Subject: Re: Text processing on plots
Posted by gunter on Fri, 14 Feb 1997 08:00:00 GMT
View Forum Message <> Reply to Message

Jeff Hicke (jhicke@al.noaa.gov) wrote:
: Hello,
:
: I am interested in whether anyone has any IDL routines which allows one
: to overplot a variable length text string in the manner of a word
: processor. In other words, if I define a box (or width of a box), then
: the routine overplots the text within this box's width. If it is longer
: than one line, it will wrap to a new line. I'm looking at adding figure
: captions to IDL plots. Can anyone help?
:
: Thanks,
: Jeff Hicke
: NOAA Aeronomy Lab
: Boulder CO

Strange, I was just working on the same thing (well, almost.) I wanted to add text to a plot interactively. I've appended two routines at the end of this post for doing such things. The first is named add_text.pro and is a widget-based routine which prompts for the text to be added (multi-lines allowed) and includes a slider for controling the text-size. Clicking the 'OK' button starts the second routine, place.pro, which moves a ghost-box around the plot window under control of the mouse. The ghost-box is set to be the size of the text that will be added. Clicking any of the mouse buttons will place the text at the cursor location. The the idea behind place.pro is stolen from curbox.pro from the ESRG routines.

Note: I've just written these codes this morning and they are sparsely documented. Also, several checks should be implemented such as making sure that you are adding text to the correct plot window (if more than one is open), etc.

Let me know if there are major errors with these. Good luck!

