Subject: Re: drawing lines in IDL
Posted by David Foster on Tue, 31 Mar 1998 08:00:00 GMT
View Forum Message <> Reply to Message

>
> Steve Hartmann (morph@vuse.vanderbilt.edu) writes:
>
>> Does anyone know how to use the mouse to draw free hand lines over a
>> displayed image?
>

You might take a look at the DEFROI function, or the DEFROI2 function
that I've included below, which is a modified and enhanced version
of DEFROI. I've also included DEFROIS, which simply allows you to
call DEFROI2 iteratively when you want multiple ROI's.

```
 ********************************************************** ********
 D E F R O I 2
 ********************************************************** ********

; DEFROI2.PRO  10-25-95
;
;+
; IMPORTANT NOTE:
;
;  THIS ROUTINE IS OUR OWN "INTERNAL" VERSION OF THE DEFROI() IDL USER'S
;    LIBRARY ROUTINE. WE HAVE MADE ADDITIONS TO THE ROUTINE, AND,
;    UNFORTUNATELY, HAVE HAD TO MAKE CORRECTIONS AS WELL!
;
;  Most notably, the indices of the roi returned, and any optional
;    vertices returned, correspond to the ORIGINAL image, without
;    the zoom factor being applied. This behavior is different
;    from IDL's DEFROI() which returns indices and vertices
;    that correspond to the zoomed image.
;
;
; NAME:        DEFROI2
; PURPOSE:  Define an irregular region of interest of an image
;       using the image display system and the cursor/mouse.
; CATEGORY: Image processing.
; CALLING SEQUENCE:
;   R = Defroi(Sx, Sy, X0, Y0)
; INPUTS:
;   Sx, Sy = Size of image, in pixels.
; Optional Inputs:
;   X0, Y0 = Coordinate of Lower left corner of image on display.
;   If omitted, (0,0) is assumed.  Screen device coordinates.
```

```
;   ZOOM = zoom factor, if omitted, 1 is assumed.
; OUTPUTS:
;   Function result = vector of subscripts of pixels inside the region.
;   Side effect: The lowest bit in which the write mask is enabled
;   is changed.
; OPTIONAL OUTPUTS:
;   Xverts, Yverts = Optional output parameters which will contain
;       the vertices enclosing the region. These always correspond
;       to the original image, without the zoom factor being applied.
; KEYWORD Parameters:
;   ALT_WINDOWS = set to window ID(s) of alternate windows in which the
;       polygon(s) will be drawn in XOR mode; these are erased at the
;       end of the routine. It is currently assumed all windows are the
;       same size.
;   COLOR = if set, polygon is displayed using specified color. Note
;       that this implies that the middle cursor button may NOT be used
;       to erase the last designation!
;   MIDDLE_QUIT = if set, the middle mouse button will close the polygon
;       and quit the polygon selection. If argument RTNVAL is supplied
;       the value of !ERR which corresponds to the button used to close
;       the polygon is returned.
;   NOFILL = if set, inhibits filling of irregular region on completion.
;   NOREGION = Setting NOREGION inhibits the return of the
;       pixel subscripts.
;   QUIET = if set, suppresses display of instructions on STDIO.
;   RESTORE = if set, original image on display is restored to its
;       original state on completion.
;   PREPEND, APPEND = set these to coordinates to insert before and
after
;       those chosen by user (use: designation of midline for example).
;   TEXT_ID = set to widget id of text widget to have current cursor
;       coordinates displayed while taking points.
;
; COMMON BLOCKS:
;   None.
; SIDE EFFECTS:
;   Display is changed if RESTORE is not set. If COLOR is set, display
;   is changed regardless.
; RESTRICTIONS:
;   Only works for interactive, pixel oriented devices with a
;       cursor and an exclusive or writing mode.
;   A region may have at most 1000 vertices.  If this is not enough
;       edit the line setting MAXPNTS.
; PROCEDURE:
;   The exclusive or drawing mode is used to allow drawing and
;   erasing objects over the original object.
;   If COLOR is set then the copy mode is used.
