
Subject: Re: widget for image display

Posted by [knight](#) on Fri, 03 Mar 1995 17:28:02 GMT

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

In article <3idauo\$8mi@testnews.ll.mit.edu>, knight@ll.mit.edu writes:

|>

|> Currently, I use Bill Thompson's image_display routines. They are great!

|> They provide the ability to display quickly images to screen or Postscript file

|> with many capabilities, including grids of images, all sorts of global

|> keywords, and overplotting of information.

And, because some people asked me, they are available via anonymous ftp from
idlastro.gsfc.nasa.gov in contrib/thompson

|>

|> I'm hungry for another capability: a widget to display an image with lots of

|> characteristics, namely cuts, histogram, regions of interest, and statistics.

|> I have a first-cut at a widget but was hoping that somebody else has already

|> done what I want. To be specific, does anybody have something like this?

Nobody responded, so I wrote look.pro, which I attach below. It works well;
see the header for details. Please note it requires plot_image.pro from
idlastro.gsfc.nasa.gov in the contrib/thompson/image_display directory. I'd
appreciate feedback.

Fred

```
;+
```

```
; Name:
```

```
; look
```

```
; Purpose:
```

```
; This procedure produces a widget to view an image or series of
```

```
; images. You get a grid of 2x2 plots with menus on both sides.
```

```
; The plots are dynamic: horizontal and vertical cuts are updated as
```

```
; you move the mouse over the tv image in the lower left and the upper
```

```
; right shows another type of plot that you choose, e.g., shade_surf,
```

```
; contour, or show3. Click and drag the left mouse to zoom and roam
```

```
; in the zoom window. Click the middle mouse to recenter.
```

```
; Read more on operation below. Here's a rendition:
```

```
;
```

```
;
```

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

```
; |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
```

```
; | m | horizontal | optional plot | m |
```

```
; | e |   cut   | zoom region | e |
```

```
; | n |   |   |   |   |   |   | n |
```

```
; | u |-----| u |
```

```
; | s |   |   |   |   |   |   | s |
```

```
; |   |   tv   | vertical |   |
```

```
; |   | full image | cut   |   |
```

```

; |   |   |   |   |
; -----
; |   instructions or comments   |
; -----
; Example:
; look ; test with dist(256)
; look,images ; display series of 2-D images
; Operation:
; 1. The basic thing is to roam around the image and look at cuts.
; 2. The optional plot buttons can be selected at any time.
; 3. The optional plot window is also the zoom window.
; 4. To zoom, click and drag the left mouse over the region of interest.
;    Then you can roam in the zoom image also.
; 5. To change the center of the zoom region, click the middle mouse.
; 6. Moving the sliders at the bottom changes the plot and tv scales.
; 7. There are guards against choosing out-of-range quantities.
; Usage:
; look[,image][,./help][,options]
; Optional Inputs:
; image = one or more 2-D images
; Keywords:
; help = flag to print the header only and return
; verbose = flag to print informational messages (debug mostly)
; group = parent of this widget
; Outputs:
; widget as rendered above
; Common blocks:
; look = variables that need to be passed between routines
; Procedure:
; There are two top-level routines: one to initialize (look) and an
; event handler (look_event). After creating the widgets,
; look places the full image in the tv window and a histogram
; in the optional plots window. The cut windows are left blank
; until the mouse is moved over the image in the tv window.
; The event loop does all the dynamic processing.
; Restrictions:
; Requires plot_image.pro from idlastro.gsfc.nasa.gov in the
; contrib/thompson/image_display directory
; A large image (>512) with a large aspect ratio (>2, e.g. 512x1024)
; will fail because there's no guard, which needs to be added as a fix.
; There is no access to color tables.
; Some quantities are not updated when a new image is loaded, e.g.,
; plot min and plot max. This can and should be fixed.
; It would be nice to have more control over the slider increments.
; Currently, the plot min slider has an increment equal to its range
; divided by its length in pixels. Sometimes, you want finer control.
; You can't plot from inside look.pro.
; ToDo:

