Subject: Re: IDL Objects Graphics cache and crash
Posted by Mark Hadfield on Sun, 10 Mar 2002 20:38:26 GMT
View Forum Message <> Reply to Message

"Altyntsev Dmitriy" <alt@iszf.irk.ru> wrote in message
news:6b9fda50.0203100016.76b8433d@posting.google.com...
> I am trying now to turn to IDL Objects Graphics so a pair of
> questions to IDL gurus.

None of the gurus seem to have replied yet, so I'll have a go.

> In IDL online help topic "The Graphics Object Hierarchy -> The
> Rendering Process" one can read about draw cache: "Subsequent draws
> of this graphic atom to the same destination can then be drawn very
> efficiently."

Hey cool! I never noticed that paragraph before. (I've never noticed
the behaviour it describes, either.)

> But it seems not to work as declared. When I change HIDE property of
> some graphic atom and then draw window it takes the same time as
> first drawing. What's wrong?

Probably the documentation.

By the way, I find the paragraph immediately before the one you refer
to particularly amusing--the one that begins, "The order in which
objects are added to the hierarchy will have an impact...". It fails
to mention the fact that the visibility of all OG atoms, except
images, is controlled by position in 3D space, not by drawing order.

> I tried different renderer and retain, but can not get any
> acceleration. I use Win98, IDL 5.4, video ASUS AGP3800.

Well, you've already anticipated my only suggestion, which was to try
hardware & software renderers.

> Using of IDLgrROI sometimes crashes IDL. It is very unpleasant. May
> be anyone can advice how avoid these crashes and other OG bugs if
> any. I use Win98, IDL 5.4

Sorry, I have no suggestion on this specific bug. Can you upgrade to
IDL 5.5? With object graphics, later is generally better. Do the
crashes occur with the software renderer? IDL display problems are
sometimes caused by the video driver and the hardware renderer
exercises this driver much more than the software one. And of course
upgrading Windows to one of the NT family (2000, XP) is generally a
good idea if you want stability.

Mark Hadfield
m.hadfield@niwa.co.nz Ka puwaha et tai nei
http://katipo.niwa.co.nz/~hadfield Hoea tatou
National Institute for Water and Atmospheric Research (NIWA)

Subject: Re: IDL Objects Graphics cache and crash
Posted by karl_schultz on Mon, 11 Mar 2002 16:48:01 GMT
View Forum Message <> Reply to Message

"Mark Hadfield" <m.hadfield@niwa.co.nz> wrote in message
news:<a6ghfk$mj6$1@newsreader.mailgate.org>...
> "Altyntsev Dmitriy" <alt@iszf.irk.ru> wrote in message
> news:6b9fda50.0203100016.76b8433d@posting.google.com...
>> I am trying now to turn to IDL Objects Graphics so a pair of
>> questions to IDL gurus.
>
> None of the gurus seem to have replied yet, so I'll have a go.
>
>> In IDL online help topic "The Graphics Object Hierarchy -> The
>> Rendering Process" one can read about draw cache: "Subsequent draws
>> of this graphic atom to the same destination can then be drawn very
>> efficiently."
>
> Hey cool! I never noticed that paragraph before. (I've never noticed
> the behaviour it describes, either.)

Most Object Graphics objects create caches of graphic primitives that
are essentially vertex lists that are suitable for direct submission
to OpenGL. Especially when working with hardware accelerated cards,
it is important to submit this vertex information to OpenGL as quickly
as possible. So, we try to avoid any computations or conversions that
can get in the way of rapid submission by creating the caches. This
was perhaps more important a few years ago when CPU's were slower.

Like so many other things in the graphics world, the acutal benefit
you see will vary greatly, depending on your hardware and exactly what
your program is doing. For some objects, like IDLgrPolyline and
IDLgrPolygon, the caches are not much different than the vertex data
stored in the object. But for other objects like IDLgrText and
IDLgrSurface, there is a significant difference between the data you
store in the object and the cache contents. So, the caches are a
bigger win for these objects.

Also (important for the discussion below), IDL can build the caches
"without notice", although it generally does so only when it needs to.

Most objects wait until they are actually drawn for the first time.
Some might do it whenever the data changes. The point of the caches
is to achieve fast drawing in situations where the application is only
changing trackball transforms, for example.