;
```

```
;   The operator marks the vertices of the region, either by
;       dragging the mouse with the left button depressed or by
;       marking vertices of an irregular polygon by clicking the
;       left mouse button, or with a combination of both.
;   The center button removes the most recently drawn points, unless
;       COLOR is specified, in which case the middle button is disabled.
;   Press the right mouse button when finished.
;   When the operator is finished, the region is filled using
;       the polyfill function, and the polyfillv function is used
;       to compute the subscripts within the region.
;
; MODIFICATION HISTORY:  DMS, March, 1987.
;   Revised for SunView, DMS, Nov, 1987.
;       Added additional argument checking, SNG April, 1991
;   Modified for devices without write masks: DMS, March, 1992.
;       Uses exclusive or mode rather than write masks.
;   Return correct subscripts when using ZOOM keyword, DSFoster,
;       Dec, 1993.
;   Corrected so that correct subscripts and vertices are
;       computed when !order=1 and ZOOM not equal to 1,
;       DSFoster, Dec, 1993.
;   Keyword QUIET added to disable instructions being sent to
;       console, DSF, Feb, 1994.
;   Keyword COLOR added to allow specification of drawing color.
;       If set, the copy graphics mode is used, and the display
;       is changed.
;   Keyword MIDDLE_QUIT added to add flexibility in calling module.
;       Implies that middle button has same function as right.
;       Done to support multiple region designations in calling
;       module. Added RTNVAL as 5th argument, to return the
;       button-press used to close the polygon. This argument
;       is optional; it will not be set if undefined. DSF, Mar, 1994.
;   Keywords PREPEND and APPEND added to introduce coordinates
;       before and after the ones chosen interactively; this is
;       used when, say, you want the user to designate a midline
;       only and have it converted into a polygon.
;
;       Improved precision of arithmetic operations to improve
;       correspondence between device coors and image coors. Added
arrays
;       to hold device coordinates for improved plotting accuracy.
;       DSF, 9-26-94.
;   Keyword TEXT_ID added to have cursor coordinates displayed in
;       text widget specified by TEXT_ID.
;
;-
;
 ;------------------------------------------------------- ---------------
```

```
;   procedure DEFROI2_PLOT_ALT
;
; Procedure to plot lines in alternate window(s) specified by keyword
; ALT_WINDOWS in call to DEFROI2().
 ;------------------------------------------------------------ ---------------

PRO defroi2_plot_alt, windows, xcoors, ycoors, color, g_mode

old_window = !d.window

if (g_mode ne 6) then device, get_graphics=tempg, set_graphics=6   ; Set
XOR

for i = 0, n_elements(windows) - 1 do begin
   if (windows(i) ne -1L) then begin
      wset, windows(i)
      plots, xcoors, ycoors, /dev, color=color, /noclip
   endif
endfor

if (g_mode ne 6) then device, set_graphics=tempg
if (old_window ne -1) then wset, old_window

return
END

 ;------------------------------------------------------------ ---------------
;   function DEFROI2
;
; Main function.
 ;------------------------------------------------------------ ---------------

Function Defroi2, Sx, Sy, Xverts, Yverts, rtnval, X0=x0, Y0=y0,
ZOOM=ZOOM, $
   NOREGION=Noregion, NOFILL=Nofill, RESTORE=restore, QUIET=quiet, $
   COLOR=color, MIDDLE_QUIT=middle_quit, PREPEND=prepend,
APPEND=append, $
   TEXT_ID=text_id, ALT_WINDOWS=alt_windows

on_error,2                          ; Return to caller if error

if (keyword_set(TEXT_ID)) then begin        ; Check for valid widget ID
(DSF)
   if (widget_info(text_id, /valid_id) eq 0) then $
      text_id = 0
endif

if (keyword_set(ALT_WINDOWS)) then begin    ; Check whether windows IDs
```

```
valid
   device, window_state=windows            ;  DSF
   n_windows = n_elements(alt_windows)
   for i = 0, n_windows - 1 do begin
      if (windows(alt_windows(i)) ne 1) then $
         alt_windows(i) = -1L
   endfor
endif

if (keyword_set(COLOR)) then begin          ; DSF
   if (color gt !d.table_size-1) then $
      message, 'Only ' + strtrim(string(!d.table_size-1),2) + $
         ' colors available ' + '(specified: ' + $
         strtrim(string(color),2) + ')'
   nc1 = color
   if (keyword_set(RESTORE)) then $
      message, 'Cannot use keywords COLOR and RESTORE together'
endif else begin
   nc1 = !d.table_size-1   ;# of colors available
endelse

if sx lt 1 or sy lt 1 then $          ; Check some obvious things
   message, 'Dimensions of the region must be greater than zero.'

