Subject: Re: IDL and X and PS
Posted by thompson on Tue, 15 Sep 1992 21:20:00 GMT
View Forum Message <> Reply to Message

In article <1992Sep14.204750.8612@sunspot.noao.edu>,
mcgraw@sunspot.sunspot.noao.edu (Robert McGraw) writes...

> I am looking for any information and/or procedures that make it
> eash to stay with the same coordinates system when going from
> set_plot,'x to/from set_plot,'ps' in IDL.
>
> For example, it seems that tv and xyouts uses a different coordinate
> system in PostScript then when in the X environment.
>
> Is there a way to make a procedure work the same when in X or PS
> without having to make a lot of changes?

It is true that TV works differently for X-windows displays and PostScript files. However, XYOUTS should normally work the same for both displays.

You can run into a problem if you try to mix and match plots and overplots. For instance, the following should work without problems:

```
SET_PLOT,'X'
PLOT,INDGEN(10)
OPLOT,INDGEN(10),PSYM=2
SET_PLOT,'PS'
PLOT,INDGEN(10)
OPLOT,INDGEN(10),PSYM=2
```

and the output should appear more or less the same on the screen and in the PostScript file. However, the following will not work properly:

```
SET_PLOT,'X'
PLOT,INDGEN(10)
SET_PLOT,'PS'
PLOT,INDGEN(10)
SET_PLOT,'X'
OPLOT,INDGEN(10),PSYM=2
SET_PLOT,'PS'
OPLOT,INDGEN(10),PSYM=2
```

(Although it may *ALMOST* work.) The reason for this is that the PLOT command defines certain system variables (e.g. !X.S, !Y.S) which describe the transformation between data coordinates and device coordinates. These system variables will be different for the two different devices, and IDL does not keep track of them as a function of device. Therefore, when the PLOT command is executed on the PostScript file, the coordinate information for the

X-windows display is lost. The same thing happens when switching between windows, or between plots generated with !P.MULTI.

However, the following routine called SETPLOT can be used to replace SET_PLOT to overcome this difficulty. I use this myself, and it fits my needs. It may work okay for you to. It simply stores some of the most used system variables in a common block as a function of plotting device and window.

The issue of TV working differently in PostScript files as opposed to X-windows files is because PostScript uses scalable pixels. The only way to control the size of the image in a PostScript file (as I understand it) is to use the XSIZE and YSIZE keywords. X-windows on the other hand always maps image pixels directly into screen pixels, and the XSIZE, YSIZE keywords are ignored.

The way I achieve device independence between PostScript files and other devices such as X-windows is to use a separate routine to display images rather than using TVSCL directly. I calculate how big I can scale the image to fit within the page, or section of the page, and I express this in device coordinates: NX, NY. I also calculate the position IX,IY to place the image to center it within the selected region, again in device coordinates. Then, if the device is a PostScript printer, I use the command

```
   TVSCL,ARRAY,IX,IY,XSIZE=NX,YSIZE=NY,/DEVICE
```

Otherwise I use the function CONGRID to rescale ARRAY to size XSIZE, i.e.

```
       TVSCL,CONGRID(ARRAY,NX,NY),IX,IY
```

Because of the IF statements involved, it's better to put this into a routine rather than call these statements directly.

