Subject: Re: Huge Maps & a device for faking a large window
Posted by dmarino on Thu, 01 Apr 2004 21:13:12 GMT
View Forum Message <> Reply to Message

<disclaimer>
Hope I'm not throwing anyone off track with this suggestion.....
I think last time I posted (years ago) I confused a dude who was justifiably angry. Grain of salt suggested, but not included :-)
</disclaimer>

anyway...
I do this stuff with fairly large (30000x30000 and larger) Quickbird images often.

I like to set up an affine transformation and use that to convert pixel/line into projected coords. No display window, no big memory use. No map_set, nothing.

I have a small C program I spwan to that takes imgx1, imgy1, mapx1, mapy1...etc for input gives you back the 6 params for the affine transform. IP rules say I can't share :(
My bet is this is super trivial for you, anyway.

I make a 6 param affine struct like so (kinda like .tfw file, well exactly like one....)

You'll have to know the projected corners locations of your tile or big image in advance to do this, so it's kind of limited...

```
aff = {
       A: affine param 1
       B: param 2
       C: param 3
       D: param 4
       E: param 5
       F: param 6
      }
```

Then I wrote a routine to do a forward transformation, and a reverse.

So you can call projected = DM_forward_affine(aff,img_x,img_y) which returns a two element result containing the projected coords.

Also works for imagecoords = DM_reverse_affine(aff, map_x, map_y), returns the image coords.

Biggest memory use is the struct.

Would that approach help?

Sorry If I'm off-base here..... I am not an expert anything.

Donnie

_____

JD Smith <jdsmith@as.arizona.edu> wrote in message
news:<pan.2004.03.31.23.38.01.813380@as.arizona.edu>...
> On Wed, 31 Mar 2004 15:59:26 -0600, Liam Gumley wrote:
>
>> The Z-buffer offers a convenient way to define map projections without
>> needing a X display, e.g.
>>
>> xsize = 43200 ; width of window
>> ysize = 21600 ; height of window
>> res = 1.0 ; Map resolution in kilometers
>> set_plot, 'Z'
>> device, set_resolution=[xsize, ysize], set_colors=256, z_buffering=0, $
>> set_character_size=[10, 12]
>> scale = res * 4.0e6
>> map_set, latcen, loncen, scale=(scale * (!d.x_px_cm / 40.0)), /lambert, $
>> position=[0, 0, 1, 1], /noerase
>>
>> The scale transformation is to account for direct graphics devices which
>> don't have the same number of pixels per centimeter as the default X device.
>>
>> That said, I've also had good luck with Xvfb.
>
>
> Thanks Liam. I had already ruled out the Z-buffer, since it allocates
> lots of memory for the display device, e.g.:
>
> IDL> help,/memory
> heap memory used: 960599, max: 1014946, gets: 7261, frees: 6933
> IDL> set_plot,'Z'
> IDL> help,/memory
> heap memory used: 960633, max: 960652, gets: 7264, frees: 6934
> IDL> device,SET_RESOLUTION=[43200,21600],Z_BUFFERING=0
> IDL> help,/memory
> heap memory used: 934080780, max: 934080810, gets: 7271, frees: 6938
>
> I fear Xvfb will just shift the memory usage outside of IDL to another
> process. Unfortunately, I don't have the memory to spare, since I
> need it to manipulate multiple ~1/4 GB tiled images. I had not heard
> of the MAP_PROJ_* functions that Ben mentioned: I'll give those a look
> (though it appears I'll essentially have to redo what MAP_IMAGE does

> using MAP_PROJ_FORWARD instead of CONVERT_COORD).
>
> JD
