Subject: Re: Bug in Box_cursor?
Posted by Liyun Wang on Wed, 09 Oct 1996 07:00:00 GMT
View Forum Message <> Reply to Message

Jarrod Paul ABBOTT wrote:
>
> Situation: 1. Two dimentional graph displayed with (say) plot.
> 2. call box_cursor.
>
> Box cursor allows movement of the cursor with no problems and allows
> resizing the cursor with the middle mouse button. The problem arises when
> resizing the box, starting with (say) the mouse next to the bottom right
> hand corner. Resizing the window beyond the left had edge of the box
> results in the box being translated.
>
> Surely resizing the box should not result in a translation? Is this an
> error in box_cursor, or am I somewhere missing the obvious...
>
> I'm using IDL 4.0.1 (sunos sparc) under SunOS 5.5.

I reported that problem two years ago to RSI and offered a modified version (which is attached in this message) with a solution to that problem. RSI promised (prior to releasing of IDL 4.0) to include my modified version in their new release. Apparently it's not in 4.0 yet. To make it distinct from box_cursor, we call it box_cursor2. Try it out and see if it solves your problem.

Good luck.

--

==========
Liyun Wang | FAX: 301-286-0264
G1 Building 26, SOHO EAF | Voice: 301-286-3126
NASA, GSFC, Code 682.3 | Liyun.Wang.1@gsfc.nasa.gov
Greenbelt, MD 20771 | http://orpheus.nascom.nasa.gov/~lwang
==========


```
;------------------------------------------------------------ ----------
; Document name: box_cursor2.pro
; Created by:    Liyun Wang, GSFC/ARC, September 9, 1994
;
; Last Modified: Mon Jun  5 12:02:16 1995 (lwang@achilles.nascom.nasa.gov)
;------------------------------------------------------------ ----------
;
PRO box_cursor2, x0, y0, nx, ny, INIT = init, FIXED_SIZE = fixed_size, $
            message = message, color = color, anywhere=anywhere, $
            event_pro=event_pro, stc4event=event
;+
; NAME:
```

```
;       BOX_CURSOR2
;
; PURPOSE:
;       Emulate the operation of a variable-sized box cursor
;
; EXPLANATION:
;       This is a better improved version of BOX_CURSOR, a standard procedure
;       from the IDL user library. The added keywords are: COLOR, ANYWHERE,
;       EVENT_PRO, and STC4EVENT.
;
; CATEGORY:
;       Interactive graphics.
;
; CALLING SEQUENCE:
;       BOX_CURSOR2, x0, y0, nx, ny [, INIT = init] [, FIXED_SIZE = fixed_size]
;                [, COLOR = color]
; INPUTS:
;       No required input parameters.
;
; OPTIONAL INPUT PARAMETERS:
;       X0, Y0, NX, NY - The initial location (X0, Y0) and size (NX, NY)
;                of the box if the keyword INIT is set.  Otherwise,
;                the box is initially drawn in the center of the
;                screen.
;       EVENT_PRO      - Name of the procedure to be called when the boxed
;                cursor is manipulated. This procedure must have one
;                and only one positional parameter which is a
;                structure. This structure is passed in with the
;                keyword STC4EVENT and must have at least two tags
;                named X and Y being the cursor position in device
;                pixels.
;       STC4EVENT      - Structure to be processed by the procedure specified
;                by EVENT_PRO. It can have any number of tags, but X
;                and Y tags are required ones.
;
; KEYWORD PARAMETERS:
;       INIT       - If this keyword is set, X0, Y0, NX, and NY contain the
;                initial parameters for the box.
;       FIXED_SIZE - If this keyword is set, nx and ny contain the initial