>
>> But it seems not to work as declared. When I change HIDE property of
>> some graphic atom and then draw window it takes the same time as
>> first drawing. What's wrong?
>
> Probably the documentation.

It is really hard to measure stuff like this, especially with some
unpredictability surrounding cache rebuilds.

I would suggest measuring the time it takes to draw your model a few
hundred times. Then hide the atom in question, and then measure
again. If the atom is sufficently complex, you should see a
difference. If the graphic atom is something really simple, like a
single polyline, you may not measure much of a difference.

>
> By the way, I find the paragraph immediately before the one you refer
> to particularly amusing--the one that begins, "The order in which
> objects are added to the hierarchy will have an impact...". It fails
> to mention the fact that the visibility of all OG atoms, except
> images, is controlled by position in 3D space, not by drawing order.

That paragraph probably assumed that the reader expected the "natural"
result of depth sorting/buffering. It could be clearer. Rendering
order makes the biggest difference when things are at the same depth
or transparency is involved.

>
>> I tried different renderer and retain, but can not get any
>> acceleration. I use Win98, IDL 5.4, video ASUS AGP3800.
>
> Well, you've already anticipated my only suggestion, which was to try
> hardware & software renderers.

We'd need to learn more about why he can't get the expected
acceleration. A lot also depends on the OpenGL implementation for
that card. Sometimes, the software implementation beats the
"hardware" implementation if the OpenGL implementation is poor. One
quick way to get a rough feel for this OpenGL implementation is to run
some other OpenGL application or demo and see how it compares. Or,
just see what the vendor claims about this product. A lot of times,
the vendor won't supply an OpenGL driver and so the system ends up

using the Microsoft OpenGL, which is a mostly software implemenation.

>
>> Using of IDLgrROI sometimes crashes IDL. It is very unpleasant. May
>> be anyone can advice how avoid these crashes and other OG bugs if
>> any. I use Win98, IDL 5.4
>
> Sorry, I have no suggestion on this specific bug. Can you upgrade to
> IDL 5.5? With object graphics, later is generally better. Do the
> crashes occur with the software renderer? IDL display problems are
> sometimes caused by the video driver and the hardware renderer
> exercises this driver much more than the software one. And of course
> upgrading Windows to one of the NT family (2000, XP) is generally a
> good idea if you want stability.

The IDLgrROI object did have a few bugs in 5.4 (fixed in 5.5), many of which can be worked around if we know enough about the problem. For example, avoiding setting the vertex data in IDLgrROI with SetProperty (only set the vertex data at Init time) helps avoid many of the problems. This means that you'd have to destroy and recreate the object if you wanted to change the data in it, but at least that's a workaround that could help until upgrading to 5.5.

Karl

Subject: Re: IDL Objects Graphics cache and crash
Posted by Rick Towler on Mon, 11 Mar 2002 18:50:47 GMT
View Forum Message <> Reply to Message

>> In IDL online help topic "The Graphics Object Hierarchy -> The
>> Rendering Process" one can read about draw cache: "Subsequent draws
>> of this graphic atom to the same destination can then be drawn very
>> efficiently."
>
> Hey cool! I never noticed that paragraph before. (I've never noticed
> the behaviour it describes, either.)

I whipped up a little hack to test this and I do see the benefits in high polygon scenes. I suspect that what is being cached are the vertex normals used to compute lighting. Low poly scenes aren't limited by these extra calculations so you don't see the benefit. My test used an orb with a density of 10. Without changing the vertex data I would get around 65 frames/second. When I did change the vertex data on each frame I got around 15 fps (you need to comment out the "self->BuildPoly" line in orb::setproperty for this to work).

>> But it seems not to work as declared. When I change HIDE property of
>> some graphic atom and then draw window it takes the same time as
>> first drawing. What's wrong?
>
> Probably the documentation.
>
> By the way, I find the paragraph immediately before the one you refer
> to particularly amusing--the one that begins, "The order in which
> objects are added to the hierarchy will have an impact...". It fails
> to mention the fact that the visibility of all OG atoms, except
> images, is controlled by position in 3D space, not by drawing order.
>

Not totally true. Visibility of OG atoms is controlled by position in 3D
space AND by drawing order. The latter only comes into play when your atoms
are transparent (textured with a image containing an alpha channel). Karl
might have to correct me but as the view heirarchy is traversed the atoms
are drawn negative z verts to positive. This has implications both in the
visibility of the scene as a whole (seeing thru one atom to another) and for
the visibility of the atoms themselves (seeing the back side of a
transparent 3d object). For a full discussion on this search the newsgroup
for "pimento problems".