```

```

; report statistics
; report pixel values when scanning over the cut plots
; allow tweaking with keyboard input, e.g. up arrow=move up one pixel
; add a display of the actual values in the array
; make it a compound widget
; Modification history:
; write, 21-26 Feb 95, FKKnight (knight@ll.mit.edu)
; add roaming in the zoom image, 27 Feb-1 Mar 95, FKK
; fix bug if middle mouse is pressed first, 1 Mar 95, FKK
;-
;
;
; =====
; =====>> COORDINATE CONVERTERS: device <--> data
; =====
;
;
; data --> device
; given data coords, return slopes & intercepts to convert device to data
pro look_si,x,y,slopes,intercepts
dev = convert_coord([x(0),x(1)],[y(0),y(1)],/data,/to_device)
slopes = [(x(1)-x(0))/(dev(0,1)-dev(0,0)),(y(1)-y(0))/(dev(1,1)-dev(1,0))]
intercepts = [x(0)-dev(0,0)*slopes(0),y(0)-dev(1,0)*slopes(1)]
return
end
; device --> data
; given device coords, return data coords
FUNCTION look_cc,coord,slope,intercept
return,fix(slope*coord+intercept+0.5) ; want integer; it's a pixel number.
END
;
;
; =====
; =====>> DECIDE IF CUTS NEED UPDATING AND DO IT
; =====
;
;
pro look_cut,tmp
common look,thisimage,zoomimage,images,ws,all,zoom,pan
wset,tmp.id
cursor,xn,yn,0,/dev
xm = look_cc(xn,tmp.slopes(0),tmp.intercepts(0))+tmp.xrange(0)
ym = look_cc(yn,tmp.slopes(1),tmp.intercepts(1))+tmp.yrange(0)
; print,format = '($,4i5,a)',xm,ym,xm-tmp.last(0),ym-tmp.last(1),':'
IF ((xm<tmp.xrange(1)>tmp.xrange(0)) EQ xm) AND ((ym<tmp.yrange(1)>tmp.yrange(0)) EQ ym)
THEN BEGIN
newy = (ym NE tmp.last(1))
IF newy THEN BEGIN
wset,tmp.hcutid
xdata = lindgen(tmp.width(0))+tmp.xrange(0)
ydata = thisimage(tmp.xrange(0):tmp.xrange(1),ym)
plot,xdata,ydata,xstyle = 1,psym = 10*(n_elements(xdata) LT 100) $

```

```

    ,ystyle = 1,position = tmp.position,yrange = tmp.tvrange,xrange = tmp.xrange
    tmp.last(1) = ym
ENDIF
newx = (xm NE tmp.last(0))
IF newx THEN BEGIN
    wset,tmp.vcutid
    xdata = thisimage(xm,tmp.yrange(0):tmp.yrange(1))
    ydata = lindgen(tmp.width(1))+tmp.yrange(0)
    IF n_elements(where( $      ; bug: need 2 pts in xrange
        (xdata GE tmp.tvrange(0)) AND (xdata LE tmp.tvrange(1)))) GE 2 THEN $
        plot,xdata,ydata,xstyle = 1,psym = 10*(n_elements(xdata) LT 100) $
        ,ystyle = 1,position = tmp.position $
        ,xrange = tmp.tvrange,yrange = tmp.yrange
    tmp.last(0) = xm
ENDIF
IF newx OR newy THEN $
    widget_control,ws.mousetext,set_value = $
    ['mouse','x='+strtrim(xm,2),'y='+strtrim(ym,2) $
    ,strtrim(thisimage(xm,ym),2)]
ENDIF
return
END
;
;
; =====
; =====>> MAKE ZOOM REGION AND DISPLAY
; =====
;
;
PRO look_zoom
common look,thisimage,zoomimage,images,ws,all,zoom,pan
;
; =====>> Make the zoom image, display it, and get coords
;
zoomimage = thisimage(zoom.xrange(0):zoom.xrange(1) $
    ,zoom.yrange(0):zoom.yrange(1))
wset,zoom.id
plot_image,zoomimage>all.tvrange(0)<all.tvrange(1),origin = [0,0],scale = 1
zoom.position = [!x.window(0),!y.window(0),!x.window(1),!y.window(1)]
look_si,[0,1],[0,1],slopes,intercepts
zoom.slopes = slopes
zoom.intercepts = intercepts
;oplot,lindgen(zoom.width(0)),lindgen(zoom.width(1))
;print,zz ; a method of stopping: print an undefined variable
;
; =====>> Report the center and width of zoom region
;
;
widget_control,ws.zoomcentertext,set_value = $
    ['zoom center','x='+strtrim(zoom.center(0),2) $
    , 'y='+strtrim(zoom.center(1),2) $

