Subject: Strange problems with IDL on my PC (memory allocation and 3d rendering)
Posted by Roberto Ricci on Tue, 26 Apr 2005 13:57:07 GMT
View Forum Message <> Reply to Message

Dear all,
even though I'm a newbie to this newsgroup, I've got some 5 years
experience in IDL programming and I somewhat got to know most of you
gurus by

reputation. In the last years I've been writing an IDL application for
the reconstruction and interactive manipulation of 3D models from range
and

intensity data acquired by using a particular laser range finder
prototype. Since final models can be very huge (some million triangles,
on the

average), I'm continuosly engaged in the challenge of improving the
performances of my application. While doing so, I noticed a quite
peculiar

behaviour with IDL on my current workstation (a Fujitsu-Siemens R610
Windows XP machine, with two Xeon 3.2 GHz processors, 4 GB RAM (!) and
a

professional nVidia FX3000 video card with 256 MB dedicated memory).
I'm pretty sure this behaviour does not depend, strictly speaking, on
IDL,

since the same application performs as expected on different computers
- even much less mighty than mine - but it only becomes manifest when I
use

IDL. That's why I'm asking to this newsgroup...
I've noticed in fact two different problems, that could possbly be
found to have some relationship with each other:

1. Despite, as I said, I can theoretically take advantage of plenty of
RAM (4GB, but I think each different application can exploit only up to
2

GB under Windows XP), my application happens to go frequently out of
memory, especially when I try to allocate large arrays. Since I've
always

been making my best to limit memory leakage, I got suspicious and
started experimenting with some memory allocation test IDL applications

I found

here and there - this newsgroup included. The results are discouraging:
I found that I cannot achieve to allocate more than approximately 700
MB

in a single block on my PC and less than 2GB overall. I report in the
following the source of one of the test programs I used together with
the

resusts obtained:

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Procedure: MEMTEST
;
; Syntax: memtest
;
; Purpose:
; This procedure was designed primarily to test the impacts of Windows
OS
; memory fragmentation behavior on IDL memory allocation.
;
; The procedure attempts to allocate 10 coexisting memory blocks of 2
GB size.
; If there is not enough memory to accomplish this, it allocates the 10
; largest coexisting blocks that it can. It stops allocating new memory
blocks
; either:
;
; - when it has allocated full 10 blocks.
; - when it cannot allocate any additional block of more than 1 MB in
size
;   (i.e. when the application has run out of available memory).
;
; Postcondition:
; This program outputs a log of its successful allocations that may
look like:
;
; Memory block # 1: 1168 Mb (total: 1168 Mb)
; Memory block # 2:  206 Mb (total: 1374 Mb)
; Memory block # 3:  143 Mb (total: 1517 Mb)
; Memory block # 4:  118 Mb (total: 1635 Mb)
; Memory block # 5:   79 Mb (total: 1714 Mb)
; Memory block # 6:   54 Mb (total: 1768 Mb)
; Memory block # 7:   41 Mb (total: 1809 Mb)
; Memory block # 8:   39 Mb (total: 1848 Mb)
; Memory block # 9:   31 Mb (total: 1879 Mb)
; Memory block #10:   16 Mb (total: 1895 Mb)
```

```
;
; (Note that the output may have fewer than 10 blocks of memory)
;-
pro memtest
  compile_opt idl2 ; set default integers to 32-bit and enforce [] for
indexing

  MB = 2^20
  currentBlockSize = MB * 2047   ; 2 GB
  print, MB, currentBlockSize
  maxIterations = 10             ; Max loop iterations
  memPtrs = ptrarr(maxIterations)
  memBlockSizes = ulonarr(maxIterations)

  for i=0, maxIterations-1 do begin
  ; Error handler
    catch, err

    ; Sepcifically designed for "Failure to allocate memory..." error
    if (err ne 0) then begin
      currentBlockSize = currentBlockSize - MB     ; ...try 1 MB
smaller allocation
      if (currentBlockSize lt MB) then break  ; Give up, if
currentBlockSize < 1 MB
    endif

  ; This 'wait' enables Ctrl-Break to interrupt if necessary (Windows).
    wait, 0.0001

  ; Allocate memory (if possible)
    memPtrs[i] = ptr_new(bytarr(currentBlockSize, /nozero), /no_copy)
    memBlockSizes[i] = currentBlockSize   ; Store the latest successful
allocation size

  ; Print the current allocated block size and the running total, in Mb
    print, format='(%"Memory block #%2d: %4d Mb (total: %4d Mb)")', $
      i + 1, ishft(currentBlockSize, -20),
ishft(ulong(total(memBlockSizes)), -20)
  endfor

  ptr_free,memPtrs
end
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

My results are:

```
IDL> memtest
     1048576  2146435072
```

```
Memory block # 1:  689 Mb (total:  689 Mb)
Memory block # 2:  314 Mb (total: 1003 Mb)
Memory block # 3:  235 Mb (total: 1238 Mb)
Memory block # 4:  213 Mb (total: 1451 Mb)
Memory block # 5:  175 Mb (total: 1626 Mb)
Memory block # 6:   64 Mb (total: 1690 Mb)
Memory block # 7:   60 Mb (total: 1750 Mb)
Memory block # 8:   36 Mb (total: 1786 Mb)
Memory block # 9:   27 Mb (total: 1813 Mb)
Memory block #10:   20 Mb (total: 1833 Mb)
```

Please note that a porting in C of the same program gives the following results on my PC:

```
megabyte = 1048576      starting block size = 2146435072

Memory block   1: 1987 Mb  (total: 1987 Mb)
Memory block   2:   45 Mb  (total: 2032 Mb)
Memory block   3:    7 Mb  (total: 2039 Mb)
```

On top of this, the same IDL applications gives much better results on different PCs, where I manage to allocate in a single block more than the

total amount of physically available RAM!

2. The second problem is even more obscure... I noticed that I obtain better performances (half rendering times) by using software OpenGL

(RENDERER=1) than hardware acceleration (RENDERER=0)!!! Please note that RETAIN is correctly set to 0 and I verified I'm really using video card

accelerator hardware (namely, by using antialiasing as a test). Once again, the same application behaves correctly on other PCs (hardware around

10 times faster than software). Nonetheless, I've had no problem whatsoever with my videocard when running other graphically heavy applications.

Even 3D benchmarks behave more or less as expected!

I hope some of you can help me solve these puzzles...
Ciao,
  Roberto