;                size of the box. This size may not be changed by the
;                user.
;       MESSAGE    - If this keyword is set, print a short message describing
;                operation of the cursor.
;       COLOR      - Index of color to be used to draw the cursor. Default:
;                !d.n_colors-1
;       ANYWHERE   - Set this keyword to allow box to be moved outside the
;                window
;
```

```
; OUTPUTS:
; x0 - X value of lower left corner of box.
; y0 - Y value of lower left corner of box.
; nx - width of box in pixels.
; ny - height of box in pixels.
;
; The box is also constrained to lie entirely within the window.
;
; COMMON BLOCKS:
; None.
;
; SIDE EFFECTS:
; A box is drawn in the currently active window.  It is erased
; on exit.
;
; RESTRICTIONS:
; Works only with window system drivers.
;
; PROCEDURE:
; The graphics function is set to 6 for eXclusive OR.  This
; allows the box to be drawn and erased without disturbing the
; contents of the window.
;
; Operation is as follows:
; Left mouse button:   Move the box by dragging.
; Middle mouse button: Resize the box by dragging.  The corner
;  nearest the initial mouse position is moved.
; Right mouse button:  Exit this procedure, returning the
;       current box parameters.
;
; KNOWN PROBLEM:
;      The box can be off the display window when resizing. More
;      checking is needed to prevent this.
;
; MODIFICATION HISTORY:
; DMS, April, 1990.
; DMS, April, 1992.  Made dragging more intutitive.
; June, 1993 - Bill Thompson
;   prevented the box from having a negative size.
;      September 1, 1994 -- Liyun Wang
;                     Added the COLOR keyword
;      September 9, 1994 -- Liyun Wang, GSFC/ARC
;                     Prevented the box from jumpping around
;                     when resizing
;      May 26, 1995 -- Liyun Wang, GSFC/ARC
;                  Added the ANYWHERE keyword
;      June 5, 1995 -- Liyun Wang, GSFC/ARC
;                  Added EVENT_PRO and STC4EVENT keywords
```

```
;-

  DEVICE, get_graphics = old, set_graphics = 6 ; Set xor

  IF N_ELEMENTS(color) EQ 0 THEN color = !d.n_colors -1
  IF N_ELEMENTS(event_pro) NE 0 THEN BEGIN
    ok = datatype(event_pro) EQ 'STR'
    IF NOT ok THEN MESSAGE, 'Error in compiling the event procedure.', /cont
  ENDIF ELSE ok = 0

  IF KEYWORD_SET(MESSAGE) THEN BEGIN
    IF KEYWORD_SET(fixed_size) THEN BEGIN
      PRINT, "Drag Left button to move box."
      PRINT, "Right button when done."
    ENDIF ELSE BEGIN
      PRINT, "Drag Left button to move box."
      PRINT, "Drag Middle button near a corner to resize box."
      PRINT, "Right button when done."
    ENDELSE
  ENDIF

  IF KEYWORD_SET(init) EQ 0 THEN BEGIN ;Supply default values for box:
    IF KEYWORD_SET(fixed_size) EQ 0 THEN BEGIN
      nx = !d.x_size/8       ;no fixed size.
      ny = !d.x_size/8
    ENDIF
    x0 = !d.x_size/2-nx/2
    y0 = !d.y_size/2-ny/2
  ENDIF

  button = 0
  GOTO, middle

  WHILE 1 DO BEGIN
    old_button = button
    cursor, x, y, 2, /dev ;Wait for a button
    button = !err
    IF (old_button EQ 0) AND (button NE 0) THEN BEGIN
      mx0 = x  ;For dragging, mouse locn...
      my0 = y
      x00 = x0              ;Orig start of ll corner
      y00 = y0
    ENDIF
    IF !err EQ 1 THEN BEGIN   ;Drag entire box?
      x0 = x00 + x - mx0
      y0 = y00 + y - my0
    ENDIF
    IF (!err EQ 2) AND (KEYWORD_SET(fixed_size) EQ 0) THEN BEGIN ;New size?
```

```
      IF old_button EQ 0 THEN BEGIN ;Find closest corner
        min_d = 1e6
        FOR i = 0,3 DO BEGIN