```

```

,strtrim(thisimage(zoom.center(0),zoom.center(1)),2)]
widget_control,ws.zoomwidthtext,set_value = $
['zoom width','x='+strtrim(zoom.width(0),2) $
,'y='+strtrim(zoom.width(1),2) $
,strtrim(thisimage(zoom.width(0),zoom.width(1)),2)]
zoom.sd = stdev(zoomimage,mean)
zoom.mean = mean
widget_control,ws.stattext,set_value = $
['full stats','mean='+strtrim(all.mean,2) $
,'sd='+strtrim(all.sd,2),'zoom stats','mean='+strtrim(zoom.mean,2) $
,'sd='+strtrim(zoom.sd,2)]
;
;
; =====>> Renew the full image with the zoom region marked
;
wset,all.id
tmpimage = thisimage
xmn = zoom.xrange(0)
xmx = zoom.xrange(1)
ymn = zoom.yrange(0)
ymx = zoom.yrange(1)
midrange = 0.5*(all.tvrange(1)-all.tvrange(0))
midway = all.tvrange(0)+midrange
fourcols = [zoom.yrange,zoom.yrange+[-1,1]]
halfperi = tmpimage(xmn:xmx,fourcols)
hilo = halfperi*0-1
upper = where(halfperi LT midway)
IF upper(0) GE 0 THEN hilo(upper) = 1 ; -1(1) for >(<) midway
tmpimage(xmn:xmx,fourcols) = halfperi+hilo*midrange
fourrows = [zoom.xrange,zoom.xrange+[-1,1]]
halfperi = tmpimage(fourrows,ymn:ymx)
hilo = halfperi*0-1
upper = where(halfperi LT midway)
IF upper(0) GE 0 THEN hilo(upper) = 1 ; -1(1) for >(<) midway
tmpimage(fourrows,ymn:ymx) = halfperi+hilo*midrange
plot_image,tmpimage>all.tvrange(0)<all.tvrange(1)
return
END
;
;
; =====
; =====>> EVENT HANDLER
; =====
;
;
pro look_event,ev
common look,thisimage,zoomimage,images,ws,all,zoom,pan
;
;
; =====>> GET NAME OF EVENT STRUCTURE
;
;
widget_control,ev.id,get_uvalue = test,get_value = datum

```

```

;
;
; =====>> Event loop preliminaries
;
;
IF n_elements(test) EQ 0 THEN test = " ; shouldn't happen
;print,format = '($,a)',test+' '
IF element(where(ws.plotmenu EQ test)) GE 0 THEN BEGIN
; widget_control,ws.comtext,set_value = 'erasing zoom widget'
  wset,zoom.id
  erase
ENDIF
;
;
; =====>> PROCESS THE EVENT USING A CASE STATEMENT ON THE UVALUE
; =====>> But treat tv and zoom separately
;
;
CASE test OF
'contour': contour,thisimage
'cutmin': BEGIN
  all.tvrange(0) = datum < all.tvrange(1) > all.imrange(0)
  zoom.tvrange(0) = datum < zoom.tvrange(1) > zoom.imrange(0)
  END
'cutmax': BEGIN
  all.tvrange(1) = datum > all.tvrange(0) < all.imrange(1)
  zoom.tvrange(1) = datum > zoom.tvrange(0) < zoom.imrange(1)
  END
'dim3': ws.d(2) = datum
'dim4': ws.d(3) = datum
'dim5': ws.d(4) = datum
'dim6': ws.d(5) = datum
'dim7': ws.d(6) = datum
'donebutton': BEGIN
  widget_control,ev.top,/destroy
  return
END
'helpbutton': xdisplayfile,'$IDL_DIR/lib/local/look.pro'
'histogram': plot,histogram(thisimage $
                    ,bins = (all.tvrange(1)-all.tvrange(0))/all.x)
'plot_image': plot_image,thisimage
'shade_surf': shade_surf,thisimage
'surface': surface,thisimage
'show3': show3,thisimage
'tvscl': tvscl,thisimage
ELSE:
ENDCASE
;
;
; =====>> Get new image if requested
;
;
IF strpos(test,'dim') EQ 0 THEN BEGIN
  thisimage = images(*,*,ws.d(2),ws.d(3),ws.d(4),ws.d(5),ws.d(6))

