UNIT 33 - SIMPLE ALGORITHMS II - POLYGONS

UNIT 33 - SIMPLE ALGORITHMS II - POLYGONS

UNIT 33 - SIMPLE ALGORITHMS II - POLYGONS

A. INTRODUCTION

- the previous unit looked at a simple algorithm for determining the intersection of two lines
- this unit considers at a second simple algorithm and at some extensions

B. POLYGON AREA

Objective

- find the area of a polygon which is defined by a sequence of vertices

Method

- common method is to use an algorithm that finds the area of a number of trapezoids bounded by:
  - a line segment
  - vertical lines dropped to the x axis
  - the x axis

  overhead - Calculating area of trapezoid

  diagram

  1. Drop vertical lines to the x axis from two adjacent vertices

     - find the area of the trapezoid so defined:

       A = (x2-x1) * (y2+y1) / 2

  diagram

     - i.e. area is difference in x's times average of y's

  2. Continue around the polygon, finding the area of each trapezoid defined by each line segment

  3. Sum the areas of all trapezoids defined to get the area of the polygon:

     - (note: must have (x(n+1),y(n+1)) = (x(1),y(1)))

       A = S (x(i+1)-x(i)) * (y(i+1)+y(i)) / 2)

     - when x(i+1) &LT x(i) the contribution to area is negative
       - this allows for trapezoids formed by the lower portion of the polygon to be subtracted so that, in the end, the total A is the area of the polygon it self

Problems

- this algorithm works fine when the polygon has holes and overhangs
- does not work with polygons whose boundaries self-cross (i.e. figure 8)
- if the polygon is digitized counterclockwise its area is negative
- problems occur when the polygon has negative y values:

    - cannot "drop" a perpendicular to the x axis
    - solutions:
        - add a large value to all y's
        - drop vertical lines to points below the lowest y value

- if the coordinate system is very precise, i.e. UTM with large numbers, will lose accuracy as a result of the comparative lack of precision of the computer
    - therefore, does not work on small polygons in large coordinate systems
    - solution:
        - move the origin temporarily up to a location near the vertices
        - this reduces the size of the coordinate values

Calculating areas of many polygons at once

- if polygons are defined by arcs with L and R polygons identified
    - can do calculations of trapezoid area for each arc
        - keep a running total of the area of each polygon
        - L polygon gets negative area
        - R polygon gets positive area

- after processing each arc, we have the areas of each polygon
    - outside world will have area equal to the negative of the total area of all polygons

C. POINT IN POLYGON ALGORITHM

Objective

- determine whether a given point lies inside or outside a given polygon

Generalization

- find out which polygon of a set contains each of a set of points

- examples
    - determine which county contains each of a set of customers identified by their coordinates, using a county boundary file
    - identify the attributes of the land use polygon containing a given house

- notation
    - the point is at (u,v)
    - the polygon has n points, (x(i),y(i)), i=1,...,n, and is closed by making (x(n+1),y(n+1)) = (x(1),y(1))

Strategy

1. draw a vertical line upwards from the point to infinity

2. count the number of times this line intersects the boundary of the polygon

- if the count is odd the point is inside
- if it is even the point is outside

Sample code

overhead - Simple point in polygon program

handout - Simple point in polygon program

ni = +1 for i=1 to n if x(i+1) &LT> x(i) then (A) if (x(i+1)-u) * (u-x(i)) >= 0 then (B) if x(i+1) &LT> u or x(i) &LT= u then (C) if x(i) &LT> u or x(i+1) >= u then (D) b = (y(i+1)-y(i)) / (x(i+1)-x(i)) a = y(i)-b * x(i) yi = a+b*u if yi > v then ni = ni*(-1) end if end if end if end if next i

- b is the slope of the line from (x(i),y(i)) to (x(i+1),y(i+1))
  - by substituting x=u in the equation y=a+b*x we get the y coordinate of the intersection between this line and the vertical line through x=u
  - then if y(i) > v the intersection must be above the point, and so is counted
- the value of ni alternates between +1 and -1
  - it is flipped every time an intersection is found
  - at the end, ni=-1 if the point is in, +1 if the point is out

Special cases

- lines (A) through (D) deal with special cases, as follows:

  A. if the line from (x(i),y(i)) to (x(i+1),y(i+1)) is vertical there can be no intersection

  diagram

  B. there can only be an intersection if the ith point is on one side of the vertical line through u and the i+1th point is on the other side

  diagram

  C.D. if either (x(i),y(i)) or (x(i+1),y(i+1)) lies exactly on the line, i.e. the point at (u,v) lies directly below a vertex, we may miscount:

  diagram

  - solution to this is:
  - if (x(i),y(i)) lies exactly on the line
    - count an intersection only if (x(i+1),y(i+1)) lies to the right, or
  - if (x(i+1),y(i+1)) lies exactly on the line
    - count an intersection only if (x(i),y(i)) lies to the left

  Polygon with isolated islands

diagram

- works fine

Polygon has hole with an island

diagram

- works fine

Concave polygons

diagram

- works fine

What if the point is on the boundary?

