Subject: Gravity ?
Posted by LINDSTROM[1] on Wed, 11 May 1994 15:34:42 GMT
View Forum Message <> Reply to Message

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.....

Subject: Re: Gravity ?
Posted by thompson on Sat, 14 May 1994 17:42:24 GMT
View Forum Message <> Reply to Message

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 think this may be another entry for the FAQ. I've come across two different routines that do this, which I include below. I believe the first is available as part of the JHU/APL IDL library, via anonymous ftp to fermi.jhuapl.edu [128.244.147.18] in /pub/idl. There may be a more up-to-date version there; this one is pretty old.

Bill Thompson

```
 ============================================================
==================
 ;------------------------------------------------------------ --
;+
; NAME:
;       MOVCROSS
; PURPOSE:
;       Interactive cross-hair on a plot.
; CATEGORY:
; CALLING SEQUENCE:
;       movcross, [x, y]
; INPUTS:
; KEYWORD PARAMETERS:
;       Keywords:
;         COLOR=c set color of line.
;         LINESTYLE=s line style.
; OUTPUTS:
;       x = data X coordinate of cross-hair.      out
;       y = data Y coordinate of cross-hair.      out
; COMMON BLOCKS:
; NOTES:
;       Note: Works in data coordinates so must make a plot first.
; MODIFICATION HISTORY:
;       R. Sterner, 1 Feb, 1991
;
; Copyright (C) 1991, Johns Hopkins University/Applied Physics Laboratory
; This software may be used, copied, or redistributed as long as it is not
; sold and this copyright notice is reproduced on each copy made.  This
; routine is provided as is without any express or implied warranties
; whatsoever.  Other limitations apply as described in the file disclaimer.txt.
;-
 ;------------------------------------------------------------ --

 pro movcross, xdt, ydt, color=color, linestyle=linestyle, $
   help=hlp

 if keyword_set(hlp) then begin
   print,' Interactive cross-hair on a plot.'
   print,' movcross, [x, y]'
   print,'   x = data X coordinate of cross-hair.      out'
   print,'   y = data Y coordinate of cross-hair.      out'
   print,' Keywords:'
   print,'   COLOR=c set color of line.'
   print,'   LINESTYLE=s line style.'
         print,' Note: Works in data coordinates so must make a plot first.'
```

```
  return
 endif

       if (total(abs(!x.crange)) eq 0.0) then begin
         print,' Error in movcross: data coordinates not yet established.'
         print,'   Must make a plot before calling movcross.'
         return
       endif

       print,' '
       print,' Interactive cross-hair.'
       print,' Move with mouse.'
       print,' Left button:   List X,Y  position.'
       print,' Middle button: Quit, erase cross-hair.'
       print,' Right button:  Quit, keep cross-hair.'
       print,' '

 flag = 0
 yl = 0
 xl = 0
 if n_elements(color) eq 0 then color = 255
 if n_elements(linestyle) eq 0 then linestyle = 0