if sx gt !d.x_size then $
   message, 'The width of the region must be less than ' + $
         strtrim(string(!d.x_size),2)

sy = sy < !d.y_size

if (keyword_set(COLOR)) then begin              ; Save current graphics
mode
   device, get_graphics=oldg, set_graphics=3    ; Copy mode (overwrite)
DSF
   g_mode = 3
endif else begin
   device, get_graphics=oldg, set_graphics=6    ; Set XOR mode
   g_mode = 6
endelse

repeat cursor,x,y,/dev,/nowait until !err eq 0    ; Flush cursor buffer
DSF


AGAIN:

if (not keyword_set(quiet)) then begin    ; DSF
   print,'Left button to mark point'
```

```
    if (keyword_set(MIDDLE_QUIT)) then begin
       print,'Middle or right button to close region'
    endif else if (not keyword_set(COLOR)) then begin
       print,'Middle button to erase previous point'
       print,'Right button to close region'
    endif else begin
       print,'Right button to close region'
    endelse
endif

n = 0                       ; Number of points taken
maxpnts = 2000              ; Max # of points.
xverts = intarr(maxpnts)      ; Arrays of coordinates
yverts = intarr(maxpnts)
dev_xverts = intarr(maxpnts)    ; Arrays for device coordinates
dev_yverts = intarr(maxpnts)
xprev = -1
yprev = -1
if n_elements(x0) le 0 then x0 = 0
if n_elements(y0) le 0 then y0 = 0
if n_elements(zoom) le 0 then zoom = 1
fzoom = float(zoom)

if (keyword_set(PREPEND)) then begin          ; Insert points at
beginning
    info = size(prepend)
    if (info(0) ne 2 or info(1) ne 2) then $
       message, 'Must set PREPEND to INTARR(2,?)'
    pts = info(2)
    xverts(0:pts-1) = fix(round(prepend(0,*) / fzoom))
    yverts(0:pts-1) = fix(round(prepend(1,*) / fzoom))
    dev_xverts(0:pts-1) = prepend(0,*)
    dev_yverts(0:pts-1) = prepend(1,*)
    n = pts
    plots,  dev_xverts(0:n-1), dev_yverts(0:n-1), /dev, color=nc1,
/noclip

    if (keyword_set(ALT_WINDOWS)) then begin    ; Plot in alternate
window(s)
       defroi2_plot_alt, alt_windows, dev_xverts(0:n-1), $
             dev_yverts(0:n-1), nc1, g_mode
    endif
endif


; Get first point. Display coordinates in text widget if specified

if (keyword_set(TEXT_ID)) then begin
```

```
   !err = 0
   while (!err eq 0) do begin
      cursor, xxx, yyy, /CHANGE, /DEV
      msg = '[' + strtrim(xxx,2) + ', ' + strtrim(yyy,2) + ']'
      WIDGET_CONTROL, text_id, set_value=msg, /no_copy
   endwhile
endif else begin
   cursor, xxx, yyy, /WAIT, /DEV
endelse

plots, [xxx,xxx], [yyy,yyy], /dev, color=nc1, /noclip     ; Mark first
point

if (keyword_set(ALT_WINDOWS)) then begin    ; Plot in alternate
window(s)
   defroi2_plot_alt, alt_windows, [xxx,xxx], [yyy,yyy+1], nc1, g_mode
endif

repeat begin
   if (n_elements(cursor_x)) then begin               ; Update cursor
coors
      cursor_x = xxx - x0
      cursor_y = yyy - x0
   endif
   xx = fix(round((xxx - x0) / fzoom))               ; To image
coords
   yy = fix(round((yyy - y0) / fzoom))
   if (xx lt sx) and (yy lt sy) and (!err eq 1) and $
      ((xx ne xprev) or (yy ne yprev)) then begin     ; New point?