```

```

wset,all.id
plot_image,thisimage > all.tvrange(0) < all.tvrange(1)
all.sd = stdev(thisimage,mean)
all.mean = mean
widget_control,ws.stattext,set_value = $
  ['full stats','mean='+strtrim(all.mean,2) $
    , 'sd='+strtrim(all.sd,2),'zoom stats','mean='+strtrim(zoom.mean,2) $
    , 'sd='+strtrim(zoom.sd,2)]
ENDIF
;
; =====>> Redraw if tvrange is changed
;
IF (test EQ 'cutmin') OR (test EQ 'cutmax') THEN BEGIN
  wset,all.id
  plot_image,thisimage>all.tvrange(0)<all.tvrange(1)
END
;
; =====>> Process the tv mouse events.
;
IF (test EQ 'alldraw') AND (n_tags(ev) EQ 8) THEN BEGIN
; widget_control,ws.comtext,set_value = string(ev,format = '(8i5)')
; IF ev.press EQ 0 THEN all.release(ev.release-1) = ev $ ; store for future
; ELSE all.press(ev.press-1) = ev
CASE 1 OF ; left mouse drag
  (ev.press EQ 1): BEGIN
    t = look_cc([ev.x,ev.y],all.slopes,all.intercepts)
    IF (t(0) GE all.xrange(0)) AND (t(0) LE all.xrange(1)) AND $
      (t(1) GE all.yrange(0)) AND (t(1) LE all.yrange(1)) THEN $
      all.down(*,ev.press-1) = t
  END
  (ev.release EQ 1): BEGIN
    b = ev.release-1
    t = look_cc([ev.x,ev.y],all.slopes,all.intercepts)
    IF (t(0) GE all.xrange(0)) AND (t(0) LE all.xrange(1)) AND $
      (t(1) GE all.yrange(0)) AND (t(1) LE all.yrange(1)) THEN BEGIN
      all.up(*,b) = t
      zoom.xrange = [(all.down(0,b)<all.up(0,b)) < (all.xrange(1)-2) $
        ,all.down(0,b)>all.up(0,b)] > all.xrange(0)
      zoom.xrange(1) = zoom.xrange(1) > (zoom.xrange(0)+2) < all.xrange(1)
      zoom.yrange = [(all.down(1,b)<all.up(1,b)) < (all.yrange(1)-2) $
        ,all.down(1,b)>all.up(1,b)] > all.yrange(0)
      zoom.yrange(1) = zoom.yrange(1) > (zoom.yrange(0)+2) < all.yrange(1)
      zoom.center = [(zoom.xrange(1)+zoom.xrange(0))/2 $
        ,(zoom.yrange(1)+zoom.yrange(0))/2]
      zoom.width = [(zoom.xrange(1)-zoom.xrange(0))+1 $
        ,(zoom.yrange(1)-zoom.yrange(0))+1]
      look_zoom
    ENDIF
  ENDIF
ENDIF

```

```

END
(ev.press EQ 2):      ; middle mouse click
(ev.release EQ 2): BEGIN
  t = look_cc([ev.x, ev.y], all.slopes, all.intercepts)
  IF (t(0) GE all.xrange(0)) AND (t(0) LE all.xrange(1)) AND $
    (t(1) GE all.yrange(0)) AND (t(1) LE all.yrange(1)) THEN BEGIN
    zoom.center = t
    zoom.width = zoom.width > 2 < ((t < (all.max-t))*2)
    zoom.xrange = (zoom.center(0)+(zoom.width(0)*[-1,1])/2) $
      > all.xrange(0) < all.xrange(1)
    zoom.yrange = (zoom.center(1)+(zoom.width(1)*[-1,1])/2) $
      > all.yrange(0) < all.yrange(1)
    look_zoom
  ENDIF
END
(ev.press EQ 4):      ; right mouse click
(ev.release EQ 4):
ELSE:
ENDCASE
ENDIF
;
; =====>> Process the tv timer events.
;
IF (test EQ 'alldraw') AND (n_tags(ev) EQ 3) THEN BEGIN
  widget_control, ev.id, timer = ws.interval
  look_cut, all
ENDIF
;
; =====>> Process the zoom timer events.
;
IF (test EQ 'zoomdraw') AND (n_tags(ev) EQ 3) THEN BEGIN
  widget_control, ev.id, timer = ws.interval
  look_cut, zoom
ENDIF
END
;
; =====
; =====>> INITIALIZING ROUTINE
; =====
;
PRO look, input, group=group, help=help, verbose = verbose
COMMON look, thisimage, zoomimage, images, ws, all, zoom, pan
;
; =====>> HELP
;
;on_error, 2
IF keyword_set(help) THEN BEGIN
  doc_library, 'look'

