Subject: Re: Drawing satellite pixels on maps?
Posted by Robert.M.Candey on Tue, 19 Nov 1996 08:00:00 GMT
View Forum Message <> Reply to Message

In article <3291C91F.7F816972@oma.be>, Philippe Peeters
<Philippe.Peeters@oma.be> wrote:

> I've already posted this question some time ago but did not get any
> answer. Here I go again:
>
> I want to draw satellite data on a map. Each ground pixel is defined by
> the latitude and longitude of the 4 corners. It is not a regular
> rectangle or square and depend on the viewing geometry of the satellite
> instrument.
> I have tried a simple polyfill,long,lat but I have serious problems with
> orthographic maps when the pixel is partly off the map. I got strange
> filled polygons from the edge of the map to the corner of the window.
> Someone on the net advice me to use a new polyfill routine which checks
> polygons boundaries before drawing it. Though slower than the original
> polyfill, it solved the problem.
> But I still have another problem with several maps when the pixel to be
> drawn is at the edge, i.e. when one or several pixel corner is on one
> side of the map (lon > -180) and the other on the other side (lon< -180)
> of the map.
> example longitude=[-179,-181,-179.5,-180.5] or [179,181,179.5,180.5]
>
> The pixel is drawn from one side to the other of the map which is pretty
> ugly. Obviously this is a 'normal' way of drawing that kind of pixel,
> polyfill is not supposed to know that it has to cut the pixel into two.
> Does anybody know how to solve this problem.
>
> And now a question related to the same topic. How can I resample the
> irregular ground pixels onto a regular (square or rectangle) grid?
>
> Philippe Peeters

I have struggled with similar problems and recently posted my attempts at a solution (auroral_image.pro; email me for a copy). To get around the problem of polyfill across the whole map, I checked to see if the endpoints of the polygon to fill are nearby in normalized coordinates and then I skipped plotting it if it was not nearby; you could split the polygon in two and plot each part, I suupose. The code is like this (for a list of triangles, with Lat1 and Lon1 defined at each corner):

```
pAll = convert_coord(Lon1, Lat1, /data, /to_normal)
pLon = pAll(0,*) & pLat = pAll(1,*)
for i=0L,n_elements(triangles(0,*))-1 do begin
  tri1 = triangles(*,i)
```

```
      Lon3 = Lon1(tri1) & Lat3 = Lat1(tri1)
;### which average scheme is best?
;      Zb1   = avg(Zb(tri1)) ; average byte values
;      Zb1   = Zb(tri1(0)) ; faster than avg and as accurate?
      Zb1 =  doByteScale([avg(Z2(tri1))],minZ1,maxZ1,Zsize,wBad,flipColor Bar,1)
; doByteScale is my routine for scaling the data values between minZ1 and maxZ1
;   with some other constraints
      if (total(abs(pLon(tri1)-shift(pLon(tri1),1))) lt 0.1) and $
      (total(abs(pLat(tri1)-shift(pLat(tri1),1))) lt 0.1) then $
        polyfill, Lon3, Lat3, color=Zb1(0), noclip=0
; you could do a "where" here instaed of "if" to find which corners are out of
; bounds
    endfor
```

As for resampling, you could use triangulate and trigrid with the sphere option (see the Map_image method in my auroral_image.pro for an example); but you are probably better off (more scientifically accurate) plotting the original polygons and not resampling

--
Robert.M.Candey@gsfc.nasa.gov
NASA Goddard Space Flight Center, Code 632
Greenbelt, MD 20771 USA 1-301-286-6707