Subject: cw_dual_slider: a slider with two slides
Posted by Mike[2] on Fri, 27 Oct 2006 20:15:11 GMT
View Forum Message <> Reply to Message

Dear IDL community,

A while back I wanted a double slider - you know, one with two slides so I can adjust an upper and lower level or slide a window around..So I wrote a compound widget that does that and am offering it here as a token of appreciation for all the freely available IDL code that has made my work so much simpler over the years.

This does have certain problems (or, um, design features). Since I use a draw widget for this, you'll have to force the widget to draw itself after it has been realized. I do this by explicitely resetting the value after realizing. Other than that, it can be used in a straight forward manner with widget_control get_value and set_value. The type of the value is taken from the value specified when the widget is created. Click in the middle of the slide to move both ends. Click on one end to move that end only. Double click in the middle to collapse it and make it similar to a widget_slider.

You are welcome to use it and abuse it at will. I'd appreciate any suggestions, bug fixes, comments, jeers, etc. There is a test program and event handler at the end of this message.

Regards, Mike

```
;+
; NAME:
; CW_DUAL_SLIDER
;
; AUTHOR:
;       Michael A. Miller,
;       Imaging Sciences, Department of Radiology, IU School of
Medicine
;       Michael.Miller5@gmail.com
;
; PURPOSE:
;       Provides a slider with two slides.  Instead of a scalar, the
;       value is a two element vector
;
; CATEGORY:
; Compound widgets.
;
; CALLING SEQUENCE:
; widget = CW_DUAL_SLIDER(parent)
```

```
;
; INPUTS:
;       PARENT - The ID of the parent widget.
;
; INPUT KEYWORDS:
;       value - a two element vector specifying the initial value for
;               the widget
;       minimum - minumum value of the slider range
;       maximum - maximum value of the slider range
;
;       scroll - distance to move the slides when the mouse is clicked
;                on either side of the slider bar
;       title - string specifying the title to be drawn under the
;               slider
;       uname - uname for the widget
;       xsize - width of the widget in pixels
;
; OUTPUTS:
;       The ID of the created widget is returned.
;
; COMMON BLOCKS:
; None.
;
; SIDE EFFECTS:
;
; PROCEDURE:
; WIDGET_CONTROL, id, SET_VALUE=value can be used to change the
;  current value displayed by the widget.
;
; WIDGET_CONTROL, id, GET_VALUE=var can be used to obtain the current
;  value displayed by the widget.
;
; MODIFICATION HISTORY:
; $Id: cw_dual_slider.pro,v 1.3 2006/10/27 19:18:32 mmiller3 Exp $
;
; Based on cw_tmpl.pro from RSI.
; Oct. 2006 - original release
;-
;
; Copyright (C) 2006, Michael A. Miller
; This software is provided as is without any warranty whatsoever.
; Permission to use, copy, modify, and distribute modified or
; unmodified copies is granted, provided this copyright and disclaimer
; are included unchanged.
;
 ;------------------------------------------------------------ ------------------
pro cw_dual_slider_draw_between, state, pos0, pos1
if pos0 ne pos1 then begin
```

```
    ;; fill in background ---
    for i = 2, state.track_height-2 do $
      plots, [pos0+1,pos1-1],
state.yBottom+state.edge_width+[i,i],/device, col=state.bg

    ;; top edge ---
    plots, [pos0+1, pos1-1], (state.yBottom+state.edge_width +
state.track_height)*[1,1], /device, col=state.light
    plots, [pos0+1, pos1-2], (state.yBottom+state.edge_width +
state.track_height)*[1,1]-1, /device, col=state.light

    ;; bottom edge ---
    plots, [pos0+2,pos1-1], (state.yBottom + state.edge_width)*[1,1],
/device, col=state.dark
    plots, [pos0+3,pos1-1], (state.yBottom + state.edge_width)*[1,1]+1,
/device, col=state.dark

    ;; left edge ---
    plots, [pos0+1,pos0+1], state.yBottom + state.edge_width +
[0,state.track_height], /device, col=state.light
    plots, [pos0+2,pos0+2], state.yBottom + state.edge_width +
[1,state.track_height], /device, col=state.light

    ;; left edge ---
    plots, [pos1-1,pos1-1], state.yBottom + state.edge_width +
[0,state.track_height-1], /device, col=state.dark
    plots, [pos1-2,pos1-2], state.yBottom + state.edge_width +
[0,state.track_height-2], /device, col=state.dark
endif
end

 ;------------------------------------------------------------ ------------------
pro cw_dual_slider_draw_slide, state, position, left=left, right=right
pos = position
if keyword_set(left) then pos = position - state.slide_length/4
if keyword_set(right) then pos = position + state.slide_length/4

;; fill in the slide ---
for i = 0, state.track_height do $
  plots, pos + state.slide_length/4 * [-1,1], $
         state.yBottom + i + state.edge_width*[1,1], /device,
