Subject: Re: FLOAT images instead of BYTE ones from IDL Object graphics ?
Posted by David Fanning on Thu, 22 Jan 2009 14:59:20 GMT
View Forum Message <> Reply to Message

Gianluca Li Causi writes:

> I'm working with 3D object rendering and I see that the oWindow ->
> GetProperty, IMAGE_DATA=Img always returns a BYTE-type image, while I
> need a FLOAT-type one, not quantized in the 256 levels.
>
> How can I do?
>
> I need this because I always need a 0_to_255 grayscale image of a
> volumetric data, while, for any values of the Opacity, I cannot
> produce a final image with a maximum gray greater than 100 (in fact
> the final gray levels depends on both the volume data, the opacity and
> the number of elements of the volume array).
> If I scale up the final byte image I get a very bad image with
> quantized grayscale...
>
> Someone can help?

It seems to me you are confusing data *display* with the
actual data. The IMAGE_DATA keyword doesn't so much return
a BYTE-type image as it returns a true-color rendition
of what you displayed in the graphics window. What you displayed
is NOT your data, it is a representation of your data, and
that is exactly what you are getting back.

If you want to further manipulate your data, I think you
are going to have to go find the source. You will not find
it in a graphics window, I'm sure of that. :-)

Cheers,

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

Subject: Re: FLOAT images instead of BYTE ones from IDL Object graphics ?
Posted by Jean H. on Thu, 22 Jan 2009 15:10:53 GMT
View Forum Message <> Reply to Message

Gianluca Li Causi wrote:

> Hi,
> I'm working with 3D object rendering and I see that the oWindow ->
> GetProperty, IMAGE_DATA=Img always returns a BYTE-type image, while I
> need a FLOAT-type one, not quantized in the 256 levels.
>
> How can I do?
[...]
>
> Thanks a lot
> Gianluca

It's all in the help file... a long 5.6 seconds search on window object!

So, have a look at IDLgrWindow::Read , then IDLgrImage.data

Jean

Subject: Re: FLOAT images instead of BYTE ones from IDL Object graphics ?
Posted by David Fanning on Thu, 22 Jan 2009 15:54:42 GMT
View Forum Message <> Reply to Message

Jean H. writes:

> It's all in the help file... a long 5.6 seconds search on window object!
>
> So, have a look at IDLgrWindow::Read , then IDLgrImage.data

Yeah, I don't think so. :-)

Cheers,

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

Subject: Re: FLOAT images instead of BYTE ones from IDL Object graphics ?
Posted by Gianluca Li Causi on Fri, 23 Jan 2009 12:47:59 GMT
View Forum Message <> Reply to Message

On Jan 22, 3:59 pm, David Fanning <n...@dfanning.com> wrote:
> Gianluca Li Causi writes:
>> I'm working with 3D object rendering and I see that the oWindow ->

>> GetProperty, IMAGE_DATA=Img always returns a BYTE-type image, while I
>> need a FLOAT-type one, not quantized in the 256 levels.
>
>> How can I do?
>
>> I need this because I always need a 0_to_255 grayscale image of a
>> volumetric data, while, for any values of the Opacity, I cannot
>> produce a final image with a maximum gray greater than 100 (in fact
>> the final gray levels depends on both the volume data, the opacity and
>> the number of elements of the volume array).
>> If I scale up the final byte image I get a very bad image with
>> quantized grayscale...
>
>> Someone can help?
>
> It seems to me you are confusing data *display* with the
> actual data. The IMAGE_DATA keyword doesn't so much return
> a BYTE-type image as it returns a true-color rendition
> of what you displayed in the graphics window. What you displayed
> is NOT your data, it is a representation of your data, and
> that is exactly what you are getting back.

No David, I'm not confusing: I just need a 16bit/channel display
instead of an 8bit/channel display, i.e. I want a float-type image
representation of my 3d data.

If this is not possible, how can I always get a well-visible image of
my volume?
In fact the final grayscale depends on the volume sampling, volume
data ancd volume opacity: it is the same problem of visualizing an
image with TV, that is solved with TVSCL: how can I get a TVSCL-like
view of an emission volume, or automatically set the OPACITY_TABLE0 in
order to always span the full grayscale range?

