Subject: Re: point inside polygon
Posted by manizade on Wed, 01 Apr 1998 08:00:00 GMT
View Forum Message <> Reply to Message

>> In article C0684CDB@oma.be, Philippe Peeters <philp@oma.be> writes:
>>> Does anybody knows of an IDL function to test whether a given point is
>>> inside a polygon?
The proposed polyfill approaches are limited by the resolution
of the bitmap used.

In general, you can decompose any polygon into a set of
triangles (using TRIANGULATE), then determine whether
the point is included in any of the triangular regions.
TRIANGULATE gives each vertex list in counterclockwise order,
so a point is inside the triangle if it is to the left of each
directed edge of the triangle.

Suppose you have a directed line from point L1 to point L2,
where the the x coordinate of L1 is L1[0] and the y coordinate
is L1[1]; L2[0], L2[1] are the (x,y) coords of L2; and px,py are
the coordinates of the point in question.
Then the boolean answer to whether the point is to the left of
the line is given by
(L1(1)-L2(1))*(L1(0)-px) LE (L1(0)-L2(0))*(L1(1)-py)

I have not seen an approach to this problem other that what I
invented (and I'm sure I am not the only one to (re)invent it).
Any other solutions out there?
--
Serdar S. Manizade <serdar.manizade@gsfc.nasa.gov>
Airborne Topographic Mapper Project
NASA/GSFC/Wallops Flight Facility, Wallops Island, VA

Subject: Re: point inside polygon
Posted by wmc on Wed, 01 Apr 1998 08:00:00 GMT
View Forum Message <> Reply to Message

> This function determines if a point is inside a polygon or not. If you
> have several points I believe you are better off with the polyfillv approach.
> Baard

> FUNCTION inside, x, y, px, py
(some bits cut)
> sx = size(px)
> sy = size(py)
> N=sx(1)

```
>     tmp_px =  [px, px[0]]                    ; Close Polygon in x
>     tmp_py =  [py, py[0]]                    ; Close Polygon in y
>     i  = indgen(N)                          ; indices 0...N-1
>     ip = indgen(N) + 1                       ; indices 1...N
>     X1 = tmp_px(i)  - x &   Y1 = tmp_py(i)  - y
>     X2 = tmp_px(ip) - x &   Y2 = tmp_py(ip) - y
>     dp = X1*X2 + Y1*Y2                        ; Dot-product
>     cp = X1*Y2 - Y1*X2                       ; Cross-product
>     theta = atan(cp,dp)
>     IF (abs(total(theta)) GT 1.0E-8) THEN return,1 ELSE return,0
> END
```

Interesting... there had to be a better way and this looks like it.
I'm now trying to work out why it works... I think you're counting up the angles going round the polygon to the point, and the sum is zero outside and 2*!pi inside.

Only one criticism: 1e-8 is too tight a test for single precision:
inside(.5,1.5,[0,1,1,0],[0,0,1,1]) returns 1
since total(theta) is -1.19e-7.
But inside(.5,1.5d0,[0,1,1,0],[0,0,1,1]) returns 0 as it should.
So I think the test should be 1e-5 or somesuch (though presumably .1 would work just as well?).

- William

William M Connolley | wmc@bas.ac.uk | http://www.nbs.ac.uk/public/icd/wmc/
Climate Modeller, British Antarctic Survey | Disclaimer: I speak for myself

---

Subject: Re: point inside polygon
Posted by Philippe Peeters on Wed, 01 Apr 1998 08:00:00 GMT
View Forum Message <> Reply to Message

Alex Schuster wrote:
>
> William Connolley wrote:
>
>> In article C0684CDB@oma.be, Philippe Peeters <philp@oma.be> writes:
>>> Does anybody knows of an IDL function to test whether a given point is
>>> inside a polygon?
>>
>> I needed to solve this recently (in a mapping context). The solution I came up
>> with works but its not elegant: use poly_fill to actually draw your polygon
>> (in a pixmap not the screen window if you prefer), then read off the pixel value
>> of your point to see if its in or out.
>>

>> This is grotesquely inelegant, but its very simple and it works. I can
>> post the code if you're interested. A better solution
>> would be to look at polyfill and see how it does the fill... but sadly
>> polyfill seems to be one of the few routines not written in IDL.
>
> POLYFILLV works similar, but does not need a pixmap. It just returns the
> subscripts of all points inside the polygon.
> This worked okay for me, but for floating point coordinates it might not
> be too accurate.

This is precisely my problem. POLYFILLV is ok to check regular grid
points within a given polygon. In my case I have a polygon with real
coordinates (a satellite pixel) and I want to check if a ground station
(lat,lon) is within the pixel.