col=state.bg

;; top edge ---
plots, pos + state.slide_length/4 * [-1,1], $
       (state.yBottom+state.edge_width + state.track_height)*[1,1],
/device, col=state.light
plots, pos + state.slide_length/4 * [-1,1] + [0,-1], $
```

```
      (state.yBottom + state.edge_width + state.track_height-1)*[1,1],
/device, col=state.light

;; bottom edge ---
plots, pos + state.slide_length/4 * [-1,1] + [1,0], $
      (state.yBottom + state.edge_width)*[1,1], /device,
col=state.dark
plots, pos + state.slide_length/4 * [-1,1] + [2,0], $
      (state.yBottom + state.edge_width+1)*[1,1], /device,
col=state.dark

;; left edge ---
plots, pos + state.slide_length/4*[-1,-1], $
      state.yBottom + state.edge_width + [0,state.track_height],
/device, col=state.light
plots, pos + state.slide_length/4*[-1,-1] + [1,1], $
      state.yBottom + state.edge_width + [1,state.track_height],
/device, col=state.light

;; left edge ---
plots, pos + state.slide_length/4*[1,1], $
      state.yBottom + state.edge_width + [0,state.track_height-1],
/device, col=state.dark
plots, pos + state.slide_length/4*[1,1] - 1, $
      state.yBottom + state.edge_width + [0,state.track_height-2],
/device, col=state.dark

end

 ;------------------------------------------------------------ ------------------
pro cw_dual_slider_draw_track, state
;; bottom edge ---
plots, state.xLeft+[0,state.track_length+2*state.edge_width],
state.yBottom+[0,0], /device, col=state.light
plots, state.xLeft+[0,state.track_length+2*state.edge_width],
state.yBottom+[1,1], /device, col=state.light

;; top edge --
plots, state.xLeft+[0,state.track_length+2*state.edge_width], $
       state.yBottom+(state.track_height+2*state.edge_width)*[1,1],
/device, col=state.dark
plots, state.xLeft+[0,state.track_length+2*state.edge_width-1], $
       state.yBottom+(state.track_height+2*state.edge_width-1)*[1,1 ],
/device, col=state.dark

; ;; right edge ---
plots, state.xLeft+state.track_length+2*state.edge_width-1*[1,1], $
      state.yBottom+[0,state.track_height+state.edge_width+1],
```

```
/device, col=state.light
plots, state.xLeft+state.track_length+2*state.edge_width*[1,1], $
       state.yBottom+[0,state.track_height+state.edge_width+2],
/device, col=state.light

;; left edge ---
plots, state.xLeft+[0,0],
state.yBottom+[0,state.track_height+2*state.edge_width], /device,
col=state.dark
plots, state.xLeft+[1,1],
state.yBottom+[1,state.track_height+state.edge_width+1], /device,
col=state.dark

;; track ---
for i = 0, state.track_height do $
  plots,
 state.xLeft+state.edge_width+[0,state.track_length-state.edg e_width], $
         state.yBottom+i+state.edge_width*[1,1], /device,
col=state.track
end

 ;------------------------------------------------------------ ------------------
function cw_dual_slider_value_to_position, state, value
slope = state.pos_width / state.width
intercept = state.min_pos - slope * state.min
position = slope * value + intercept
return, position
end

 ;------------------------------------------------------------ ------------------
function cw_dual_slider_position_to_value, state, position
slope = state.width / state.pos_width
intercept = state.min - slope * state.min_pos
value = slope * position + intercept
return, value
end

 ;------------------------------------------------------------ ------------------
function cw_dual_slider_click_on_slider, state, event
y = event.y - state.yBottom - state.edge_width
result = 0
if (y gt 0) and (y lt state.track_height) then begin
    if event.x gt (state.xLeft + state.edge_width) $
      and event.x lt (state.xLeft + state.edge_width +
state.track_length) then result = 1
endif
return, result
end
```

```
;------------------------------------------------------------ ------------------
function cw_dual_slider_click_on_slide, state, event
;; Calculate pixel position for slides and labels
pos0 = cw_dual_slider_value_to_position(state, state.value[0])
pos1 = cw_dual_slider_value_to_position(state, state.value[1])
x = event.x - state.xLeft - state.edge_width
result = 0
if cw_dual_slider_click_on_slider(state, event) then begin
   if (x-pos0 gt 0) and (x-pos1 lt 0) then result = 3 ; click between
sliders
   if abs(x-pos0) lt state.slide_length/4 then result = 1 ; click on
left slider
   if abs(x-pos1) lt state.slide_length/4 then result = 2 ; click on
right slider
endif
return, result
end

;------------------------------------------------------------ ------------------
function cw_dual_slider_click_type, state, event
;; Calculate pixel position for slides and labels
pos0 = cw_dual_slider_value_to_position(state, state.value[0])
pos1 = cw_dual_slider_value_to_position(state, state.value[1])
result = ''
;print, 'click type> ', event.x, pos0, pos1, pos1+state.slide_length/2
