
Subject: xmanager, /managed

Posted by [Benjamin Hornberger](#) on Wed, 11 May 2005 22:40:02 GMT

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

Hi all,

I found a new tip on David's website, which sounded great to me:

http://www.dfanning.com/widget_tips/managed.html

However, if I do

xmanager, ..., /managed

I get an error "Keyword MANAGED not allowed in call to: XMANAGER".

This is IDL 6.1.1 on Windows.

I checked the source code of xmanager.pro, and couldn't find any trace of that keyword having ever existed. Any clues? Any other way to prevent a draw widget from becoming the current graphics window?

Thanks,
Benjamin

Subject: Re: xmanager, /managed

Posted by [Benjamin Hornberger](#) on Thu, 07 Jul 2005 22:31:06 GMT

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

Two months later, I want to report another experience with this issue. By the way, David seems to have removed the article from his website -- he doesn't seem to like the thing any more :-).

Even though I don't really understand why (see the previous messages in this thread), the "widget_control, /managed" did solve one issue I found quite disturbing before: If you have a non-blocking widget program with a draw widget on the display, but then want to plot something from the command line, it will be plotted in the draw widget unless you manually open a new plot window first. With the statement above (see code attached), a new plot window pops up automatically as desired.

However, by pure chance I discovered now that this statement caused another problem I had been chasing for weeks now: I am using a GUI program with a draw widget to display images. The user can mark a rectangle in the image by dragging the mouse with the left mouse button held down. On Windows, everything worked fine, but on Linux the rectangle was not shown while dragging, only the final rectangle

appeared after releasing the left mouse button. After commenting out the "widget_control, tlb_id, /managed" line, it works in Linux as well.

Attached is a sample program with the relevant code extracted from my real application, for the ones who care. It requires David's fsc_color(). Try to drag a rectangle in the image. On Windows, everything should be fine, but on Linux the rectangle doesn't show up while dragging (at least that's how it behaves for me). Commenting out the "widget_control, /managed" near the end solves this (but introduces the problem mentioned in the beginning).

I should mention that besides keeping track of colors in the GUI program, as David recommends in his book, I also try not to mess with the command line's color vectors and decomposition state. Even while the widget program is running, I want to be able to quickly tvscl another array from the command line with a nice b-w color table rather than the GUI's color table with some plot line colors loaded at the top. Unfortunately, the code becomes quite ugly. Maybe object graphics is the way to go, but I haven't had the time to learn that.

Benjamin

```
;;-----  
PRO test_mouse_update_display, info  
  
  old_window = !d.window  
  
  ;; Store the currently active color vectors and decomposition state,  
  ;; so that we don't mess with the command line's colors.  
  tvlct, old_r, old_g, old_b, /get  
  device, get_decomposed=old_decomposed  
  
  ;; Now load sm_gui's color vectors  
  tvlct, *info.r, *info.g, *info.b  
  device, decomposed=0  
  
  wset, info.draw_wid  
  tv, bytscl(*info.image, top=info.ncolors-1)+1  
  
  ;; draw rectangle  
  IF info.draw_rectangle THEN BEGIN  
    x1 = info.rectangle_firstcorner[0]  
    y1 = info.rectangle_firstcorner[1]  
    x2 = info.rectangle_secondcorner[0]  
    y2 = info.rectangle_secondcorner[1]  
    plots, [x1, x1, x2, x2, x1], [y1, y2, y2, y1, y1], $  
    /device, color=info.green
```

ENDIF

```
;; Restore the previous color state in case somebody's working
;; on the command line in parallel
device, decomposed=old_decomposed
tvlct, old_r, old_g, old_b
wset, old_window
```

