Subject: Color 'n linestyle picker for Plot
Posted by Pavel A. Romashkin on Thu, 08 Mar 2001 23:14:23 GMT
View Forum Message <> Reply to Message

Hello everybody,

Being stranded from IDL for a while due to the preparation for the wildest of the field expeditions we've ever undertaken, I finally got back to my Mac (sorry, David). Exclusively for the relaxation purposes, I quickly typed up the short code that I had no real purpose for, but a colleague of mine found useful. I am respectfully asking the more advanced of us (those who condemn Mac-borne simple-minded code :-) to ignore this program. However, I anticipate that it may even work on a Linux machine (considered here to be at the top of the food chain, mainly due to the set-up difficulties, I guess :-).
Basically, the attached program allows to pick a color and linestyle via a user interface. As you can tell, the last two programs in the attachement are not needed at all, they are just a quick hack to demonstrate how the chooser works. To test, (compile the attached module and) type:

device, decomposed=0 ; for color compatibility
loadct, 39 ; for nicer looks :-)
example_plot

You will see a window with default background. I recommend then typing

plot, findgen(10), backgr=255, color=1

on the command line under the plot area, then hit Enter. A plot should appear. Now use Style and Color buttons to set the style and color of subsequent plots. After using the buttons, on the plot's command line, type something like

oplot, findgen(10)-0.5

or anything else that will appear in the given data range, and hit Enter. The new plot should look like you've specified.
I though this might be useful to simple-minded widget programmers who want a simple way to control the main visual plot settings. The code will have to be adapted to the individual requirements, of course, but it is so small that anybody will be able to do this in minutes.