if cw_dual_slider_click_on_slider(state, event) then begin
   result = 'between'          ; click between slides
   if event.x lt pos0-state.slide_length/2 then result = 'minus' ;
click below both slides
   if event.x gt pos1+state.slide_length/2 then result = 'plus' ;
click above both slides
   if result eq 'between' then begin ; sort out where the slides have
been clicked...
      if pos0 ne pos1 then begin ; slides are well separated
         if event.x lt pos0 then result = 'lower' ; click on left
slide
         if event.x gt pos1 then result = 'upper' ; click on rightt
slide
      endif else begin        ; give a little space between slides to
select both...
         if event.x lt pos0 - state.slide_length/4 then result =
'lower' ; click on left slide
         if event.x gt pos1 + state.slide_length/4 then result =
'upper' ; click on right slide
      endelse
   endif
endif
```

```
return, result
end

 ;------------------------------------------------------------ ------------------
pro cw_dual_slider_draw, state
;; Set window to the proper device:
widget_control, state.drawID, get_value=wID
wset, wID

;; Save the existing color table
tvlct, r, g, b, /get

;; Set up the color table
tvlct, reform([  0,   0,   0],1,3), state.black
tvlct, reform([163, 163, 163],1,3), state.track
tvlct, reform([106, 106, 106],1,3), state.dark
tvlct, reform([228, 228, 228],1,3), state.light
tvlct, reform([192, 192, 192],1,3), state.bg

erase, col=state.bg

;; Label:
xyouts, !d.x_size/2, 5, state.title, /device, font=0, alignment=0.5,
col=state.black

;; Draw the track:
cw_dual_slider_draw_track, state

;; Calculate pixel position for slides and labels
pos0 = cw_dual_slider_value_to_position(state, state.value[0])
pos1 = cw_dual_slider_value_to_position(state, state.value[1])

;print, state.value, pos0, pos1

;; draw left slide
cw_dual_slider_draw_slide, state, pos0, /left

;; draw right slide
cw_dual_slider_draw_slide, state, pos1, /right

;; fill in bewteen slides...
cw_dual_slider_draw_between, state, pos0, pos1

;; Value label(s)
if pos1 ne pos0 then begin
   xyouts, pos0, !d.y_size-15, strtrim(state.value[0],2), $
         /device, font=0, alignment=1, col=state.black
   xyouts, pos1, !d.y_size-15, strtrim(state.value[1],2), $
```

```
        /device, font=0, alignment=0, col=state.black

;    plots, [pos0+3, pos1-3],
 state.yBottom+state.edge_width+state.track_height/2+[17,17], /device,
col=state.black, thick=2
endif else begin
   xyouts, pos0, !d.y_size-15, strtrim(state.value[0],2), $
        /device, font=0, alignment=0.5, col=state.black
endelse

;; Restore the original color table:
tvlct, r, g, b
end


PRO dual_slider_set_value, id, value
;; This routine is used by WIDGET_CONTROL to set the value for
;; your compound widget.  It accepts one variable.
;; You can organize the variable as you would like.  If you have
;; more than one setting, you may want to use a structure that
;; the user would need to build and then pass in using
;; WIDGET_CONTROL, compoundid, SET_VALUE = structure.

;; Don't show up in HELP output unless HIDDEN keyword is used.
COMPILE_OPT hidden

;; Return to caller.
ON_ERROR, 2

;; Retrieve the state.
stash = WIDGET_INFO(id, /CHILD)
WIDGET_CONTROL, stash, GET_UVALUE=state, /NO_COPY

;print, '>>>>>>>>>>>>> '
;help, state, /structure

;; Set the value here.
if n_elements(value) eq 2 then begin
   if (value[0] ge state.min) and (value[0] le state.value[1]) then
state.value[0] = value[0]
   if (value[1] le state.max) and (value[1] ge state.value[0]) then
state.value[1] = value[1]
endif

if n_elements(value) eq 4 then begin
   state.min = value[2]
   state.max = value[3]
   if (value[0] ge state.min) and (value[0] le state.max) then
```

```
state.value[0] = value[0]
   if (value[1] le state.max) and (value[1] ge state.min) then
state.value[1] = value[1]
   state.width = float(state.max-state.min)
endif

cw_dual_slider_draw, state

;print, '>>>>>>>>>>>>> '
;help, state, /structure

;; Restore the state.
WIDGET_CONTROL, stash, SET_UVALUE=state, /NO_COPY

END



FUNCTION dual_slider_get_value, id

;; This routine is used by WIDGET_CONTROL to get the value from
;; your compound widget.  As with the set_value equivalent,
;; you can only pass one value here so you may need to load
;; the value by using a structure or array.

;; Don't show up in HELP output unless HIDDEN keyword is used.
COMPILE_OPT hidden

;; Return to caller.
ON_ERROR, 2

;; Retrieve the structure from the child that contains the sub ids.
stash = WIDGET_INFO(id, /CHILD)
WIDGET_CONTROL, stash, GET_UVALUE=state, /NO_COPY

;; Get the value here
result = state.value

;; Restore the state.
WIDGET_CONTROL, stash, SET_UVALUE=state, /NO_COPY

;; Return the value here.
return, result

END


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

