

Posted by [Dennis J. Boccippio](#) on Thu, 13 Jul 2000 07:00:00 GMT

A question for the IDL gurus:

- I have data in the form of irregular polygons, each polygon has an associated value (let's call it an amplitude). I want a composite image of the sum of all these polygons' amplitudes.

- My current approach is to :

- (1) create a (large) WINDOW,.../PIXMAP,
- (2) render each polygon using POLYFILL
- (3) TVRD() the pixmap window
- (4) add this to an accumulation array

... iterate (1)-(4) until all polygons have been rendered

This works, but is painfully slow. Profiling the code shows that by far the most significant logjam is the TVRD() of the pixmap.

So: does anyone know of more efficient ways to do this? Is the Z device an option - it seems like it can be used for internal frame storage, but would still have to be probed by TVRD()...?

Thanks,

Dennis

--

/ Dennis J. Boccippio /
 / <http://fly.hiwaay.net/~djboccip/Dennis.html> /

Posted by [wrb1000](#) on Fri, 14 Jul 2000 07:00:00 GMT

[View Forum Message](#) <> [Reply to Message](#)

Hi Dennis,

I actually encountered a different problem and am currently using the solution that you hinted at. I read Randall's hint and I don't think it applies to me as I am summing multiple samples of a waveform to create a color-coded plot of a waveform. Here's a cut-and-paste of some of the code I used. It's highly abbreviated, but will give you an

idea of have the Zbuffer works and will let you develop your own test case. It aint super speedy (uses TVRD), but it works just fine.

```
intensity_array = uintarr(540, 459) ; image array
current_clip = !P.CLIP ; Copy current clipping boundaries
```

```
set_plot, 'z'
DEVICE, Z_BUFFERING = 0
device, set_resolution = [540,459]
!P.CLIP = current_clip ; Make Z-buffer clip same boundaries
```

```
; Setup new color table for Z-buffer image
table = intarr(256)
table[1] = 255
tvlct, table, table, table
```

```
FOR i = <1st plot>, <2nd plot>, incr DO BEGIN
  plots, x_data, y_data, color = 1
  intensity_array = temporary(intensity_array) + tvrd()
ENDFOR
```

```
device, /close
set_plot, 'win'
```

The 'intensity_array' now contains your summed data.
Hope this helps,

Bill B.

--

"They don't think it be like it is, but it do."

Oscar Gamble, NY Yankees

Sent via Deja.com <http://www.deja.com/>
Before you buy.

Subject: Re: optimization question: a faster way to PIXMAP?
Posted by [Dennis J. Boccippio](#) on Sat, 15 Jul 2000 07:00:00 GMT
[View Forum Message](#) <> [Reply to Message](#)

Thanks to both Randall and Bill for the tips...

I've found a temporary workaround which is only enabled by the fact that my polygons are much smaller than the summation grid ... I allocate much

smaller drawing windows, which tremendously speeds up TVRD(), and accumulate them into the appropriate summation grid subarrays. However, this is obviously case-specific, and doesn't solve the general problem of full-image accumulation. (Indeed, once this kludge is implemented, the initial PLOT used to set up each temporary frame's coordinate bounds becomes the bottleneck... it seems the graphics functions are just [relatively] slow).

Non-graphics and POLYFILLV sounds promising... will check that shortly.

Bill: I've benched your suggested code using both PIXMAP and the Z-buffer. The Z-buffer (at least on a Mac) seems to win out significantly:

	Z_buf	PIXMAP
	-----	-----
main	95.72	151.45
tvr	17.04	38.45
plots	14.07	49.95
randomu	1.38	1.34
sin	1.35	1.37
findgen	0.38	0.59

Surprising ... I'm curious how the guts of drawing to the Z-buf are different from the guts of drawing to a PIXMAP...

