Subject: Re: Reverse engineering the new graphics PLOT() margin property?
Posted by Jim Pendleton on Fri, 13 Apr 2012 20:46:46 GMT
View Forum Message <> Reply to Message

On Thursday, April 12, 2012 11:37:45 AM UTC-6, Sean Davis wrote:
> I was wondering if anyone has figured out how the default margins are determined in new graphics?
>
> On a related note, it is infuriating that there is no way to return the values of INIT properties like margin, position, etc. (unless I'm missing something!)
>
> What I'd really like to be able to do is the following --
>
> p = plot(findgen(20))
>
> print, p.margin

If you are fearless, don't mind rolling up your sleeves and perhaps your pantlegs, and enjoy the thought of diving deep (well, just 6 routines deep) into the "internal" .pro routines, it's pretty straightforward to answer the question "How?"

If you have none of these qualities, remain ashore and read no further. Contact support@exelisvis.com with a feature request, and return to the safety of your home.

There's no reverse engineering since the .pro code is exposed. But there is hacking.

You have set sail to Terra Incognita, where a secret dialect is fluently spoken only by the natives known as The Second Floor*. Here there be undoc'ed IDL.

As a shortcut to getting to the crux of your question, execute the following slightly illegal statement from the Workbench

IDL> p = plot(/test, margin=[0,0,0], /debug)

This will cause IDL to stop on an error
% MARGIN must have 1 or 4 elements.

All the display geometry calculation is in an editor window showing the routine in which you've stopped.

Set a breakpoint in the method, recompile, RETALL, then rerun with some valid input.

If you have the mad IDL debugging skilz, you'll see that the MARGIN value is likely discarded when other values are derived from it. By itself it never appears to be stored as a property of your plot object, such as in an obvious member variable.

If you back your way slowly and carefully out of the call stack (Show no fear! New Graphics can smell fear!), you may be able to discern how the derived values are used in subsequent

calculations.

Of course the harder questions to answer are, "How do I change this to do what I want?" and the more generic "Why was it implemented this way?" These are exercises left for the reader.

Jim P.

*I'm on the third floor. We don't get capitalization up here.

Subject: Re: Reverse engineering the new graphics PLOT() margin property?
Posted by David Fanning on Fri, 13 Apr 2012 21:00:51 GMT
View Forum Message <> Reply to Message

jimmyleependleguy@gmail.com writes:

> If you are fearless, don't mind rolling up your
> sleeves and perhaps your pantlegs, and enjoy the
> thought of diving deep (well, just 6 routines deep)
> into the "internal" .pro routines, it's pretty
> straightforward to answer the question "How?"

Thanks for explaining the innards of the function
graphics system. It's the first time I've ever heard
it explained with good humor instead of foul language,
and I can't tell you how grateful I am for that!

This may go down as one of those "classic" posts
of the IDL newsgroup. I hope so. :-)

Cheers,

David
--
David Fanning, Ph.D.
Fanning Software Consulting, Inc.
Coyote's Guide to IDL Programming: http://www.idlcoyote.com/
Sepore ma de ni thui. ("Perhaps thou speakest truth.")

Subject: Re: Reverse engineering the new graphics PLOT() margin property?
Posted by David Fanning on Fri, 13 Apr 2012 21:10:19 GMT
View Forum Message <> Reply to Message

jimmyleependleguy@gmail.com writes:

> Set a breakpoint in the method, recompile, RETALL, then rerun with some valid input.

By the way, I just point out that in Windows 7, you can
set a breakpoint, recompile, etc., etc., but you will
never stop at the breakpoint. The Windows 7 "permissions"
don't allow you to write in the ITTVIS directories.
(Although you will get no warning that this is the case
or that anything is amiss.)

To set a breakpoint in an IDL-supplied program, you have
to copy the file to a directory you own and modify it
there. It's just another way saying, "Don't do this!"
without being too obvious about it. :-)

Cheers,

David

--
David Fanning, Ph.D.
Fanning Software Consulting, Inc.
Coyote's Guide to IDL Programming: http://www.idlcoyote.com/
Sepore ma de ni thui. ("Perhaps thou speakest truth.")

Subject: Re: Reverse engineering the new graphics PLOT() margin property?
Posted by lecacheux.alain on Sun, 15 Apr 2012 11:15:36 GMT
View Forum Message <> Reply to Message

On 13 avr, 23:10, David Fanning <n...@idlcoyote.com> wrote:
> jimmyleependle...@gmail.com writes:
>> Set a breakpoint in the method, recompile, RETALL, then rerun with some valid input.
>
> By the way, I just point out that in Windows 7, you can
> set a breakpoint, recompile, etc., etc., but you will
> never stop at the breakpoint. The Windows 7 "permissions"
> don't allow you to write in the ITTVIS directories.
> (Although you will get no warning that this is the case
> or that anything is amiss.)
>
> To set a breakpoint in an IDL-supplied program, you have
> to copy the file to a directory you own and modify it
> there. It's just another way saying, "Don't do this!"
> without being too obvious about it. :-)
>
> Cheers,
>

