Subject: Re: Line-Mouse widget tool
Posted by David Fanning on Tue, 29 Mar 2005 23:25:34 GMT
View Forum Message <> Reply to Message

Robert Barnett writes:

> I render a direct graphics plot onto a widget_draw.
> How do you transform a mouse click event which is specified in pixels to
> data-space when using direct graphics? I can compute it from the
> xmargin, ymargin, xtick_get, ytick_get, xsize and ysize. But, surely
> there must be an easier way?

Well, remember that every object I create and every-
thing I draw into a window has a coordinate system
associated with it. This is, effectively, the *data*
coordinate space. So, a medical image, for example,
might have a coordinate system in mm or cm, something
like that. And typically, these objects (images are
good examples, as are plots) know something about
where they are located in the display window.

So, these objects have a Pixel_To_Value method that
simply takes a window pixel X and Y value and can
return the proper "data" coordinate. (And also tell
me if I am inside or outside the object.)

So a call looks like this:

```
data = theObject -> Pixel_To_Value(event.x, event.y, Inside=in)
IF in THEN BEGIN
   Print, 'Data Coordinates are: ', data
ENDIF ELSE Print, 'Outside of object'
```

The Pixel_To_Value method looks something like this (stripped
to the essentials):

```
PRO myObject::Pixel_To_Value, x, y, INSIDE=inside
inside = 0
self -> ApplyCoordinates ; Set up data coordinate space.
c = Convert_Coord(x, y, /Device, /To_Data)
retval = retValue = [c[0,0], c[1,0]]
IF (retValue[0] GE !X.CRange[0]) AND $
   (retValue[0] LE !X.Crange[1]) AND $
   (retValue[1] GE !Y.CRange[0]) AND $
   (retValue[1] LE !Y.Crange[1]) THEN inside = 1
RETURN, retval
END
```

The "inside" test varies a little bit, depending upon the object. What I've shown you is for some kind of plot object. If this were an image, I would be checking to see if the point is inside the image's "location" in the window.

The notion of a coordinate system associated with every object is amazingly powerful. Today I built a "ruler" object for an image, which is just a small scale that you want to display on an image to give you some idea of the size of features you are looking at. The ruler has it's own coordinate system, so I can position it in a window, and it has a holder for the parent coordinate system. It uses the parent coordinate system to see how big it needs to be, and its own coordinate system to display itself.

So I just popped it into an image object and it grows or retracts in size as I zoom in and out of the image, all the while staying in the same place in the window. Amazing!! Magic!

Let's just say there are days when you've *got* to love IDL!

Cheers,

David

--
David Fanning, Ph.D.
Fanning Software Consulting, Inc.
Coyote's Guide to IDL Programming: http://www.dfanning.com/