```
FUNCTION dual_slider_event, event

;; This routine handles all the events that happen in your
;; compound widget and if the events need to be passed along
;; this routine should return the new event.  If nobody needs
;; to know about the event that just occured, this routine
;; can just return 0.  If your routine never needs to pass
;; along an event, this routine can be a procedure instead
;; of a function.  Whichever type used must be set below in the
;; WIDGET_BASE call using either the EVENT_PROC or EVENT_FUNC
;; keyword.  An event function that returns a scalar 0 is
;; essentially an event procedure.

;; Don't show up in HELP output unless HIDDEN keyword is used.
COMPILE_OPT hidden

;; Retrieve the structure from the child that contains the sub ids.
parent = event.handler
stash = WIDGET_INFO(parent, /CHILD)
WIDGET_CONTROL, stash, GET_UVALUE=state, /NO_COPY

case tag_names(event, /structure_name) of
   'WIDGET_TRACKING':begin
      if event.enter then begin
         device, /cursor_original
      endif else begin
         device, /cursor_crosshair
      endelse
      ;; Restore the state structure and return
      WIDGET_CONTROL, stash, SET_UVALUE=state, /NO_COPY
      return, 0
   end

   'WIDGET_DRAW': begin
      ;print, '-------------------- cw_dual_slider
----------------------------'
      ;help, event, /structure
      ;help, state, /structure
      ;print,
 '------------------------------------------------------------ ------'

      cw_dual_slider_draw, state

      if event.press then begin
         case cw_dual_slider_click_type(state, event) of
            'minus': begin  ; increment negative
               ;print, 'minus'
               state.value = state.value - state.scroll
```

```
                if state.value[0] lt state.min then state.value[0]
= state.min
                if state.value[1] lt state.min then state.value[1]
= state.min
                cw_dual_slider_draw, state
            end
            'plus': begin   ; increment positive
                ;print, 'plus ', state.value,
state.value+state.scroll
                state.value = state.value + state.scroll
                if state.value[0] gt state.max then state.value[0]
= state.max
                if state.value[1] gt state.max then state.value[1]
= state.max
                cw_dual_slider_draw, state
            end
            'between': begin
                ;print, 'between'
                if event.clicks eq 2 then begin
                    if event.modifiers eq 0 then begin ;; double
click ==> colapse to middle of range
                        midpoint = float(state.value[0] +
state.value[1])/2
                        state.value = [midpoint, midpoint]
                    endif
                    if event.modifiers eq 1 then begin ;;
shift+double click ==> expand to full range
                        state.value = [state.min, state.max]
                    endif
                    cw_dual_slider_draw, state
                endif else begin
                    state.mouse_state = 'between' ; move both
slides
                    widget_control, state.drawID,
draw_motion_events=1
                endelse
            end
            'lower': begin
                ;print, 'lower'
                state.mouse_state = 'lower' ; move lower slide
                widget_control, state.drawID, draw_motion_events=1
            end
            'upper': begin
                ;print, 'upper'
                state.mouse_state = 'upper' ; move upper slide
                widget_control, state.drawID, draw_motion_events=1
            end
            else: begin
```

```
                print, 'cw_dual_slider: invalid click type [' +
cw_dual_slider_click_type(state, event) + ']'
            end
          endcase
        endif

        if event.type eq 2 then begin ; handle motion events
          case state.mouse_state of
            'between': begin ; move both slides
              midpoint = float(state.value[0] + state.value[1])/2
              range = float(state.value[1] - state.value[0])
              new_midpoint =
cw_dual_slider_position_to_value(state, event.x)
              if (new_midpoint - range/2 ge state.min) and
(new_midpoint + range/2 le state.max) then begin
                state.value = new_midpoint + [-0.5,0.5]*range
              endif else begin
                if (new_midpoint - range/2 lt state.min) then
begin
                  state.value = [state.min, state.min+range]
                endif
                if (new_midpoint + range/2 gt state.max) then
begin
                  state.value = [state.max-range, state.max]
                endif
              endelse
              cw_dual_slider_draw, state
            end
            'lower': begin  ; move just the lower slide
              new_point = cw_dual_slider_position_to_value(state,
event.x)
              ;print, state.value, new_point
              if (new_point ge state.min) and (new_point le
state.value[1]) then begin
                state.value = [new_point, state.value[1]]
                cw_dual_slider_draw, state
              endif
            end
            'upper': begin  ; move just the lower slide
              new_point = cw_dual_slider_position_to_value(state,
event.x)
              if (new_point le state.max) and (new_point ge
state.value[0]) then begin
                state.value = [state.value[0], new_point]
                cw_dual_slider_draw, state
              endif
            end
            else: begin
```