      xprev = xx
      yprev = yy
      if (n ge (maxpnts-1)) then begin
         print,'^GDEFROI2(): Too many points'
         n = n-1
      endif
      xverts(n) = xx                           ; Image
coordinates
      yverts(n) = yy
      dev_xverts(n) = xxx                        ; Device
coordinates
      dev_yverts(n) = yyy

      if (n ne 0) then begin
         plots, dev_xverts(n-1:n), dev_yverts(n-1:n), /dev, $
               color=nc1, /noclip
         if (keyword_set(ALT_WINDOWS)) then begin    ; Alternate
window(s)
            defroi2_plot_alt, alt_windows, dev_xverts(n-1:n), $
```

```
                    dev_yverts(n-1:n), nc1, g_mode
          endif
       endif

       n = n + 1
    endif

    ; We use 2 or 5 for the middle button because some Microsoft
    ; compatible mice use 5.

    if ((!err eq 2) or (!err eq 5)) and (n gt 0) and $
          not keyword_set(MIDDLE_QUIT) then begin
       if (not keyword_set(COLOR)) then begin         ; DSF
          n = n-1
          if (n gt 0) then begin                    ; Remove a vertex
             plots, dev_xverts(n-1:n), dev_yverts(n-1:n), $
                color=nc1, /dev, /noclip
             if (keyword_set(ALT_WINDOWS)) then begin    ; Alternate
window
                defroi2_plot_alt, alt_windows, dev_xverts(n-1:n), $
                      dev_yverts(n-1:n), nc1, g_mode
             endif
          endif
       endif
    endif
    if (keyword_set(TEXT_ID)) then begin
       !err = 0
       while (!err eq 0) do begin
          cursor, xxx, yyy, /CHANGE, /DEV
          msg = '[' + strtrim(xxx,2) + ', ' + strtrim(yyy,2) + ']'
          WIDGET_CONTROL, text_id, set_value=msg, /no_copy
       endwhile
    endif else begin
       cursor, xxx, yyy, /WAIT, /DEV
    endelse
endrep until ( !err eq 4 or $
       ((!ERR eq 2 or !ERR eq 5) and keyword_set(MIDDLE_QUIT)) )


if (keyword_set(APPEND)) then begin          ; Points inserted at end
    info = size(append)
    if (info(0) ne 2 or info(1) ne 2) then $
       message, 'Must set APPEND to INTARR(2,?)'
    pts = info(2)
    xverts(n:n+pts-1) = fix(round(append(0,*) / fzoom))
    yverts(n:n+pts-1) = fix(round(append(1,*) / fzoom))
    dev_xverts(n:n+pts-1) = append(0,*)
    dev_yverts(n:n+pts-1) = append(1,*)
```

```
   plots, dev_xverts(n-1:n+pts-1), dev_yverts(n-1:n+pts-1), $
          /dev, color=nc1, /noclip

   if (keyword_set(ALT_WINDOWS)) then begin    ; Alternate window(s)
      defroi2_plot_alt, alt_windows, dev_xverts(n-1:n+pts-1), $
            dev_yverts(n-1:n+pts-1), nc1, g_mode
   endif
   n = n + pts
endif

if (n_params() gt 4) then rtnval = !ERR    ; Return button-press

if (n lt 3) then begin
   print,'ROI - Must have 3 points for region.  Try again.'
   goto, AGAIN
endif
xverts = xverts(0:n-1)                    ; Truncate
yverts = yverts(0:n-1)
dev_xverts = dev_xverts(0:n-1)
dev_yverts = dev_yverts(0:n-1)

; Erase polygons in alternate window if necessary.

