
Subject: Invalid Object Reference!

Posted by [Adam Rankin](#) on Fri, 21 Sep 2001 18:56:02 GMT

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

Hey guys, another one that seems to baffle me.

What I'm doing is loading mutiple images into a single WIDGET_DRAW using object graphics.

My first confession: I don't get object graphics, just know how to use em.

My second confession: I'm not getting paid enough to learn something more effecient than common blocks so =P

Now, back to business.

This code returns this error:

```
% IDLGRSRCDEST::DRAW: Invalid object reference:
```

```
    <ObjHeapVar8776>.
```

```
% Execution halted at: DISPLAYIMAGES    104
```

```
etc.. etc...
```

when i make a call to it via:

displayimages, image

where image is the data

array... [128,128,X] (x can be anynumber of images etc...)

however, when i call displayimages from loadimages (the first one)

it works like a charm.

so long short of it is, if i call displayimages from loadimages it

works... if i call it from anywhere else, it doesn't

HELP!

-Adam

here is the code:

sorry bout that formatting hope you can forgive me...

i took the two procedures that were valid, if you need to see others, just
reply

```
*****  
;  
;this procedure will load the images in from the ABL  
;this procedure reverses the images right side up
```

```
.*****  
,
```

PRO loadimages, Ev

```
COMMON graphics, ptrGraphStruct  
COMMON display, oWindow  
COMMON param, ptrParams  
COMMON data, nb, ns, nl, wl, oPalette, numRows, image
```

```
;create a palette to make it easy  
oPalette = obj_new('IDLgrPalette')  
;load the black and white color table in to the palette  
oPalette -> LoadCT, 0
```

```
ENVI_SELECT, title='Choose Images to Display', fid=fid, dims=dims,  
$  
  pos=pos, /FILE_ONLY
```

```
IF (fid eq -1) THEN RETURN ; return if "cancel" selected
```

```
ENVI_FILE_QUERY, fid, fname=fname, bname=bname, nb=nb, ns=ns,  
nl=nl, $  
  data_type=dtype, wl=wl
```

```
;define the number of rows to display  
numRows=FIX(nb/(*ptrParams).imagePerRow)
```

```
;check to see if one needs to be added  
IF ((FLOAT(nb)/FLOAT(*ptrParams).imagePerRow)) GE  
(FIX(nb/(*ptrParams).imagePerRow)+0.5)) THEN BEGIN  
  numRows = numRows + 1  
ENDIF
```

```
image=DBLARR(ns, nl, nb)
```

```
;load in the images into image and flip them around  
FOR i=0, nb-1 DO BEGIN  
  image[*,*,i] = ENVI_GET_DATA(fid=fid, dims=dims, pos=i)  
  image[*,*,i] = REVERSE(image[*,*,i], 2)  
END
```

```
(*ptrGraphStruct).loadImages='yes'
```

```
displayimages, image  
END
```

```
.*****  
,
```

```
;this procedure displays the images to a WIDGET_DRAW using oGraphics (lol)
```

```
.*****  
,
```

PRO displayimages, image

```
COMMON graphics, ptrGraphStruct  
COMMON display, oWindow  
COMMON param, ptrParams  
COMMON data, nb, ns, nl, wl, oPalette, numRows
```

```
image[*,*,*]=image[*,*,*]*((*ptrParams).brightness/100)
```

```
;create the view here
```

```
oImageView = obj_new('IDLgrView', LOCATION=[0,0],  
DIMENSIONS=[ns*(*ptrParams).imagePerRow,nl*numRows], $
```

```
VIEWPLANE_RECT=[0,0,ns*(*ptrParams).imagePerRow,nl*numRows], ZCLIP=[1,  
-1])
```

```
;define these as the image positions in the view
```

```
xImagePos=0  
yImagePos=(numRows*nl)-nl  
label='b='
```

```
;this statement creates an array of image objects
```

```
oImage=REPLICATE(obj_new('IDLgrImage', PALETTE=oPalette), nb)
```

```
;this loop defines and adds the images to the model
```

```
FOR i=0, nb-1 DO BEGIN  
  oImage[i] -> SetProperty, DATA=image[*,*,i]  
  oText = obj_new('IDLgrText',  
label+STRTRIM(STRING(FIX(wl[i])),2), $  
  LOCATION=[xImagePos+10, yImagePos+10],  
COLOR=(*ptrParams).color)  
  oImageModel = obj_new('IDLgrModel')  
  oImage[i] -> SetProperty, LOCATION=[xImagePos, yImagePos]
```

```
;this here decided it doesnt want to work...
```

```
oImageModel -> Add, oImage[i], /ALIAS  
oImageModel -> Add, oText  
oImageView -> Add, oImageModel  
xImagePos=xImagePos+ns  
IF (xImagePos GE ns*(*ptrParams).imagePerRow) THEN BEGIN  
  xImagePos=0  
  yImagePos=yImagePos-nl  
ENDIF  
END
```

```
WIDGET_CONTROL, (*ptrGraphStruct).draw, GET_VALUE=oWindow
```

```
;kill the window and redraw it.
OBJ_DESTROY, oWindow
draw = WIDGET_DRAW((*ptrGraphStruct).subBase, /BUTTON_EVENTS, $
  GRAPHICS_LEVEL=2,
XSIZE=ns*(*ptrParams).imagePerRow, YSIZE=nl*numRows)
(*ptrGraphStruct).draw = draw
WIDGET_CONTROL, (*ptrGraphStruct).subBase,
XSIZE=ns*(*ptrParams).imagePerRow+10
WIDGET_CONTROL, (*ptrGraphStruct).subBase, YSIZE=nl*numRows+10

;create and load the new image(s)
WIDGET_CONTROL, (*ptrGraphStruct).draw, GET_VALUE=oWindow
oWindow -> SetProperty, GRAPHICS_TREE=oImageView
oWindow -> Draw, oImageView

;clear out the crap that is no longer needed
OBJ_DESTROY, olImage[*]
OBJ_DESTROY, olImageView
OBJ_DESTROY, oPalette

;make sure that everyone knows that the draw's are now drawn.
(*ptrGraphStruct).drawFlag='1'
```

END