```
            end
          endcase
        endif

        if event.release then begin
          ;; Turn off motion events
          state.mouse_state = 'do nothing'
          widget_control, state.drawID, draw_motion_events=0
        endif
    end
endcase
;print, state.value
value = state.value
;print, state.value

;; Restore the state structure
WIDGET_CONTROL, stash, SET_UVALUE=state, /NO_COPY

;; You may need to add more tags to the event structure for your
;; compound widget.  If so do it after the first three which are
;; required and preserve the order of the first three.

RETURN, { ID:parent, TOP:event.top, HANDLER:0L, value:value }
END

 ;------------------------------------------------------------ ------------------
FUNCTION cw_dual_slider, parent, value=value, minimum=minimum,
maximum=maximum, $
  scroll=scroll, title=title, uname=uname, xsize=xsize ;, $
;  integer=integer, long=long, floating=floating,
motion_events=motion_events

;; You should not use the user value of the main base for
;; your compound widget as the person using your compound widget
;; may want it for his or her own use.
;; You also should not use the user value of the first widget you
;; install in the base as it is used to keep track of the state.

;; state structure for your compound widget.

IF (N_PARAMS() EQ 0) THEN MESSAGE, 'Must specify a parent for
cw_dual_slider'

ON_ERROR, 2                     ;return to caller

;; Set default parameters

if n_elements(minimum) eq 0 then minimum = 0.0
```

```
if n_elements(maximum) eq 0 then maximum = 100.0
if n_elements(value) ne 2 then value = [0.0,0.0]

if n_elements(scroll) ne 1 then scroll = 1

if n_elements(xsize) eq 0 then xsize = 200

;; Defaults for keywords
IF NOT (KEYWORD_SET(uval))  THEN uval = 0
IF NOT (KEYWORD_SET(uname))  THEN uname = 'CW_DUAL_SLIDER_UNAME'
IF NOT (KEYWORD_SET(title))  THEN title = 'dual slider'

;; Rather than use a common block to store the widget IDs of the
;; widgets in your compound widget, put them into this structure so
;; that you can have multiple instances of your compound widget.
state = { id:0, $
        title: title, $
        value: value, $
        min: minimum, $
        max: maximum, $
        scroll: scroll, $
        width: float(maximum-minimum), $
        track_height: 11, $
        edge_width: 2, $
        slide_length: 32, $
        track_length: xsize, $
        xLeft: 10, $
        xRight: 10, $
        yTop: 20, $
        yBottom: 20, $
        min_pos: 0, $
        max_pos: 0, $
        pos_width: 0, $
        drawID: 0L, $
        mouse_state: '', $
        track: 1, $
        dark: 2, $
        light: 3, $
        bg: 4, $
        black: 5 $
       }
state.min_pos = state.xLeft + state.edge_width + state.slide_length/2
state.max_pos = state.xLeft + state.edge_width + state.track_length -
state.slide_length/2
state.pos_width = state.max_pos - state.min_pos

;; Here the widget base that encompasses your compound widget's
;; individual components is created.  This is the widget ID that
```

```
;; is passed back to the user to represent the entire compound
;; widget.  If it gets mapped, unmapped, sensitized or otherwise
;; effected, each of its individual subcomponents will also be
;; effected.  You can see that the event handler is installed here.
;; As events occur in the sub-components of the compound widgets,
;; the events are passed up the tree until they hit this base
;; where the event handler you define above handles them.  Similarily
;; whenever WIDGET_CONTROL, SET/GET_VALUE is called on this base,
;; the routine defined by the FUNC_GET/PRO_SET_VALUE is called to
;; set the value of the compound widget.  None of the three keywords
;; that override the standard behaviour are not required so it
;; depends on your usage whether they are needed.
mainbase = WIDGET_BASE(parent, UVALUE = uval, UNAME = uname, $
                  EVENT_FUNC = "dual_slider_event", $
                  FUNC_GET_VALUE = "dual_slider_get_value", $
                  PRO_SET_VALUE = "dual_slider_set_value")

;; Here you would define the sub-components of your widget.  There
;; is an example component which is just a label.
;state.id = WIDGET_LABEL(mainbase, VALUE = "Compound Widget Template")

state.drawID = widget_draw(mainbase, /button_events, /tracking_events,
$
                     xsize=state.xLeft + 2*state.edge_width +
state.track_length + state.xRight, $
                     ysize=state.yBottom + 2*state.edge_width +
state.track_height + state.yTop)