Bill Thompson

P.S. The file SETPLOT.PRO, which contains several procedures, follows.

```
 ============================================================
==================
 PRO ADD_DEVICE, NAME, SV
;
;  Called from SETPLOT.  Used to add devices to the PLOTFILE common block.
;  Also used to set default values for the various devices.
;
 ON_ERROR,1
 COMMON PLOTFILE,NAMES,SAVE
;
;  Define some of the defaults for various devices.
;
 ;********************************************************* ******************
```

```
;     These represent my own personal preferences.  You can change these, or
;     omit this section entirely.  I make these settings take effect by
;     putting the command SETPLOT,!D.NAME in my IDL startup file.
;************************************************************ ********************
;
 IF NAME EQ 'QMS'   THEN BEGIN
  SV.THICK     = 3
  SV.CHARTHICK = 3
  SV.XTHICK    = 3
  SV.YTHICK    = 3
 ENDIF
 IF NAME EQ 'REGIS' THEN BEGIN
  SV.COLOR  = 3
  SV.FONT   = 0
 ENDIF
 IF NAME EQ 'TEK'   THEN SV.FONT   = 0
;
;  Check to see if the common block variables have been initialized.  Either
;  initialize the common block with this device, or add this device to the
;  common block.
;
 IF N_ELEMENTS(NAMES) EQ 0 THEN BEGIN
  NAMES = NAME
  SAVE = SV
 END ELSE BEGIN
  NAMES = [NAMES,NAME]
  SAVE = [SAVE,SV]
 ENDELSE
;
 RETURN
 END
;************************************************************ ********************
;  ****   ****   ****   ****   ****   ****   ****   ****   ****   ****
;************************************************************ ********************
 PRO STORE_INTO_SV, SV
;
;  Called from SETPLOT.  Used to initialize SV, and to store the various system
;  variables in it.
;
 SV = {SV_PLT, CHARSIZE: !P.CHARSIZE,  $
   FONT: !P.FONT,   $
   COLOR: !P.COLOR,  $
   BACKGROUND: !P.BACKGROUND, $
   REGION: !P.REGION,  $
   CLIP: !P.CLIP,   $
   POSITION: !P.POSITION,  $
   THICK: !P.THICK,  $
   CHARTHICK: !P.CHARTHICK, $
```

```
  XTYPE: !X.TYPE,   $
  XCRANGE: !X.CRANGE,  $
  XS: !X.S,   $
  XMARGIN: !X.MARGIN,  $
  XWINDOW: !X.WINDOW,  $
  XREGION: !X.REGION,  $
  XTHICK: !X.THICK,  $
  YTYPE: !Y.TYPE,   $
  YCRANGE: !Y.CRANGE,  $
  YS: !Y.S,   $
  YMARGIN: !Y.MARGIN,  $
  YWINDOW: !Y.WINDOW,  $
  YREGION: !Y.REGION,  $
  YTHICK: !Y.THICK,  $
  ZTYPE: !Z.TYPE,   $
  ZCRANGE: !Z.CRANGE,  $
  ZS: !Z.S,   $
  ZMARGIN: !Z.MARGIN,  $
  ZWINDOW: !Z.WINDOW,  $
  ZREGION: !Z.REGION}
;
 RETURN
 END
 ;************************************************************ ********************
;  ****   ****   ****   ****   ****   ****   ****   ****   ****   ****
 ;************************************************************ ********************
 PRO SETPLOT,DEVICE
;+
; NAME:
; SETPLOT
; PURPOSE:
; Switches among the various available plotting devices. The plotting
; variables for each device are saved in a common block so that the user
; retains the ability to reset to a previously used device and do over-
; plots, even if plots were produced on another device in the meantime.
;
; Calling SETPLOT with the name of the currently selected device resets
; the system variables to either default values, or those from the last
; time SETPLOT was called.
;
; CALLING SEQUENCE:
; SETPLOT, DEVICE
; INPUTS:
; DEVICE - Name of the plotting device one is changing to.
; OPTIONAL INPUT PARAMETERS:
; None.
; OUTPUTS:
; None.
```

```
; OPTIONAL OUTPUT PARAMETERS:
; None.
; COMMON BLOCKS:
; PLOTFILE - Saves system variables for later retrieval.  Not to be used
; by other routines.
; SIDE EFFECTS:
; Many system variables are manipulated by this routine--in particular
; !P.CHARSIZE and !P.FONT.
;
; The first time the routine is called, SET_PLOT is called in turn for
; each plotting device and the results saved in the common block.
;
; RESTRICTIONS:
; The procedure will not work correctly unless it is used exclusively to
; change the plotting device.
; PROCEDURE:
; Many system variables that would be changed by the command SET_PLOT
; are saved in a common block.  SET_PLOT is called, and then the saved
; values are restored.
; MODIFICATION HISTORY:
; W.T.T., Sept. 1987.
; William Thompson, February, 1990.
;-
;
 ON_ERROR,1
 COMMON PLOTFILE,NAMES,SAVE
;
;  Check the number of parameters.
;
 IF N_PARAMS(0) EQ 0 THEN BEGIN
  PRINT,'*** SETPLOT must be called with one parameter:'
  PRINT,'                 DEVICE'
  RETURN
 ENDIF
;
;  Define the structure.
;
 STORE_INTO_SV, SV
;
;  Check to see if the common block variables have been initialized.
;
 IF N_ELEMENTS(NAMES) EQ 0 THEN ADD_DEVICE,!D.NAME,SV
;
;  Get the number of the current plot device.
;
 I_DEVICE = WHERE(NAMES EQ !D.NAME, N_FOUND)
 IF N_FOUND EQ 0 THEN BEGIN
  ADD_DEVICE,!D.NAME,SV
```

```
  I_DEVICE = WHERE(NAMES EQ !D.NAME)
 ENDIF
 I_DEVICE = I_DEVICE(0)
;
;  If the device is being changed, then store the current system variables.
;
 IF STRUPCASE(DEVICE) NE !D.NAME THEN BEGIN
  SAVE(I_DEVICE) = SV
;
;  Change the plotting device.
;
  SET_PLOT,DEVICE
;
;  Get the number of the new plotting device.
;
  I_DEVICE = WHERE(NAMES EQ !D.NAME,N_FOUND)
  IF N_FOUND EQ 0 THEN BEGIN
   STORE_INTO_SV, SV
   ADD_DEVICE,!D.NAME,SV
   I_DEVICE = WHERE(NAMES EQ !D.NAME)
  ENDIF
  I_DEVICE = I_DEVICE(0)
 ENDIF
;
;  Restore the system variables from the saved arrays.  This is done even if
;  the plotting device is not changed.  By using SETPLOT on the device one is
;  already set to, one can reinitialize the system variables.
;
 SV = SAVE(I_DEVICE)
 !P.CHARSIZE   = SV.CHARSIZE
 !P.FONT       = SV.FONT
 !P.COLOR      = SV.COLOR
 !P.BACKGROUND = SV.BACKGROUND
 !P.REGION     = SV.REGION
 !P.CLIP       = SV.CLIP
 !P.POSITION   = SV.POSITION
 !P.THICK      = SV.THICK
 !P.CHARTHICK  = SV.CHARTHICK
 !X.TYPE       = SV.XTYPE
 !X.CRANGE     = SV.XCRANGE
 !X.S       = SV.XS
 !X.MARGIN     = SV.XMARGIN
 !X.WINDOW     = SV.XWINDOW
 !X.REGION     = SV.XREGION
 !X.THICK      = SV.XTHICK
 !Y.TYPE       = SV.YTYPE
 !Y.CRANGE     = SV.YCRANGE
 !Y.S       = SV.YS
```

```
 !Y.MARGIN     = SV.YMARGIN
 !Y.WINDOW     = SV.YWINDOW
 !Y.REGION     = SV.YREGION
 !Y.THICK      = SV.YTHICK
 !Z.TYPE       = SV.ZTYPE
 !Z.CRANGE     = SV.ZCRANGE
 !Z.S      = SV.ZS
 !Z.MARGIN     = SV.ZMARGIN
 !Z.WINDOW     = SV.ZWINDOW
 !Z.REGION     = SV.ZREGION
;
 RETURN
 END
```