>> I tried different renderer and retain, but can not get any
>> acceleration. I use Win98, IDL 5.4, video ASUS AGP3800.
>

How do you know that you aren't seeing any acceleration? In simple scenes
you will probably not see a difference. Have you tried an OG benchmark?
The TNT2 chipset on that card is fairly decent and should provide an
improvement over the software renderer. I suggest running time_test_gr2.pro
with both the software and hardware renderers. (if you don't have a copy,
myself or others on the list can provide one)

On the subject of general stability, I second Mark's comments. The video
driver can be one source of problems but only when rendering in hardware
mode. If you use the software renderer and still run into problems it is
probably not your driver. Also, win9x is very unforgiving when you are
sloppy with OO programming in IDL. I have had much better luck with Win2k.

-Rick

Subject: Re: IDL Objects Graphics cache and crash

Posted by karl_schultz on Tue, 12 Mar 2002 01:11:01 GMT

View Forum Message <> Reply to Message

"Rick Towler" <rtowler@u.washington.edu> wrote in message
news:<a6iuea$oc6$1@nntp6.u.washington.edu>...
>>> In IDL online help topic "The Graphics Object Hierarchy -> The
>>> Rendering Process" one can read about draw cache: "Subsequent draws
>>> of this graphic atom to the same destination can then be drawn very
>>> efficiently."
>>
>> Hey cool! I never noticed that paragraph before. (I've never noticed
>> the behaviour it describes, either.)
>
> I whipped up a little hack to test this and I do see the benefits in high
> polygon scenes. I suspect that what is being cached are the vertex normals
> used to compute lighting. Low poly scenes aren't limited by these extra
> calculations so you don't see the benefit. My test used an orb with a
> density of 10. Without changing the vertex data I would get around 65
> frames/second. When I did change the vertex data on each frame I got around
> 15 fps (you need to comment out the "self->BuildPoly" line in
> orb::setproperty for this to work).

Right. I forgot about the normals when I said that caching didn't
help polygons as much. Nice results.

>>> But it seems not to work as declared. When I change HIDE property of
>>> some graphic atom and then draw window it takes the same time as
>>> first drawing. What's wrong?
>>
>> Probably the documentation.
>>
>> By the way, I find the paragraph immediately before the one you refer
>> to particularly amusing--the one that begins, "The order in which
>> objects are added to the hierarchy will have an impact...". It fails
>> to mention the fact that the visibility of all OG atoms, except
>> images, is controlled by position in 3D space, not by drawing order.
>>
>
> Not totally true. Visibility of OG atoms is controlled by position in 3D
> space AND by drawing order. The latter only comes into play when your atoms
> are transparent (textured with a image containing an alpha channel). Karl
> might have to correct me but as the view heirarchy is traversed the atoms
> are drawn negative z verts to positive. This has implications both in the
> visibility of the scene as a whole (seeing thru one atom to another) and for
> the visibility of the atoms themselves (seeing the back side of a
> transparent 3d object). For a full discussion on this search the newsgroup
> for "pimento problems".

Oh no, not the pimento!!!! :-)

The graphic atom drawing order depends completely on the order in
which models are placed in the view, and the order in which atoms are
placed into models.

Within an atom, the order in which vertices are drawn happens to
depend on the order that they are described in the primitive, and not
by their Z ordering. For example, if you have an IDLgrPolygon object
with a connectivity list that starts: [4, 200,201,202,203...], the
first polygon drawn will use verts 200, 201, 202, and 203, regardless
of the Z value of these vertices as compared to the other verts in the
object.

As Rick mentions, this is really only gets important when working with
transparency.

Part of the pimento discussion was about the Orb object. The Orb
object generates a mesh which happens to draw the triangles from
negative Z to positive Z. That is a property of the Orb object and
nothing else.

I have a little test program that draws three parallel polygons with
transparency. The program uses a trackball and changes the order of
the polygon drawing as the model "flips" from one side to another so
that the transparency always looks right. The problem is pretty easy
to solve for simple data like this, but gets really messy for more
complex scenes. I'll clean up the program and send it to anyone who
is interested.

