Subject: Re: IDL 8.2.2 released
Posted by tom.grydeland on Fri, 26 Apr 2013 10:06:50 GMT
View Forum Message <> Reply to Message

On Tuesday, April 16, 2013 4:08:24 AM UTC, Chris Torrence wrote:

> This has been fixed for IDL 8.2.3.
> [...]

> Note that if you don't disable the refresh, then each time a new plot is added, IDL will recompute the plot range to make sure the axes cover all of the plots.

Obviously. What I tried to say was that the quadratic behaviour suggests to me that this recomputation queries _all_ existing objects in the plot, something which is quite unnecessary. The plot range necessary for all _previously_ added objects does not change as a result of adding another one.

This is quite orthogonal to whether refresh is disabled or not.

((In pseudocode of an AXIS object of some description, assuming BOX will compute the union of two bounding-box arguments, and REDUCE works as one would expect, this can be expressed as

```
  self.bbox = BOX(self.bbox, newOjb.bbox)   ; bbox kept in property of AXIS
```

instead of

```
  bbox = REDUCE('BOX', self.objects)        ; bbox recomputed and not kept
```

))

> the time for radial_antenna(12) went from 40 seconds down to around 4 seconds.

Obviously a great improvement, but I cannot shake a feeling that there must be at least another factor of 10 within reach.

For comparison, on our old compute-server (Dual-core AMD Opteron, 2.6 GHz, 5200 bogomips) and with X over the network, an old version of That Other Vectorized Language (r2008a) does radial_antenna with time per antenna of 0.75 ms consistently for n from 10 to 30 (100 to 900 antennas in the pattern).

On my desktop (i5, 3.2 GHz, 6400 bogomips), the time per antenna is 70 ms for n=6, 130 ms for n=10, 234 ms for n=14, and that is as far as my patience extends for now.

If you run the instrumented code below, the final plot should show a roughly constant value, not one that increases quadratically. Does it in 8.2.3?

```
function radial_antenna, n, length=length
```

```
  if n_elements(length) eq 0 then length = 0.9

  ii = indgen(n)-(n-1.)/2
  xx = ii[*,ii]
  yy = transpose(xx)
  cg = plot(xx[*], yy[*], 'D', aspect_ratio=1)
  cg.refresh, /disable

  nx = n_elements(xx)
  if nx mod 2 then begin
    ;; if there is an element in the centre, eliminate it
    tmp = [indgen(nx/2), nx/2+1+indgen(nx/2)]
    xx = xx[tmp]
    yy = yy[tmp]
  endif
  th = atan(yy, xx)

  tic
  for ii=0L, n_elements(xx)-1 do begin
    cx = length/2*[-1, 1]*cos(th[ii])
    cy = length/2*[-1, 1]*sin(th[ii])
    !null = plot(xx[ii]+cx, yy[ii]+cy, 'k', /current, /overplot)
    !null = plot(xx[ii]-cy, yy[ii]+cx, 'r', /current, /overplot)
  endfor

  cg.refresh
  return, toc() / n^2
end


;; main routine

nn = 6 + 4*indgen(3)
tt = 0. * nn

foreach ii, nn, idx do tt[idx] = radial_antenna(ii)

!null = plot(nn, tt)


end
```

> Chris

--T