
Subject: Once again, an error I don't understand
Posted by [Adam Rankin](#) on Fri, 30 Nov 2001 17:51:28 GMT
[View Forum Message](#) <> [Reply to Message](#)

% IDLGRMODEL::ADD: Objects can only have one parent at a
time: <ObjHeapVar3833(IDLGRIMAGE)>

sigh, wish I wasn't a n00b.
hate to waste your time, but the formation of an IDL programmer takes
time... (yukyuk)

-Adam

this is the code in which it takes place...

```
*****  
;  
;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)
  olImageModel = obj_new('IDLgrModel')
  olImage[i] -> SetProperty, LOCATION=[xImagePos, yImagePos]

;this here decided it doesnt want to work...
olImageModel -> Add, olImage[i], /ALIAS
olImageModel -> Add, oText
olImageView -> Add, olImageModel
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,
RETAIN=2, $
  GRAPHICS_LEVEL=2,
XSIZE=ns*(ptrParams).imagePerRow, YSIZE=nI*numRows)
(*ptrGraphStruct).draw = draw
WIDGET_CONTROL, (*ptrGraphStruct).subBase,
XSIZE=ns*(ptrParams).imagePerRow+10
WIDGET_CONTROL, (*ptrGraphStruct).subBase, YSIZE=nI*numRows+10

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

;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

;*****
;this procedure clears the widget draw and draws another one with no
images

```

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

PRO clearimages, Ev

```
COMMON display, oWindow  
COMMON graphics, ptrGraphStruct
```

```
IF ((*ptrGraphStruct).drawFlag EQ '1') THEN BEGIN  
  OBJ_DESTROY, oWindow  
  draw = WIDGET_DRAW((*ptrGraphStruct).subBase, $  
    GRAPHICS_LEVEL=2, XSIZE=400, YSIZE=200)  
  (*ptrGraphStruct).draw = draw  
  WIDGET_CONTROL, (*ptrGraphStruct).subBase, XSIZE=410  
  WIDGET_CONTROL, (*ptrGraphStruct).subBase, YSIZE=210  
ENDIF
```

END

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

;this simple procedure quits the application by calling /DESTROY to widget control

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

PRO quit, ev

```
COMMON graphics, ptrGraphStruct  
COMMON param, ptrParams  
COMMON display, oWindow
```

```
;clear out that stuff  
OBJ_DESTROY, oWindow  
PTR_FREE, ptrGraphStruct  
PTR_FREE, ptrParams  
WIDGET_CONTROL, Ev.top, /DESTROY  
END
```

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

;This procedure will load up a box to modify the preferences

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

PRO prefs, Ev

```
COMMON param, ptrParams
```

```
;create the gui that will have the preferences on it  
userValueBase = WIDGET_BASE(title='Please enter the parameters',  
/COLUMN)  
userValueSubBase = WIDGET_BASE(userValueBase, /COLUMN, XSIZE=250,
```

```

YSIZE=200)
  userValueImagePerRow = CW_FIELD(userValueSubBase, title='Number of
Images per Row', $
  /INTEGER, value=(*ptrParams).imagePerRow, XSIZE=5,
YSIZE=1)

;possible color choices
colors = [ 'Color', $
  'Blue', $
  'Green', $
  'Red', $
  'Black', $
  'White', $
  'Purple']

userValueColorBase = WIDGET_BASE(userValueSubBase, /ROW, XSIZE=50)
  userValueColorList = WIDGET_DROPLIST(userValueColorBase,
value=colors, title='Text Color:', $
  EVENT_PRO='paramsEvent', UVALUE='colorlist')
  userValueBrightSlider = WIDGET_SLIDER(userValueSubBase,
EVENT_PRO='paramsEvent', $
  MAXIMUM=200, MINIMUM=0, /FRAME, title='Contrast
Factor(%):', value=100, $
  UVALUE='slider', XSIZE=300)
  userValueButtonBase = WIDGET_BASE(userValueSubBase, /ROW,
XSIZE=75, YSIZE=30)
  userValueAccept = WIDGET_BUTTON(userValueButtonBase,
value='Accept', UVALUE='Accept', $
  EVENT_PRO='paramsEvent', YSIZE=25)
  userValueCancel = WIDGET_BUTTON(userValueButtonBase,
value='Cancel', UVALUE='Cancel', $
  EVENT_PRO='paramsEvent', YSIZE=25)

;get the previous data so it's not lost
WIDGET_CONTROL, userValueImagePerRow, GET_VALUE=imagePerRow
WIDGET_CONTROL, userValueBrightSlider, GET_VALUE=brightness
firstx=(*ptrParams).firstx
firsty=(*ptrParams).firsty
lastx=(*ptrParams).lastx
lasty=(*ptrParams).lasty
color=(*ptrParams).color

PTR_FREE, ptrParams

;create the structure to hold the necessary data
params= { imagePerRow:imagePerRow,
userValueBase:userValueBase, $
  userValueBrightSlider:userValueBrightSlider, $

```

```

    userValueImagePerRow:userValueImagePerRow, $
    userValueColorList:userValueColorList,
color:color, $
    brightness:brightness, $
    firstx:firstx, firsty:firsty, lastx:lastx,
lasty:lasty}

```

```
ptrParams = PTR_NEW(params, /NO_COPY)
```

```

WIDGET_CONTROL, userValueBase, /REALIZE
END

```

```

;*****
;
;this code retrives the inputted values and output to params.
;*****
;

```

```
PRO paramsEvent, Ev
```

```

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