Karl

Subject: Re: IDL Objects Graphics cache and crash
Posted by Rick Towler on Tue, 12 Mar 2002 01:49:51 GMT
View Forum Message <> Reply to Message

Many thanks to Karl. Always dishing out the OG good stuff. I know you work
with this all day and answering our questions can be tedious.

All of us OG users appreciate your participation in this newsgroup.

-Rick

Subject: Re: IDL Objects Graphics cache and crash
Posted by btupper on Tue, 12 Mar 2002 13:32:13 GMT
View Forum Message <> Reply to Message

On Mon, 11 Mar 2002 17:49:51 -0800, "Rick Towler"
<rtowler@u.washington.edu> wrote:

> Many thanks to Karl. Always dishing out the OG good stuff. I know you work
> with this all day and answering our questions can be tedious.
>
> All of us OG users appreciate your participation in this newsgroup.
>
> -Rick
>
>

Hi,

I second that!

Ben

P.S. And many thanks to you, too, Rick.

Subject: Re: IDL Objects Graphics cache and crash
Posted by Karl Schultz on Tue, 12 Mar 2002 16:30:53 GMT

View Forum Message <> Reply to Message

"Karl Schultz" <karl_schultz@yahoo.com> wrote in message
news:e415b359.0203111711.159d69ee@posting.google.com...
> I have a little test program that draws three parallel polygons with
> transparency. The program uses a trackball and changes the order of
> the polygon drawing as the model "flips" from one side to another so
> that the transparency always looks right. The problem is pretty easy
> to solve for simple data like this, but gets really messy for more
> complex scenes. I'll clean up the program and send it to anyone who
> is interested.

