Subject: blocking user from changing graphics properties in widget_window
Posted by PMan on Mon, 28 Oct 2013 23:55:04 GMT
View Forum Message <> Reply to Message

I was playing the example code of the widget_window in the IDL documentation (See below). Works ok, but the user can rotate and zoom on the image in the widget window. For this sort application, I would want to lock down the graphic so the user could not zoom or rotate. Is there anyway to lock certain properties so they cannot be changed?

```
;+

; The event handler called when any mouse button is pressed.

;-

FUNCTION WIDGET_WINDOW_EX_MOUSE_DOWN_EVENT, $

  win, x, y, button, keymods, clicks

  COMPILE_OPT idl2



  state = win.uvalue

  IF(button eq 1 && keymods eq 2) then begin

    state['drag'] = 1

    state['start'] = [x,y]

    ; Do not do the default handling.

    RETURN, 0

  ENDIF

  ; Continue with the default handling.

  return, 1

END



;+

; The event handler called when any mouse button is released.
```

```
;-
FUNCTION WIDGET_WINDOW_EX_MOUSE_UP_EVENT, win, x, y, button
COMPILE_OPT idl2
state = win.uvalue
state['drag'] = 0
RETURN, 1
END


;+
; The event handler called when the mouse is moved.
;-
FUNCTION WIDGET_WINDOW_EX_MOUSE_MOTION_EVENT, win, x, y, keymods
  COMPILE_OPT idl2


  state = win.uvalue
  IF(state['drag']) then begin
    difference = [x,y] - state['start']
    new_center = state['center'] + difference[0]
    new_width = state['width'] + difference[1]
    new_brain = SET_WINDOW(state['image'], new_center, new_width)
    win['display'].setdata, new_brain
    return, 0
  ENDIF
  RETURN, 1
```

```
END

;+
; Helper routine. Sets window center and width for an image.
;-
FUNCTION SET_WINDOW, image, center, width
  COMPILE_OPT idl2
  lo = center - 0.5*width
  hi = center + 0.5*width
  RETURN, BYTSCL((image > lo) < hi)
END

;+
; The launch routine. Sets up UI & state variable, loads data.
;-
PRO WIDGET_WINDOW_EX
  COMPILE_OPT idl2

  ; An MR image with center and width.
  brain = READ_IMAGE(FILE_WHICH('mr_brain.dcm'))
  center = 1011 ; from file
  width = 2021 ; from file
  brain_scaled = SET_WINDOW(brain, center, width)
```

```
wtop = WIDGET_BASE(/COLUMN, $
  TITLE='Graphics event handling with IDL widgets')
wdraw = WIDGET_WINDOW(wtop, $
  MOUSE_DOWN_HANDLER='widget_window_ex_mouse_down_event', $
  MOUSE_UP_HANDLER='widget_window_ex_mouse_up_event', $
  MOUSE_MOTION_HANDLER='widget_window_ex_mouse_motion_event')
WIDGET_CONTROL, wtop, /REALIZE

; Extract the window reference from the draw widget,
; and display the first image resampled to 150% original size.
WIDGET_CONTROL, wdraw, GET_VALUE=w
g = IMAGE(brain_scaled, /CURRENT, TITLE='MR: Brain', NAME='display')
g.SCALE, 1.5, 1.5, 1.0

t = TEXT(0.5,0.02, $
  ['Hold <Ctrl> and drag horizontally to change brightness,', $
  'drag vertically to change contrast'], $
  ALIGNMENT='center')

; Make a state variable.
state = HASH()
state['image'] = brain_scaled
state['drag'] = 0
state['start'] = [0,0]
```

```
  state['center'] = center
  state['width'] = width

  ; Attach the state variable to the window's user value. Attach the
  ; window to the top-level base's user value. This ensures that state
  ; information can be communicated to both the widget and event handlers.
  w.uvalue = state
  WIDGET_CONTROL, wtop, SET_UVALUE=w

  XMANAGER, 'widget_window_ex', wtop, /NO_BLOCK
END
```