> David
>
> --
> David Fanning, Ph.D.
> Fanning Software Consulting, Inc.
> Coyote's Guide to IDL Programming:http://www.idlcoyote.com/
> Sepore ma de ni thui. ("Perhaps thou speakest truth.")
>
>

Debugging is not mandatory for trying to understand "graphics", nor "itools" from which graphics is derived. Studying the *.pro files as distributed in \lib\graphics and \lib\itools directories would be enough, in principle. I acknowledge Exelis to have made this software entirely open.
But such a hacking task cannot be reasonably requested from a "normal" (and paying) user. Exelis should have produced a decent documentation too, with a precise description of the adopted conventions as well as the used principles. Only such a documentation would make any user really able to understand and efficiently use by himself the rich capabilities offered by the IDL object graphics. This lack of documentation is a pity: while IDL might be one of the best existing multipurpose scientific software, there is the risk that a growing number of IDL users only "sub-use" IDL (and then question the licence cost) or, even, leave IDL completely out (e.g. the "python" effect, etc...).
alx.

Subject: Re: Reverse engineering the new graphics PLOT() margin property?
Posted by David Fanning on Sun, 15 Apr 2012 14:00:20 GMT
View Forum Message <> Reply to Message

alx writes:

> Exelis should have produced a decent
> documentation too, with a precise description of the adopted
> conventions as well as the used principles. Only such a documentation
> would make any user really able to understand and efficiently use by
> himself the rich capabilities offered by the IDL object graphics.

I don't have any special knowledge about this, but
my guess is that the people who have enough knowledge
about this system to document it properly are too busy
trying to make it work correctly to write about it.
Maybe a couple of years from now there will be time
to do a better job of it.

The real problem with such a complex system, it seems
to me, is that "fixing" a problem often puts
two more problems into the work queue. It's no wonder
it can't be documented. It would be a thankless
and never-ending job! I can't imagine anyone signing
up for it. (Although, as Coyote reminds me, he has
done these kinds of jobs for years as our inadequately
paid webmaster, and, except for a persistent drinking
problem, he is doing great!)

Cheers,

David

--
David Fanning, Ph.D.
Fanning Software Consulting, Inc.
Coyote's Guide to IDL Programming: http://www.idlcoyote.com/
Sepore ma de ni thui. ("Perhaps thou speakest truth.")

Subject: Re: Reverse engineering the new graphics PLOT() margin property?
Posted by lecacheux.alain on Sun, 15 Apr 2012 17:49:37 GMT
View Forum Message <> Reply to Message

> David Fanning writes:
> I don't have any special knowledge about this, but
> my guess is that the people who have enough knowledge
> about this system to document it properly are too busy
> trying to make it work correctly to write about it.
> Maybe a couple of years from now there will be time
> to do a better job of it.
>
What you guess makes sense.
But I disagree with the view that the new graphics system would be too
much complex to work. I am now using it extensively, in real
scientific work, since several months. I learned a lot! I can quickly
produce figures of publishable quality (faster and better than by
using direct graphics); widget programming is oversimplified by using
"widget_window"; I can easily build and use movies when I want to do
some animated presentations; 3-D plotting is no longer a technical
difficulty; and many other things that I did not discover yet.
Of course, there are still some hard limitations, too much frequently
encountered and really frustrating. But most often, when I get a
problem with new graphics, I have no way to decide whether the problem
comes from an actual bug or because I am misusing it ! An usable
documentation would fix that and, by the way, would certainly help
Exelis in getting more useful returns from users.

alx.

Subject: Re: Reverse engineering the new graphics PLOT() margin property?
Posted by David Fanning on Sun, 15 Apr 2012 19:15:08 GMT
View Forum Message <> Reply to Message

alx writes:

> What you guess makes sense.
> But I disagree with the view that the new graphics system would be too
> much complex to work. I am now using it extensively, in real
> scientific work, since several months. I learned a lot! I can quickly
> produce figures of publishable quality (faster and better than by
> using direct graphics); widget programming is oversimplified by using
> "widget_window"; I can easily build and use movies when I want to do
> some animated presentations; 3-D plotting is no longer a technical
> difficulty; and many other things that I did not discover yet.
> Of course, there are still some hard limitations, too much frequently
> encountered and really frustrating. But most often, when I get a
> problem with new graphics, I have no way to decide whether the problem
> comes from an actual bug or because I am misusing it ! An usable
> documentation would fix that and, by the way, would certainly help
> Exelis in getting more useful returns from users.

Well, Alain, why don't *you* write about it! :-)

You are probably one of the few users who understand it.
I'm happy to give you free reign to write as many articles
as you like and post them on my web page. I'd like to have
some positive articles about function graphics to balance
my own experience with them. (They don't lend themselves
to the type of scientific work I've been doing lately,
I guess.)

This offer goes to anyone who would like to write an article.
You can find an article template here:

http://www.idlcoyote.com/template.php

Cheers,

David

--
David Fanning, Ph.D.
Fanning Software Consulting, Inc.

Coyote's Guide to IDL Programming: http://www.idlcoyote.com/
Sepore ma de ni thui. ("Perhaps thou speakest truth.")

