Subject: 8-bit vs. 24-bit color on Windows
Posted by thompson on Fri, 22 Jan 1999 08:00:00 GMT
View Forum Message <> Reply to Message

I know this question has been asked many times before, but I'm afraid I don't remember what the answer is. Is there any way to convince IDL to use 8-bit pseudo-color on a Windows computer with a 16-bit or higher display? I know that in other operating systems this is done by using

DEVICE,PSEUDO_COLOR=8

However, this is not supported under Windows. I tried

DEVICE,DECOMPOSED=0

which the documentation claims will make routines work like they did before, but this doesn't appear to be the whole story. With DECOMPOSED=0, data will come up with the correct color, but only if they are displayed after the color table is loaded. This is unacceptable. There must be another step to convince IDL to use 8-bit pseudo-color, and if there isn't then RSI must address this.

I've checked David Fanning's Coyote Guide (http://www.dfanning.com/), but the only suggestion I could find there that meets my needs is to set Windows to run at 256 colors. I'm perfectly happy to do that, but it would be nice to be able to use 16-bit or 24-bit for those programs which need it, and 8-bit color for IDL. This is possible in other operating systems; can it be done in Windows?

William Thompson

Subject: Re: 8-bit vs. 24-bit color on Windows
Posted by thompson on Wed, 27 Jan 1999 08:00:00 GMT
View Forum Message <> Reply to Message

davidf@dfanning.com (David Fanning) writes:

> Oh, heck. I'm a sucker for a little newsgroup controversy
> every now and again....

> William Thompson (thompson@orpheus.nascom.nasa.gov) writes:

>> This seems to be a strictly widget-oriented solution. Not everything is
>> widgets!!!! Nor should it be--a lot of specialized data analysis is not done
>> in a widget environment. Quite a bit of it, in fact, is done directly from the
>> command line. Most scientific users who are writing programs their own use
>> don't bother to go to the trouble of writing widget programs.

> While I'm dubious about "most scientific users" working at the command

> line, I'll let that pass. But color table manipulation is NOT
> strictly a widget-oriented solution. It is simply easier to
> implement with widgets. (As are most user-interactive type of
> programs. After all, that is the *point* of a widget program.)

> XCOLORS, for example, works equally well with any object
> that has a "DRAW" method. Give XCOLORS a structure that has
> an object reference, the name of the "DRAW" method to call,
> and the window you want to draw into (even a regular old
> graphics window) and without doing much else you have a window
> that can update itself when the colors in the color table change.

> But if Bill's right, most scientific programmers aren't
> using XCOLORS or XLOADCT anyway. They are building their
> own color table vectors and loading them with TVLCT the
> way God intended. What help can we offer to them?

I didn't say that people weren't using XLOADCT. What I'm saying is that a lot of work (myself included, and the people I know around me) is done using non-widget software. The first thing you do, when you get some data that you don't know what it's going to look like, is you read it and display it. For example, I might say

```
 IDL> fxread, 'mydata.fits', array, header ;Generic FITS reader
 IDL> tvscl, array
```

Then I want to start playing with the color table, possibly using just LOADCT, possibly with XLOADCT, or possibly with some home-grown equivalent. It's EXTREMELY frustating if the colors don't change automatically.

To you, maybe that's a minor annoyance which you're willing to put up with for perceived advantages of a more complicated 3-plane color system. For you that's great, and IDL should be able to provide that for you. However, for me, a color-table methodology is much simpler and more appropriate. I argue that IDL should be taking me into account too, and all those more basic users who don't subscribe to this newsgroup.

>> One thing I don't understand is why the graphic needs to be regenerated.
>> Couldn't one just read in the byte values from all active windows and write
>> them back out again?

> Yes. If the byte value represented the color of the pixel,
> which it simply does not in a 24-bit true-color system
> (*any* 24-bit true-color system).