Thanks
Gianluca

Subject: Re: FLOAT images instead of BYTE ones from IDL Object graphics ?
Posted by David Fanning on Fri, 23 Jan 2009 12:56:08 GMT
View Forum Message <> Reply to Message

Gianluca Li Causi writes:

> If this is not possible, how can I always get a well-visible image of
> my volume?
> In fact the final grayscale depends on the volume sampling, volume
> data ancd volume opacity: it is the same problem of visualizing an

> image with TV, that is solved with TVSCL: how can I get a TVSCL-like
> view of an emission volume, or automatically set the OPACITY_TABLE0 in
> order to always span the full grayscale range?

You are going to have to spell this out for me. I
cannot imagine why you think the OPACITY_TABLE0
limits you to 100 values of gray scale. Sorry for
being on the dense side of the opacity scale. :-(

Cheers,

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

Subject: Re: FLOAT images instead of BYTE ones from IDL Object graphics ?
Posted by Gianluca Li Causi on Fri, 23 Jan 2009 15:31:40 GMT
View Forum Message <> Reply to Message

> You are going to have to spell this out for me. I
> cannot imagine why you think the OPACITY_TABLE0
> limits you to 100 values of gray scale. Sorry for
> being on the dense side of the opacity scale. :-(
>
> Cheers,
>
> David

Ok, so I think that a practical example could help: the following
program makes a spherical volume made of concentric shells, following
a sin(radius) law.

I use a linear grayscale as RGB_TABLE so that the maximum value is
white and the minimum is black.
Then I also use a linear Opacity from 0 to 15, because I want to well
view throug all the shells until the center.

As you see, the final image is a byte array and its maximum value is
142.
In my application the maximum gray is in the range 10 to 20, but I
want an image with 256 gray levels not only 10 or 20 grays.

Thanks for your help
Gianluca

Here is the sample program:

```
x_pix = 100.
y_pix = x_pix
z_pix = x_pix

Volume_Color = (findgen(256)/255) # [255,255,255]
Volume_Opacity = (findgen(256)/255) * 15

;Make a spherical volume with density in shells:
Volume_Data = fltarr(x_pix,y_pix,z_pix)
FOR x = 0., x_pix-1 DO BEGIN
 FOR y = 0., y_pix-1 DO BEGIN
  FOR z = 0., z_pix-1 DO BEGIN
   Volume_Data[x,y,z] = SQRT((x_pix/2.-x)^2 + (y_pix/2.-y)^2 + (z_pix/
2.-z)^2) ;distance from center
  ENDFOR
 ENDFOR
ENDFOR
Volume_Data = Volume_Data * (Volume_Data LE (x_pix/2.))  ;cut at x_pix/
2 radius
Volume_Data = SIN(Volume_Data)        ;make sinusoidal shells

;convert to byte as required by IDLgrVolume
Volume_Data_byte = BYTE((Volume_Data / max(Volume_Data)) * 255.)


;3d graphic objects
oWindow = OBJ_NEW('IDLgrWindow', RETAIN=2, DIMENSIONS=[400,400])
oView = OBJ_NEW('IDLgrView', COLOR=[0,0,0])
oModel = OBJ_NEW('IDLgrModel')


;Create Volume Object
oVolume = OBJ_NEW('IDlgrVolume', Volume_Data_byte, LIGHTING_MODEL=0,
INTERPOLATE=1, $
   OPACITY_TABLE0=Volume_Opacity, COMPOSITE_FUNCTION=0,
ZERO_OPACITY_SKIP=1, ZBUFFER=1)

oModel -> Add, oVolume


;set display window
Display_XRANGE = [0, x_pix-1]
Display_YRANGE = [0, y_pix-1]
```

```
Display_ZRANGE = [0, z_pix-1]

Display_xSize = x_pix
Display_ySize = y_pix
Display_zSize = z_pix
Display_Diagonal = SQRT(Display_xSize^2 + Display_ySize^2 +
Display_zSize^2)

Display_xCenter = x_pix/2
Display_yCenter = y_pix/2
Display_zCenter = z_pix/2

oModel -> TRANSLATE, -Display_xCenter, -Display_yCenter, -
Display_zCenter

oModel -> ROTATE, [1,0,0], -90
oModel -> ROTATE, [0,1,0], -60
oModel -> ROTATE, [1,0,0], 30

oView -> SetProperty, VIEWPLANE_RECT=Display_Diagonal*[-.5,-.5,1,1],
ZCLIP=Display_Diagonal*[.5,-.5], EYE=Display_Diagonal

oView -> Add, oModel

;Display Object Hierarchy
oWindow -> Draw, oView

;Catch the Window
oWindow -> GetProperty, IMAGE_DATA=IMAGE

help, IMAGE
print, max(IMAGE)

stop

OBJ_DESTROY, oView
OBJ_DESTROY, oWindow
```

Subject: Re: FLOAT images instead of BYTE ones from IDL Object graphics ?
Posted by David Fanning on Fri, 23 Jan 2009 16:29:20 GMT
View Forum Message <> Reply to Message

Gianluca Li Causi writes:

> Ok, so I think that a practical example could help: the following
> program makes a spherical volume made of concentric shells, following
> a sin(radius) law.