```

```

return
ENDIF
;
; =====>> SET DEFAULTS
;
IF n_elements(verbose) EQ 0 THEN verbose = 0
IF n_elements(input) EQ 0 THEN input = dist(256)
images = input ; I don't know how to avoid this copy.
si = size(input)
CASE 1 OF
    si(0) LT 2: message,'2-D or larger arrays only! Type look,/help.'
    si(0) EQ 2: thisimage = input
    si(0) GT 2: thisimage = input(*,*,0)
ENDCASE
quiet = !quiet
!quiet = 1
x = si(1) ; 4 conveniences
y = si(2)
xgen = lindgen(x)
ygen = lindgen(y)
zoomimage = thisimage ; Use full image to start
xrange = [0,x-1]
yrange = [0,y-1]
device,get_screen_size = scrdims ; ~plot+borders
dims = [1.14,1.16]*[x,y] < scrdims/2 > scrdims/4
setflag,/noexact ; use most of the draw widget
interval = 0.1 ; [s] timer interval for pixel location
id = bytarr(2,2) ; plotting window indices
xlast = -1 ; previous cursor coordinates
ylast = -1
plotmenu = ['contour','histogram','plot_image','surface','shade_surf',' show3','tvscf']
imrange = [min(thisimage,max = mx),mx]
tvrange = imrange
down = [-1,-1]
center = down
width = down
;
; =====>> CREATE AND REGISTER WIDGETS: top, columns, parts of columns
;
IF xregistered("look") GT 0 THEN message,'Another look exists! Type rtall & xmanager, then
look,...'
looktop = widget_base(title='look',/column)
lookbase = widget_base(looktop,/row)
col1base = widget_base(lookbase,/column)
donebutton = widget_button(col1base,value='DONE',uvalue = 'donebutton')
helpbutton = widget_button(col1base,value='HELP',uvalue = 'helpbutton')
mousetext = widget_text(col1base,xsize = 11,ysize = 4,value = 'mouse')
zoomcentertext = widget_text(col1base,xsize = 11,ysize = 4,value = 'zoom center')

```

```

zoomwidthtext = widget_text(col1base,xsize = 11,ysize = 3,value = 'zoom width')
stattext = widget_text(col1base,xsize = 11,ysize = 6,value = 'stats')
col2base = widget_base(lookbase,/column)
col2draw1 = widget_draw(col2base,xsize=dims(0),ysize=dims(1))
col2draw2 = widget_draw(col2base,xsize=dims(0),ysize=dims(1),uvalue='all draw',/button)
col2min = widget_slider(col2base,title = 'plot min',min = tvrange(0) $
    ,uvalue = 'cutmin',max = tvrange(1),value = tvrange(0))
col3base = widget_base(lookbase,/column)
col3draw1 = widget_draw(col3base,xsize=dims(0),ysize=dims(1),uvalue = 'zoomdraw',/button)
col3draw2 = widget_draw(col3base,xsize=dims(0),ysize=dims(1))
col3max = widget_slider(col3base,title = 'plot max',min = tvrange(0) $
    ,uvalue = 'cutmax',max = tvrange(1),value = tvrange(1))
col4base = widget_base(lookbase,/column)
xmenu,plotmenu,col4base,/exclusive,/frame,uvalue = plotmenu
col4sliders = -1
IF si(0) GT 2 THEN BEGIN
    col4sliders = lonarr(si(0)-2)
    FOR i = 3,si(0) DO col4sliders(i-3) = widget_slider(col4base,title = 'dim '+strtrim(i,2),min = 0,max
= si(i)-1,value = 0,uvalue = 'dim'+strtrim(i,2))
    ENDIF
comtext = widget_text(lookTop,xsize = 50,ysize = 1,value = 'Roam in tv image. Click and drag left
mouse to zoom. Click middle to recenter.')
widget_control,lookbase,/realize ; PUT THEM ON THE SCREEN
;
; =====>> Get window indices and store in common
;
widget_control,get_value = id_c2d1,col2draw1
widget_control,get_value = id_c2d2,col2draw2
widget_control,get_value = id_c3d1,col3draw1
widget_control,get_value = id_c3d2,col3draw2
ids = [[id_c2d1,id_c2d2],[id_c3d1,id_c3d2]]
;
; =====>> Allocate two timers: for the tv window and zoom window
;
widget_control,col2draw2,timer = interval
widget_control,col3draw1,timer = interval
;
; =====>> Fill the image widget & calculate scale (dev->data)
;
wset,id_c2d2
plot_image,thisimage,origin = [0,0],scale = 1
position = [!x.window(0),!y.window(0),!x.window(1),!y.window(1)]
look_si,xgen,ygen,slopes,intercepts
wset,id_c3d1
h = histogram(thisimage,binsize = (tvrange(1)-tvrange(0))/x)
plot,xgen,h,xstyle = 1,ystyle = 1,position = position
wset,id_c2d2
;

