Subject: disk-based animation timings
Posted by caron on Wed, 20 Jul 1994 21:22:05 GMT
View Forum Message <> Reply to Message

Here are a few recent timings that I did that I thought others might
be interested in, or like to comment on. These numbers were generated with
IDL running on an RS6000 model 560, using an RS6000 model 320H as the X
server.

Basically I wanted to know how fast I can do animation in idl, if
there are too many animation frames to fit into memory. Note that
xinteranimate requires images to be first loaded into (virtual) memory on
the X server (I assume using pixmaps).

I generated 48 1000x500 pixel images, and saved them to disk, using tvrd().
Then I read them back and used tv to display them. I could get about
1 frame/sec rate.

I thought it should be faster, so I wrote an Xlib program to see how idl
compared. Using the exact same images, and the XPutImage() routine, I
could display 48 frames in about 30 secs. Trying to seperate the disk I/O
from the X cost, I tested just reading from disk (~16 sec) and just calling
XPutImage() 48 times (~21 secs). Note the sum doesnt add to doing both the
read and display at once.

Anyway, I concluded the idl overhead is not excessive.

Since my real problem is to visualize gridded data, I decided to compare
this to a simpler rendering technique, namely false color rectangles. I have
a 96 x 48 grid. Instead of countouring in idl and saving the 250 KByte image,
I just take the 4608 data values and assign a color index based on a data
interval (I used 10 intervals, same as the contour values). Now I save the
48 x 4096 bytes (=221Kbytes) in a file. I read it into my Xlib program and
use XFillRectangle(), and I can do about 5 frames per second.

Then I note that by comparing the data values of succesive grids, I only need
to redraw the ones that change. On my particular set (probably not realistic)
this increased the rate to 50 frames/sec.

In order to keep a high rate, I used a bit plane for the world map overlay.

Finally, I suspect that a simple run-length encoding of the data would reduce
the file size by another factor of 5 to 20.

In conclusion, taking advantage of the underlying simplicity of the data, data
compression, and using a more primitive rendering technique, may allow for
animation-speed visualization. I dropped into Xlib for proof of concept. I
dont know if I will try to implement the same thing in idl; i havent seen

any documentation on bit plane overlays, essential to keeping the frame rate high.