Here's the program. It is pretty simple, but should show the general idea.

```
;;
;; Simple opacity and rendering order demonstration
;;
;; The small blue square is opaque.  The red and green squares are
translucent.
;; Rotate the scene to observe that you can always see the other two squares
;; through the front square.  This is accomplished by changing the drawing
order
;; of the squares as the trackball transform changes their "stacking" order
;; relative to the viewer.
;;
;; Run with /NOFLIP to observe what happens when the drawing order is not
```

```
changed.
;;
;; S. Houston / K. Schultz RSI
;;

PRO OPACITY_TEST_EVENT, sEvent

   widget_control, sEvent.id, GET_UVALUE=uval

   ; Handle KILL requests.
   IF TAG_NAMES(sEvent, /STRUCTURE_NAME) EQ 'WIDGET_KILL_REQUEST' THEN
BEGIN
      WIDGET_CONTROL, sEvent.top, GET_UVALUE=sState

     ; Destroy the objects.
     OBJ_DESTROY, sState.oView
     OBJ_DESTROY, sState.oTrack
     WIDGET_CONTROL, sEvent.top, /DESTROY
     RETURN
   ENDIF

   ; Handle other events
   CASE uval OF
      'DRAW': BEGIN
         widget_control, sEvent.top, GET_UVALUE=sState, /NO_COPY

         ; expose
         IF (sEvent.type EQ 4) THEN BEGIN
            sState.oWindow->Draw, sState.oView
            widget_control, sEvent.top, SET_UVALUE=sState, /NO_COPY
            RETURN
         ENDIF

         ; Trackball updates
         bHaveTransform = sState.oTrack->Update(sEvent, TRANSFORM=qmat)
         if (bHaveTransform NE 0) THEN BEGIN
            sState.oTopZPlus->GetProperty, TRANSFORM=t
            t = t # qmat
            sState.oTopZPlus->SetProperty, TRANSFORM=t
            sState.oTopZMinus->SetProperty, TRANSFORM=t
            print, "Z direction of trackball:", t[2,2]
            ;;
            ;; If the Z direction of the trackball is positive, hide
            ;; the model that draws the polygons sorted from +Z to -Z
            ;; and draw the model that draws the polygons sorted from -Z
to +Z.
            ;; Vice-versa for a trackball Z direction that is negative.
            if not sState.noflip then begin
```

```
                if t[2,2] ge 0 then begin
                    print, "Draw green square before red square"
                    sState.oTopZPlus->SetProperty, HIDE=0
                    sState.oTopZMinus->SetProperty, HIDE=1
                endif else begin
                    print, "Draw red square before green square"
                    sState.oTopZPlus->SetProperty, HIDE=1
                    sState.oTopZMinus->SetProperty, HIDE=0
                endelse
            endif else begin
                print, "Draw green square before red square"
            endelse
            sState.oWindow->Draw, sState.oView
        ENDIF

        ; button press
        IF (sEvent.type EQ 0) THEN BEGIN
            sState.btndown = 1b
            widget_control, sState.wDraw, /DRAW_MOTION
            sState.oWindow->Draw, sState.oView
        ENDIF

        ; button release
        IF (sEvent.type EQ 1) THEN BEGIN
            IF (sState.btndown EQ 1b) THEN $
                sState.oWindow->Draw, sState.oView
            sState.btndown = 0b
            widget_control, sState.wDraw, DRAW_MOTION=0
        ENDIF

        WIDGET_CONTROL, sEvent.top, SET_UVALUE=sState, /NO_COPY
    END
  ENDCASE
END

;  ------------------------------------------------------------ --------------
-
pro opacity_test, NOFLIP=noflip

  noflip = KEYWORD_SET(noflip)

  xdim = 512
  ydim = 512

  ; Create the widgets
  wBase = WIDGET_BASE(/COLUMN, XPAD=0, YPAD=0, $
                    TITLE="Opacity Test", $
                    /TLB_KILL_REQUEST_EVENTS)
```

```
   wDraw = WIDGET_DRAW(wBase, XSIZE=xdim, YSIZE=ydim, UVALUE='DRAW', $
                 RETAIN=0, /EXPOSE_EVENTS, /BUTTON_EVENTS, $
                 GRAPHICS_LEVEL=2)

   ; Realize the widgets
   widget_control, wBase, /REALIZE

   ; Get the is of the drawable
   widget_control, wDraw, GET_VALUE=oWindow

   oView = OBJ_NEW('IDLgrView', COLOR=[255,255,255], PROJECTION=2)
   oTopZPlus = OBJ_NEW('IDLgrModel')
   oTopZMinus = OBJ_NEW('IDLgrModel')
   oTrack = OBJ_NEW('Trackball', [xdim/2, ydim/2], xdim/2)


   ;; Red transparent polygon
   texData = BYTARR(4,64,64)
   texData[0,*,*] = 255
   texData[1,*,*] = 0
   texData[2,*,*] = 0
   texData[3,*,*] = 64

   oImage1 = OBJ_NEW('IDLgrImage', texData, INTERLEAVE=0)
   oPoly1 = OBJ_NEW('IDLgrPolygon', [-0.5, 0.5, 0.5, -0.5], $
                            [-0.5, -0.5, 0.5, 0.5], $
                            [0.5, 0.5, 0.5, 0.5], $
                            COLOR=[255,255,255], $

TEXTURE_COORD=[[0,0],[1,0],[1,1],[0,1]], $
                            TEXTURE_MAP=oImage1)
   oModel1 = OBJ_NEW('IDLgrModel')
   oModel1->Add, oPoly1

   ;; Green transparent polygon
   texData[0,*,*] = 0
   texData[1,*,*] = 255
   texData[2,*,*] = 0
   texData[3,*,*] = 64

   oImage2 = OBJ_NEW('IDLgrImage', texData, INTERLEAVE=0)
   oPoly2 = OBJ_NEW('IDLgrPolygon', [-0.5, 0.5, 0.5, -0.5], $
                            [-0.5, -0.5, 0.5, 0.5], $
                            [-0.5, -0.5, -0.5, -0.5], $
                            COLOR=[255,255,255], $

TEXTURE_COORD=[[0,0],[1,0],[1,1],[0,1]], $
```

```
                              TEXTURE_MAP=oImage2)
   oModel2 = OBJ_NEW('IDLgrModel')
   oModel2->Add, oPoly2

   ;; Blue opaque polygon
   oPoly3 = OBJ_NEW('IDLgrPolygon', $
                             [-0.2, 0.2, 0.2, -0.2], $
                             [-0.2, -0.2, 0.2, 0.2], $
                             COLOR=[0,0,255])
   oModel3 = OBJ_NEW('IDLgrModel')
   oModel3->Add, oPoly3


   oTopZPlus->Add, oModel3 ; Non transparent first
   oTopZPlus->Add, oModel2 ; Green (back) next
   oTopZPlus->Add, oModel1 ; Red (front) last
   oView->Add, oTopZPlus
   oTopZMinus->Add, oModel3, /ALIAS ; Non transparent first
   oTopZMinus->Add, oModel1, /ALIAS ; Red (front, but back when rotated)
next
   oTopZMinus->Add, oModel2, /ALIAS ; Green (back, but front when rotated)
last
   oView->Add, oTopZMinus

   ; Hide the back-oriented model
   oTopZMinus->SetProperty, HIDE=1

   sState = {  btndown: 0b, $
            wDraw: wDraw, $
            oWindow: oWindow, $
            oView: oView, $
            oTopZPlus: oTopZPlus, $
            oTopZMinus: oTopZMinus, $
            oTrack: oTrack, $
            noflip: noflip }

   widget_control, wBase, SET_UVALUE=sState, /NO_COPY
   xmanager, 'opacity_test', wBase, /NO_BLOCK

end
```