But when I say "tvscl, array", that's what I'm assuming is happening to the image. I'm not working with separate red, green, and blue color values, I'm working with arrays of numbers which I want to appear in a color scheme that

can be manipulated to best show what's in the data.

> I don't honestly know. Perhaps RSI *could* hack a PSUEDOCOLOR
> like thing together for the Windows platform. But it sure as
> hell would be non-standard Windows programming and a gigantic
> pain in the neck to maintain, I suspect.

> More to the point, would I want them spending their resources
> doing this, so that we can continue to work with the old
> tired 8-bit strategy far into the future, or do I want them
> adding new features to the language that will allow us to
> write better programs for better and faster computers now?
> Personally, I vote for better programs going forward. And
> which strategy do you think is more likely to result in
> RSI even being around in the future?

Well I don't think this is going to break the back of RSI, and I certainly vote for a pseudo-color capability on Windows just like the other platforms. Almost everybody I know of with a 24-bit graphics cards on a Unix box opts for pseudo-color in IDL. I'm not saying that pseudo-color is more important than 24-bit color, but don't tell me it's less important either!

>> One thing I don't understand is why the graphic needs to be regenerated.
>> Couldn't one just read in the byte values from all active windows and write
>> them back out again?

> Yes. If the byte value represented the color of the pixel,
> which it simply does not in a 24-bit true-color system
> (*any* 24-bit true-color system).

Which is precisely why 24-bit color is more confusing to deal with when you're dealing with what is essentially grey-scale data. It should be simple for RSI to maintain in memory an 8-bit image which then gets translated through the color table into 24-bits when displayed. In pseudo-color mode, any changes to the color table could retrigger this translation automatically, rather than making us have to do it ourselves.

> I hate to tell you this, Bill. But the problem here is
> that RSI is hacking operating system code to make it
> *look* like another operating system, all in the name
> of cross-platform compatibility. If I recall correctly,
> that is the solution you are proposing here for yet one
> more operating system difference. I, for one, could do without
> any more of these kinds of "solutions". :-)

Without interplatform compatibility, IDL would be a useless product for us. Give that up, and we'd just give up on IDL and use something else instead.

Subject: Re: 8-bit vs. 24-bit color on Windows
Posted by davidf on Wed, 27 Jan 1999 08:00:00 GMT
View Forum Message <> Reply to Message

Oh, heck. I'm a sucker for a little newsgroup controversy
every now and again....

William Thompson (thompson@orpheus.nascom.nasa.gov) writes:

> This seems to be a strictly widget-oriented solution. Not everything is
> widgets!!!! Nor should it be--a lot of specialized data analysis is not done
> in a widget environment. Quite a bit of it, in fact, is done directly from the
> command line. Most scientific users who are writing programs their own use
> don't bother to go to the trouble of writing widget programs.

While I'm dubious about "most scientific users" working at the command
line, I'll let that pass. But color table manipulation is NOT
strictly a widget-oriented solution. It is simply easier to
implement with widgets. (As are most user-interactive type of
programs. After all, that is the *point* of a widget program.)

XCOLORS, for example, works equally well with any object
that has a "DRAW" method. Give XCOLORS a structure that has
an object reference, the name of the "DRAW" method to call,
and the window you want to draw into (even a regular old
graphics window) and without doing much else you have a window
that can update itself when the colors in the color table change.

But if Bill's right, most scientific programmers aren't
using XCOLORS or XLOADCT anyway. They are building their
own color table vectors and loading them with TVLCT the
way God intended. What help can we offer to them?

I don't honestly know. Perhaps RSI *could* hack a PSUEDOCOLOR
like thing together for the Windows platform. But it sure as
hell would be non-standard Windows programming and a gigantic
pain in the neck to maintain, I suspect.

More to the point, would I want them spending their resources
doing this, so that we can continue to work with the old
tired 8-bit strategy far into the future, or do I want them
adding new features to the language that will allow us to
write better programs for better and faster computers now?
Personally, I vote for better programs going forward. And
which strategy do you think is more likely to result in

RSI even being around in the future?