```

```

WIDGET_CONTROL, Ev.id, GET_UVALUE=uvalue
WIDGET_CONTROL, (*ptrParams).userValueColorList, get_uvalue=value

```

```
CASE uvalue OF
```

```

    "Accept":BEGIN
        ;for some reason it wants me to do this in two
        steps... dunno why.
        WIDGET_CONTROL, (*ptrParams).userValueImagePerRow,
GET_VALUE=temp
        (*ptrParams).imagePerRow=temp

```

```

        ;clear dat shit.
        WIDGET_CONTROL, (*ptrParams).userValueBase,
/DESTROY

```

```

    IF ((*ptrGraphStruct).loadImages EQ 'yes') THEN
BEGIN
    ; displayimages, image
    ENDIF
END

```

```

    "Cancel":BEGIN
        WIDGET_CONTROL, (*ptrParams).userValueBase,
/DESTROY
    END

```

```

"slider":BEGIN
  WIDGET_CONTROL,
(*ptrParams).userValueBrightSlider, GET_VALUE=temp
  (*ptrParams).brightness=temp
END

"colorlist":BEGIN
  selcolor =
WIDGET_INFO((*ptrParams).userValueColorList, /DROPLIST_SELECT)

; yada yada yada, if it's X in the list make it X
color... RGB format
  IF (selcolor Eq 1) THEN BEGIN
;    R G B
    (*ptrParams).color=[0,0,255]
  ENDIF ELSE IF (selcolor Eq 2) THEN BEGIN
;    R G B
    (*ptrParams).color=[0,255,0]
  ENDIF ELSE IF (selcolor Eq 3) THEN BEGIN
;    R G B
    (*ptrParams).color=[255,0,0]
  ENDIF ELSE IF (selcolor Eq 4) THEN BEGIN
;    R G B
    (*ptrParams).color=[0,0,0]
  ENDIF ELSE IF (selcolor Eq 5) THEN BEGIN
;    R G B
    (*ptrParams).color=[255,255,255]
  ENDIF ELSE BEGIN
;    R G B
    (*ptrParams).color=[255,0,255]
  ENDELSE

END ;colorlist

END ;case

END

;*****
;
;this is here because the object graphics passes events to this procedure
;it wont pass it to what you want it too...
;*****

PRO multidisplay_event, ev

COMMON param, ptrParams
COMMON data, nb, ns, nl, wl, oPalette, numRows, image

```

```

IF (ev.type EQ 0) THEN BEGIN
  (*ptrParams).firstx=ev.x
  (*ptrParams).firsty=ev.y
ENDIF

```

```

IF (ev.type EQ 1) THEN BEGIN
  (*ptrParams).lastx=ev.x
  (*ptrParams).lasty=ev.y

```

```

IF ((*ptrParams).lasty LT (*ptrParams).firsty) THEN BEGIN
  ;decrease brightness
  (*ptrParams).brightness=$

```

```

  (*ptrParams).brightness/(((ptrParams).firsty-(ptrParams).lasty)/100)
ENDIF ELSE IF((*ptrParams).lasty GT (*ptrParams).firsty)

```

```

THEN BEGIN
  ;increase brightness
  (*ptrParams).brightness=$

```

```

(((ptrParams).lasty-(ptrParams).firsty)/100)*(ptrParams).brightness
ENDIF ELSE BEGIN
  ;do nothing
ENDELSE

```

```

ENDIF

```

```

END

```

```

.*****
,
;this is the main procedure, it loads the widgets and draws the
application
.*****
,

```

```

PRO multidisplay, Ev

```

```

;pass the common block
COMMON graphics, ptrGraphStruct
COMMON display, oWindow
COMMON param, ptrParams

```

```

topBase = WIDGET_BASE(Title='Multiple Displays', /COLUMN,
MBAR=bar)
subBase = WIDGET_BASE(topBase, XSIZE=600, YSIZE=480, /COLUMN)

```

```

fileMenu = WIDGET_BUTTON(bar, /MENU, value='File')
editMenu = WIDGET_BUTTON(bar, /MENU, value='Edit')

```

```

loadButtonMenu = WIDGET_BUTTON(fileMenu, Value='Load Images',
EVENT_PRO='loadimages')
clearButtonMenu = WIDGET_BUTTON(fileMenu, Value='Clear Images',
EVENT_PRO='clearimages')
quitButtonMenu = WIDGET_BUTTON(fileMenu, Value='Quit',
EVENT_PRO='quit')

prefButtonMenu = WIDGET_BUTTON(editMenu, Value='Preferences',
EVENT_PRO='prefs')

draw = WIDGET_DRAW(subBase, GRAPHICS_LEVEL=2, XSIZE=600,
YSIZE=480)

;first time through, and there are no images so set flag to 0
drawFlag='0'
loadImages='no'

;make a structure with all the widgets and pass the pointer
graphStruct = {topBase:topBase, subBase: subBase, draw:draw,
drawFlag:drawFlag, loadImages:loadImages}
ptrGraphStruct = PTR_NEW(graphStruct, /NO_COPY)

imagePerRow=5
color=[255,255,255]
brightness=100
firstx=0.0
firsty=0.0
lastx=0.0
lasty=0.0

params={imagePerRow:imagePerRow, color:color,
brightness:brightness, $
firstx:firstx, firsty:firsty, lastx:lastx, lasty:lasty}
ptrParams=PTR_NEW(params, /NO_COPY)

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

WIDGET_CONTROL, topBase, /REALIZE
XMANAGER, 'multidisplay', topBase

END

```