- Dennis

Test code below:

```

pro testzbuf

intensity_array = uintarr(540, 459) ; image array
current_clip = !P.CLIP ; Copy current clipping boundaries

set_plot, 'z'
DEVICE, Z_BUFFERING = 0
device, set_resolution = [540,459]
!P.CLIP = current_clip ; Make Z-buffer clip same boundaries

; Setup new color table for Z-buffer image
table = intarr(256)
table[1] = 255
tvlct, table, table, table

plot, 1!*pi*findgen(1000)/1000, sin(4!*pi*findgen(1000)/1000) + $
    randomu(seed, 1000), color=1, /nodata

```

```

FOR i = 0, 4000, 1 DO BEGIN
    plots,1*!pi*findgen(1000)/1000,sin(4*!pi*findgen(1000)/1000) + $
        randomu(seed,1000),color=1
    intensity_array = temporary(intensity_array) + tvrd()
ENDFOR

device, /close
set_plot, 'mac'

end

pro testpixmap

    set_plot,'mac'
    intensity_array = uintarr(540, 459) ; image array
    window,0,xsize=540,ysize=459,/pixmap
    plot,1*!pi*findgen(1000)/1000,sin(4*!pi*findgen(1000)/1000) + $
        randomu(seed,1000),color=1,/nodata
    table = intarr(256)
    table[1] = 255
    tvlct, table, table, table

    FOR i = 0, 4000, 1 DO BEGIN
        plots,1*!pi*findgen(1000)/1000,sin(4*!pi*findgen(1000)/1000) + $
            randomu(seed,1000),color=1
        intensity_array = temporary(intensity_array) + tvrd()
    ENDFOR

    set_plot, 'mac'

end

```

Subject: Re: optimization question: a faster way to PIXMAP?
 Posted by [Dennis Boccippio](#) on Mon, 17 Jul 2000 07:00:00 GMT
[View Forum Message](#) <> [Reply to Message](#)

In my actual (polygon-based) application, using the Z-buffer improved significantly over the pixmap. Now that I've got a reasonably-working algorithm, I'll experiment with POLYFILLV and post the results...

FWIW, I also found that using iterative calls to PLOT,/NODATA to set my PIXMAP or Z-buffer coordinate bounds used a LOT of overhead. Directly setting !P.S and !X.RANGE, !Y.RANGE turned out (not surprisingly) to be much more efficient...

Bless the 5.3 code profiler functionality!!! Between that and the project manager, it's almost like using CodeWarrior...

DJB

In article <8kv3kg\$nnu\$1@nnrp1.deja.com>, wrb1000@my-deja.com wrote:

> Dennis,
>
> Guessing - the pixmap function interacts with the video card.
> Utilizing the Z-buffer, the process is probably just a local memory
> allocation/deallocation exercise. Curious to learn the results of the
> POLYFILLV exercise.
>
> Bill B.
>
> --
> "They don't think it be like it is, but it do."
>
> Oscar Gamble, NY Yankees
>
>
> Sent via Deja.com <http://www.deja.com/>
> Before you buy

Subject: Re: optimization question: a faster way to PIXMAP?
Posted by [wrb1000](#) on Mon, 17 Jul 2000 07:00:00 GMT
[View Forum Message](#) <> [Reply to Message](#)

Dennis,

Guessing - the pixmap function interacts with the video card.
Utilizing the Z-buffer, the process is probably just a local memory
allocation/deallocation exercise. Curious to learn the results of the
POLYFILLV exercise.

Bill B.

--
"They don't think it be like it is, but it do."

Oscar Gamble, NY Yankees

Sent via Deja.com <http://www.deja.com/>
Before you buy.

Subject: Re: optimization question: a faster way to PIXMAP?

Posted by [Struan Gray](#) on Tue, 18 Jul 2000 07:00:00 GMT

[View Forum Message](#) <> [Reply to Message](#)

Dennis Boccippio, djboccip@hotmail.com writes:

- > In my actual (polygon-based) application, using
- > the Z-buffer improved significantly over the pixmap.
- > Now that I've got a reasonably-working algorithm,
- > I'll experiment with POLYFILLV and post the results...

This is just idle musing on my part, but have you tried object graphics? Plotting semi-transparent polygons successively offset along the Z-axis towards the viewer should built up density in the right way. You only need to read the pixmap at the end and you can take advantage of the fact that OpenGL is optimised for fast polygon plotting (and often accelerated to boot).

Struan