> One thing I don't understand is why the graphic needs to be regenerated.
> Couldn't one just read in the byte values from all active windows and write
> them back out again?

Yes. If the byte value represented the color of the pixel,
which it simply does not in a 24-bit true-color system
(*any* 24-bit true-color system).

> One thing that seems to be happening with IDL lately is that people are being
> expected to use more complicated programming techniques to solve problems that
> used to be much simpler.

I don't think this problem is specific to IDL. Have you
looked at any new word processing software that has come to
market in the last five years? What about just typing one
character after the other until you get to the end of the
line and hit a carriage return? Microsoft Word is so damn
complicated I haven't even worked up the courage to figure
it out. Squiggly red lines everywhere!

> The problem of displaying traditional pseudo-color
> images on the more advanced graphics cards is one such example (although only
> on Windows platforms). Another is the changes within modal widgets, where one
> is asked to use complicated things like timer events for things that just used
> to work without thinking about it.

I hate to tell you this, Bill. But the problem here is
that RSI is hacking operating system code to make it
look like another operating system, all in the name
of cross-platform compatibility. If I recall correctly,
that is the solution you are proposing here for yet one
more operating system difference. I, for one, could do without
any more of these kinds of "solutions". :-)

Cheers,

David

Subject: Re: 8-bit vs. 24-bit color on Windows
Posted by thompson on Wed, 27 Jan 1999 08:00:00 GMT
View Forum Message <> Reply to Message

davidf@dfanning.com (David Fanning) writes:

> Liam Gumley (Liam.Gumley@ssec.wisc.edu) writes:

>> When using IDL under Windows with a 24 bit display setting, the only way
>> around this problem is to re-display your graphic after changing the
>> color table. That's why David's XCOLORS program
>> (http://www.dfanning.com/programs/xcolors.pro) includes a keyword which
>> enables you to notify an external event handler that the color table has
>> changed.

> Note that XLOADCT now has a similar capability to call
> an IDL procedure and pass it some "data" when the color
> tables change. (Someone at RSI must *certainly* be
> reading this newsgroup! :-)

(rest deleted)

This seems to be a strictly widget-oriented solution. Not everything is widgets!!!! Nor should it be--a lot of specialized data analysis is not done in a widget environment. Quite a bit of it, in fact, is done directly from the command line. Most scientific users who are writing programs for their own use don't bother to go to the trouble of writing widget programs.

In any case, redisplaying a graphic is a really silly way to go. Some complicated graphics could take anywhere from seconds to minutes to display--that's a hell of a lot of time and computing power to waste on simply changing a color table.

If the problem on Windows is caused by the Windows environment itself, which I suspect is the case, I still think it should be possible for RSI to emulate the pseudo-color mode we're familiar with.

One thing I don't understand is why the graphic needs to be regenerated. Couldn't one just read in the byte values from all active windows and write them back out again? That wouldn't depend on the window being part of a widget program (or part of a different widget program than the one you told XLOADCT about, for instance), and could be implemented within any color manipulation program without requiring the complication that information about the running programs.

One thing that seems to be happening with IDL lately is that people are being expected to use more complicated programming techniques to solve problems that used to be much simpler. The problem of displaying traditional pseudo-color images on the more advanced graphics cards is one such example (although only on Windows platforms). Another is the changes within modal widgets, where one is asked to use complicated things like timer events for things that just used to work without thinking about it. It's good that IDL has these more sophisticated tools for those who can make use of them, but we must remember that IDL is supposed to be a tool for people who are not professional programmers. It's not supposed to be a environment for programmers to write

canned solutions for other people.