>
> I use a linear grayscale as RGB_TABLE so that the maximum value is
> white and the minimum is black.
> Then I also use a linear Opacity from 0 to 15, because I want to well
> view throug all the shells until the center.
>
> As you see, the final image is a byte array and its maximum value is
> 142.
> In my application the maximum gray is in the range 10 to 20, but I
> want an image with 256 gray levels not only 10 or 20 grays.

OK, two things. First, there is no requirement that the volume argument is required to be a byte array. A float array, scaled from 0 to 65535 works just as well in your example.

To *display* the data, of course, it has to be scaled into the range of 0 to 255, as all data does to be displayed in a graphics window of a normal, off-the-shelf computer. Naturally, the values you read *out* of the window will be in this range.

The *particular* value you read back from the window is indicative of the shade of gray that was used to render that particular pixel. This rendering choice is a function of the color of the objects, the way the opacity table is being used to modify the value of the pixels, the composite function, and the lighting you are using on your model.

I see nothing here to cause me to retract my earlier statement that you are confusing data with the display of the data. Sorry.

Cheers,

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

Subject: Re: FLOAT images instead of BYTE ones from IDL Object graphics ?
Posted by Gianluca Li Causi on Fri, 23 Jan 2009 16:54:00 GMT
View Forum Message <> Reply to Message

> OK, two things. First, there is no requirement that the
> volume argument is required to be a byte array. A float
> array, scaled from 0 to 65535 works just as well in your
> example.

Yes you're right, it can be also a float.

> To *display* the data, of course, it has to be scaled
> into the range of 0 to 255, as all data does to be
> displayed in a graphics window of a normal, off-the-
> shelf computer. Naturally, the values you read *out*
> of the window will be in this range.
>
> The *particular* value you read back from the window
> is indicative of the shade of gray that was used to
> render that particular pixel. This rendering choice
> is a function of the color of the objects, the
> way the opacity table is being used to modify the
> value of the pixels, the composite function, and the
> lighting you are using on your model.

So, what can I do to automatically get a 0 to 255 full-range display
of my volumes where I can look through all the volume data?

As you see I'm still a newcomer in volume rendering...
Gianluca

Subject: Re: FLOAT images instead of BYTE ones from IDL Object graphics ?
Posted by David Fanning on Fri, 23 Jan 2009 17:07:57 GMT
View Forum Message <> Reply to Message

Gianluca Li Causi writes:

> So, what can I do to automatically get a 0 to 255 full-range display
> of my volumes where I can look through all the volume data?
>
> As you see I'm still a newcomer in volume rendering...

Yes, well, I'm at least one page ahead of you. :-)

I don't know if you can "automatically" do much of anything
useful in IDL. I presume, though, that you could modify
the color of the volume, use lights, and change the composite
function to modify the values your surface is rendered in.

You are probably NOT going to find the information you need
to do this convincingly in the IDL documentation. I think you

are going to have to get a good OpenGL book and see how all
of this works. Then you will have to make assumptions, half
of them wrong, about how IDL implemented OpenGL properties.

In the end, you will have spent an enormous amount of time
and you may be a bit wiser. (Although I wouldn't risk too
much money betting on this.) In other words, you are going
to have to learn object graphics just like the rest of us. :-)

Cheers,

David

P.S. It might help to get Rick Towler season tickets to the
Seattle Seahawk's games, but you will have to discuss this
with him privately. :-)

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

Subject: Re: FLOAT images instead of BYTE ones from IDL Object graphics ?
Posted by Karl[1] on Mon, 26 Jan 2009 17:57:36 GMT
View Forum Message <> Reply to Message

Well, your composite function is

dest' = src * srcalpha + dest * (1 - srcalpha)

srcalpha comes from your opacity table, which has a max value of 15.

So, for a voxel at max value, 255, the brightest that a single pixel
can be against a black background is 15. Of course all voxels along
the ray would contribute to the brightness in an additive way.

With a max opacity of 15, you are going to be getting "mostly
background" with each blend. It will take a lot of high voxel values
along to ray to get them to add up to close to 255.