;help, state, /structure

;; Save out the initial state structure into the first childs UVALUE.
WIDGET_CONTROL, WIDGET_INFO(mainbase, /CHILD), SET_UVALUE=state,
/NO_COPY

;; Return the base ID of your compound widget.  This returned
;; value is all the user will know about the internal structure
;; of your widget.
RETURN, mainbase

END


;;------- cw_dual_slider_test.pro
pro cw_dual_slider_test_event, event
;print, 'in cw_dual_slider_test_event... ', widget_info(event.id,
/uname)
;help, event, /structure
case event.id of
```

```
    widget_info(event.top, find_by_uname='button1'): $
      widget_control, widget_info(event.top, find_by_uname='dual1'),
set_value=[300,700]
    widget_info(event.top, find_by_uname='button2'): $
      widget_control, widget_info(event.top, find_by_uname='dual1'),
set_value=[500,500]
    widget_info(event.top, find_by_uname='exit'): exit
;    widget_info(event.top, find_by_uname='dual1'): begin
;        print, 'dual1 -----------'
;        help, event, /structure
;        print, event.value
;     end
    else:
endcase
end

pro cw_dual_slider_test
base = widget_base(title='test', /column)
button = widget_button(base, value='300,700', uname='button1')
button = widget_button(base, value='500,500', uname='button2')
button = widget_button(base, value='exit', uname='exit')
slider = widget_slider(base, value=500, min=0, max=1000)
;slider = widget_slider(base, value=50, min=0, max=100, /vertical)
ds1 = cw_dual_slider(base, minimum=-500, maximum=1000,
value=fix([500,500]), uname='dual1', title='integer')
ds2 = cw_dual_slider(base, minimum=-500, maximum=1000,
value=long([500,500]), uname='dual3', title='long')
ds3 = cw_dual_slider(base, minimum=-500, maximum=1000,
value=float([500,500]), uname='dual2', title='float')
ds4 = cw_dual_slider(base, minimum=-500, maximum=1000,
value=double([500,500]), uname='dual4', title='double')

widget_control, base, /realize
widget_control, ds1, set_value=[500,500]
widget_control, ds2, set_value=[500,500]
widget_control, ds3, set_value=[500,500]
widget_control, ds4, set_value=[500,500]

