Subject: Re: 8 to 24 bit conversion
Posted by Liam Gumley on Tue, 09 Nov 2004 22:40:37 GMT
View Forum Message <> Reply to Message

Ken Mankoff wrote:
>
> I am trying to convert an 8 bit image created in the Z buffer to a 24
> bit image. Is this possible? I would think so. But I am having trouble
> getting it to work.
>
> I based my code off of this algorithm:
> http://groups.google.com/groups?hl=en&lr=&selm=c0jq0
j%24i40%241%40nntp6.u.washington.edu
>
> But I don't want to use the COLOR_QUAN bit of code that reduces it back
> to 8 bits.
>
> Can anyone point me to a library that does this, or point out some of
> the errors in the code below? Please don't point out the embarrassing
> optimization errors I know are there...
>
> Thanks
>
> -k.
>
>
> image = TVRD()
> TVLCT, R, G, B, /GET
>
> s = SIZE(image, /DIMENSIONS)
> rImage = BYTARR(s)
> gImage = BYTARR(s)
> bImage = BYTARR(s)
> ; replace this section with a HISTOGRAM statement
> for n=0, 255 do begin
> idx = WHERE(image eq n, count)
> if (count gt 0) then begin
> rImage[idx] = R[n]
> gImage[idx] = G[n]
> bImage[idx] = B[n]
> endif
> endfor
>
> newImage = bytarr(s(0), s(1), 3)
> newImage[*,*,0] = rImage
> newImage[*,*,1] = gImage
> newImage[*,*,2] = bImage

```
dims = size(image, /dimensions)
true_image = bytarr(3, dims[0], dims[1])
true_image[0, *, *] = r[image]
true_image[1, *, *] = g[image]
true_image[2, *, *] = b[image]
```

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

Subject: Re: 8 to 24 bit conversion
Posted by Liam Gumley on Tue, 09 Nov 2004 22:49:20 GMT
View Forum Message <> Reply to Message

Liam Gumley wrote:
> dims = size(image, /dimensions)
> true_image = bytarr(3, dims[0], dims[1])
> true_image[0, *, *] = r[image]
> true_image[1, *, *] = g[image]
> true_image[2, *, *] = b[image]

This might be a little bit faster:

```
dims = size(image, /dimensions)
true_image = bytarr(dims[0], dims[1], 3)
true_image[0, 0, 0] = r[image]
true_image[0, 0, 1] = g[image]
true_image[0, 0, 2] = b[image]
```

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

Subject: Re: 8 to 24 bit conversion
Posted by KM on Tue, 09 Nov 2004 23:21:38 GMT
View Forum Message <> Reply to Message

On Tue, 9 Nov 2004, Liam Gumley wrote:
>
> Ken Mankoff wrote:
>>
>> I am trying to convert an 8 bit image created in the Z buffer to
>> a 24 bit image. Is this possible? I would think so. But I am

>> having trouble getting it to work.
>>
>> I based my code off of this algorithm:
>> http://groups.google.com/groups?hl=en&lr=&selm=c0jq0
j%24i40%241%40nntp6.u.washington.edu
>> But I don't want to use the COLOR_QUAN bit of code that reduces
>> it back to 8 bits.
>>
>> image = TVRD()
>> TVLCT, R, G, B, /GET
> dims = size(image, /dimensions)
> true_image = bytarr(3, dims[0], dims[1])
> true_image[0, *, *] = r[image]
> true_image[1, *, *] = g[image]
> true_image[2, *, *] = b[image]

OK, that is quite a bit more succinct. But the colors still aren't right when I "TV, true_image, /true"

Thinking it could be a display issue (decomposed, true_color, pseudo, etc.) I tested it by writing true_image out to a png and jpeg, but they don't look write.

FYI, the above function I called toRGB, and my test code is:

```
loadct, 39
tv, congrid(indgen(16,16),640,512)
x = torgb()
;loadct, 0 ; tried w/ & w/o this commented
tv,x,/true  ; produces color, but not same as last tv command
```

Subject: Re: 8 to 24 bit conversion
Posted by Robert Barnett on Tue, 09 Nov 2004 23:36:06 GMT
View Forum Message <> Reply to Message

Firstly,

If there was anything I could wish for, it would be a 24bit buffer. I've had so much trouble trying to capture color snapshots of IDL windows (to send to a medical imaging PACS). This is particularily when I've rendered text and lines whose colors are not in the LUT. If anyone has a nice way around this, which is backwards compatible to IDL 5.1 I'd be so happy.

