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