xmanager, 'cw_dual_slider_test', base
end
;;------- end of cw_dual_slider_test.pro
```

---

Subject: Re: cw_dual_slider: a slider with two slides
Posted by David Fanning on Fri, 27 Oct 2006 23:41:18 GMT
View Forum Message <> Reply to Message

Hi Mike

You write:

> I have the same problem if I run it on windows
>
> If I toss a device, retain=2, decomposed=0 into my startup file (it was
> already there on my linux systems) then it works, although the colors
> are not quite right. So I'll add "use widget_info(/system_colors)" to
> the list of things to do.

Humm. OK, looks promising. Here are a couple of ideas.

First, since you are saving and restoring the color table, you assume we have 24-bit graphics cards. Why then do you want us to cripple them to run this program by setting DEVICE, DECOMPOSED=0? If you *must* use indexed color, don't bother us with it. Let us use all the colors we want. Get the current decomposed state (GET_DECOMPOSED) when you get the colors, and restore it when you restore the colors. :-)

Second, the numbers on top of the bar disappear when I move the sliders towards the ends. You could either put the numbers at either end of the bar (with the appropriate ALIGN keyword) or, if you liked the numbers moving, you could calculate how wide the number is (use a negative value to the WIDTH keyword on XYOUTS) so you could always keep it in view. You might need a pixmap to do this properly, but see 3, below.

Third, the slider "flashes" as I move it. You could get rid of most of the flashing if you "buffered" the output in a pixmap, and then moved it over to the window with the DEVICE, COPY technique.

I like the way it works, though. Good job! -)

Cheers,

David

P.S. I've done a similar window/level colorbar widget. (Seen in Catalyst.) But I like the natural look of this one better.
--
David Fanning, Ph.D.
Fanning Software Consulting, Inc.
Coyote's Guide to IDL Programming: http://www.dfanning.com/

Sepore ma de ni thui. ("Perhaps thou speakest truth.")

---

Subject: Re: cw_dual_slider: a slider with two slides
Posted by Mike[2] on Tue, 31 Oct 2006 18:25:59 GMT
View Forum Message <> Reply to Message

David Fanning wrote:
> If you *must* use indexed color, don't bother us with it. Let us use all
> the colors we want. Get the current decomposed state (GET_DECOMPOSED)
> when you get the colors, and restore it when you restore the colors. :-)

Done. This has been good for my continuing growth in the area of IDL colors. Instead of setting decomposed, I replaced all the color specifications with calls to fsc_color and color24 (with arguments come from widget_system_colors). I've also pixmap buffered the whole thing. See http://www.indyrad.iupui.edu/public/mmiller3/cw_dual_slider for the latest which now looks sort of windows-ish on windows, *nix-ish on *nix and who-knows-what-ish on macs.

Mike

P.S. I'm curious to see what it does look like on a mac if anyone wants to send a screen capture...

---

Subject: Re: cw_dual_slider: a slider with two slides
Posted by David Fanning on Tue, 31 Oct 2006 19:22:25 GMT
View Forum Message <> Reply to Message

Mike writes:

> See http://www.indyrad.iupui.edu/public/mmiller3/cw_dual_slider for
> the latest which now looks sort of windows-ish on windows, *nix-ish on
> *nix and who-knows-what-ish on macs.

Those links don't seem to work for me. :-(

Cheers,

David
--
David Fanning, Ph.D.
Fanning Software Consulting, Inc.
Coyote's Guide to IDL Programming: http://www.dfanning.com/
Sepore ma de ni thui. ("Perhaps thou speakest truth.")

---

Subject: Re: cw_dual_slider: a slider with two slides
Posted by JD Smith on Tue, 31 Oct 2006 20:50:23 GMT
View Forum Message <> Reply to Message

On Tue, 31 Oct 2006 10:25:59 -0800, Mike wrote:

>
> P.S. I'm curious to see what it does look like on a mac if anyone
> wants to send a screen capture...

Mac IDL uses Motif under X11 just like the other unixes, so it will look identical. Sometimes this is a nice thing.

JD

---

Subject: Re: cw_dual_slider: a slider with two slides
Posted by Mike[2] on Tue, 31 Oct 2006 22:30:18 GMT
View Forum Message <> Reply to Message

David Fanning wrote:
> Mike writes:
>
>> See http://www.indyrad.iupui.edu/public/mmiller3/cw_dual_slider for
>> the latest which now looks sort of windows-ish on windows, *nix-ish on
>> *nix and who-knows-what-ish on macs.
>
> Those links don't seem to work for me. :-(

Sigh... that is a feature of our brain dead web server. Try this instead:
http://mypage.iu.edu/~mmiller3/cw_dual_slider/

--Mike

---

Subject: Re: cw_dual_slider: a slider with two slides
Posted by David Fanning on Tue, 31 Oct 2006 22:41:18 GMT
View Forum Message <> Reply to Message

Mike writes:

> Sigh... that is a feature of our brain dead web server. Try this
> instead:
> http://mypage.iu.edu/~mmiller3/cw_dual_slider/

Perfect. Thanks. :-)

David
--
David Fanning, Ph.D.
Fanning Software Consulting, Inc.
Coyote's Guide to IDL Programming: http://www.dfanning.com/
Sepore ma de ni thui. ("Perhaps thou speakest truth.")

---

Subject: Re: cw_dual_slider: a slider with two slides
Posted by Paul Van Delst[1] on Tue, 31 Oct 2006 22:49:17 GMT
View Forum Message <> Reply to Message

Mike wrote:
> David Fanning wrote:
>> Mike writes:
>>
>>> See http://www.indyrad.iupui.edu/public/mmiller3/cw_dual_slider for
>>> the latest which now looks sort of windows-ish on windows, *nix-ish on
>>> *nix and who-knows-what-ish on macs.
>> Those links don't seem to work for me. :-(
>
> Sigh... that is a feature of our brain dead web server. Try this
> instead:
> http://mypage.iu.edu/~mmiller3/cw_dual_slider/

I get this:

```
lnx:/usr/tmp : idl
IDL Version 6.2 (linux x86 m32). (c) 2005, Research Systems, Inc.
Installation number: 100131.
Licensed for use by: NOAA/NCEP/EMC