Subject: Re: Reverse engineering the new graphics PLOT() margin property?
Posted by Mark Piper on Tue, 17 Apr 2012 01:54:42 GMT
View Forum Message <> Reply to Message

On Thursday, April 12, 2012 11:37:45 AM UTC-6, Sean Davis wrote:
> I was wondering if anyone has figured out how the default margins are determined in new graphics?
>
> On a related note, it is infuriating that there is no way to return the values of INIT properties like margin, position, etc. (unless I'm missing something!)
>
> What I'd really like to be able to do is the following --
>
> p = plot(findgen(20))
>
> print, p.margin

Hi Sean,

I agree, we should be able to get these plot properties that we can set on init. I'll enter a bug report.

In the meantime, I think this works (though I haven't tested it thoroughly):

```
IDL> p = plot(findgen(10)^2)
IDL> r = transpose([[p.xrange], [p.yrange]])
IDL> m = p.convertcoord(r, /data, /to_normal)
IDL> print, m
     0.13000000      0.13000000      0.00000000
     0.92000000      0.89000000      0.00000000
```

Check:

```
IDL> print, p.convertcoord(m, /normal, /to_data)
  8.8817842e-016      0.00000000      0.00000000
       10.000000       100.00000      0.00000000
```

mp

Subject: Re: Reverse engineering the new graphics PLOT() margin property?
Posted by Sean[1] on Wed, 18 Apr 2012 21:15:40 GMT
View Forum Message <> Reply to Message

On Monday, 16 April 2012 19:54:42 UTC-6, Mark Piper wrote:
> On Thursday, April 12, 2012 11:37:45 AM UTC-6, Sean Davis wrote:
>> I was wondering if anyone has figured out how the default margins are determined in new graphics?
>>
>> On a related note, it is infuriating that there is no way to return the values of INIT properties like margin, position, etc. (unless I'm missing something!)
>>
>> What I'd really like to be able to do is the following --
>>
>> p = plot(findgen(20))
>>
>> print, p.margin
>
> Hi Sean,
>
> I agree, we should be able to get these plot properties that we can set on init. I'll enter a bug report.
>
> In the meantime, I think this works (though I haven't tested it thoroughly):
>
> IDL> p = plot(findgen(10)^2)
> IDL> r = transpose([[p.xrange], [p.yrange]])
> IDL> m = p.convertcoord(r, /data, /to_normal)
> IDL> print, m
>   0.13000000   0.13000000   0.00000000
>   0.92000000   0.89000000   0.00000000
>
> Check:
>
> IDL> print, p.convertcoord(m, /normal, /to_data)
>   8.8817842e-016   0.00000000   0.00000000
>   10.000000   100.00000   0.00000000
>
> mp

Mark,

Yup, that's the solution I came up with, but there are two complications that are currently undocumented (and will hopefully be fixed in 8.2!!!)

1. For log, axes, data coordinates are returned as the log of the value

```
;make a plot with a logarithmic y axis
IDL> p = plot(findgen(10), yrange=[1e3,10.], ylog=1)
% Loaded DLM: XML.
;try to return the normal coordinates of the lower left corner
IDL> print, p.convertcoord(0,1e3, /data,/to_normal)
```

```
      0.13000000      -378.73000       0.0000000
;yikes, the y-value of the lower corner doesn't look right -- try feeding in the log10 of the lower
corner
IDL> print, p.convertcoord(0,alog10(1e3), /data,/to_normal)
      0.13000000       0.13000000       0.0000000
;that looks better!
```

2. For multiplots using the LAYOUT keyword, the margin values are not normal relative to the whole window, but just to the fraction of the window that is taken up by the given plot. This is kind of hard to explain, but here's an example:

```
IDL> p = plot(findgen(10), yrange=[1e3,10.], ylog=1, layout=[2,2,1],title='Margins not specified')


;Now get the lower left and upper  right corners of the plot box
IDL> print, p.convertcoord(0,3, /data,/to_normal)
     0.085000000       0.57000000       0.0000000
IDL> print, p.convertcoord(10,1, /data,/to_normal)
      0.46000000       0.94500000       0.0000000

;Try to reproduce the first plot by feeding in the margins that are implied (assuming the layout
keyword allots a box going from 0.-0.5 in x and 0.5-1.0 in y (normal coordinates)
IDL> p2 = plot(findgen(10), yrange=[1e3,10.], ylog=1, layout=[2,2,1],title='Margin =
[.085,.07,.04,.055]',margin=[.085,.07,.04,.055])
;This doesn't work! Why? Because apparently the margins are interpreted as being normal
coordinates relative to the plot region allotted by the layout keyword!

;Instead, since we know that the 2 x 2 layout gives a plot region that is half the size of the full
window in both x- and y-directions, we multiply the margin by 2
IDL> p3 = plot(findgen(10), yrange=[1e3,10.], ylog=1, layout=[2,2,1],title='Margin =
[.085,.07,.04,.055]*2',margin=[.085,.07,.04,.055]*2)
;Now we've created a plot that is the same as the original plot!
```

Sean