- some point in polygon routines deal with this explicitly, but this does not
  - sometimes it finds the point inside, sometimes outside

Fuzzy boundaries

- Blakemore (see Blakemore, 1984) described a "fuzzy" generalization of the point in polygon problem:
  - location of the boundary is uncertain
  - a band of width epsilon is drawn around the boundary, and represents the band within which the true boundary is assumed to lie
  - points can now be classified as "definitely out", "probably out", "probably in" and "definitely in"

Many points in many polygons

- polygons will likely be stored by arc
- check all arcs against vertical lines drawn upward from each point overhead - Many points in many polygons
  - if an arc between polygons A and B intersects, record an intersection with the boundary of both A and B, irrespective of which side of the arc A and B lie on
  - this will require a counter for each point/polygon combination
    - not very efficient
- a better algorithm is as follows:
- when an arc is found to intersect a vertical line, record the y value of the intersection and the polygon lying below the intersection (right polygon if the arc runs from W to E, left polygon otherwise)
  - for each point, keep account of: (a) the lowest such intersection found, and

(b) the polygon below that intersection

- after all arcs have been examined, the polygon below the lowest intersection is the containing polygon
- now have just two things to store for each point

- will still need to check every arc against every point, unless some sorting is done first

D. CENTROID LOCATION

overhead - Examples of centroid locations

- the centroid is potentially useful as a representative, central point in the polygon

- the centroid can be defined as the point about which the polygon would balance if it were cut out in plywood of uniform thickness and suspended
  - unfortunately the centroid is not always inside the polygon, which reduces its usefulness as a central point

- some programs calculate the centroid by averaging the x and y coordinates of the polygon vertices
  - this does not give the centroid

diagram

Method

- a better algorithm uses trapezoids dropped to the axes, as in the area algorithm
  - since each trapezoid consists of a triangle and a rectangle, the centroid of the polygon can be found from a weighted average of triangle and rectangle centroids

overhead - Centroid location code

handout - (cont) Centroid location code

$X = S ((y(i) - y(i+1)) (x(i)2 + x(i)x(i+1) + x(i+1)2)/6A)$ $Y = S ((x(i+1) - x(i)) (y(i)2 + y(i)y(i+1) + y(i+1)2)/6A)$ where A is the area of the polygon

- note: as with the area algorithm, the polygon must be coded clockwise and all y coordinates must be non- negative

E. SKELETON

- is the network of lines inside a polygon constructed so that each point on the network is equidistant from the nearest two edges in the polygon boundary
  - nodes on the network are equidistant from the three nearest edges

- a skeleton is what remains when a polygon contracts

each of its straight edges moves inward at a constant rate
- can be thought of as the opposite of the buffer operation

overhead - Polygon skeleton
- at the convex corners of the polygon, the bisectors of their adjacent edges form lines that are traced inward
- at each concave corner, the reduced polygon consists of the arc of a circle centered on the corner
- as the process continues the bisectors and arcs eventually merge, forming a tree-like structure

- as the polygon gets smaller, it may form into two or more islands
- ultimately, the polygon is reduced to a point
  - this point is:
    - furthest from the original boundary
    - the center of the largest circle one could draw inside the original polygon

Applications

- finding good locations for labels for polygons
  - label might be drawn:
    - along the skeleton axis of a polygon
    - at the point remaining after the polygon has been shrunk to a point
- breaking a city block into polygons, one for each face of the block, for labeling

REFERENCES

Blakemore, M., 1984, "Generalization and error in spatial databases," Cartographica 21:131-9.

Shamos, M.I., and F.P. Preparata, 1985. Computational Geometry, Springer-Verlag, Berlin. The standard but technically complex work on geometric algorithms.

DISCUSSION AND EXAM QUESTIONS

1. Discuss the results of using the polygon area and point in polygon algorithms when the polygon is not correctly structured (e.g. unclosed, figure-of-eight).

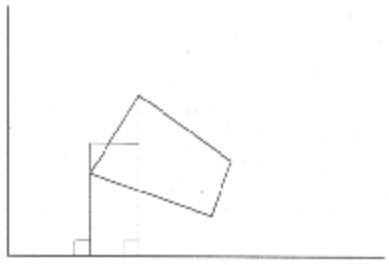
2. Modify the point in polygon algorithm to determine correctly if the point lies on the boundary of the polygon, in addition to inside or outside.