I have tried to adapt a C code from Graphic Gems
http://www.acm.org/tog/GraphicsGems/
which is the CrossingMultiply from Haines
in C:

```
/* ======= Crossings Multiply algorithm
================================== */

/*
 * This version is usually somewhat faster than the original published
in
 * Graphics Gems IV; by turning the division for testing the X axis
crossing
 * into a tricky multiplication test this part of the test became
faster,
 * which had the additional effect of making the test for "both to left
or
 * both to right" a bit slower for triangles than simply computing the
 * intersection each time.  The main increase is in triangle testing
speed,
 * which was about 15% faster; all other polygon complexities were
pretty much
 * the same as before.  On machines where division is very expensive
(not the
 * case on the HP 9000 series on which I tested) this test should be
much
 * faster overall than the old code.  Your mileage may (in fact, will)
vary,
 * depending on the machine and the test data, but in general I believe
this
 * code is both shorter and faster.  This test was inspired by
unpublished
 * Graphics Gems submitted by Joseph Samosky and Mark Haigh-Hutchinson.
```

```
 * Related work by Samosky is in:
 *
 * Samosky, Joseph, "SectionView: A system for interactively specifying
and
 * visualizing sections through three-dimensional medical image data",
 * M.S. Thesis, Department of Electrical Engineering and Computer
Science,
 * Massachusetts Institute of Technology, 1993.
 *
 */

/* Shoot a test ray along +X axis.  The strategy is to compare vertex Y
values
 * to the testing point's Y and quickly discard edges which are entirely
to one
 * side of the test ray.  Note that CONVEX and WINDING code can be added
as
 * for the CrossingsTest() code; it is left out here for clarity.
 *
 * Input 2D polygon _pgon_ with _numverts_ number of vertices and test
point
 * _point_, returns 1 if inside, 0 if outside.
 */
int CrossingsMultiplyTest( pgon, numverts, point )
double pgon[][2] ;
int numverts ;
double point[2] ;
{
register int j, yflag0, yflag1, inside_flag ;
register double ty, tx, *vtx0, *vtx1 ;

    tx = point[X] ;
    ty = point[Y] ;

    vtx0 = pgon[numverts-1] ;
    /* get test bit for above/below X axis */
    yflag0 = ( vtx0[Y] >= ty ) ;
    vtx1 = pgon[0] ;

    inside_flag = 0 ;
    for ( j = numverts+1 ; --j ; ) {

 yflag1 = ( vtx1[Y] >= ty ) ;
 /* Check if endpoints straddle (are on opposite sides) of X axis
  * (i.e. the Y's differ); if so, +X ray could intersect this edge.
  * The old test also checked whether the endpoints are both to the
  * right or to the left of the test point.  However, given the faster
  * intersection point computation used below, this test was found to
```

```
 * be a break-even proposition for most polygons and a loser for
 * triangles (where 50% or more of the edges which survive this test
 * will cross quadrants and so have to have the X intersection computed
 * anyway).  I credit Joseph Samosky with inspiring me to try dropping
 * the "both left or both right" part of my code.
 */
if ( yflag0 != yflag1 ) {
    /* Check intersection of pgon segment with +X ray.
     * Note if >= point's X; if so, the ray hits it.
     * The division operation is avoided for the ">=" test by checking
     * the sign of the first vertex wrto the test point; idea inspired
     * by Joseph Samosky's and Mark Haigh-Hutchinson's different
     * polygon inclusion tests.
     */
    if ( ((vtx1[Y]-ty) * (vtx0[X]-vtx1[X]) >=
     (vtx1[X]-tx) * (vtx0[Y]-vtx1[Y])) == yflag1 ) {
 inside_flag = !inside_flag ;
    }
}

/* Move to the next pair of vertices, retaining info as possible. */
yflag0 = yflag1 ;
vtx0 = vtx1 ;
vtx1 += 2 ;
  }

  return( inside_flag ) ;
}
```

I code it in IDL as

```
function ptpoly,pgonx,pgony,x,y

numverts=n_elements(pgonx)
if numverts ne n_elements(pgony) then message,'X & Y must have same
size'
if numverts lt 3 then message,'At least 3 vertex'

   tx = x
   ty = y

   vtx0x = pgonx[numverts-1]
   vtx0y = pgony[numverts-1]
   ; get test bit for above/below X axis
   yflag0 = ( vtx0y ge ty ) ;
   vtx1x = pgonx[0]
   vtx1y = pgony[0]
```

```
   inside_flag = 0
   for j = 1,numverts-1 do begin

 yflag1 = ( vtx1y ge ty ) ;

 if  yflag0 ne yflag1 then begin

     if ( ((vtx1y-ty) * (vtx0x-vtx1x) ge $
      (vtx1x-tx) * (vtx0y-vtx1y)) eq yflag1 ) then $
  inside_flag = not(inside_flag)

 endif

 yflag0 = yflag1
 vtx0x = vtx1x
 vtx0y = vtx1y
 vtx1x = pgonx[j]
 vtx1y = pgony[j]
   endfor

   return,inside_flag
end
```

I tried it with a square px=[0,1,1,0] and py=[0,0,1,1] and random points with 0<x<2 and 0<y<2. It works quite well with that polygon but fail if I rotate the polygon vertices using shift(px,1) and shift(px,2). The polygon is the same, only the vertices ordering has changed.

But...

