Subject: Re: Huge Maps & a device for faking a large window
Posted by btt on Wed, 31 Mar 2004 20:46:33 GMT
View Forum Message <> Reply to Message

JD Smith wrote:
>

> a) Does anyone know of a way to access the mapping transformations
> directly (aside from re-coding them yourself), independent of any
> particular window geometry? Why shouldn't I be able to perform an
> arbitrary coordinate transformation using one of the many mapping
> transforms MAP_SET offers? Coupling this to a specific display
> device size is an unnecessary limitation.

Hi JD,

I think the MAP_PROJ_XXXX routines introduced in v 5.6 are supposed to let you configure !MAP and make map data transformations without rendering to an output window.

Ben

Subject: Re: Huge Maps & a device for faking a large window
Posted by Michael Wallace on Wed, 31 Mar 2004 21:09:59 GMT
View Forum Message <> Reply to Message

Here's a trick I've used before on *nix. If you're on Windows, you can stop reading right here. There's a program called Xvfb which emulates a dumb framebuffer using virtual memory. If it's not already installed on your machine, there are plenty of places across the internet where you can download it.

Once you have Xvfb, you can create a framebuffer with any dimension or color depth you want. Then, make your IDL code use this new display instead your actual X window session. IDL thinks that there is a valid display available, however you do not see a thing. You can then create a super-huge IDL window and do what you need to do with your map. However, I don't know how Xvfb manages memory or how efficient it is. This might not be a viable solution, but I don't know since I've never used it in this way.

Even if this doesn't work for you, there are some good uses of the "Xvfb trick," especially for automation. There have been times I wanted to automate a task, but the program insisted on having a display even though it didn't really need one. There have been other times when I've wanted to automate programs that used the 'X' device. Instead of

changing code around to use the Z buffer and remove all of the references to windows devices, I could just create a virtual framebuffer and send all IDL graphical output to it. Now I can run those programs even if no "real" display is available without changing a line of code.
I have a server that doesn't even have a monitor attached and I can run graphical IDL code with no problem (assuming that there doesn't need to be any graphical interaction with the program).

-Mike

Subject: Re: Huge Maps & a device for faking a large window
Posted by Liam Gumley on Wed, 31 Mar 2004 21:59:26 GMT
View Forum Message <> Reply to Message

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.

Cheers,
Liam.
Practical IDL Programming
http://www.gumley.com/

"JD Smith" <jdsmith@as.arizona.edu> wrote in message
news:pan.2004.03.31.20.02.23.598950@as.arizona.edu...
>
>
> Any makers of large map projection images here? I'm having a
> conceptual problem creating a very large (~1Gpix) projected image. I
> bin a large data set into small bins tiling the entire range of
> latitude and longitude (43200x21600). I do this in a series of
> "tiles" to avoid working with the entire data set at once. So far so
> good. If I then want to warp this image to a given projection (like

> Aitoff), it seems I must first use MAP_SET to specify the projection
> details, *and* have a window open of the desired output size. The
> problem is, I have no intention of actually displaying the projected
> image (too large!), so all of the memory allocated for creating that
> big window is wasted (which is more than a nuisance when building such
> huge images).
>
> Unfortunately, MAP_PATCH (possibly via an undocumented keyword to
> TRIGRID -- MAP) and MAP_IMAGE (via CONVERT_COORDS) rely on a presently
> set window to dictate the size of the projected image. If
> TRIGRID(MAP=) were documented, perhaps I could do this myself, but it
> seems likely it also internally consults the current window to set the
> size for the coordinate transform. I see two ways out:
>
> a) Does anyone know of a way to access the mapping transformations
> directly (aside from re-coding them yourself), independent of any
> particular window geometry? Why shouldn't I be able to perform an
> arbitrary coordinate transformation using one of the many mapping
> transforms MAP_SET offers? Coupling this to a specific display
> device size is an unnecessary limitation.
>
> b) Barring this, is there a device in which a window can be
> established which does not consume any memory or accept display
> commands, but simply provides a dummy framework from which
> CONVERT_COORDS etc. can take window info?
>
> Thanks,
>
> JD
>

Subject: Re: Huge Maps & a device for faking a large window
Posted by JD Smith on Wed, 31 Mar 2004 23:38:02 GMT
View Forum Message <> Reply to Message

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

Subject: Re: Huge Maps & a device for faking a large window
Posted by Jack Saba on Thu, 01 Apr 2004 13:55:13 GMT
View Forum Message <> Reply to Message

JD Smith wrote:

>
> Any makers of large map projection images here? I'm having a
> conceptual problem creating a very large (~1Gpix) projected image. I
> bin a large data set into small bins tiling the entire range of
> latitude and longitude (43200x21600). I do this in a series of
> "tiles" to avoid working with the entire data set at once. So far so
> good. If I then want to warp this image to a given projection (like
> Aitoff), it seems I must first use MAP_SET to specify the projection
> details, *and* have a window open of the desired output size. The
> problem is, I have no intention of actually displaying the projected
> image (too large!), so all of the memory allocated for creating that

> big window is wasted (which is more than a nuisance when building such
> huge images).
>
> Unfortunately, MAP_PATCH (possibly via an undocumented keyword to
> TRIGRID -- MAP) and MAP_IMAGE (via CONVERT_COORDS) rely on a presently
> set window to dictate the size of the projected image. If
> TRIGRID(MAP=) were documented, perhaps I could do this myself, but it
> seems likely it also internally consults the current window to set the
> size for the coordinate transform. I see two ways out:
>
> a) Does anyone know of a way to access the mapping transformations
> directly (aside from re-coding them yourself), independent of any
> particular window geometry? Why shouldn't I be able to perform an
> arbitrary coordinate transformation using one of the many mapping
> transforms MAP_SET offers? Coupling this to a specific display
> device size is an unnecessary limitation.
>
> b) Barring this, is there a device in which a window can be
> established which does not consume any memory or accept display
> commands, but simply provides a dummy framework from which
> CONVERT_COORDS etc. can take window info?
>
> Thanks,
>
> JD
>

If the problem is space, what about postscript mode with filename='/dev/null'?
Assuming you are on a unix system, of course.

Jack Saba
--
jack / saba at gsfc / nasa / gov

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