--
david gunter
http://www.mcs.anl.gov/people/gunter/

"When you are a Bear of Very Little Brain, and you Think of Things, you find sometimes that a Thing which seemed very Thingish inside you is quite different when it gets out into the open and has other people looking at it."
- A.A. Milne, "The House At Pooh Corner"

```
 ------------------------------------------------------------ -----------------
pro add_text_event, event

widget_control, event.top, get_uvalue=info

eventname = tag_names(event, /STRUCTURE_NAME)

if eventname EQ 'WIDGET_BUTTON' then begin
   widget_control, event.id, get_value=button

   case button of

      ' Cancel ': begin
         widget_control, event.top, /destroy
      end

      ' OK ': begin
         widget_control, info.text, get_value=text
         n_lines = n_elements(text)

         if n_lines LT 1 then return  ; No text was entered.

; Remove NULL elements if at the end

         if text(n_lines-1) EQ '' then begin
            text = text(0:n_lines-2)
            n_lines=n_lines-1
         endif

; Make sure that some text has been entered
         text_idx = where(text NE '', hits)
         if hits GT 0 then begin

; Concatenate all lines, putting !c's for cariage returns.
            all_text=''
            for i=0, n_lines-1 do $
              all_text = all_text+text(i)+'!c'

; Get the value of the slider.
            widget_control, info.char_sizer, get_value=char_size
; Put the slider in the text window. (I didn't like the way the slider
; displayed it's own value.)
            widget_control, info.char_size, set_value=char_size

; Determine the height (in pixels) of a string.
; (Thanks go out to David Fanning!)
            box_height = !D.Y_CH_SIZE * n_lines * char_size * 1.1
```

```
; Convert to /NORMAL coordinates
            box_height = float(box_height)/float(!D.Y_VSIZE)

; The above fails for string widths because of spacing.  Instead use xyouts
; with charsize=-1 to get the width returned in /NORMAL coordinates.  No
; output is actually displayed. (Thanks again David!)
            xyouts, 0, 0, all_text, width=box_width, size=char_size, $
              charsize=-1
; Assign coordinates for interactive box display (in /NORMAL coordinates).
            left = 0.1
            right = left + box_width
            bottom = 0.9
            top =  bottom + box_height

; Call the interactive display routine.
            place, left, right, bottom, top, /UPPER_LEFT

; Output the text.
            xyouts, left, bottom-float(box_height/n_lines), all_text, $
              size=char_size, /NORMAL

          endif
          widget_control, event.top, /DESTROY
        end

    endcase

endif else if eventname EQ 'WIDGET_SLIDER' then begin

; Update the slider info.
    widget_control, info.char_sizer, get_value=char_size
    widget_control, info.char_size, set_value=char_size

endif

end

 ;************************************************************ *
;** Add_Text lets a user interactively add text to a graph **
 ;************************************************************ *

pro add_text, group_leader=group_leader

tlb = widget_base(title='Add Text', /COLUMN)
text_label = widget_label(tlb, value='Enter text below:', /ALIGN_LEFT)
text = cw_field(tlb, title=' ', value='', xsize=50, ysize=10)

char_sz_base = widget_base(tlb, /COLUMN, /FRAME, /ALIGN_CENTER)
```

```
char_size = cw_field(char_sz_base, title='Character Size:', xsize=2, $
                value=1, /NOEDIT)
char_sizer = widget_slider(char_sz_base, scroll=1, minimum=1, $
                    maximum=15, value=1, /SUPPRESS_VALUE, /DRAG)


exit_base = widget_base(tlb, /ROW, space=15)
ok_button = widget_button(exit_base, value=' OK ')
cancel_button = widget_button(exit_base, value=' Cancel ', /ALIGN_RIGHT)

widget_control, tlb, /REALIZE

info = {text:text,              $
       char_size:char_size,    $
       char_sizer:char_sizer   $
      }

widget_control, tlb, set_uvalue=info

xmanager, 'add_text', tlb, /MODAL, group_leader=group_leader

end

 ------------------------------------------------------------ -------------------
; Interactively choose a place to put a box (i.e. a legend box).
; [left, right, bottom, top] describe the edges of the box.
; By default, the box is dragged around the screen by the lower left corner,
; setting the UPPER_LEFT keyword drags it by the upper left corner.

pro place, left, right, bottom, top, upper_left=upper_left

; Save the current device setup, turn on XOR for "erase" feature.
device, get_graphics = prev_dvc, set_graphics = 6

x_size = !d.x_size
y_size = !d.y_size
x_vsize = !d.x_vsize
y_vsize = !d.y_vsize

; /NORMAL window coordinate ranges
xrange = [0.,1.]
yrange = [0.,1.]

if n_params() EQ 4 then begin

  Box_width = left > right - left < right
  Box_height = bottom > top  -  bottom < top
```

```
endif else begin

  Box_width = 0.2
  Box_height = Box_width * x_vsize/y_vsize

endelse

; Position the cursor.
; Note: !d.x_size is given in pixels.

x_curr = 0.5 * x_size          ; Horizontal center of graphics window.
y_curr = 0.5 * y_size          ; Vertical center.
tvcrs, x_curr, y_curr

width = x_vsize * Box_width
height = y_vsize * Box_height

if keyword_set(upper_left) then height = -height

; Set up box coordinates and draw it.

Box_x=[x_curr, x_curr+width, x_curr+width, x_curr, x_curr]
Box_y=[y_curr, y_curr, y_curr+height, y_curr+height, y_curr]
plots, Box_x, Box_y, /DEVICE

; Increment amounts

dx=1
dy=1

while 1 do begin

    x_old = x_curr
    y_old = y_curr

; Get latest cursor position
    cursor, x_curr, y_curr, /CHANGE, /DEVICE

    button_stat = !err

    if ( ((x_curr NE x_old) OR (y_curr NE y_old))  OR $
         (button_stat NE 0) ) then begin

; Erase the old box.
        plots, Box_x, Box_y, /DEVICE
        empty

; Keep the box in-bounds.
```

```
        xnew = x_curr > 0 < (x_size - width)
        ynew = y_curr > 0 < (y_size - height)

        if keyword_set(upper_left) then ynew = y_curr > abs(height) < y_size

        dx = xnew EQ x_curr
        dy = ynew EQ y_curr

        x_curr = xnew
        y_curr = ynew

; Draw the box in it's new position
        Box_x = [x_curr, x_curr+width, x_curr+width, x_curr, x_curr]
        Box_y = [y_curr, y_curr, y_curr+height, y_curr+height, y_curr]

        plots, Box_x, Box_y,/device
        empty

; Quit.
; If any mouse button was pressed, we quit.
; Note: for 2-button mice, remove the last condition (button_stat EQ 4).

        if (button_stat EQ 1) OR (button_stat EQ 2) or $
          (button_stat EQ 4) then begin
            plots, Box_x, Box_y, /DEVICE
            empty
            device, set_graphics = prev_dvc, cursor_standard=30

            left = poly(x_curr/x_vsize, xrange)
            right = left + Box_width
            bottom = poly(y_curr/y_vsize, yrange)
            top = bottom + Box_height

            return

        endif
    endif
endwhile

end
```