Now I have tried Crossing algorithm. this one seems to work. The only thing is that it can gives a "divide by zero" when two vertices have the same Y coordinates. In my case of satellite pixels, align vertices are very improbable.

--
Philippe Peeters
 ---------- ----------
Belgian Institute for Space Aeronomy Tel: +32-2-373.03.81
Institut d'Aeronomie Spatiale de Belgique Fax: +32-2-374.84.23
3 Avenue Circulaire Email: philp@oma.be
B-1180 Brussels, Belgium http://www.oma.be/BIRA-IASB/

Subject: Re: point inside polygon

Alex Schuster wrote:
>
> William Connolley wrote:
>
>> In article C0684CDB@oma.be, Philippe Peeters <philp@oma.be> writes:
>>> Does anybody knows of an IDL function to test whether a given point is
>>> inside a polygon?
>>
>> I needed to solve this recently (in a mapping context). The solution I came up
>> with works but its not elegant: use poly_fill to actually draw your polygon
>> (in a pixmap not the screen window if you prefer), then read off the pixel value
>> of your point to see if its in or out.
>>
>> This is grotesquely inelegant, but its very simple and it works. I can
>> post the code if you're interested. A better solution
>> would be to look at polyfill and see how it does the fill... but sadly
>> polyfill seems to be one of the few routines not written in IDL.
>

This function determines if a point is inside a polygon or not. If you
have
several points I believe you are better off with the polyfillv approach.
Baard

```
FUNCTION inside, x, y, px, py

   sx = size(px)
   sy = size(py)
   IF (sx[0] EQ 1) THEN NX=sx[1] ELSE return,-1    ; error if px not a
vector
   IF (sy[0] EQ 1) THEN NY=sy[1] ELSE return,-1    ; error if py not a
vector
   IF (NX EQ NY) THEN N = NX ELSE return,-1       ; Incompatible
dimensions

   tmp_px =  [px, px[0]]                    ; Close Polygon in x
   tmp_py =  [py, py[0]]                    ; Close Polygon in y

   i  = indgen(N)                          ; indices 0...N-1
   ip = indgen(N) + 1                       ; indices 1...N

   X1 = tmp_px(i)  - x &   Y1 = tmp_py(i)  - y
   X2 = tmp_px(ip) - x &   Y2 = tmp_py(ip) - y

   dp = X1*X2 + Y1*Y2                       ; Dot-product
```

```
   cp = X1*Y2 - Y1*X2                    ; Cross-product
   theta = atan(cp,dp)

   IF (abs(total(theta)) GT 1.0E-8) THEN return,1 ELSE return,0
END
```

Subject: Re: point inside polygon
Posted by Alex Schuster on Wed, 01 Apr 1998 08:00:00 GMT
View Forum Message <> Reply to Message

William Connolley wrote:

> In article C0684CDB@oma.be, Philippe Peeters <philp@oma.be> writes:
>> Does anybody knows of an IDL function to test whether a given point is
>> inside a polygon?
>
> I needed to solve this recently (in a mapping context). The solution I came up
> with works but its not elegant: use poly_fill to actually draw your polygon
> (in a pixmap not the screen window if you prefer), then read off the pixel value
> of your point to see if its in or out.
>
> This is grotesquely inelegant, but its very simple and it works. I can
> post the code if you're interested. A better solution
> would be to look at polyfill and see how it does the fill... but sadly
> polyfill seems to be one of the few routines not written in IDL.

POLYFILLV works similar, but does not need a pixmap. It just returns the
subscripts of all points inside the polygon.
This worked okay for me, but for floating point coordinates it might not
be too accurate.

Alex
--
Alex Schuster Wonko@weird.cologne.de PGP Key available
alex@pet.mpin-koeln.mpg.de

Subject: Re: point inside polygon
Posted by wmc on Wed, 01 Apr 1998 08:00:00 GMT
View Forum Message <> Reply to Message

In article C0684CDB@oma.be, Philippe Peeters <philp@oma.be> writes:
> Does anybody knows of an IDL function to test whether a given point is
> inside a polygon?

I needed to solve this recently (in a mapping context). The solution I came up

with works but its not elegant: use poly_fill to actually draw your polygon
(in a pixmap not the screen window if you prefer), then read off the pixel value
of your point to see if its in or out.

This is grotesquely inelegant, but its very simple and it works. I can
post the code if you're interested. A better solution
would be to look at polyfill and see how it does the fill... but sadly
polyfill seems to be one of the few routines not written in IDL.

- William

William M Connolley | wmc@bas.ac.uk | http://www.nbs.ac.uk/public/icd/wmc/
Climate Modeller, British Antarctic Survey | Disclaimer: I speak for myself

---

Subject: Re: point inside polygon
Posted by LC's No-Spam Newsread on Thu, 09 Apr 1998 07:00:00 GMT
View Forum Message <> Reply to Message

In article C0684CDB@oma.be, Philippe Peeters <philp@oma.be> writes:
>> Does anybody knows of an IDL function to test whether a given point is
>> inside a polygon?

NO, but insideness testing is a native postscript operator, and is described
in the Adobe postscript manual (aka the red book)
