Subject: Re: Simple animation?
Posted by JD Smith on Wed, 11 Sep 2002 16:26:58 GMT
View Forum Message <> Reply to Message

On Wed, 11 Sep 2002 07:47:49 -0700, Liam E. Gumley wrote:

```
>  David Fanning wrote:
>>  Liam E. Gumley (Liam.Gumley@ssec.wisc.edu) writes:
>>
>>>  Shawn wrote:
>>>>      I recently tried to animate the change in my data with time.  I
>>>>  found that if I plotted to the same window, it would over plot the
>>>>  old data, which was good, but it also redraws the axis every time
>>>>  as well, which causes a lot of "flashing".  Is it possible to tell
>>>>  IDL to only redraw the data and to leave the axis alone?
>>>
>>>  Try plotting the first dataset with PLOT, and subsequent datasets
>>>  with OPLOT.
>>
>>  Well, that wouldn't animate the plot, really. Just show more and more
>>  data piled on top of it.
>>
>>  Try this. Create a pixmap window the same size as your display window.
>>  Draw your plots in the pixmap window, then use the DEVICE COPY
>>  technique to copy the contents of the pixmap window to the display
>>  window. This will result in a flicker-free animation. :-)
>
>  I initially started writing a reply that included this exact advice. But
>  on rereading the original post, I thought a simpler solution might be
>  required.
>
>  However as David says, DEVICE COPY will give you true animation. Here's
>  an example of Brownian motion animation from chapter 5 of my book:
>
>
```

Liam:

I was interested to see that you don't use true double buffering, but
instead are just "erasing" the plot axes using the pixmap version of the
dataless plot.  For such a simple example, this gets the job done (in
fact, just using

```
erase & plot, [0], /nodata, xrange=[-1, 1], yrange=[-1, 1]
```

would also do it about as well, except the axes would flicker a bit more).
However, if you had a much more complicated plot, or a multiple command
plot sequence, the plot drawing commands would themselves cause flicker

and lag.  This is especially noticeable if driven by motion events.  In
that case, it pays to use double buffering, i.e. draw *everything* to a
pixmap, and, when finished, copy over to the visible window in one go.
This produces by far the smoothest animation and motion for multi-element
plots.  It would go something like:

```
wset,pixwin
plot,...
countour,..,/noerase
oplot,.....
tv,...,/noerase
wset,viswin
device,copy=[0, 0, xsize, ysize, 0, 0, pixwin]
```

Obviously, if you can avoid redrawing certain things by offloading them to
another pixmap and zapping that into the double-buffer pixmap, you can
increase frame rates, but I find this simple technique excels for even
very complicated plots.

JD