Cheers,
Pavel

```
;************************************************************ **************
pro PARPlotConfig_setLinestyle, event, init=init
compile_opt IDL2, OBSOLETE
```

```
if keyword_set(init) then begin
 ; Use EVENT parameter here as widget ID to define what are we doing.
 widget_control, event, set_uvalue='SET_LINESTYLE'
 tmp=bytarr(112, 16)+255b
 dash_index = lindgen(112)+112*7
 solid = lindgen(112)
 dotted = solid*(lindgen(112) mod 4 gt 1)
 dashed = solid*(reform(rebin(long([1,1,1,1,0,0,0]), 7, 16), 112))
 dash_dot = 3 + solid*(reform(rebin(long([1,1,1,1,0,0,1,0,0]), 9, 12), 108))
 dash_ddd =  solid*(reform(rebin(long([1,1,1,1,1,0,0,1,0,0,1,0,0,1,0,0]), 16, 7), 112))
 l_dash = solid*(lindgen(112) mod 10 gt 3)
 patterns = {a : solid, b : dotted, c : dashed, d : dash_dot, e : dash_ddd, f : l_dash}
 ; Fill the window with line patterns.
 tv, bytarr(112, 112)+255b
 for i = 0, 5 do begin
  tmp[dash_index[patterns.(i)]] = 1b
  tv, tmp, i
  tmp[*] = 255b
 endfor
endif else begin
 ; Get the user value. It is a pointer to plot_extra structure to be used by PLOT.
 widget_control, event.top, get_uvalue=stored_value
 plot_extra = *stored_value
 ; Set focus to the popup window.
 widget_control, event.id, get_value=new_window
 wset, new_window
 ; Figure out where the click occurred.
 loc = 6 - fix((event.y) / 16)
 ; Draw a rectangle around the selected color.
 tmp = bytarr(112, 16)+255b
 border_h = lindgen(112)
 border_v = lindgen(16) * 112
 tmp[[border_h, border_v, border_h+112*15, border_v+111]] = 0b
 tv, tmp, loc
 ; Wait so that the user can see the rectangle around the chosen line style.
 wait, 0.1
 ; Set plot_extra 's LINESTYLE property.
 temp = where('LINESTYLE' eq Tag_names(plot_extra), count)
 if count ne 0 then plot_extra.linestyle = loc else $
  plot_extra = create_struct(plot_extra, 'LINESTYLE', loc)
 ; Put plot_extra back to accessible location.
 *stored_value = plot_extra
endelse
end
 ;************************************************************ **************
pro PARPlotConfig_setColor, event, init=init
compile_opt IDL2, OBSOLETE
if keyword_set(init) then begin
```

```
 ; Use EVENT parameter here as widget ID to define what are we doing.
 widget_control, event, set_uvalue='SET_COLOR'
 tmp=bytarr(7, 7)
 for i = 0, !d.table_size-1 do begin
  tv, tmp, i
  tmp = tmp + 1B
 endfor
endif else begin
 ; Get the user value. It is a poiter to plot_extra structure to be used by PLOT.
 widget_control, event.top, get_uvalue=stored_value
 plot_extra = *stored_value
 ; Set focus to the popup window.
 widget_control, event.id, get_value=new_window
 wset, new_window
 ; Figure out where the click occurred.
 loc = fix((event.x) / 7) + ((15-fix((event.y) / 7)) * 16)
 ; Draw a rectangle around the selected color.
 tmp = bytarr(7, 7) + loc
 border = lindgen(7)
 if loc lt 80 then tmp[[border, border+42, border*7, border*7+6]] = 255B $
 else tmp[[border, border+42, border*7, border*7+6]] = 0B
 tv, tmp, loc
 ; Wait so that the user can see the rectangle around the chosen color.
 wait, 0.1
 ; Set plot_extra 's COLOR property.
 temp = where('COLOR' eq Tag_names(plot_extra), count)
 if count ne 0 then plot_extra.color = loc else $
  plot_extra = create_struct(plot_extra, 'COLOR', loc)
 ; Put plot_extra back to accessible location.
 *stored_value = plot_extra
endelse
end
;************************************************************ **************
pro PARPlotConfig_event, event
compile_opt IDL2, OBSOLETE

 if event.release then return ; Do nothing on RELASE of mouse button.
 widget_control, event.id, get_uvalue=action

 if action eq 'SET_COLOR' then PARPlotConfig_setColor, event ; Set COLOR
 if action eq 'SET_LINESTYLE' then PARPlotConfig_setLinestyle, event ; Set COLOR

 widget_control, event.top, /destroy ; Kill the popup window.
end
;************************************************************ **************
pro PARPlotConfig, plot_extra=plot_extra, $
 set_color=set_color, set_linestyle=set_linestyle
compile_opt IDL2, OBSOLETE
```

```
 stored_value = ptr_new(plot_extra) ; Create a commonly accessible location.
 old_window = !D.WINDOW ; Remember the old plot window.
 ; Create a popup window for defining plot parameters.
 top_ID = widget_base(title='Color', tlb_frame_attr=3, event_pro='PARPlotConfig_event')
 draw_ID = widget_draw(top_ID, xsize=112, ysize=112, /button)
 widget_control, top_ID, /realize, set_uvalue=stored_value
 widget_control, draw_ID, get_value = new_window
 wset, new_window ; Give focus to popup window.

 if keyword_set(set_color) then PARPlotConfig_setColor, draw_ID, /init
 if keyword_set(set_linestyle) then PARPlotConfig_setLinestyle, draw_ID, /init

 wset, old_window ; Return focus to the window that had it before.
 ; Dead_Top_ID is not being used. It is here to describe what that means.
 temp = widget_event(top_ID, bad_ID=dead_top_ID)
 ; Retrieve the modified plot_extra.
 plot_extra = *stored_value
 ptr_free, stored_value
end
 ;************************************************************ **************
pro example_plot_event, event
widget_control, event.top, get_uvalue = state
widget_control, state.(1), get_uvalue = plot_extra
case event.id of
state.(0) : PARplotConfig, plot_extra=plot_extra, /set_linestyle
state.(1) : PARplotConfig, plot_extra=plot_extra, /set_color
state.(2) : begin
 widget_control, state.(3), get_value=win_id
 wset, win_id
 widget_control, event.id, get_value=temp
 temp = execute(strjoin(temp)+', _extra=plot_extra')
 widget_control, event.id, set_value=''
 end
endcase
widget_control, state.(1), set_uvalue = plot_extra
end
 ;************************************************************ **************
pro example_plot
top_base = widget_base(row=2)
draw = widget_draw(top_base, xsize=600, ysize=450)
bottom_base = widget_base(top_base, /row)
linestyle = widget_button(bottom_base, value='Style')
color = widget_button(bottom_base, value='Color')
entry = widget_text(bottom_base, /editable)
state = {linestyle : linestyle, color : color, entry : entry, draw : draw}
plot_extra = {void_field : 'void'}
widget_control, top_base, set_uvalue = state
widget_control, color, set_uvalue = plot_extra
```

```
widget_control, top_base, /realize
xmanager, 'Example_plot', top_base, /no_block
end
```

File Attachments

1) example_plot.pro, downloaded 188 times