Currently I do as you do. I capture the screen and put it into my own byte array. I currently capture rendered text and lines with their own LUT separate from the image. Then I render the medical image in its LUT

and capture that as well. I actually use 4 channels so as to include an alpha channel. I merge the two images together and then ship them off to wherever they are meant to go. It's very complicated and slow but it works and looks half reasonable. Retrospectivley I would have split my colortable into two and scaled the medical image so it only uses its part of the color table.

Anyway, enough whining. Ken, is this what you wanted to do?

```
; Load the color table (blue)
loadct, 1
; Draw something at x=0, y=0
tvscl, dist(100)
; Read the entire screen
image = tvrd()
; Read the look up table
tvlct, red, green, blue, /get
; Get the size of the snapshot of the screen
size = size(image)
; Assign a rgb image
rgb = bytarr(3,size[1],size[2])
; Fill in each channel based on the byte value of the image
rgb[0,*,*] = red[image]
rgb[1,*,*] = blue[image]
rgb[2,*,*] = green[image]
; Load a BW color table
loadct, 0
; Display each channel
tvscl, rgb[0,0:100,0:100], 0 ; Red
tvscl, rgb[1,0:100,0:100], 1 ; Green
tvscl, rgb[2,0:100,0:100], 2 ; Blue
```

Ken Mankoff wrote:

>
> I am trying to convert an 8 bit image created in the Z buffer to a 24
> bit image. Is this possible? I would think so. But I am having trouble
> getting it to work.
>
> I based my code off of this algorithm:
> http://groups.google.com/groups?hl=en&lr=&selm=c0jq0
j%24i40%241%40nntp6.u.washington.edu
>

```
>  But I don't want to use the COLOR_QUAN bit of code that reduces it back
>  to 8 bits.
>
>  Can anyone point me to a library that does this, or point out some of
>  the errors in the code below? Please don't point out the embarrassing
>  optimization errors I know are there...
>
>  Thanks
>
>    -k.
>
>
>  image = TVRD()
>  TVLCT, R, G, B, /GET
>
>  s = SIZE(image, /DIMENSIONS)
>  rImage = BYTARR(s)
>  gImage = BYTARR(s)
>  bImage = BYTARR(s)
>  ; replace this section with a HISTOGRAM statement
>  for n=0, 255 do begin
>      idx = WHERE(image eq n, count)
>      if (count gt 0) then begin
>          rImage[idx] = R[n]
>          gImage[idx] = G[n]
>          bImage[idx] = B[n]
>      endif
>  endfor
>
>  newImage = bytarr( s(0), s(1), 3 )
>  newImage[*,*,0] = rImage
>  newImage[*,*,1] = gImage
>  newImage[*,*,2] = bImage
>
```

--

nrb@
Robbie Barnett
imag
Research Assistant
wsahs
Nuclear Medicine & Ultrasound
nsw
Westmead Hospital
gov
Sydney Australia

au
+61 2 9845 7223

Subject: Re: 8 to 24 bit conversion
Posted by Robert Barnett on Tue, 09 Nov 2004 23:49:26 GMT
View Forum Message <> Reply to Message

Sorry I didn't press refresh on the newsgroup
How about this.

```
erase
; Load the color table (blue)
loadct, 2
; Draw something at x=0, y=0
tvscl, dist(100)
; Read the entire screen
image = tvrd()
; Read the look up table
tvlct, red, green, blue, /get
; Get the size of the snapshot of the screen
size = size(image)
; Assign a rgb image
rgb = bytarr(3,size[1],size[2])
; Fill in each channel based on the byte value of the image
rgb[0,*,*] = red[image]
rgb[2,*,*] = blue[image]
rgb[1,*,*] = green[image]
; Load a BW color table
loadct, 0
; Display each channel
tvscl, rgb, /true
;tvscl, rgb[0,0:100,0:100], 0 ; Red
;tvscl, rgb[1,0:100,0:100], 1 ; Green
;tvscl, rgb[2,0:100,0:100], 2 ; Blue
```

For some reason I have to swap the blue and green positions.