Number of colors is     16777216
Color table size is         256
% Compiled module: COLORS.
IDL> cw_dual_slider_test
% Compiled module: CW_DUAL_SLIDER_TEST.
% Compiled module: CW_DUAL_SLIDER.
% Variable is undefined: COLOR24.
% Error occurred at: DUAL_SLIDER_SET_VALUE  311 /usr1/wd20pd/tmp/cw_dual_slider.pro
%                     WIDGET_CONTROL
%                     CW_DUAL_SLIDER_TEST   32 /usr1/wd20pd/tmp/cw_dual_slider_test.pro
%                     $MAIN$
% Execution halted at: CW_DUAL_SLIDER_TEST   32 /usr1/wd20pd/tmp/cw_dual_slider_test.pro
```

A dependency issue? The header docs didn't mention any requirements.

paulv

--
Paul van Delst Ride lots.
CIMSS @ NOAA/NCEP/EMC Eddy Merckx
Ph: (301)763-8000 x7748
Fax:(301)763-8545

---

Subject: Re: cw_dual_slider: a slider with two slides
Posted by Mike[2] on Tue, 31 Oct 2006 22:58:25 GMT
View Forum Message <> Reply to Message

Paul van Delst wrote:
> A dependency issue? The header docs didn't mention any requirements.

Hi Paul,

It now depends directly on fsc_color and color24 from David's coyote library, and indirectly on whatever they depend on. You can get the whole collection at www.dfanning.com. I'll add that into the docs.

Mike

---

Subject: Re: cw_dual_slider: a slider with two slides
Posted by Robbie on Wed, 01 Nov 2006 00:09:43 GMT
View Forum Message <> Reply to Message

Works perfectly on IDL 6.1 Windows 2000.

I like the novel use of system colors to simulate a real widget.

I've uploaded the .sav file of an an OO equivalent of "A slider with two slides", but it's not as good as Mike's.

http://barnett.id.au/idl/

Robbie

---

Subject: Re: cw_dual_slider: a slider with two slides
Posted by badjelly.witch on Wed, 01 Nov 2006 19:08:25 GMT
View Forum Message <> Reply to Message

Mike wrote:
>

> Try this instead:
> http://mypage.iu.edu/~mmiller3/cw_dual_slider/

Great stuff!

A quibble with your cw_dual_slider_test procedure. The event-handler calls the EXIT routine if the "exit" button is pressed. This quits the IDL session (luckily it asks first). I doubt that this is what you intended :-)

---

Subject: Re: cw_dual_slider: a slider with two slides
Posted by edward.s.meinel@aero. on Thu, 02 Nov 2006 16:54:07 GMT
View Forum Message <> Reply to Message

Mark Hadfield wrote:
> Mike wrote:
>>
>> Try this instead:
>> http://mypage.iu.edu/~mmiller3/cw_dual_slider/
>
> Great stuff!
>
> A quibble with your cw_dual_slider_test procedure. The event-handler
> calls the EXIT routine if the "exit" button is pressed. This quits the
> IDL session (luckily it asks first). I doubt that this is what you
> intended :-)

Replace "exit" with "WIDGET_CONTROL, event.top, /DESTROY" and it works just fine.

Way cool! Keep up the good work.

Ed

---

Subject: Re: cw_dual_slider: a slider with two slides
Posted by greg michael on Thu, 02 Nov 2006 19:00:38 GMT
View Forum Message <> Reply to Message

Hi Mike,

Looks great now. I found a minor quirk though: if you're in the slider extending mode, and you hit the end of the scale, you're typically left with a number somewhere short of the limit. On my system you have to approach dead slow to get that limit number. Perhaps off-the-end drag events should set the value to the limit?

regards,
Greg

---

Subject: Re: cw_dual_slider: a slider with two slides
Posted by Braedley on Fri, 03 Nov 2006 15:08:27 GMT
View Forum Message <> Reply to Message

Mike wrote:
> David Fanning wrote:
>> If you *must* use indexed color, don't bother us with it. Let us use all
>> the colors we want. Get the current decomposed state (GET_DECOMPOSED)
>> when you get the colors, and restore it when you restore the colors. :-)
>
> Done. This has been good for my continuing growth in the area of IDL
> colors. Instead of setting decomposed, I replaced all the color
> specifications with calls to fsc_color and color24 (with arguments come
> from widget_system_colors). I've also pixmap buffered the whole thing.
> See http://www.indyrad.iupui.edu/public/mmiller3/cw_dual_slider for
> the latest which now looks sort of windows-ish on windows, *nix-ish on
> *nix and who-knows-what-ish on macs.
>
> Mike
>
> P.S. I'm curious to see what it does look like on a mac if anyone
> wants to send a screen capture...

A vast improvement over the original version, although I did have a little problem with it when I had a color table loaded (the color table is a necessary evil for some routines that I use, and therefore I have to set device, decomposed=0). It still worked, but grabbed the colors from the color table, so it looked a little funky. Keep up the good work!

Braedley

---

Subject: Re: cw_dual_slider: a slider with two slides
Posted by David Fanning on Fri, 03 Nov 2006 15:15:18 GMT
View Forum Message <> Reply to Message

Braedley writes:

> A vast improvement over the original version, although I did have a
> little problem with it when I had a color table loaded (the color table
> is a necessary evil for some routines that I use, and therefore I have

> to set device, decomposed=0). It still worked, but grabbed the colors
> from the color table, so it looked a little funky. Keep up the good
> work!

This is more likely a problem with your own programs
not following the standard rule of "if you want your
colors to be right, load them your own damn self!"
And, preferably, just before you plan to use them. :-)

Cheers,

David
--
David Fanning, Ph.D.
Fanning Software Consulting, Inc.
Coyote's Guide to IDL Programming: http://www.dfanning.com/
Sepore ma de ni thui. ("Perhaps thou speakest truth.")

---