END

```
;;-----
PRO test_mouse_event, event

catch, error
IF error NE 0 THEN BEGIN
  catch, /cancel
  ok = error_message(/traceback, /error)
  IF n_elements(info) GT 0 THEN $
    widget_control, event.top, set_uvalue=info, /no_copy
  return
ENDIF

widget_control, event.top, get_uvalue=info, /no_copy

old_window = !d.window
wset, info.draw_wid

CASE event.id OF
  info.draw_id: BEGIN
    CASE event.type OF
      0: BEGIN ;; button pressed
        CASE event.press OF
          1B: BEGIN ;; left button pressed
            ;; store the current color vectors (will be
            ;; restored after button release)
            tvlct, r, g, b, /get
            device, get_decomposed=decomposed
            *info.old_r = r
            *info.old_g = g
            *info.old_b = b
            info.old_decomposed = decomposed
            device, decomposed=0
            tvlct, *info.r, *info.g, *info.b
            info.mouseclick_x = event.x
            info.mouseclick_y = event.y
            ;; to erase any previous rectangle
            info.draw_rectangle = 0
```

```

info.drawing_rectangle = 0
test_mouse_update_display, info
info.drawing_rectangle = 1
;; create a pixmap window to store the
;; original image (without the box)
window, /free, /pixmap, $
    xsize=200, ysize=200
info.pixmap_window_id = !d.window
device, copy=[0, 0, 200, 200, $
    0, 0, info.draw_wid]
info.rectangle_firstcorner = [event.x, event.y]
test_mouse_update_display, info
END ;; left button pressed
ELSE: BEGIN ;; middle or right button pressed
    ;; do nothing
ENDELSE ;; middle or right button pressed
ENDCASE ;; event.press
END ;; button pressed
1: BEGIN ;; button released
CASE event.release OF
    1B: BEGIN ;; left button released
        IF info.drawing_rectangle THEN BEGIN
            info.rectangle_secondcorner = [event.x,
event.y]

            ;; destroy pixmap
wdelete, info.pixmap_window_id
info.drawing_rectangle = 0
            IF (info.rectangle_firstcorner[0] NE $
                info.rectangle_secondcorner[0]) AND $
                (info.rectangle_firstcorner[1] NE $
                info.rectangle_secondcorner[1]) THEN $
                info.draw_rectangle = 1 ELSE $
                info.draw_rectangle = 0
            test_mouse_update_display, info
        ENDIF
        ;; restore previous color vectors
tvlct, *info.old_r, *info.old_g, *info.old_b
device, decomposed=info.old_decomposed
END ;; left button released
ELSE: BEGIN ;; middle or right button released
    ;; do nothing
ENDELSE ;; middle or right button released
ENDCASE ;; event.release
END ;; button released
2: BEGIN ;; cursor motion
IF info.drawing_rectangle THEN BEGIN
    wset, info.draw_wid
    ;; erase the last box

```

```

        device, copy=[0, 0, 200, 200, $
            0, 0, info.pixmap_window_id]
        ;; draw new box
        x1 = info.rectangle_firstcorner[0]
        y1 = info.rectangle_firstcorner[1]
        x2 = event.x
        y2 = event.y
        plots, [x1, x1, x2, x2, x1], [y1, y2, y2, y1, y1], $
            /device, color=info.green
    ENDIF ;; drawing rectangle
END ;; cursor motion
ENDCASE ;; event.type
END
info.update_id: BEGIN
    test_mouse_update_display, info
END
ENDCASE

```

```

wset, old_window

```

```

widget_control, event.top, set_uvalue=info, /no_copy

```

```

END

```

```

;;-----
PRO test_mouse_cleanup, tlb_id

```

```

    widget_control, tlb_id, get_uvalue=info, /no_copy

```

```

    ptr_free, info.image
    ptr_free, info.r
    ptr_free, info.g
    ptr_free, info.b
    ptr_free, info.old_r
    ptr_free, info.old_g
    ptr_free, info.old_b

```

```

END

```

```

;;-----
PRO test_mouse

```

```

    on_error, 2

```

```

    ;; First, store the existing color vectors and decomposition state
    device, get_decomposed=old_decomposed