William Thompson

Subject: Re: 8-bit vs. 24-bit color on Windows
Posted by davidf on Thu, 28 Jan 1999 08:00:00 GMT
View Forum Message <> Reply to Message

Liam Gumley (Liam.Gumley@ssec.wisc.edu) writes:

>> Almost everybody I know of with a 24-bit graphics cards on a Unix box opts for
>> pseudo-color in IDL.
>
> As I said earlier, this is how everyone here works on Unix boxes.

All right. I concede I am a romantic when it comes to software.
How else would you dare write a book? :-)

Cheers,

David

Subject: Re: 8-bit vs. 24-bit color on Windows
Posted by Liam Gumley on Thu, 28 Jan 1999 08:00:00 GMT
View Forum Message <> Reply to Message

William Thompson wrote:
> I didn't say that people weren't using XLOADCT. What I'm saying is that a lot
> of work (myself included, and the people I know around me) is done using
> non-widget software. The first thing you do, when you get some data that you
> don't know what it's going to look like, is you read it and display it. For
> example, I might say
>
> IDL> fxread, 'mydata.fits', array, header ;Generic FITS reader
> IDL> tvscl, array
>
> Then I want to start playing with the color table, possibly using just LOADCT,
> possibly with XLOADCT, or possibly with some home-grown equivalent. It's
> EXTREMELY frustating if the colors don't change automatically.

I know many people in my own organization who do a lot of work in IDL in
just this fashion. Most people here have Unix consoles (8 bit, or 24 bit
in pseudo mode), and therefore have never seen the problem where you are
busy moving sliders around in XLOADCT, but nothing happens to the image
that is on-screen. There are a few people here who use PCs running

Xserver software to access Unix boxes. Some of them had 24 bit displays when their new PCs arrived, but as soon as they saw colr tables in IDL (and another in-house application) behaving oddly, they switched to 8 bit displays.

> But when I say "tvscl, array", that's what I'm assuming is happening to the
> image. I'm not working with separate red, green, and blue color values, I'm
> working with arrays of numbers which I want to appear in a color scheme that
> can be manipulated to best show what's in the data.

This is an important point. Most users of IDL have a conceptual model of how color tables work that goes something like this:
1. I display an image
2. I invoke XLOADCT
3. I move the sliders in XLOADCT, and it causes the colors in my image to change.
They've seen this work reliably on their Unix boxes, so if it doesn't work on a PC or Mac, then something is 'wrong'. At this point, the only response they get is "Well you don't understand how color tables work", which is unsatisfactory. After all, they've seen the color tables work 'correctly' on their Unix box, so why shouldn't it work on their PC or Mac?

> Almost everybody I know of with a 24-bit graphics cards on a Unix box opts for
> pseudo-color in IDL.

As I said earlier, this is how everyone here works on Unix boxes.

Cheers,
Liam.

Liam E. Gumley
Space Science and Engineering Center, UW-Madison
1225 W. Dayton St., Madison WI 53706, USA
Phone (608) 265-5358, Fax (608) 262-5974
http://cimss.ssec.wisc.edu/~gumley

Subject: Re: 8-bit vs. 24-bit color on Windows
Posted by Martin Schultz on Fri, 29 Jan 1999 08:00:00 GMT
View Forum Message <> Reply to Message

Stein Vidar Hagfors Haugan wrote:
>
> In article <MPG.111a547f221756b5989683@news.frii.com> davidf@dfanning.com
> (David Fanning) writes:
>

>> Liam Gumley (Liam.Gumley@ssec.wisc.edu) writes:
>>>> Almost everybody I know of with a 24-bit graphics cards on a Unix box opts
>>>> for pseudo-color in IDL.
>>>
>>> As I said earlier, this is how everyone here works on Unix boxes.
>> All right. I concede I am a romantic when it comes to software.
>> How else would you dare write a book? :-)
>
> Just to stress the point a bit, I think Bill Thompson is right
> wrt. how many (most?) scientists work with data/color tables,
> at least some of the time.
>
> To them (and me), the "I" in IDL stands for exactly what he
> described (interactive use of commands and xloadct in unison),
> and no widget programmer can ever imagine in advance *everything* a
> scientist would want to do with his data before/during/after
> displaying them (the scientist doesn't know in advance, either:-),
> so the interactive command line is an extremely valuable part of IDL.
>
> Not having the possibility of exploring an image by tweaking
> the color table in pseudo-color mode would take away a lot of the
> original appeal of IDL, IMHO, so this should be taken very seriously
> by RSI.
>

So true, so true !

