Subject: Re: Gravity ?
Posted by joel on Tue, 17 May 1994 15:06:00 GMT
View Forum Message <> Reply to Message

In article <2qqtui$180@gould.ualr.edu>, LINDSTROM@acs.harding.edu (Greg Lindstrom) writes...
> Greeting All-
>
> I am running PV-Wave4.2CL on a Sun SPARC IPC (SunOS 4.1.3) and
> X11R5. What I would really like to do is set "gravity" on the
> cursor in my display window so that the crosshair lines will
> extend to the edges of the display window. I have seen it done
> in "X", but not in WAVE. Can it be done? Can you tell me how?
>
> Thanks,
>
> Greg Lindstrom
> Harding University
>
> BTW- My grant runs out this summer. If you are looking for a
> programmer/administrator.....

I have been working off-and-on on a full-screen cursor procedure. I submitted a preliminary version to the UIT IDLASTRO library, but I'm not 100% happy with it. I used the DEVICE,SET_GRAPHICS=6, but could not control the color of the overplot (the XOR defined by SET_GRAPHICS=6 is not a *true* XOR, but the color translation table is also involved somehow...).

I decided to make some changes similar to what is in the JHUAPL library routine MOVCROSS. I'm still not 100% satisfied (see comments under "BUGS" below), but it works well enough.