          d = FLOAT(px(i)-x)^2 + FLOAT(py(i)-y)^2
          IF d LT min_d THEN BEGIN
            min_d  = d
            corner = i
          ENDIF
        ENDFOR
        nx0 = nx          ;Save sizes.
        ny0 = ny
      ENDIF
      dx = x-mx0 & dy = y-my0 ;Distance dragged...
 ;-------------------------------------------------------- ----------------
;       The major change was made here. After the closest corner is
;       found, the opposite corner is fixed. This prevents the box
;       from jumping around              -- Liyun Wang, GSFC/ARC
 ;-------------------------------------------------------- ----------------
      CASE corner OF
        0: BEGIN
          IF (dx GT nx0) THEN BEGIN
            x0 = x00+nx0
            nx = dx-nx0
          ENDIF ELSE BEGIN
            x0 = x00+dx
            nx = nx0-dx
          ENDELSE
          IF (dy GT ny0) THEN BEGIN
            y0 = y00+ny0
            ny = dy-ny0
          ENDIF ELSE BEGIN
            y0 = y00+dy
            ny = ny0-dy
          ENDELSE
        END
        1: BEGIN
          IF (dx LE -nx0) THEN BEGIN
            nx = -(nx0+dx)
            x0 = x00-nx
          ENDIF ELSE BEGIN
            nx = nx0+dx
            x0 = x00
          ENDELSE
          IF (dy GT ny0) THEN BEGIN
            y0 = y00+ny0
            ny = dy-ny0
          ENDIF ELSE BEGIN
            y0 = y00+dy
```

```
            ny = ny0-dy
          ENDELSE
        END
        2: BEGIN
          IF (dx LE -nx0) THEN BEGIN
            nx = -(nx0+dx)
            x0 = x00-nx
          ENDIF ELSE BEGIN
            nx = nx0+dx
            x0 = x00
          ENDELSE
          IF (dy LE -ny0) THEN BEGIN
            ny = -(ny0+dy)
            y0 = y00-ny
          ENDIF ELSE BEGIN
            ny = ny0+dy
            y0 =  y00
          ENDELSE
        END
        3: BEGIN
          IF (dx GT nx0) THEN BEGIN
            x0 = x00+nx0
            nx = dx-nx0
          ENDIF ELSE BEGIN
            x0 = x00+dx
            nx = nx0-dx
          ENDELSE
          IF (dy LE -ny0) THEN BEGIN
            ny = -(ny0+dy)
            y0 = y00-ny
          ENDIF ELSE BEGIN
            ny = ny0+dy
            y0 = y00
          ENDELSE
        END
      ENDCASE
    ENDIF
    PLOTS, px, py, col=color, /dev, thick=1, lines=0 ;Erase previous box
    EMPTY                    ;Decwindow bug

    IF !err EQ 4 THEN BEGIN   ;Quitting?
      DEVICE,set_graphics = old
      RETURN
    ENDIF

middle:

    IF NOT KEYWORD_SET(anywhere) THEN BEGIN
```

```
 ;------------------------------------------------------- ----------
;       Never allow the box to be outside window
 ;------------------------------------------------------- ----------
        x0 = x0 > 0
        y0 = y0 > 0
        x0 = x0 < (!d.x_size-1 - nx)
        y0 = y0 < (!d.y_size-1 - ny)
      ENDIF ELSE BEGIN
        x0 = x0 > (-nx)
        y0 = y0 > (-ny)
        x0 = x0 < (!d.x_size-1)
        y0 = y0 < (!d.y_size-1)
      ENDELSE
      IF ok THEN BEGIN
        event.x = x0
        event.y = y0
        CALL_PROCEDURE, event_pro, event
      ENDIF

      px = [x0, x0 + nx, x0 + nx, x0, x0] ;X points
      py = [y0, y0, y0 + ny, y0 + ny, y0] ;Y values

      PLOTS,px, py, col=color, /dev, thick=1, lines=0 ;Draw the box
      wait, .1                ;Dont hog it all
    ENDWHILE
END

 ;------------------------------------------------------- ----------
; End of 'box_cursor2.pro'.
 ;------------------------------------------------------- ----------
```

File Attachments

1) box_cursor2.pro, downloaded 99 times