Subject: Re: memory consumption when drawing an idlgrscene object
Posted by David Fanning on Fri, 04 Jul 2003 22:52:29 GMT
View Forum Message <> Reply to Message

Jan writes:

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

What version of IDL is this? And how do you check
the memory? You don't happen to have a test program,
do you. :-)

I see some memory usage, but nothing of this
magnitude, I don't think.

> PS: Do you guys have a problem with getting spammed quite a bit when
> writing to this list?
> I've written to this list a couple of times before, and I got some good
> answers, but also a lot of spam :-(

Last month a received about 8800 e-mails, 94% of them
identified as spam. And 80% of those offered to increase
the size of my ... well, let's just say it wasn't memory,
so perhaps you aren't interested. Fortunely, with my
MailArmory spam filter, I actually saw a couple of handfuls of
those messages. Talk to your ISP about MailArmory. It
is *fabulous*!

Cheers,

David

--
David W. Fanning, Ph.D.
Fanning Software Consulting, Inc.
Phone: 970-221-0438, E-mail: david@dfanning.com
Coyote's Guide to IDL Programming: http://www.dfanning.com/
Toll-Free IDL Book Orders: 1-888-461-0155

Subject: Re: memory consumption when drawing an idlgrscene object
Posted by Karl Schultz on Mon, 07 Jul 2003 15:16:15 GMT

"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

Subject: Re: memory consumption when drawing an idlgrscene object
Posted by Jan[1] on Mon, 07 Jul 2003 22:38:54 GMT
View Forum Message <> Reply to Message

Hi guys

Thanks for your help. The problem seems to be fixed, so I will try to summarise a little bit:

I am using IDL 5.6 for Linux. The way I checked the memory consumption was basically that I ran my program in IDL, and at the same time I run top to monitor the memory consumption. By running the program twice, once when drawing the scene object, and once without drawing the scene (but doing everything else), I observed the ~18 MB difference in memory consumption. Another option is to use IDLDE and go through the program step by step while observing the memory consumption in top. In any case, it was clear that the actual drawing of the scene would require a lot of memory.

As suggested by some of you, I should make a test program. So I did, and

then by stepping through that in IDLDE I got a good feeling for what is happening. Obviously, when drawing the scene object, a lot of memory might be required. When destroying the scene object, this memory is then released again, or at least most of it. Karl Schultz mentioned that IDL would cache some information, and I suppose this is what happens here. By redoing the scene object several times with different arrays, and drawing the scene each time, the memory consumption will more or less stabilize on some level.

Also, one of my cleanup routines was not working properly, which meant that ~18 MB of memory was taken each time a scene was drawn. I did not look into that before, since I didn't consider it to be the problem. I still don't understand why an object all of a sudden takes up a lot of memory just by drawing it (and not when it is not drawn). Any ideas on that? Because of this, I suppose there is no way of seeing how much memory each object requires?

If you still want to see my test program, you are welcome, just post a message. But I think I figured out what happened to the memory, so there's nothing spooky here. Once again: thanks guys.

Regards,
Jan Staff

Subject: Re: memory consumption when drawing an idlgrscene object
Posted by Karl Schultz on Tue, 08 Jul 2003 14:33:00 GMT
View Forum Message <> Reply to Message

"Jan" <staffNOSPAM@fys.ku.dk> wrote in message
news:Pine.LNX.4.44.0307072320400.2931-100000@johansen.fys.ku .dk...
> Hi guys
>
> Thanks for your help. The problem seems to be fixed, so I will try to
> summarise a little bit:
>
> I am using IDL 5.6 for Linux. The way I checked the memory consumption was
> basically that I ran my program in IDL, and at the same time I run top to
> monitor the memory consumption. By running the program twice, once when
> drawing the scene object, and once without drawing the scene (but doing
> everything else), I observed the ~18 MB difference in memory consumption.
> Another option is to use IDLDE and go through the program step by step
> while observing the memory consumption in top. In any case, it was clear
> that the actual drawing of the scene would require a lot of memory.

Top is OK, but you should also investigate the memory() function built into IDL.

> As suggested by some of you, I should make a test program. So I did, and
> then by stepping through that in IDLDE I got a good feeling for what is
> happening. Obviously, when drawing the scene object, a lot of memory might
> be required. When destroying the scene object, this memory is then
> released again, or at least most of it. Karl Schultz mentioned that IDL
> would cache some information, and I suppose this is what happens here. By
> redoing the scene object several times with different arrays, and drawing
> the scene each time, the memory consumption will more or less stabilize on
> some level.

This stabilizing is an important factor when conducting memory usage investigations. Some of the tables inside IDL may grow slightly to reach "high-water marks" in terms of variable and object utilization. One very common practice used here at RSI is to run the code suspected of leaking twice. The first time, ignore the memory measurements, and the second time pay attention to them.

```
pro proc_under_test
   obj = OBJ_NEW("myobj")
   OBJ_DESTROY, obj
end

pro test
   proc_under_test
   mem1 = memory()
   proc_under_test
   mem2 = memory()
   print, mem1, mem2
end
```

In fact, to see what I am talking about a little easier, start a fresh IDL session and:

```
IDL> print, memory()
458966 378 132 460726
IDL> print, memory()
458990 380 133 458990
IDL> print, memory()
458990 381 134 458990
```

Note that there is a slight increase in memory between the first and second invocations and none between the second and third.

This is really important when analyzing memory problems, because a lot of people misinterpret the memory delta between the first and second invocations as a real problem.

> Also, one of my cleanup routines was not working properly, which meant

> that ~18 MB of memory was taken each time a scene was drawn. I did not
> look into that before, since I didn't consider it to be the problem.

Glad you found it.

> I still don't understand why an object all of a sudden takes up a lot of
> memory just by drawing it (and not when it is not drawn). Any ideas on
> that?

Sure. Because of the caches I mentioned. When a graphics object like IDLgrContour is instantiated, it really only stores the properties and data that you supply to it. In the case of IDLgrContour, this might be just the 2D array that you pass in as the first argument. Often, nothing else happens until you draw the object. At this point, IDL converts the data that you have supplied into a form that is optimized for rendering and stores this data into a cache that remains until the object (or window it was drawn on) is destroyed. This way, subsequent drawings of the scene containing the object go much faster. This improves the performance of animations and trackball manipulations.

The amount of benefit the caches provide depend on the object. IDLgrContour and IDLgrSurface benefit greatly because the caches require quite a bit of computation to compute - more than you would want to do on each redraw. In the case of IDLgrContour, the contouring algorithm is applied to generate the contour lines.

> Because of this, I suppose there is no way of seeing how much memory
> each object requires?

Not really. The caches can be of varying size, depending on the properties that are set and a bunch of other factors. Also, each object has its own caching policy - some may not even cache at all. The whole caching thing is private to the object implementation and should be invisible to the user, except for the memory consumption, as you have noticed. Unless there is an object implementation bug that causes a leak, a memory problem might lie elsewhere, as you have experienced.

> If you still want to see my test program, you are welcome, just post a
> message. But I think I figured out what happened to the memory, so there's
> nothing spooky here. Once again: thanks guys.

No need. But I think the memory() function will be more useful to you in future investigations.

Karl