Subject: Re: David Fanning's rubberband box
Posted by davidf on Fri, 13 Mar 1998 08:00:00 GMT
View Forum Message <> Reply to Message

Joseph Scott Stuart (nospam@ll.mit.edu) writes:

> I've recently gotten David Fanning's book, and I like it quite a bit.
> Good job, David! I am having a problem though with the rubberband box
> code on page 117. I'm using IDL 5.0.2 on an SGI. When I am finished
> drawing the rubberband box and release the mouse button, the program
> does not exit the repeat loop. I have to push another button before it
> will register that I've released the first button. I added the
> following line just before the ENDREP:
>
>   cursor, x, y, /NoWait
> ENDREP UNITL !Mouse.Button NE 1
>
> That mostly fixed the problem, but it still seems to only work
> sporadically. That is, sometimes I'll draw a box, release the button,
> and it ends the loop as it is supposed to, other times it does not,
> and I have to either redraw the box or push another button. This is a
> problem because I was planning to use the other mouse buttons for
> other things (middle button to zoom out, right button to exit the
> program). This behavior persists when I run the program on the SGI
> with the display on the console or on a PC (NT) with X-Win32

Whoops, blush. On getting home and having a look at this
code, I realized that you could probably get what you want
just by changing the keyword on the CURSOR command that is
inside the loop to CHANGE from WAIT. In this state, the
box is drawn as long as you hold the left button down and
drag, but you exit the program when you release the button.
(Because !Mouse.Button is set to 0.)

Here is a poor man's box cursor that draws a rubberband
box in any window. (You can pass the window's ID as a
parameter.) Moreover, keywords specify the color index used
to draw the box (COLOR) and whether you want the output
box coordinates to be in data coordinates (DATA) or normalized
coordinates (NORMAL). The default is to report the box
coordinates in DEVICE coordinates. The box coordinates
are ordered like the are in the Position keyword
(i.e, [smallest_X, smallest_Y, largest_X, largest_Y]).
I've written this as a function because I like to get
the box coordinates back explicitly.

To see it work, try this:

```
  image = BytScl( Dist(200), Top=149)
  TVLCT, 255, 255, 0, 150
  Window, XSize=200, YSize=200
  TV, image
  box = DrawBox(Color=150)
```

Cheers,

David

```
 ************************************************************ ****
Function DrawBox, $
  wid, $        ; ID of window where box is drawn. (Default: !D.Window)
  Color=color, $ ; The color index of the box. (Default: !D.N_Colors-1)
  Data=data, $   ; Box coordinates returned as DATA coordinates.
  Normal=normal  ; Box coordinates returned as NORMAL coordinates.

  ; Check for parameters.

IF N_Params() EQ 0 THEN wid = !D.Window > 0
IF N_Elements(color) EQ 0 THEN color = (!D.N_Colors - 1) < 255

  ; Make requested window active. Get size of window.

WSet, wid
xsize = !D.X_VSize
ysize = !D.Y_VSize

  ; Create a pixmap for erasing the box. Copy window
  ; contents into it.

Window, /Pixmap, /Free, XSize=xsize, YSize=ysize
pixID = !D.Window
Device, Copy=[0, 0, xsize, ysize, 0, 0, wid]

  ; Get the first location in the window. This is the
  ; static corner of the box.

WSet, wid
Cursor, sx, sy, /Down, /Device

  ; Go into a loop.

REPEAT BEGIN

    ; Get the new cursor location (dynamic corner of box).

  Cursor, dx, dy, /Change, /Device
```

```
      ; Erase the old box.

   Device, Copy=[0, 0, xsize, ysize, 0, 0, pixID]

      ; Draw the new box.

   PlotS, [sx, sx, dx, dx, sx], [sy, dy, dy, sy, sy], $
      /Device, Color=color

ENDREP UNTIL !Mouse.Button NE 1

   ; Erase the final box.

Device, Copy=[ 0, 0, xsize, ysize, 0, 0, pixID]

   ; Delete the pixmap.

WDelete, pixID

   ; Order the box coordinates.

sx = Min([sx,dx], Max=dx)
sy = Min([sy,dy], Max=dy)

   ; Need coordinates in another coordinate system?

IF Keyword_Set(data) THEN BEGIN
   coords =  Convert_Coord([sx, dx], [sy, dy], /Device, /To_Data)
   RETURN, [coords[0,0], coords[1,0], coords[0,1], coords[1,1]]
ENDIF

IF Keyword_Set(normal) THEN BEGIN
   coords =  Convert_Coord([sx, dx], [sy, dy], /Device, /To_Normal)
   RETURN, [coords[0,0], coords[1,0], coords[0,1], coords[1,1]]
ENDIF

   ; Return device coordinates, otherwise.

RETURN, [sx, sy, dx, dy]
END
```

David Fanning, Ph.D.
Fanning Software Consulting
E-Mail: davidf@dfanning.com
Phone: 970-221-0438
Coyote's Guide to IDL Programming: http://www.dfanning.com/