Subject: Re: memory for pixmaps: big vs many
Posted by Jim Pendleton, ITT Vi on Thu, 21 Jun 2007 22:12:36 GMT
View Forum Message <> Reply to Message

You may find it worth testing whether or not you even need
pixmaps since physical memory transfer to the card is generally
fast enough these days to display "appropriately sized" animations
smoothly, and will also generally provide you with significantly more
room to save frames, assuming you don't induce a paging condition.

That is, if you have an RGB image cube in memory (and it's not in
a paging condition) the following action (in Object Graphics notation)

```
oImage->SetProperty, $
   Data = Reform(Frames[*, *, *, CurrentFrame])
oDrawWindow->Draw
```

can be comparable to pixmap display speed. On more modern
hardware and "reasonable" image sizes, you can often exceed the
refresh rate of the graphics card using this technique.

I haven't tested this theory with the TV command in Direct Graphics,
so your mileage may vary.

Alternatively, in Object Graphics you can pre-load a series of
IDLgrImage objects into models then use the IDLgrModel's SelectTarget
property to specify which to draw. The overhead here is that the ::Draw
method
must be called once on each image for it to be loaded into graphics memory,
making the initial display of each image sequence slow-ish; but subsequent
displays will be much faster. Unlike the physical memory option, though,
you'll be limited to the size of the graphics memory; If you exceed this
limit, then parts of the graphics memory may be sent back to physical
memory and the OS then has the option of (shudder) paging them off
to disk, which will destroy any performance gain you may have hoped
to get.

Jim P.

"Richard Clark" <rclark@hindmost.lpl.arizona.edu> wrote in message
news:f5ej1bhgg1@onion.ccit.arizona.edu...
> In using offscreen pixmaps to preload a sequence of images for smooth
> display in an animation you can run up against memory limitations in the
> graphics hardware.
>
> Two aproaches are to set up a very large pixmap that is
> xsize*total_images by ysize and store the images along the x length
> of the pixmap,

> or
> to set up total_images xsize by ysize pixmaps, one to an image.
>
> The latter aproach gives your OS and graphics hardware more of a chance
> to make use virtual memory. Works like a charm on my linux system while
> the big pixmap movie loop hits the graphics card memory limit.
>
> But a search through some discussion on the topic suggests that there
> will likely be some OS and/or graphics card specific differences in
> how well it works.
>
> Does anyone have experience with this on multiple platforms and could
> comment? For instance, any OS-wide generalizations that can be made,
> or is it hopelessly graphics card dependent?
>
> On a related note, the animation I'm interested in displaying is made
> up of b&w images. So using an 8 bit rather than 24 bit mode should
> make room for more images to be loaded in the card's memory.
>
> Does using an 8 bit mode make room for 3x, 4x, or 1x times as many
> images?
>
> Richard Clark
> rclark@lpl.arizona.edu