> I think a large market segment would feel left out if the color-table
> methodology got lost...
>

Didn't someone say lately that RSI sells (was it ?) 60% as PC/Windows versions these days? Don't let me repeat this GENPLOT experience again. This was once a very nice program that just never made it to windows (not to speak of Unix), because the programmer held a flag for IBM in the OS/2 vs. Win3.1 war. In some ways, I still mourn over that loss, because it was even simpler to do this command line data exploration with GENPLOT than it is with IDL (it's all these little inconsistencies that make IDL such a "human" language ;-). Now, I really don't want to loose IDL to the clicking community (programs that run automatically without requiring any user interaction do have their virtues(!) and this old batch processing principle just runs counter to all widgeting (and maybe even objects). It's great to have both options in IDL, just as it is great to have X windows that allow you to type commands instead of DOS boxes that you have to close when they are done. Unfortunately, it's us scientists who have the greatest demands in terms of what a program should be able to do (i.e. graphics, numerical algorithms) and this in a very incoherent fashion. And, it is usually us scientists who can't

afford spending too much on software (how many people are still using 4.x versions of IDL?). So, I just hope that neither IDL for Unix nor the colortable scheme will ever be abandoned!!

A little besides the point, but since this thread has broadened anyhow ...
Let me unravel a little dream here that I had again lately: Wouldn't it be great if a program would have enough intelligence to notice inconsistencies in user input? Example: you draw a map and you specify both hatched and filled continents. Instead of somehow selecting either one, a window would pop up and present you with the possible options. Likewise, all parameters and keywords from a program could be displayed (and queried) automatically either by setting a special keyword (e.g. draw,/query) or as some kind of error handler (e.g. if (n_params() eq 0) then query,...). This could be helpful to run programs that you are not too familar with (now, what were these keywords again ??), and it would take care of those countless situations when you run a program in different contexts or with different data sets where for instance one variable is missing that was explicitely defined in the other data set (or it is named differently). The mechanism would still be "invisible" if you specify all parameters and keywords correctly, and it would of course have to "know" the keywords and parameters automatically so that you don't have to pass a long name list to query when you call it. The "interactive variable assigner" should allow to either enter values manually, or select from all variables that are "visible" on the caller level (and maybe even globally)...

OK. Gotta get back to work now,
Martin.

--

 -- -------
Dr. Martin Schultz
Department for Engineering&Applied Sciences, Harvard University
109 Pierce Hall, 29 Oxford St., Cambridge, MA-02138, USA

phone: (617)-496-8318
fax : (617)-495-4551

e-mail: mgs@io.harvard.edu
Internet-homepage: http://www-as.harvard.edu/people/staff/mgs/
 -- -------

Subject: Re: 8-bit vs. 24-bit color on Windows

Posted by steinhh on Fri, 29 Jan 1999 08:00:00 GMT
View Forum Message <> Reply to Message

In article <MPG.111a547f221756b5989683@news.frii.com> davidf@dfanning.com (David Fanning) writes:

> Liam Gumley (Liam.Gumley@ssec.wisc.edu) writes:
>>> Almost everybody I know of with a 24-bit graphics cards on a Unix box opts
>>> for pseudo-color in IDL.
>>
>> As I said earlier, this is how everyone here works on Unix boxes.
> All right. I concede I am a romantic when it comes to software.
> How else would you dare write a book? :-)

Just to stress the point a bit, I think Bill Thompson is right
wrt. how many (most?) scientists work with data/color tables,
at least some of the time.