if (keyword_set(ALT_WINDOWS)) then begin
   defroi2_plot_alt, alt_windows, dev_xverts, dev_yverts, nc1, g_mode
endif

if keyword_set(RESTORE) then begin
   plots, dev_xverts, dev_yverts, /dev, color=nc1, /noclip
endif else if keyword_set(NOFILL) then begin
   plots, [dev_xverts(0),dev_xverts(n-1)],
[dev_yverts(0),dev_yverts(n-1)], $
          /dev, color = nc1,/noclip          ; Complete polygon
endif else begin
   polyfill, dev_xverts, dev_yverts, /dev, color=nc1, /noclip
endelse

; DSFoster modification: divide SY by ZOOM below. Before if
;    !order=1 and ZOOM<>1 then YVERTS inverted incorrectly;
;    you have to scale the size of the image (SY) by ZOOM.
;    Use FIX() to avoid conversion to LONG.

if (!order ne 0) then $
   yverts = fix(sy/ZOOM) - 1 - yverts       ; Invert Y?

device,set_graphics=oldg                    ; Re-enable original
graphics mode
```

```
; DSFoster modification: divide SX and SY by ZOOM below so that the
;   correct subscripts are returned when using ZOOM.

if keyword_set(NOREGION) then begin
   a = 0
endif else begin
   a = polyfillv(xverts,yverts,sx/ZOOM,sy/ZOOM)   ; Get subscripts
inside area
endelse

if (keyword_set(TEXT_ID)) then $
   WIDGET_CONTROL, text_id, set_value=''

return, a
end


 ************************************************************ ********
 D E F R O I S
 ************************************************************ ********

;   DEFROIS.PRO   11-18-97  DSFoster
;
; Routine to allow user to choose multiple regions of interest from
; the current graphics device. Calls DEFROI2.PRO, our own modified
; version of the IDL DEFROI procedure. Returns the indices of the
; ROI's matrix locations, just like DEFROI.
; An array of long integers will be returned as argument PIXELS, and
; this array will contain the number of pixels chosen in each
; region designated; its dimension will be the number of regions chosen.

;  9-26-94 DSF Add keywords PREPEND and APPEND so they may be passed
;         on to DEFROI2()  (see DEFROI2 for details).
; 12-01-94 DSF Add TEXT_ID keyword to pass to DEFROI2() to print
;         current cursor locations in specified text widget.
;  5-18-95 DSF Add keyword ALT_WINDOWS to enable the plotting of
;         polygons on alternate windows in DEFROI2().
; 11-18-97 DSF Add keyword NOFILL to prevent polygon from being filled.


FUNCTION defrois, xdim,ydim,xcoors,ycoors,pixels,ZOOM=zoom,COLOR=color,
$
       PREPEND=prepend, APPEND=append, TEXT_ID=text_id, $
       ALT_WINDOWS=alt_windows, NOFILL=nofill

if (not keyword_set(TEXT_ID)) then text_id = 0
if (not keyword_set(ALT_WINDOWS)) then alt_windows = 0
```

```
max_poly = 100
pixels = LONARR(max_poly)
DEVICE, CURSOR_STANDARD=68
chosen = 0

AGAIN:

roi =  defroi2(xdim,ydim,xc,yc,ret,zoom=zoom,color=color,/middle_qu it, $
        PREPEND=prepend, APPEND=append, TEXT_ID=text_id, $
        ALT_WINDOWS=alt_windows, NOFILL=nofill)

pixels(chosen) = n_elements(roi)
if (n_elements(all_rois) eq 0) then begin
    all_rois = roi
    xcoors = xc
    ycoors = yc
endif else begin
    all_rois = [all_rois, roi]
    xcoors = [xcoors, xc]
    ycoors = [ycoors, yc]
endelse

chosen = chosen + 1
if (chosen eq max_poly) then begin
    message, 'Maximum of ' + strtrim(max_poly,2) + ' polygons allowed',
$
            /continue
endif else if (ret eq 2 or ret eq 5) then begin
    goto, AGAIN
endif

DEVICE, /CURSOR_ORIGINAL
pixels = pixels(0:chosen-1)

return, all_rois

END
```

--

~~~~~~ ~~~~~~
David S. Foster   Univ. of California, San Diego
Programmer/Analyst   Brain Image Analysis Laboratory
foster@bial1.ucsd.edu Department of Psychiatry
(619) 622-5892   8950 Via La Jolla Drive, Suite 2240
La Jolla, CA 92037
~~~~~~ ~~~~~~

Subject: Re: drawing lines in IDL
Posted by davidf on Tue, 31 Mar 1998 08:00:00 GMT
View Forum Message <> Reply to Message

Steve Hartmann (morph@vuse.vanderbilt.edu) writes:

> Does anyone know how to use the mouse to draw free hand lines over a
> displayed image?