3. Derive the equations for polygon area and centroid from first principles.

4. Modify the polygon area algorithm to test for and accommodate negative y coordinates.

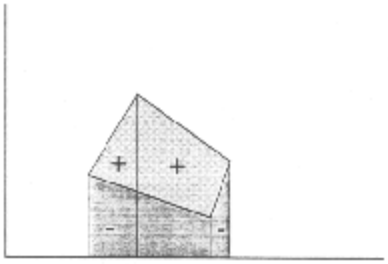
Last Updated: August 30, 1997.

UNIT 33 IMAGES



Trapezoid Construction

Component Trapezoids



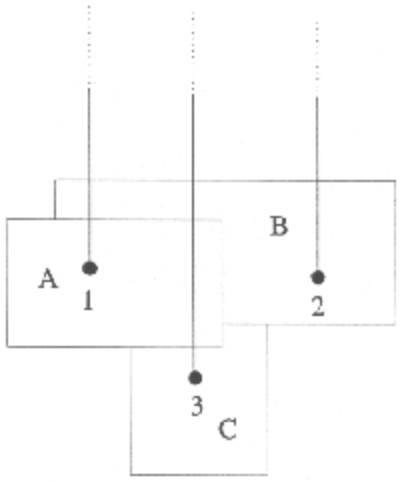
```
ni = +1
for i=1 to n
if x(i+1) <> x(i) then                                    (A)
    if (x(i+1)-u) * (u-x(i)) >= 0 then                    (B)
        if x(i+1) <> u or x(i) <= u then                  (C)
            if x(i) <> u or x(i+1) >= u then              (D)
                b = (y(i+1)-y(i)) / (x(i+1)-x(i))
                a = y(i)-b * x(i)
                yi = a+b*u
                if yi > v then
                    ni = ni*(-1)
                end if
            end if
        end if
    end if
end if
next i
```

```
ni = +1
for i=1 to n
if x(i+1) <> x(i) then  (A)
    if (x(i+1)-u) * (u-x(i)) >= 0 then                    (B)
        if x(i+1) <> u or x(i) <= u then                  (C)
            if x(i) <> u or x(i+1) >= u then              (D)
                b = (y(i+1)-y(i)) / (x(i+1)-x(i))
                a = y(i)-b * x(i)
                yi = a+b*u
                if yi > v then
                    ni = ni*(-1)
                end if
            end if
        end if
    end if
end if
next i
```

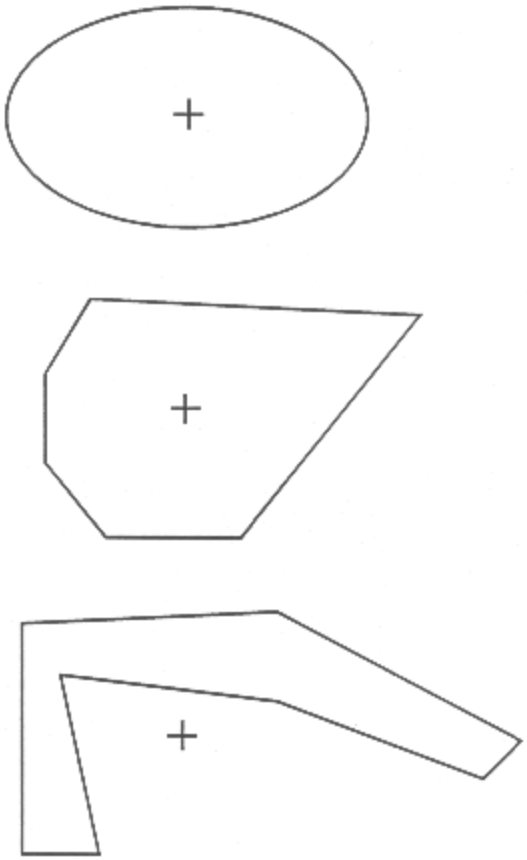
Centroid location code

$$X = \sum ((y(i) - y(i+1))(x(i)^2 + x(i)x(i+1) + x(i+1)^2)/6A)$$
$$Y = \sum ((x(i+1) - x(i))(y(i)^2 + y(i)y(i+1) + y(i+1)^2)/6A)$$

where A is the area of the polygon



Point	Polygon	# Intersections
1	A	1
1	B	2
1	C	0
2	A	0
2	B	1
2	C	0
3	A	2
3	B	2
3	C	1



$$X = \sum ((y(i) - y(i+1))\ (x(i)^2 + x(i)x(i+1) + x(i+1)^2)/6A)$$

$$Y = \sum ((x(i+1) - x(i))\ (y(i)^2 + y(i)y(i+1) + y(i+1)^2)/6A)$$

where A is the area of the polygon