Subject: more log-scaled images
Posted by ben.bighair on Tue, 16 Dec 2008 03:00:55 GMT
View Forum Message <> Reply to Message

Hi All,

I have been refreshing my memory on how to show an image with one (or both) dimensions log-scaled. I want to use these to work with some 2-d histograms, where one of the dimensions has all the action at low values but I don't want to lose the high value info. Additionally, I want to work in direct graphics, but I am more than willing to leverage IDL's object graphics in the background.

I am using an IDLgrSurface with an image texture map to transform the image into log space. My problem seems to be matching the viewport size to the input image size as the result always has an white band around one side or the other. Where the white band ends up depends upon how much fiddling I do the viewport size - but I always get a white band. Grrrr.

Here's the basic outline of my steps...

1. Plot the pixel coordinate vectors - in this case I have a square image so I simply create indexed vectors that range from 1-n where n is the image size along either dimension.

2. Convert the plot region size (!X.window and !Y.window) to pixels. These determine the *output* image size.

3. Create an IDLgrSurface that is the same size as the *input* image. Texture map it with the image.

4. Create an IDLgrView with a viewport that covers the *input* data space (1-n in each dimension.)

5. Draw the view on an IDLgrBuffer that is the size of the desired *output* image.

6. Read the image from the buffer, TV it and then replot the axes and data.

The example code pasted below shows my troubles.

Thanks and cheers,
Ben
Mac OSX 10.4.11 with IDL 6.3

***START

```
PRO testlog, n
  if n_elements(n) EQ 0 then n=100
  image = BYTSCL(REPLICATE(1,n) # BINDGEN(n), TOP = 180)

  ;make a patch within the image to follow
  n2 = n/2.0
  dx = [-1,0,1,-1,0,1,-1,0,1]
  dy = [1,1,1,0,0,0,-1,-1,-1]
  image[n2+dx,n2+dy] = 0B

  ;determine the image size
  dim = SIZE(image,/DIM)
  if (SIZE(dim,/DIM)) EQ 2 then begin
    nx = dim[0] & ny = dim[1]
  endif else begin
    nx = dim[1] & ny = dim[2]
  endelse

  ;make up vectors of the pixel locations and
  ;create a plot
  ;add one to x and y just to avoid log of 0
  x = findgen(nx)+1
  y = findgen(ny)+1
  PLOT, x,y,/NODATA, /YLOG, $
    XSTYLE = 1, YSTYLE = 1

  ;determine the data region in pixels
  xyz = CONVERT_COORD(!X.window, !Y.window, $
    /NORM, /TO_DEV)
  onx = ROUND((xyz[0,1]-xyz[0,0]) + 1)
  ony = ROUND((xyz[1,1]-xyz[1,0]) + 1)

  ;create a surface
  s = REPLICATE(1,nx,ny)
  o = OBJ_NEW("IDLgrSurface", s, x, alog10(y), $
    color = [255,255,255], style = 2, $
    texture_map = obj_new("IDLgrImage", image))

  ;create a model
  model = OBJ_NEW('IDLgrModel')
  model->Add,o
  scale = ny / alog10(ny)
  model->Scale, 1.0, scale, 1.0

  ;create a view that is bounded by
  ;the data space - which is 1 to n
  view = OBJ_NEW('IDLgrView', $
    viewplane = [1,1,nx,ny])
```

```
  view->Add, model

  ;create a destination
  buffer = OBJ_NEW('IDLgrBuffer', $
    DIM = [onx,ony], graphics_tree = view)
  buffer->Draw,view

  ;get the image data
  image = buffer->Read()
  image->GetProperty, data = data
  OBJ_DESTROY, [view, buffer, image]

  ;show the image and data
  TV, data, xyz[0], xyz[1], $
    TRUE = 1, XSIZE = onx, YSIZE = ony
  PLOT, x,y, /YLOG, $
    XSTYLE = 1, YSTYLE = 1, /NOERASE
  ;PLOTS, x, y
end
***END
```