Subject: Re: memory consumption when drawing an idlgrscene object
Posted by Karl Schultz on Mon, 07 Jul 2003 15:16:15 GMT
View Forum Message <> Reply to Message

"Jan" <staffNOSPAM@fys.ku.dk> wrote in message
news:Pine.LNX.4.44.0307050021240.13655-100000@johansen.fys.k u.dk...
> Hi
>
> I have a problem with an idlgrscene object that I want to draw. When I
> draw it, it consumes about 18 MB of memory, and I can't find a way of
> getting it back. Anyone has any ideas?

I think that a sample program would help this discussion a great deal. We have to consider many subtle details when analyzing memory consumption.

> To destroy the window releases a little bit of memory, but not close to 18
> MB.

When retain=2, this can be about the size of the window times 3 or 4 bytes. This could account for a few meg.

> The scene object itself does not require that much memory, and destroying
> that will therefore not release much memory.

Right. The contents of the scene are probably more important.

> The scene object contains filled idlgrcontour objects. Not filling them
> takes somewhat less memory, but it is not really an option.

The number of contour objects and the nature of the contour information is important here.

> I have set retain=2 in the draw widget where I want to draw the scene,
> changing this to 1 or 0 reduces the required memory somewhat, but not
> enough, and it is also a bad solution.

You could probably get away from retain=2 if you add an expose event handler. If memory is really that tight, this might help.

> Now we are dealing with memory consumption, anyone has any idea to why the
> following line will steal about half a megabyte of memory, and how to get
> it back:
> oContour->GetProperty, XRANGE=xr, YRANGE=yr
> ?
> The ocontour object is an idlgrcontour object.

My guess is that you are calling this before you actually draw the contour. The contour object generates a lot of geometry information such as vertex

and connectivity lists to represent the contours that you may have specified with perhaps only a compact 2D array. Depending on the size of the data and other properties of the contour, the size of this geometry information can easily approach a half meg. IDL caches this information because it requires some time to compute. This speeds up redrawing the geometry.

Usually, IDL builds these caches the first time the object is drawn. But if you ask for the range first, IDL will build the caches on that request, instead of waiting for the first draw, because it needs to know the geometry extents to return the range values. All you are really doing is building the caches a little sooner, resulting in really no net increase in overall memory consumption.

If you are that concerned about the memory consumption, you could destroy the object and recreate it before each draw, or store much simpler contour data into it so that there is little or no geometry information in the cache. But I don't think there is a big win here. You are still going to need this memory anyway to draw the contour objects.

In summary, your 18 meg are probably being used up by a combination of:

- a copy of the frame buffer due to retain=2
- the IDLgrContour geometry caches
- other variables or data
- a relatively small amount by the widgets and other misc objects.

Again, a code sample would help. If you have a dozen complex contours, then that could explain it.

Karl