```

```

tvlct, r_old, g_old, b_old, /get

;; Load b-w linear color table into all available colors to make
;; sure plot and background colors are right
device, decomposed=0
loadct, 0
ncolors = !d.table_size-8 ;; number of colors for image display
;; Now, load b-w linear color table, but leave plot, background and
;; some other colors untouched.
loadct, 0, bottom=1, ncolors=ncolors
black = fsc_color('black', !d.table_size-2)
white = fsc_color('white', !d.table_size-3)
green = fsc_color('green', !d.table_size-4)
red = fsc_color('red', !d.table_size-5)
blue = fsc_color('blue', !d.table_size-6)
tvlct, r, g, b, /get

;; And restore the old color vectors in case somebody wants to work
;; on the command line in between
tvlct, r_old, g_old, b_old
device, decomposed=old_decomposed

;; store the current graphics window
old_window = !d.window

;; define the widgets
tlb_id = widget_base(/col)
draw_id = widget_draw(tlb_id, xsize=200, ysize=200, $
    /button_events, /motion_events, $
    retain=2)
update_id = widget_button(tlb_id, value='Update Display')

widget_control, tlb_id, /realize
widget_control, draw_id, get_value=draw_wid

info = {tlb_id: tlb_id, $
    draw_id: draw_id, $
    draw_wid: draw_wid, $
    update_id: update_id, $
    image: ptr_new(dist(200)), $
    ncolors: ncolors, $
    r: ptr_new(r), $
    g: ptr_new(g), $
    b: ptr_new(b), $
    black: black, $
    white: white, $
    green: green, $
    red: red, $

```

```
blue: blue, $
old_r: ptr_new(/allocate_heap), $
old_g: ptr_new(/allocate_heap), $
old_b: ptr_new(/allocate_heap), $
old_decomposed: 0, $
mouseclick_x: 0L, $
mouseclick_y: 0L, $
draw_rectangle: 0, $
drawing_rectangle: 0, $
pixmap_window_id: 0L, $
rectangle_firstcorner: [0L, 0L], $
rectangle_secondcorner: [0L, 0L]}
```

```
test_mouse_update_display, info
```

```
widget_control, tlb_id, set_uvalue=info, /no_copy
```

```
:: this will avoid that one of the draw widgets will become the
```

```
:: current graphics window for command line use
```

```
widget_control, tlb_id, /managed
```

```
wset, old_window
```

```
xmanager, 'test_mouse', tlb_id, /no_block, $
```

```
    event_handler='test_mouse_event', $
```

```
    cleanup='test_mouse_cleanup'
```

END

Subject: Re: xmanager, /managed

Posted by [David Fanning](#) on Fri, 08 Jul 2005 06:27:52 GMT

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

Benjamin Hornberger writes:

```
> Two months later, I want to report another experience with this
> issue. By the way, David seems to have removed the article from his
> website --he doesn't seem to like the thing any more :-).
```

With the exception of articles written by JD, I try not

to publish articles on my web page that I don't understand.

People have the most embarrassing habit of asking questions... :-(

```
> Even though I don't really understand why (see the previous messages
> in this thread), the "widget_control, /managed" did solve one issue I found
> quite disturbing before: If you have a non-blocking widget program
> with a draw widget on the display, but then want to plot something
> from the command line, it will be plotted in the draw widget unless
```

- > you manually open a new plot window first. With the statement above
- > (see code attached), a new plot window pops up automatically as
- > desired.

I'll have to explain it to you in a private e-mail (I wouldn't *think* of risking it here), but that's why I use it, too.

- > However, by pure chance I discovered now that this statement caused
- > another problem I had been chasing for weeks now: I am using a GUI
- > program with a draw widget to display images. The user can mark a
- > rectangle in the image by dragging the mouse with the left mouse
- > button held down. On Windows, everything worked fine, but on Linux
- > the rectangle was not shown while dragging, only the final rectangle
- > appeared after releasing the left mouse button. After commenting out
- > the "widget_control, tlb_id, /managed" line, it works in Linux as
- > well.

Humm. I think I saw this behavior, too, just shortly after I released that Catalyst save file for public consumption. I think I solved the problem in another way, but I can't think now how I did it. Smoke and mirrors and pixmaps, I expect.

- > Attached is a sample program with the relevant code extracted from my
- > real application, for the ones who care. It requires David's
- > fsc_color(). Try to drag a rectangle in the image. On Windows,
- > everything should be fine, but on Linux the rectangle doesn't show up
- > while dragging (at least that's how it behaves for me). Commenting
- > out the "widget_control, /managed" near the end solves this (but
- > introduces the problem mentioned in the beginning).