Subject: Re: IDL Objects Graphics cache and crash
Posted by alt on Wed, 13 Mar 2002 14:07:33 GMT

View Forum Message <> Reply to Message

Thanks to all of you and especially to Karl Schultz for almost
exhaustive information about OG cache and drawing order. I see it was

useful and appreciated by many people. I join to others thanks.

But I asked this question on other matter. Actually I am not engage in 3D stuff. I have a GUI application with interface that reminds GIS like ArcView. It is completely 2D. It has the main graphics window which displays images, overlaid polylines, polygons, symbols, ROI, zoom box. They organized as layers with names, properties and so on. There is an interface to change visibility and properties of each layer. It has floating zoom window. The other interface stuff is specialized for our purposes so I suppose it is not advantageous to use ENVI displays functions or any GIS.

One of the problems in this case is how to overlay different vector objects on image in most effective way. I mean when you change visibility (or position, color, one vertex coord...) of some layer the result of this change should be displayed as soon as possible. If it happens really fast it creates very good impression from work. Otherwise you just begin to avoid "clicking" because of latency. To illustrate the task, imagine the bundle of transparency films with pictures on them.

In Direct Graphics, if some property is changed, the picture should be drawn from the beginning. I expected that Object Graphics is smarter in this case and caches "visual representation" as SOMETHING (e.g. images with mask or some image indexes to be drawn) and then renderer (hardware) sticks layers together very fast. But it seems OG draws the picture from the beginning too.

Let's look at the small test that moves polyline over image.

```
pro test
  szx = 1000L
  szy = 650L
  img = bytscl(dist(szx,szy))
  N = 100L
  x = randomu(seed,N) * szx
  y = randomu(seed,N) * szy

  w = obj_new('IDLgrWindow', dim = [szx,szy], retain = 0, render = 1)
  v = obj_new('IDLgrView', view = [0,0,szx,szy])
  m = obj_new('IDLgrModel')
  im = obj_new('IDLgrImage', img)
  pl = obj_new('IDLgrPolyline', x, y, color = [255,0,0] )
  v -> add, m
  m -> add, im
  m -> add, pl
  t0 = systime(1)
  for dx = 0, 100 do begin
```

```
    pl -> SetProperty, xcoord = [dx,1]
    w -> draw, v
  endfor
  print, 'Time = ', systime(1) - t0

end
```

It takes 13.45 sec on my Pentium-III 700. It is very very slow. With render = 0 it is even slower. I am sure that modern video cards allow moving this polyline with the speed of bullet. Simple redrawing without moving takes the same time. It seems that image does not even moved as object in video memory. Probably because IDLgrImage is not actual 3D object. Or may be this entire task is not OpenGL purpose. Or I have some problems with my OpenGL driver? Although it works fine with outside OpenGL tests and games. time_test_gr2 seems to not work properly, but IDL demo works fine.

OG as a concept and utilities library is very very suitable for this task programming and it would be pity not to use it because of speed.

I have feeling that I missed something very trivial. How can I make this stuff faster?

Thank you in advance.

Best regards,
Altyntsev Dmitriy
Remote Sensing Center, ISTP
Irkutsk, Russia
http://ckm.iszf.irk.ru