Here is an example that involved some simple modifications of
the example program to draw rubberband boxes that appears on my
web page.

Click and draw with the LEFT mouse button to draw free-hand
lines on the image. Click and drag with the RIGHT mouse
button to draw straight lines on the image. Among other
things this example illustrates is how you might go about
collecting points for creating a free-hand region of interest
using pointers.

Cheers,

David

```
************************************************************
PRO Example_Button_Events, event

; All program button events handled here.

Widget_Control, event.top, Get_UValue=info, /No_Copy

   ; Which button caused this event?

Widget_Control, event.id, Get_Value=thisButtonValue
CASE thisButtonValue OF

   'Quit': BEGIN
      Widget_Control, event.top, /Destroy
      RETURN
      ENDCASE

   'Erase Lines': BEGIN
      WSet, info.wid
      TV, BytScl(info.image, Top=info.drawColor-1)
      ENDCASE

ENDCASE
Widget_Control, event.top, Set_UValue=info, /No_Copy
END
```

```
;------------------------------------------------------------ -------


PRO Example_Draw_Events, event

; All program draw widget events handled here.

   ; Deal only with DOWN, UP, and MOTION events.

IF event.type GT 2 THEN RETURN

   ; Get the info structure.

Widget_Control, event.top, Get_UValue=info, /No_Copy

   ; What kind of event is this?

eventTypes = ['DOWN', 'UP', 'MOTION']
thisEvent = eventTypes[event.type]

whichButton = ['NONE', 'LEFT', 'MIDDLE', 'NONE', 'RIGHT']

CASE thisEvent OF

  'DOWN': BEGIN

      ; Which button was used? LEFT or RIGHT?

    info.buttonUsed = whichButton(event.press)

      ; Turn motion events on for the draw widget.

    Widget_Control, info.drawID, Draw_Motion_Events=1

      ; Create a pixmap. Store its ID. Copy window contents into it.

    Window, /Free, /Pixmap, XSize=info.xsize, YSize=info.ysize
    info.pixID = !D.Window
    Device, Copy=[0, 0, info.xsize, info.ysize, 0, 0, info.wid]

      ; Initialize the starting coordinates of the line.

    IF info.buttonUsed EQ 'RIGHT' THEN BEGIN
      info.xstart = event.x
      info.ystart = event.y
    ENDIF ELSE BEGIN
      info.xvalues = Ptr_New([event.x])
```

```
      info.yvalues = Ptr_New([event.y])
    ENDELSE

    ENDCASE

  'UP': BEGIN

      ; Erase the last line drawn. Destroy the pixmap.

    WSet, info.wid
    Device, Copy=[0, 0, info.xsize, info.ysize, 0, 0, info.pixID]
    WDelete, info.pixID

      ; Turn draw motion events off. Clear events queued for widget.

    Widget_Control, info.drawID, Draw_Motion_Events=0, Clear_Events=1

      ; Draw the final line.

    IF info.buttonUsed EQ 'RIGHT' THEN BEGIN

      PlotS, [info.xstart, event.x], [info.ystart, event.y], $
        /Device, Color=info.drawColor

      ; Reinitialize the line starting coordinates.