Yes, well, I'll have Karsten chase this problem down. He has his brand new Mac configured as a Mac, a PC, a UNIX machine, a German Umpah Band, and I don't know what else! If he can't get to the bottom of this, no one can. :-)

- > I should mention that besides keeping track of colors in the GUI
- > program, as David recommends in his book, I also try not to mess with
- > the command line's color vectors and decomposition state. Even while
- > the widget program is running, I want to be able to quickly tvscl
- > another array from the command line with a nice b-w color table
- > rather than the GUI's color table with some plot line colors loaded
- > at the top. Unfortunately, the code becomes quite ugly. Maybe object
- > graphics is the way to go, but I haven't had the time to learn that.

Object graphics would certainly solve your problem (as well as putting your marriage at risk with the months you will spend learning it), but it seems like overkill to me. I think your main problem is all those damn TV commands.

That is what is causing you so much pain. You have to live in an 8-bit environment in a 24-bit world. I would set that DECOMPOSED keyword permanently to 1 and get a sensible replacement for TV (TVIMAGE, TVSCALE, IMGDISP, PLOTIMAGE). I've lived for nearly two years now neither knowing or caring much about my color decomposition state. (Except for filled contour plots, of course. Sigh...)

My advice is to write your programs so they are totally indifferent to the DECOMPOSED keyword. Life is *much* simpler that way. :-)

Cheers,

David

P.S. And if you *are* forced to go the object route (always a good idea, in my opinion), direct graphics objects are a LOT simpler to learn than object graphics and are more useful in a lot of circumstances.

Subject: Re: xmanager, /managed
Posted by [David Fanning](#) on Fri, 08 Jul 2005 09:06:22 GMT
[View Forum Message](#) <> [Reply to Message](#)

Benjamin Hornberger wrote:

> However, by pure chance I discovered now that this statement caused
> another problem I had been chasing for weeks now: I am using a GUI
> program with a draw widget to display images. The user can mark a
> rectangle in the image by dragging the mouse with the left mouse button
> held down. On Windows, everything worked fine, but on Linux the
> rectangle was not shown while dragging, only the final rectangle
> appeared after releasing the left mouse button. After commenting out the
> "widget_control, tlb_id, /managed" line, it works in Linux as well.

Humm, I think you are right. It appears on UNIX that managing a window prevents DEVICE COPYs to that window from a pixmap. That's nasty. :-(

Think that is a bug? RSI?

Cheers,

David

P.S. To RSI's credit, they do mention that user applications should

not use the MANAGED keyword directly. This is probably why.

Subject: Re: xmanager, /managed

Posted by [Benjamin Hornberger](#) on Fri, 08 Jul 2005 12:11:10 GMT

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

David Fanning wrote:

> Benjamin Hornberger writes:

>

>> I should mention that besides keeping track of colors in the GUI
>> program, as David recommends in his book, I also try not to mess with
>> the command line's color vectors and decomposition state. Even while
>> the widget program is running, I want to be able to quickly tvscl
>> another array from the command line with a nice b-w color table
>> rather than the GUI's color table with some plot line colors loaded
>> at the top. Unfortunately, the code becomes quite ugly. Maybe object
>> graphics is the way to go, but I haven't had the time to learn that.

>

>

> Object graphics would certainly solve your problem (as
> well as putting your marriage at risk with the months you
> will spend learning it), but it seems like overkill to me.

Well, there's no marriage to put at risk yet, but wanting to graduate in a reasonable time ... After all, you don't get a PhD for beautiful code (at least not in my field).

> I think your main problem is all those damn TV commands.
> That is what is causing you so much pain. You have to live
> in an 8-bit environment in a 24-bit world. I would set that
> DECOMPOSED keyword permanently to 1 and get a sensible
> replacement for TV (TVIMAGE, TVSCALE, IMGDISP, PLOTIMAGE). I've
> lived for nearly two years now neither knowing or caring much
> about my color decomposition state. (Except for filled
> contour plots, of course. Sigh...)
>
> My advice is to write your programs so they are totally
> indifferent to the DECOMPOSED keyword. Life is *much*
> simpler that way. :-)

There's two things:

1. The widget program has to use indexed color mode (decomposed=0) because I want to be able to use those pretty predefined color tables. Or are you telling me one can do that with decomposed color as well? Also, I am using the same code for screen and PS output ...