Any suggestions always appreciated,
Joel Parker

----------snip snip----------

```
pro RDPLOT, x, y, WaitFlag, DATA=Data, DEVICE=Device, NORMAL=Normal, $
   NOWAIT=NoWait, WAIT=Wait, DOWN=Down, CHANGE=Change, $
   PRINT=Print, XTITLE=XTitle, YTITLE=YTitle,  FULLCURSOR=FullCursor, $
   LINESTYLE=Linestyle, NOCLIP=NoClip, COLOR=Color, CROSS=Cross
 ;********************************************************** ********************
;+
; NAME:
; RDPLOT
;
; PURPOSE:
; This program is designed to essentially mimic the IDL CURSOR command,
```

```
; but with the additional options of continuously printing out the data
; values of the cursor's position, and using a full-screen cursor rather
; than a small cross cursor.  The Full screen cursor uses PLOTS and
; TV/TVRD commands to make the large cursor.
;
; CALLING SEQUENCE:
; rdplot, [X, Y, WaitFlag], [/DATA, /DEVICE, /NORMAL,
;  /NOWAIT, /WAIT, /DOWN, /CHANGE,   PRINT=, XTITLE=, YTITLE=,
;  /FULLCURSOR, LINESTYLE=, THICK=, /NOCLIP, COLOR=, /CROSS]
;
; REQUIRED INPUTS:
; None.
;
; OPTIONAL INPUTS:
; WAITFLAG = if equal to zero it sets the NOWAIT keyword {see below}
;
; OPTIONAL KEYWORD PARAMETERS:
; DATA = Data coordinates are returned.
; DEVICE = device coordinates are returned.
; NORMAL = normal coordinates are returned.
; NOWAIT = if non-zero the routine will immediately return the cursor's
;  present position.
;       WAIT = if non-zero will wait for a mouse key click before returning.
;       DOWN = equivalent to WAIT
;       CHANGE = returns when the mouse is moved OR if a key is clicked.
; PRINT = if non-zero will continuously print out the data values of the
;  cursor's position, if PRINT>1 will printout a brief header
;  describing the mouse button functions.
; XTITLE = label used to describe the values of the abscissa if PRINT>0
; YTITLE = label used to describe the values of the ordinate if PRINT>0
; FULLCURSOR = if non-zero default cursor is blanked out and full-screen
;  (or full plot window, depending on the value of NOCLIP) lines
;  are drawn; their intersecton is centered on the cursor position.
;       LINESTYLE = style of line that makes the full-screen cursor.
;       NOCLIP = if non-zero will make a full-screen cursor, otherwise it will
;  default to the value in !P.NOCLIP.
; COLOR = color of the full-screen cursor.
; CROSS = if non-zero will show the regular cross AND full screen cursors.
;
; NOTES:
; Note that this procedure does not allow the "UP" keyword/flag...which
; doesn't seem to work too well in the origianl CURSOR version anyway.
;
;       If a data coordinate system has not been established, then RDPLOT
;  will create one identical to the device coordinate system.   Note
; that this kluge is required even if the user specified /NORMAL
; coordinates, since CURFULL makes use of the OPLOT procedure.  This new
; data coordiante system is effectively "erased" (!X.CRange and !Y.CRange
```

```
; are both set to zero) upon exit of the routine so as to not change the
; plot status from the user's point of view.
;
; Only tested on X-windows systems.    If this program is interrupted,
; the graphics function might be left in a non-standard state.   Type
; DEVICE,SET_GRAPHICS=3 to return the standard graphics function.
;
; PROCEDURE:
; Basically is a bells-n-whistles version of the CURSOR procedure.  All
; the details are covered in the above discussion of the keywords.
;
; BUGS:
; If a part of the plotting window is covered by another window, the
; TVRD and PLOTS commands used in FULLCORSOR mode will not work correctly
; in the area covered by the other window.  It will tend to erase/smudge
; lables and lines, and add all sorts of noise to the plot.
;
; The response is a bit slow overall and jittery because of the plotting
;  and tv-reading/overwriting, but that's how it goes...
;
; MODIFICATION HISTORY:
; Written by J. Parker  22 Nov 93 [originally called CURCROSS]
; Create data coordinates if not already present, W. Landsman Nov. 93
; Modified to add continuous printout of data values, COLOR keyword, and
;  FULLCURSOR keyword (so that default is that it acts just like
;  the cursor command).  Renamed RDPLOT.  J. Parker  20 Apr 94
; Modified to use TVRD and PLOTS commands (as well as a number of other
;  modifications) patterened after the JHUAPL library's procedure
;  MOVCROSS.  J. Parker   17 May 94
;-
 ;********************************************************** ********************
On_error,2
if ((!D.Flags and 256) ne 256) then FullCursor = 0
FullCursor = keyword_set(FullCursor)

;
;   If plotting coordinates are not already established, and the NORMAL
; keyword is not set, then use device coordinates.
;   Note that even if this procedure was called with the DATA keyword set, that
; the DEVICE keyword will always take precedence over the DATA keyword in the