--

nrb@
Robbie Barnett
imag
Research Assistant
wsahs

Nuclear Medicine & Ultrasound
nsw
Westmead Hospital
gov
Sydney Australia
au
+61 2 9845 7223

Subject: Re: 8 to 24 bit conversion
Posted by KM on Wed, 10 Nov 2004 02:17:15 GMT
View Forum Message <> Reply to Message

On Tue, 9 Nov 2004, Ken Mankoff wrote:

> On Tue, 9 Nov 2004, Liam Gumley wrote:
>>
>> Ken Mankoff wrote:
>>>
>>> I am trying to convert an 8 bit image created in the Z buffer to
>>> a 24 bit image. Is this possible? I would think so. But I am
>>> having trouble getting it to work.
>>>
>>> I based my code off of this algorithm:
>>> http://groups.google.com/groups?hl=en&lr=&selm=c0jq0
j%24i40%241%40nntp6.u.washington.edu
>>> But I don't want to use the COLOR_QUAN bit of code that reduces
>>> it back to 8 bits.
>>>
>>> image = TVRD()
>>> TVLCT, R, G, B, /GET
>> dims = size(image, /dimensions)
>> true_image = bytarr(3, dims[0], dims[1])
>> true_image[0, *, *] = r[image]
>> true_image[1, *, *] = g[image]
>> true_image[2, *, *] = b[image]
>
> OK, that is quite a bit more succinct. But the colors still aren't
> right when I "TV, true_image, /true"
>

OK, that code is correct. My mistake was that I was testing it in
the X buffer, not Z buffer, so the TVRD() command wasn't working
properly (24 bit display and all that...).

Thanks,

-k.

Subject: Re: 8 to 24 bit conversion
Posted by KM on Wed, 10 Nov 2004 02:26:57 GMT
View Forum Message <> Reply to Message

On Wed, 10 Nov 2004, Robert Barnett wrote:
> If there was anything I could wish for, it would be a 24bit
> buffer. I've had so much trouble trying to capture color snapshots
> of IDL windows (to send to a medical imaging PACS). This is
> particularily when I've rendered text and lines whose colors are
> not in the LUT. If anyone has a nice way around this, which is
> backwards compatible to IDL 5.1 I'd be so happy.
I agree. It looks like you can get one if you need it on a single system, but cross-platform it becomes more difficult. There is the WINDOW, /PIXMAP on X, or SET_PLOT,'CGM', and there is SET_PLOT,'METAFILE' on windows. But none of those work for me so I am stuck in 8 bit Z.

> Currently I do as you do. I capture the screen and put it into my
> own byte array. I currently capture rendered text and lines with
> their own LUT separate from the image. Then I render the medical
> image in its LUT and capture that as well. I actually use 4
> channels so as to include an alpha channel. I merge the two images
> together and then ship them off to wherever they are meant to go.
> It's very complicated and slow but it works and looks half
> reasonable. Retrospectivley I would have split my colortable into
> two and scaled the medical image so it only uses its part of the
> color table.
I didn't explain exactly why I was doing this, but you are right. I am producing a nice color image in RGB using 256 colors (w/ colorbar), and then the text, axes, title, colorbar axes, etc. in grayscale get added in at the byte level.

I am interested as to why you say you would do it the other way with a split colorbar, and not the way we currently are.

I also don't know if I can do it the other way. Not only am I doing the above, but I am doing it all at 4x as large as it needs to be, and then using REBIN() (Dr. Fannings trick) to get the text and continents anti-aliased. I don't think this trick will work if the colorbar is split in two, will it?

As you said it is slow. Unfortunately, my boss wants it both looking nice (anti-aliased) _and_ fast. :/

> Anyway, enough whining. Ken, is this what you wanted to do?
Yes, the code you provided works well. Thank you.

Subject: Re: 8 to 24 bit conversion
Posted by Robert Barnett on Wed, 10 Nov 2004 06:27:41 GMT
View Forum Message <> Reply to Message

> I also don't know if I can do it the other way. Not only am I doing the
> above, but I am doing it all at 4x as large as it needs to be, and then
> using REBIN() (Dr. Fannings trick) to get the text and continents
> anti-aliased. I don't think this trick will work if the colorbar is
> split in two, will it?

