Subject: Re: Map Projection Woes
Posted by savoie on Thu, 13 Apr 2006 19:44:50 GMT
View Forum Message <> Reply to Message

David Fanning <davidf@dfanning.com> writes:

> Folks,
>
> I took on this day/night terminator project to learn a
> little more about map projections. I thought I would learn
> something by doing it. I know better, and should be careful
> what I wish for.
>
> Here is my problem (I think), and you map projection
> experts probably know the answer. My program calculates
> a field of view, essentially, from the point of view
> of the sun looking towards the Earth. You could think
> of it as a circular polygon.
>
> If the subsolar point on the Earth is inside this
> polygon, that is the light part of the map. If it is
> outside, that is the dark part of the map, with
> respect to the day/night terminator.
>
> But...I don't always get a reliable answer to my question:
> "Is the subsolar point inside the polygon that describes
> the terminator?" The problem comes (surprise, surprise)
> when the polygon crosses the international date line
> and there is a jump from 180 degrees of longitude to -180
> degrees of longitude.
>
> My question is how is this handled, normally? (If you
> are an Aussie or a Kiwi this problem probably comes up
> daily.)

Don't know if this helps. But I know a website...

Directly from geospatialmethods.org:

http://geospatialmethods.org/spheres/MiscAlgorithms.html#PSP

Point in Spherical Polygon:

Determining whether a given point is inside or outside a given spherical polygon is rather difficult. Several effective algorithms exist for making the same determination on a plane, but those don't transfer well to the surface of the sphere. In the spheres package we have adapted the "point-at-a-distance" algorithm to work on the sphere for "reasonable"

polygons, but it cannot be made absolutely foolproof.

The plane is infinitely large so finding a point outside a given polygon is relatively easy on the plane. One can simply pick (maximumX + 1, maximumY + 1) or, to be really safe, one can pick (maximumX * 10, maximumY * 10) where maximumX and maximumY are the maximum values of the given corner points. Indeed the algorithm is frequently refered to as the point-at-an-infinite-distance algorithm because developers just use the point (infinity, infinity), or substitute some absurdly large number for "infinity", no matter what polygon they are working with.

But the sphere is a closed surface so when you go far enough away from a given point you end up back where you started. Consequently there is no easy way to pick a point known to be external to the polygon. The spheres package employs a number of heuristics to guess an external point as outlined below, and in cases where that guess it likely to be wrong the SphericalPolygon class has a setExternalPoint() method to allow users to designate the known external point.

Given a point known to be external to the polygon determining if a given point is inside a given polygon is a simple matter.

1. Connect the point to the known external point with a great circle arc.

2. For each great circle arc that is a side of the spherical polygon test if it intersects the arc constructed in step #1 and count the number of intersections.

3. If the total number of intersections is odd the given point is inside the spherical polygon. if the total number of intersections is even the point is outside the spherical polygon.

Behavior is undefined if the given point is a corner point of, or on an edge of, the spherical polygon. In cases where the great circle arc constructed in step #1 just touches an edge of the polygon the algorithm may wrongly designate an external point as internal. But those cases should be excessively rare.

--
Matthew Savoie - Scientific Programmer
National Snow and Ice Data Center
(303) 735-0785 http://nsidc.org