; cursor command.  However, if the NORMAL and DEVICE keywords are both set,
; then very strange values are returned.
;
UndefinedPlot = (total(abs(!X.CRange)) eq 0)
if UndefinedPlot then plot, [0,!D.X_Size], [0,!D.Y_Size], /NODATA, $
   XSTYLE=5, YSTYLE=5, XMARGIN=[0,0], YMARGIN=[0,0]

;
```

```
;   Check to see if the user does not want to wait.
;
if (N_Params() eq 3) then NoWait = (WaitFlag eq 0)
if keyword_set(NoWait) then begin
   cursor, X, Y, /NOWAIT, DATA=Data, DEVICE=Device, NORMAL=Normal
   return
endif

;
;   Set up carriage return and line feed variables for the formatted printout.
;   If Print>1, then printout the informative header.
;
if keyword_set(Print) then begin
   CR = string("15b)
   LF = string("12b)
   if not(keyword_set(XTitle)) then XTitle = "X"
   if not(keyword_set(YTitle)) then YTitle = "Y"
   Format  = "($,'  " + XTitle + " = ',A13, '    " + YTitle + " = ',A13, A)"
   Blanks  = "                    "
endif else Print = 0

if (Print gt 1) then begin
   print, ' '
   print, 'Mouse Button:    LEFT         MIDDLE        RIGHT'
   print, 'Result Action:   New Line     Nothing       Exit'
   print, ' '
endif



;
;   If using the full-screen cursor, set up the linestyle, clipping, and color
; keywords for the plots commands.  Blank out the regular cross cursor if the
; CROSS keyword is not set.
;
if FullCursor then begin
   if not(keyword_set(Linestyle)) then Linestyle = 0
   NoClip = keyword_set(NoClip)
   if not(keyword_set(Color)) then Color = !D.N_Colors - 1
   if not(keyword_set(Cross)) then device, CURSOR_IMAGE=intarr(16)
endif

;
;   If the Change keyword isn't set and if the cursor is beyond the boundaries
; of the plot, then wait until the cursor is moved within the plot.  Then read
; the cursor's values in the desired coordinate system.
;
Change = keyword_set(Change)
cursor, X, Y, /NOWAIT
```

```
if ( not(Change) and ((X lt !X.CRange(0)) or (X gt !X.CRange(1)) or $
   (Y lt !Y.CRange(0)) or (Y gt !Y.CRange(1))) ) then cursor, X, Y, /CHANGE
cursor, X, Y, /NOWAIT, DATA=Data, DEVICE=Device, NORMAL=Normal

;
;   Initialize the !Err variable.  The value of !Err corresponds to the BYTE
; value of the buttons on the mouse from left to right, lowest bit first.  So,
; the left button gives !Err = 1, next button gives !Err = 2, then 4.
;   Begin the loop that will repeat until a button is clicked (or a change if
; that is what the user wanted).
;   Wait for a change (movement or key click).  Delete the old lines, and
; if we don't exit the loop, repeat and draw new lines.
;
!Err   = 0
repeat begin

;
;   Determine the cursor's device coordinates.  If doing a full-screen cursor,
; overplot two full-screen lines intersecting at that position.
;
   DevPos =  convert_coord(X,Y,DATA=Data,DEVICE=Device,NORMAL=Normal,/TO_ DEV)
   DevPos = (DevPos > 0) < ([!D.X_Size, !D.Y_Size] - 1)
   if FullCursor then begin
      CutCol = tvrd(DevPos(0),0,1,!D.Y_Size)
      CutRow = tvrd(0,DevPos(1),!D.X_Size,1)
      plots, DevPos(0), [0,!D.Y_Size], /DEVICE, NOCLIP=NoClip, COLOR=Color, $
         LINESTYLE=Linestyle
      plots, [0,!D.X_Size], DevPos(1), /DEVICE, NOCLIP=NoClip, COLOR=Color, $
         LINESTYLE=Linestyle
   endif

;
;   If printing out data values, do so.
;
   if (Print gt 0) then begin
      if (!Err eq 1) then begin         ; signal for a new line
         print, LF, format="($,a)"
         while (!Err ne 0) do begin     ; if button is held down, don't print
            wait, 0.1
            cursor, X, Y, /NOWAIT
         endwhile
      endif
      print, strtrim(X,2)+Blanks, strtrim(Y,2)+Blanks, CR, format=format
   endif

;
;   Check to see that the cursor's current position is really the last measured
; position (the mouse could have moved during a delay in the last section).  If
```

```
; so, then go on.  If not, then wait for some change in the mouse's status
; before going on.
;  In either case, once we are going on, then if doing a full-screen cursor,
; "overplot" the previous lines with the tv command.  Repeat until exit signal.
;
   cursor, XX, YY, /NOWAIT, DATA=Data, DEVICE=Device, NORMAL=Normal
   if ((XX ne X) or (YY ne Y)) then begin
      X = XX
      Y = YY
   endif else cursor, X, Y, /CHANGE, DATA=Data, DEVICE=Device, NORMAL=Normal

   if FullCursor then begin
      tv, CutCol, DevPos(0), 0
      tv, CutRow, 0, DevPos(1)
   endif

endrep until (Change or ((!Err ne 0) and (Print eq 0)) or (!Err eq 4))

if (Print gt 0) then print, LF

;
;   Go back to the default TV and cursor in case it was changed.  Also erase the
; plot ranges if they originally were not defined.
;
device, /CURSOR_CROSSHAIR

if UndefinedPlot then begin
   !X.CRange = 0
   !Y.CRange = 0
endif

return
end   ;   RDPLOT   by   Joel Parker   16 May 94
```