      info.xstart = -1
      info.ystart = -1

    ENDIF ELSE BEGIN

      PlotS, *info.xvalues, *info.yvalues, /Device, $
         Color=info.drawColor

      ; Reinitialize the pointers.

      Ptr_Free, info.xvalues
      Ptr_Free, info.yvalues
      info.xvalues = Ptr_New()
      info.yvalues = Ptr_New()

    ENDELSE
    ENDCASE

  'MOTION': BEGIN

      ; Here is where the actual line is drawn and erased.
      ; First, erase the last line.
```

```
      WSet, info.wid
      Device, Copy=[0, 0, info.xsize, info.ysize, 0, 0, info.pixID]

      IF info.buttonUsed EQ 'RIGHT' THEN BEGIN

        ; Draw a straight line.

      PlotS, [info.xstart, event.x], [info.ystart, event.y], /Device, $
          Color=info.drawColor

      ENDIF ELSE BEGIN

        ; Get the points of the new free-hand line and draw it.

      *info.xvalues = [*info.xvalues, event.x]
      *info.yvalues = [*info.yvalues, event.y]
      PlotS, *info.xvalues, *info.yvalues, /Device, Color=info.drawColor

      ENDELSE

      ENDCASE

ENDCASE

   ; Store the info structure.

Widget_Control, event.top, Set_UValue=info, /No_Copy
END
 ;------------------------------------------------------------- -------



PRO Example

   ; Open an image data set.

file = Filepath(SubDirectory=['examples','data'], 'ctscan.dat')
OpenR, lun, file, /Get_Lun
image = BytArr(256, 256)
ReadU, lun, image
Free_Lun, lun

xsize = (Size(image))[1]
ysize = (Size(image))[2]

   ; Create the TLB.
```

```
tlb = Widget_Base(Title='Free-Hand Lines in a Widget Program', $
   MBar=menubase)

   ; Create some menu bar buttons.

fileID = Widget_Button(menubase, Value='File', $
   Event_Pro='Example_Button_Events')
eraseID = Widget_Button(fileID, Value='Erase Lines')
quitID = Widget_Button(fileID, Value='Quit')

   ; Create the draw widget graphics window. Turn button events ON.

drawID = Widget_Draw(tlb, XSize=xsize, YSize=ysize, Button_Events=1, $
   Event_Pro='Example_Draw_Events')

   ; Realize widgets and make draw widget the current window.

Widget_Control, tlb, /Realize
Widget_Control, drawID, Get_Value=wid
WSet, wid

   ; Load drawing color and display the image.

drawColor = !D.N_Colors-1
TVLCT, 255, 255, 0, drawColor
TV, BytScl(Image, Top=drawColor-1)

   ; Create an "info" structure with information to run the program.

info = { image:image, $       ; The image data.
         wid:wid, $           ; The window index number.
         drawID:drawID, $     ; The draw widget identifier.
         pixID:-1, $          ; The pixmap identifier.
         xsize:xsize, $       ; The X size of the graphics window.
         ysize:ysize, $       ; The Y size of the graphics window.
         xstart:-1, $         ; The starting X coordinate of the line.
         ystart:-1, $         ; The starting Y coordinate of the line.
         xvalues:Ptr_New(), $  ; The X coordinates of the free-hand line.
         yvalues:Ptr_New(), $  ; The Y coordinates of the free-hand line.
         buttonUsed:'NONE', $  ; A flag to indicate which button is used.
         drawColor:drawColor } ; The rubberband box color.

   ; Store the info structure.

Widget_Control, tlb, Set_UValue=info, /No_Copy

   ; Start the program going.
```

```
XManager, 'example', tlb, /No_Block
END
```

David Fanning, Ph.D.
Fanning Software Consulting
E-Mail: davidf@dfanning.com
Phone: 970-221-0438
Coyote's Guide to IDL Programming: http://www.dfanning.com/