```

```

; =====>> Put all the information in structures to pass to event routine
;
all = { $ ; structure for entire image
    ,id:ids(1,0) $ ; draw widget id
    ,hcutid:ids(0,0) $ ; draw id for horizontal cut
    ,vcutid:ids(1,1) $ ; draw id for vertical cut
    ,position:position $ ; lo.left & up.right of plot region
    ,min:[xrange(0),yrange(0)] $ ; minimum x&y coordinate values: (0,0)
    ,max:[xrange(1),yrange(1)] $ ; maximum x&y coordinate values
    ,mean:0. $
    ,sd:0. $
    ,xrange:xrange $ ; [min(0),max(0)] = x coordinate range
    ,yrange:yrange $ ; [min(1),max(1)] = y coordinate range
    ,center:[x,y]/2 $ ; center x&y coordinates
    ,width:[x,y] $ ; # of x&y pixels
    ,slopes:slopes $ ; device-->data x&y slopes
    ,intercepts:intercepts $ ; device-->data x&y intercepts
    ,imrange:imrange $ ; range of pixel values
    ,tvrangle:tvrangle $ ; range of tv and cut plots
    ,last:[-1,-1] $ ; last mouse position
    ,x:x,y:y $ ; # of x&y pixels
;    ,press:replicate(dbe,4) $ ; widget_event structures for 3 mouse
;    ,release:replicate(dbe,4) $ ; button events: press and release
;                ; (left,middle,right) in (0,1,3)
;                ; corresponding to (1,2,4) bits
    ,down:lonarr(2,4)-1 $ ; data coords for mouse button presses
    ,up:lonarr(2,4)-1 $ ; data coords for mouse button releases
}
zoom = all ; same structure for zoom draw widget
zoom.id = ids(0,1) ; but change id
zoom.hcutid = ids(1,1) ; draw id for horizontal cut
zoom.vcutid = ids(0,0) ; draw id for vertical cut
pan = all ; same structure for pan draw widget
pan.id = ids(0,1) ; this will need to be another id
ws = {interval:interval $ ; timer interval
    ,ids:ids $ ; draw widget ids
    ,d:lonarr(7) $ ; current dims to browse in series
    ,col4sliders:col4sliders $ ; widget ids for dimension sliders
    ,comtext:comtext $ ; id for comment text widget
    ,mousetext:mousetext $ ; id for mouse coords text widget
    ,stattext:stattext $ ; id for statistics text widget
    ,plotmenu:plotmenu $ ; names of optional plots
    ,zoomcentertext:zoomcentertext $ ; id for zoom center text widget
    ,zoomwidthtext:zoomwidthtext $ ; id for zoom width text widget
;    ,pancentertext:pancentertext $ ; id for pan center text widget
;    ,panwidthtext:panwidthtext $ ; id for pan width text widget
}
;

```

```
; =====>> Report Statistics
;
all.sd = stdev(thisimage,mean)
all.mean = mean
widget_control,ws.stattext,set_value = $
  ['full stats','mean='+strtrim(all.mean,2) $
    , 'sd='+strtrim(all.sd,2), 'zoom stats','mean='+strtrim(zoom.mean,2) $
    , 'sd='+strtrim(zoom.sd,2)]
;
; =====>> TURN OVER CONTROL TO XMANAGER
;
xmanager,'look',lookbase,event_handler = 'look_event'
!quiet = quiet
end
--
=Fred Knight (knight@ll.mit.edu) (617) 981-2027
C-483\\MIT Lincoln Laboratory\\244 Wood Street\\Lexington, MA 02173
```
