficult to get the two systems to work well together. (This is why I have been emphasising color protection schemes in my IDL courses lately. If your IDL program can't protect its own colors, you are going to be in a world of hurt as you begin running your program along with programs that use object graphics.)

In any case, knowing when and how to use "buffering" is not just an issue with respect to direct verses object graphics. It is also an issue with respect to 8-bit verses 24-bit color with direct graphics. Images, for example, have to be redisplayed on a 24-bit system after the color table has been changed. This can be a "buffering" issue. At least thinking of it as a buffering issue makes it easier to understand, I think. This issue is also becoming more prominent as most computers these days are purchased with 24-bit graphics cards. Are your IDL programs going to run in a 24-bit system?

I have been preoccupied with many of these issues for the past six months or so. In fact, if I weren't distracting myself writing newsgroup articles I would be working on this very chapter in my IDL book!

Those of you who are particularly interested and are as tired (as I am) of waiting for this damn book to be finished can hear my latest thoughts on the subject by attending my IDL Programming Techniques course Nov 18-21 in Columbia, Maryland. About half the course will be devoted to these topics. There are just a few seats left. You can find details on my web page.

I have learned, by the way, that speeding up object graphics is a high priority for the folks at RSI in their next release of IDL.

Cheers,

David

David Fanning, Ph.D.

Fanning Software Consulting
E-Mail: davidf@dfanning.com
Phone: 970-221-0438
Coyote's Guide to IDL Programming: http://www.dfanning.com/