2. In my programs, I am trying to handle colors properly and use `tvimage`. However, when using the command line, things have to be quick. And `"tvscl, array"` can be typed much faster than `"tvimage, bytscl(array), /tv"`. Yes, I could write a wrapper, but even that doesn't help if I am in `decomposed=0` mode and some program loaded extra colors somewhere in the color table. Maybe `decomposed=1` is the way to go on the command line -- usually, I don't need fancy color tables in that case.

Benjamin

Subject: Re: xmanager, /managed

Posted by [David Fanning](#) on Mon, 11 Jul 2005 06:57:16 GMT

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

Benjamin Hornberger wrote:

- > There's two things:
- >
- > 1. The widget program has to use indexed color mode (`decomposed=0`)
- > because I want to be able to use those pretty predefined color tables.
- > Or are you telling me one can do that with decomposed color as well?
- > Also, I am using the same code for screen and PS output ...

Well, this is the point of `TVImage`, `IMGDIST`, etc. You can work in `DECOMPOSED=1` mode and *still* use color tables because the programs are smart enough to put you in the correct mode when they display the image. And, of course, they are also smart enough to know when they are outputting to a device like PS and so can do the proper thing there, too. (Including, not so incidentally, sizing and positioning the image on the PS page.)

- > 2. In my programs, I am trying to handle colors properly and use
- > `tvimage`. However, when using the command line, things have to be quick.
- > And `"tvscl, array"` can be typed much faster than `"tvimage,`
- > `bytscl(array), /tv"`. Yes, I could write a wrapper, but even that doesn't
- > help if I am in `decomposed=0` mode and some program loaded extra colors
- > somewhere in the color table. Maybe `decomposed=1` is the way to go on the
- > command line -- usually, I don't need fancy color tables in that case.

Forget writing the wrapper. I've already done it for you: :-)

<http://www.dfanning.com/programs/tvscale.pro>

- > Maybe `decomposed=1` is the way to go on the
- > command line -- usually, I don't need fancy color tables in that case.

Well, it is the way I go. I think. As I say, I haven't really looked in a couple of years. :-)

Cheers,

David

Subject: Re: xmanager, /managed

Posted by [Benjamin Hornberger](#) on Mon, 11 Jul 2005 19:52:25 GMT

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

David Fanning wrote:

> Benjamin Hornberger wrote:

>

>> There's two things:

>>

>> 1. The widget program has to use indexed color mode (decomposed=0)

>> because I want to be able to use those pretty predefined color tables.

>> Or are you telling me one can do that with decomposed color as well?

>> Also, I am using the same code for screen and PS output ...

>

>

> Well, this is the point of TVImage, IMGDIST, etc. You can work

> in DECOMPOSED=1 mode and *still* use color tables because the

> programs are smart enough to put you in the correct mode when

> they display the image. And, of course, they are also smart

> enough to know when they are outputting to a device like PS

> and so can do the proper thing there, too. (Including, not so

> incidentally, sizing and positioning the image on the PS page.)

I'm still not convinced that with tvimage I could do what I want without worrying about anything else. Anyway, it works now, and as it had been mentioned before, one shouldn't try to optimize working code just for the sake of it ... Next time I write an image display program I'll think about it again.

>

>> 2. In my programs, I am trying to handle colors properly and use

>> tvimage. However, when using the command line, things have to be

>> quick. And "tvsc1, array" can be typed much faster than "tvimage,

>> bytscl(array), /tv". Yes, I could write a wrapper, but even that

>> doesn't help if I am in decomposed=0 mode and some program loaded

>> extra colors somewhere in the color table. Maybe decomposed=1 is the

>> way to go on the command line -- usually, I don't need fancy color

>> tables in that case.

>

>

> Forget writing the wrapper. I've already done it for you: :-)
>
> <http://www.dfanning.com/programs/tvscale.pro>

Hmm, somehow I missed this even though I use programs from your library quite often ...

Benjamin