Where did you get 15?

If you change that to something like 100, you will get closer to the
full range in the final image.

Karl

On Jan 23, 9:31 am, Gianluca Li Causi <lica...@mporzio.astro.it>
wrote:
>> You are going to have to spell this out for me. I
>> cannot imagine why you think the OPACITY_TABLE0
>> limits you to 100 values of gray scale. Sorry for
>> being on the dense side of the opacity scale. :-(
>
>> Cheers,
>
>> David
>
> Ok, so I think that a practical example could help: the following
> program makes a spherical volume made of concentric shells, following
> a sin(radius) law.
>
> I use a linear grayscale as RGB_TABLE so that the maximum value is
> white and the minimum is black.
> Then I also use a linear Opacity from 0 to 15, because I want to well
> view throug all the shells until the center.
>
> As you see, the final image is a byte array and its maximum value is
> 142.
> In my application the maximum gray is in the range 10 to 20, but I
> want an image with 256 gray levels not only 10 or 20 grays.
>
> Thanks for your help
> Gianluca
>
> Here is the sample program:
>
> x_pix = 100.
> y_pix = x_pix
> z_pix = x_pix
>
> Volume_Color = (findgen(256)/255) # [255,255,255]
> Volume_Opacity = (findgen(256)/255) * 15
>
> ;Make a spherical volume with density in shells:
> Volume_Data = fltarr(x_pix,y_pix,z_pix)
> FOR x = 0., x_pix-1 DO BEGIN
> FOR y = 0., y_pix-1 DO BEGIN
> FOR z = 0., z_pix-1 DO BEGIN
> Volume_Data[x,y,z] = SQRT((x_pix/2.-x)^2 + (y_pix/2.-y)^2 +
(z_pix/
> 2.-z)^2) ;distance from center
> ENDFOR
> ENDFOR

```
>  ENDFOR
>  Volume_Data = Volume_Data * (Volume_Data LE (x_pix/2.))        ;cut at x_pix/
>  2 radius
>  Volume_Data = SIN(Volume_Data)
            ;make sinusoidal shells
>
>  ;convert to byte as required by IDLgrVolume
>  Volume_Data_byte = BYTE((Volume_Data / max(Volume_Data)) * 255.)
>
>  ;3d graphic objects
>  oWindow = OBJ_NEW('IDLgrWindow', RETAIN=2, DIMENSIONS=[400,400])
>  oView = OBJ_NEW('IDLgrView', COLOR=[0,0,0])
>  oModel = OBJ_NEW('IDLgrModel')
>
>  ;Create Volume Object
>  oVolume = OBJ_NEW('IDlgrVolume', Volume_Data_byte, LIGHTING_MODEL=0,
>  INTERPOLATE=1, $
>                      OPACITY_TABLE0=Volume_Opacity,
COMPOSITE_FUNCTION=0,
>  ZERO_OPACITY_SKIP=1, ZBUFFER=1)
>
>  oModel -> Add, oVolume
>
>  ;set display window
>  Display_XRANGE = [0, x_pix-1]
>  Display_YRANGE = [0, y_pix-1]
>  Display_ZRANGE = [0, z_pix-1]
>
>  Display_xSize = x_pix
>  Display_ySize = y_pix
>  Display_zSize = z_pix
>  Display_Diagonal = SQRT(Display_xSize^2 + Display_ySize^2 +
>  Display_zSize^2)
>
>  Display_xCenter = x_pix/2
>  Display_yCenter = y_pix/2
>  Display_zCenter = z_pix/2
>
>  oModel -> TRANSLATE, -Display_xCenter, -Display_yCenter, -
>  Display_zCenter
>
>  oModel -> ROTATE, [1,0,0], -90
>  oModel -> ROTATE, [0,1,0], -60
>  oModel -> ROTATE, [1,0,0], 30
>
>  oView -> SetProperty, VIEWPLANE_RECT=Display_Diagonal*[-.5,-.5,1,1],
>  ZCLIP=Display_Diagonal*[.5,-.5], EYE=Display_Diagonal
>
```

```
>  oView -> Add, oModel
>
>  ;Display Object Hierarchy
>  oWindow -> Draw, oView
>
>  ;Catch the Window
>  oWindow -> GetProperty, IMAGE_DATA=IMAGE
>
>  help, IMAGE
>  print, max(IMAGE)
>
>  stop
>
>  OBJ_DESTROY, oView
>  OBJ_DESTROY, oWindow
```