Ha ha! I *also* use rebin to make my p's look pretty.

You are correct that splitting the colortable means that you can't rebin, otherwise the edges of the letters have funny shades. In fact I found that I get the nicest image by rebining each color channel after flattening the layers.

I've only just started to get to know the IDLgr* classes. IDLgrBuffer is probably the best solution. However, it's not much use if you've already written your code using tvscl, plot etc... Or I could just recode my project as an exercise.

I have promised myself that my next project will make use of Graphics Object Classes. I have to admit that they are pretty useful and much simpler than expected. But then again, any OO task may look easy after trying to extend a Java Swing component. Just take a look at some of the nasty Java API's and you'll come squirming to David Fanning land.

Cheers
--

nrb@
Robbie Barnett
imag
Research Assistant
wsahs
Nuclear Medicine & Ultrasound
nsw
Westmead Hospital
gov
Sydney Australia
au
+61 2 9845 7223

Subject: Re: 8 to 24 bit conversion

Posted by David Fanning on Wed, 10 Nov 2004 07:04:48 GMT
View Forum Message <> Reply to Message

Robert Barnett writes:

> I've only just started to get to know the IDLgr* classes. IDLgrBuffer is
> probably the best solution. However, it's not much use if you've already
> written your code using tvscl, plot etc... Or I could just recode my
> project as an exercise.
>
> I have promised myself that my next project will make use of Graphics
> Object Classes. I have to admit that they are pretty useful and much
> simpler than expected. But then again, any OO task may look easy after
> trying to extend a Java Swing component. Just take a look at some of the
> nasty Java API's and you'll come squirming to David Fanning land.

I'll just reiterate--again--that you can make some
really useful objects and not go anywhere near graphics
object classes. One thing RSI has done extremely well
is erroneously make object graphics synonymous with
object programming. It is most frustrating when you are
in the field trying to convert the masses. :-(

Cheers,

David
--
David W. Fanning, Ph.D.
Fanning Software Consulting, Inc.
Coyote's Guide to IDL Programming: http:/www.dfanning.com/
Phone: 970-221-0438, IDL Book Orders: 1-888-461-0155

Subject: Re: 8 to 24 bit conversion
Posted by KM on Wed, 10 Nov 2004 13:45:06 GMT
View Forum Message <> Reply to Message

On Wed, 10 Nov 2004, David Fanning wrote:
> Robert Barnett writes:
>> I've only just started to get to know the IDLgr* classes.
>> IDLgrBuffer is probably the best solution. However, it's not much
>> use if you've already written your code using tvscl, plot etc...
>> Or I could just recode my project as an exercise.
From what I understand, its also not much use if I am using the
MAP_SET, MAP_*, routines, which is what this project is about. So I
am stuck in direct grahics. (Also, I don't have 6.1 w/ iMap).

These limitations in the context of my current project were recently

discussed here:
http://groups.google.com/groups?hl=en&lr=&selm=Pine.OSX.4.61.0410221805220.19262%40gouda.local

> I'll just reiterate--again--that you can make some really useful
> objects and not go anywhere near graphics object classes.
I think this is the approach I'll take. Some nice objects dealing with my map and data that do direct graphics Z-buffer calls.

Regards,

-Ken.

Subject: Re: 8 to 24 bit conversion
Posted by David Fanning on Wed, 10 Nov 2004 14:10:02 GMT
View Forum Message <> Reply to Message

Ken Mankoff writes:

> From what I understand, its also not much use if I am using the
> MAP_SET, MAP_*, routines, which is what this project is about. So I
> am stuck in direct grahics. (Also, I don't have 6.1 w/ iMap).

Well, the MAP_PROJ_*** routines are quite good and can be used with either direct or object graphics. And of course you get a much more complete set of map projections and options when you use them. I think I agree with you, though, that map projections don't gain much when done in object graphics.

> I think this is the approach I'll take. Some nice objects dealing
> with my map and data that do direct graphics Z-buffer calls.

Well, yes, objects here are certain to make your life a lot easier. But the Z-buffer. That's pretty 1970s, isn't it. :-)

Cheers,

David

P.S. Have you written your nice letter to RSI pleading for nice fonts in direct graphics yet? That would certainly save us from all this rigamarole, wouldn't it. :-)

--
David W. Fanning, Ph.D.