To them (and me), the "I" in IDL stands for exactly what he
described (interactive use of commands and xloadct in unison),
and no widget programmer can ever imagine in advance *everything* a
scientist would want to do with his data before/during/after
displaying them (the scientist doesn't know in advance, either:-),
so the interactive command line is an extremely valuable part of IDL.

Not having the possibility of exploring an image by tweaking
the color table in pseudo-color mode would take away a lot of the
original appeal of IDL, IMHO, so this should be taken very seriously
by RSI.

I think Bill has a good point in saying that:

> To you, maybe that's a minor annoyance which you're willing to put up
> with for perceived advantages of a more complicated 3-plane color
> system. For you that's great, and IDL should be able to provide that
> for you. However, for me, a color-table methodology is much simpler
> and more appropriate. I argue that IDL should be taking me into
> account too, and all those more basic users who don't subscribe to this
> newsgroup.

I think a large market segment would feel left out if the color-table
methodology got lost...

Regards,

Stein Vidar

Subject: Re: 8-bit vs. 24-bit color on Windows
Posted by Phillip & Suzanne on Tue, 09 Feb 1999 08:00:00 GMT
View Forum Message <> Reply to Message

Martin Schultz wrote:
>
> Stein Vidar Hagfors Haugan wrote:
>>
>> To them (and me), the "I" in IDL stands for exactly what he
>> described (interactive use of commands and xloadct in unison),
>> and no widget programmer can ever imagine in advance *everything* a
>> scientist would want to do with his data before/during/after
>> displaying them (the scientist doesn't know in advance, either:-),
>> so the interactive command line is an extremely valuable part of IDL.
>>
>> Not having the possibility of exploring an image by tweaking
>> the color table in pseudo-color mode would take away a lot of the
>> original appeal of IDL, IMHO, so this should be taken very seriously
>> by RSI.
>>
>
> So true, so true !

I work in a shop of about 80 people, roughly half programmers and half analysts. The analysts almost all use IDL for a lot of their work, but get by with programmer support from only 4 of us! While the four of us really appreciate all of the wonderful niceties provided by each upgrade, most of our users get more annoyed than anything else.

In particular, direct graphics into a window is the typical M.O. of most of these people. When they figure out what they want, THEN they consider creating a basic widget to automate the process in the future IF they're pretty saavy with IDL.

> Let me unravel a little dream here that I had again lately: Wouldn't it
> be great if a program would have enough intelligence to notice
> inconsistencies in user input? Example: you draw a map and you specify
> both hatched and filled continents. Instead of somehow selecting either
> one, a window would pop up and present you with the possible options.
> Likewise, all parameters and keywords from a program could be displayed
> (and queried) automatically either by setting a special keyword (e.g.
> draw,/query) or as some kind of error handler (e.g. if (n_params() eq 0)
> then query,...). This could be helpful to run programs that you are not
> too familar with (now, what were these keywords again ??), and it would
> take care of those countless situations when you run a program in
> different contexts or with different data sets where for instance one
> variable is missing that was explicitely defined in the other data set
> (or it is named differently). The mechanism would still be "invisible"
> if you specify all parameters and keywords correctly, and it would of

> course have to "know" the keywords and parameters automatically so that
> you don't have to pass a long name list to query when you call it. The
> "interactive variable assigner" should allow to either enter values
> manually, or select from all variables that are "visible" on the caller
> level (and maybe even globally)...

While I'm not volunteering to do so, it should be simple to write just such a program. Pass the command line for the command you want to a routine that has a case statement to check each of the possible commands, determines if the calling sequence is reasonable, and either invokes the requested command or pops up a Dialog_Message box indicating what options are available. Since most users really only use a small subset of the commands, you wouldn't have to make it complete (unless you're RSI doing this...:-). Instead, only include the commands you have trouble with. Good luck.

Phillip