       tmp = [!y.range, !y.crange]
       yy = [min(tmp), max(tmp)]
       tmp = convert_coord([0,0],yy,/to_dev)
       yydv = transpose(tmp(1,0:1))
       tmp = [!x.range, !x.crange]
       xx = [min(tmp), max(tmp)]
       tmp = convert_coord( xx, [0,0], /to_dev)
       xxdv = transpose(tmp(0,0:1))

loop: tvrdc, xdt, ydt, 0, /data ; Read data coordinates of cursor.
 xdt = xdt > xx(0) < xx(1)
 ydt = ydt > yy(0) < yy(1)
 if !err ne 0 then begin  ; Button pressed?
        if !err eq 1 then begin
          print,' X,Y = ',xdt,ydt
          wait, .1
        endif
        if !err eq 4 then return
        if !err eq 2 then begin
          tv, tx, xl, 0  ; Erase X line on exit.
          tv, ty, 0, yl   ; Erase Y line on exit.
          return
        endif
 endif
 wait, 0    ; Pause.
 tmp = convert_coord(xdt, ydt, /to_dev) ; Convert data to device coord.
```

```
 xdv = tmp(0)  & ydv = tmp(1)
 if flag eq 0 then ty = tvrd(0,ydv,!d.x_size,1)   ; row at ydv 1st time.
       if flag eq 0 then tx = tvrd(xdv,0,1,!d.y_size)   ; col at xdv 1st time.
 if flag eq 1 then begin  ; Not first time?
   if (ydv ne yl) or (xdv ne xl) then begin ; Cursor moved?
           tv, tx, xl, 0                ; Replace old column at xl.
     tv, ty, 0, yl     ; Replace old row at yl.
           tx = tvrd(xdv,0,1,!d.y_size)   ; Read new column at xdv.
     ty = tvrd(0, ydv,!d.x_size,1)  ; Read new row at ydv.
   endif
 endif
 flag = 1   ; Set first time flag.
       xl = xdv                     ; Remember where column read.
 yl = ydv   ; Remember where row read.
       ;---  Plot vertical at new = xdv. ----
       if (xdt ge xx(0)) and (xdt le xx(1)) then begin
         plots, [xdv, xdv], yydv, color=color, $
           linestyle=linestyle, /dev
 endif
 ;---  Plot horizontal at new = ydv. ----
 if (ydt ge yy(0)) and (ydt le yy(1)) then begin
   plots, xxdv, [ydv, ydv], color=color, $
     linestyle=linestyle, /dev
 endif
 goto, loop
 end

 ============================================================
==================
Newsgroups: comp.lang.idl-pvwave
Path:  nsisrv!ames!sun-barr!cs.utexas.edu!zaphod.mps.ohio-state.edu
!news.acns.nwu.edu!osssenu.astro.nwu.edu!pendleton
From: pendleton@ossenu.astro.nwu.edu (Jim Pendleton)
Subject: Re: Giant crosshairs (with arrows)
Message-ID: <1992Aug29.174136.1@ossenu.astro.nwu.edu>
Lines: 123
Sender: usenet@news.acns.nwu.edu (Usenet on news.acns)
Organization: GRO/OSSE Northwestern U.
References: <laa-270892173708@berserk.c3.lanl.gov> <Btpv88.4ss@csn.org>
Date: Sat, 29 Aug 1992 22:41:36 GMT

In article <Btpv88.4ss@csn.org>, james@teal.csn.org (Jim Phillips) writes:
>  In article <laa-270892173708@berserk.c3.lanl.gov> laa@lanl.gov (Lee Ankeny) writes:
>> Hello Wave gurus,
>>
>> I'd like to implement a "giant crosshair", like the old tek terminals used
>> to have. It would extend from top edge to bottom edge, and left edge to
>> right edge of the window, intersecting at the current mouse pointer
```

>> location.
>>
>
> Hi Lee. You might give this a try. Not guaranteed to be bullet proof :-)
> But it should get you going in the right direction. Courtesy of
> Mike 4 Mayer, PVI field Technical Sales Engineer.
>
> James K. Phillips Sprocket Scientist pvi!james@boulder.colorado.edu
> Precision Visuals, Inc. (303) 530-9000
> 6260 Lookout Rd Boulder - A quaint little town nestled between
> Boulder, CO 80301 the ROCKIES and REALITY.

I've modified BIGHAIR to optionally accept arrow keys from the keyboard (your escape sequences may vary...I'm using ESC[A, B, C, D). Also the blips from the continuous overplotting have been eliminated by selectively plotting only when the cursor position changes.

> ----8<----------CUT HERE -----8<---------------CUT HERE-------------

```
PRO BIGHAIR, x, y, keyboard=keyboard, accelerate=accelerate
;
; Procedure to get cursor location in current window,
; using cursor lines across the entire window.
; The X and Y cursor locations (in pixels) are returned.
; Further modification could allow Normalized or
; Data coordinates to be returned.
; /Keyboard accepts arrow keys input from keyboard, rather than mouse
; Accelerate = number of pixels to jump in "/keyboard" mode.
;
; Mike Mayer, 8/28/92
; Modified:
; Jim Pendleton, Dept. Physics & Astronomy
; Northwestern U.
; 8/29/92
;
; Usage:
; WAVE> BIGHAIR, x, y [,/Keyboard [,Accelerate=# pixels]]
;
DEVICE, Get_Graphics = oldmode
DEVICE, Set_Graphics = 6 ; XOR write mode.
xmax = !D.X_Vsize
ymax = !D.Y_Vsize
!Err = 0 ; Reset.
oldx = 0
oldy = 0
If (not Keyword_Set(Accelerate)) then Begin
 Accelerate = 1.
EndIf
If (not Keyword_Set(KeyBoard)) then Begin
```

```
 PRINT, 'Press left mouse button to quit...'
 CURSOR, oldx, oldy, /Device, /Change
EndIf Else Begin
 PRINT, 'Press space bar to quit...'
 oldx = xmax/2
 oldy = ymax/2
 x = oldx
 y = oldy
EndElse
PLOTS, [oldx, oldx], [0, ymax], /Device  ; Draw first crosshairs.
PLOTS, [0, xmax], [oldy, oldy], /Device  ; Draw first crosshairs.
EMPTY
Up = String(27B) + '[A'
Down = String(27B) + '[B'
Right = String(27B) + '[C'
Left = String(27B) + '[D'
Clear = Get_Kbrd(0)
Null = ''
WHILE !Err NE 1 DO BEGIN
   If (Keyword_Set(Keyboard)) then Begin
    A = String(Byte(Get_Kbrd(0)))
    While ((A ne String(27B)) and (StrUpCase(A) ne ' ')) Do Begin
     A = String(Byte(Get_Kbrd(0)))
    EndWhile
    B = String(Byte(Get_Kbrd(0)))
    C = String(Byte(Get_Kbrd(0)))
    Key = A + B + C
   EndIf Else Begin
    CURSOR, x, y, /Device, /Change
   EndElse
   PLOTS, [oldx, oldx], [0, ymax], /Device  ; Erase last crosshairs.
   PLOTS, [0, xmax], [oldy, oldy], /Device ; Erase last crosshairs.
   If (Keyword_Set(Keyboard)) then Begin
    !Err = 0
    Case Key Of
  Up : y = oldy + Accelerate
  Down : y = oldy - Accelerate
  Left : x = oldx - Accelerate
  Right : x = oldx + Accelerate
  Null : x = x
  Else : !Err = 1
    EndCase
   EndIf
   If ((x ne oldx) or (y ne oldy)) then Begin
    PLOTS, [x, x], [0, ymax], /Device ; Draw new crosshairs.
    PLOTS, [0, xmax], [y, y], /Device ; Draw new crosshairs.
    oldx = x
    oldy = y
```

```
    EMPTY
   EndIf
ENDWHILE
If (not Keyword_Set(Keyboard)) then Begin
;
; Already erased in Keyboard mode.
;
 PLOTS, [oldx, oldx], [0, ymax], /Device  ; Erase last crosshairs.
 PLOTS, [0, xmax], [oldy, oldy], /Device ; Erase last crosshairs.
EndIf
Empty
DEVICE, Set_Graphics = oldmode
PRINT, 'X = ' + STRTRIM(x, 2), '   Y = ' + STRTRIM(y, 2)
END
```
> ----8<----------CUT HERE -----8<---------------CUT HERE----------

Jim Pendleton, SysMgr, etc.
GRO/OSSE, Dept. Physics & Astronomy
Northwestern U.
pendleton@ossenu.astro.nwu.edu

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
```