Computational Geometry

Dr. Marco Pellegrini
Lecture 5: point-in-polygon

1

Summary of lecture 5

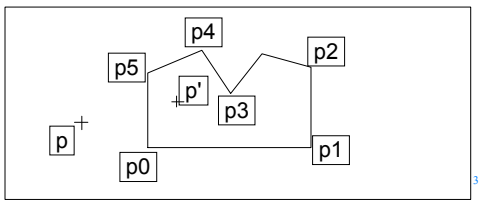
- Summary of lecture 4:
  Making shapes,
  Convex Hull
  Jarvis's March, Graham's scan.
- Summary of lecture 5.
  Searching in simple polygons:
  point-in-polygon queries.
  Searching in convex polygons:
  point-in-polygon queries.

2

Point-in-polygon queries

(for simple polygons)

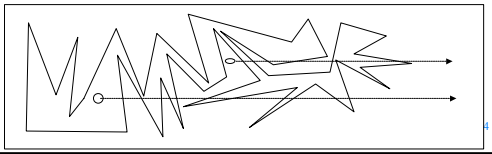
- We are given a simple polygon P as a list of its vertices (p0,p1,...,pn).
- Afterwards a new point p is given (e.g. by clicking the mouse on the screen.



3

Point-in-polygon, 2.

- If we click inside the polygon P we expect the computer to perform some action; if we click outside we expect a different action.
- So we have to decide the point-in-polygon query.
- The *crossing-number* method:



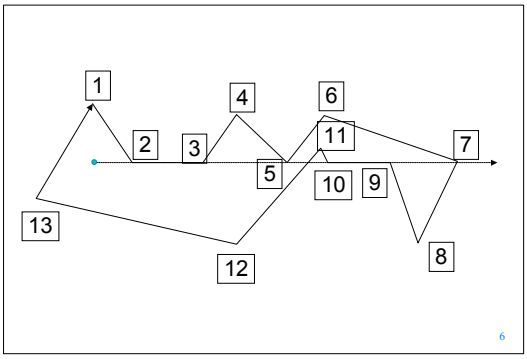
4

The ray crossings method

- The idea of this method is to COUNT how many edges of P are crossed by a ray starting from the query point p and directed along the positive x direction.
- If we do not hit any vertex of P, then when p is outside we get an even number of crossings, when p is inside we get an odd number of crossings.
- Special cases: the ray pass through a vertex, the ray overlaps an edge.

5

Special cases



6

How to deal with the special cases

- If the ray R overlays the edge (p[i],p[i+1]) we check the points p[i-1] and p[i+2], if they are on the same side of R we do not have a crossing, if they are on different sides we have a crossing.
- If the ray R intersect edge (p[i],p[i+1]) at p[i+1], then we have to check whether p[i+2] and p[i] are on different sides of the ray, in such case we have a crossing.
- We need to modify the segment intersection detection procedure so that it returns more than TRUE/FALSE: we need the values: NO-INTERSECTION, OVERLAY, HIT-SECOND-VERTEX, PROPER-INTERSECTION.
- We can give R by a pair (p, (+infinity,py)).

7

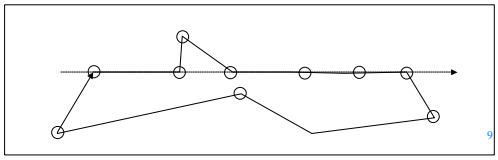
Point-in-simple-polygon-test

- Point-in-polygon((p,q): ray; P[0,..,n-1]: Polygon)
  Cross:= 0;
  FOR i:=0 TO n-1
  DO x := INTERSECT((p,q),(P[i],P[i+1])
  CASE x OF
  PROPER-INT : Cross:=Cross+1
  NO-INT : do nothing
  HIT-SECOND-V :
  IF (left(p,q,P[i]) ≠ left(p,q,P[i+2]))
  THEN Cross:=Cross+1
  END IF
  OVERLAY:
  IF (left(p,q,P[i-1]) ≠ left(p,q,P[i+2]))
  THEN Cross := Cross+1
  END IF
  END FOR
  IF EVEN(Cross) THEN RETURN False,
  ELSE RETURN True
  END IF

8

Observations and exercise

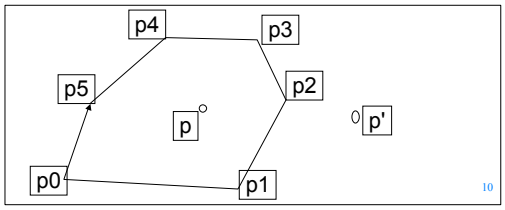
- We assume that all index arithmetic is done MOD n, so to be able to wrap around the array.
- The code does work properly only if we assume that there are not 2 collinear consecutive edges.
- Exercise: modify so to make it work when there are collinear consecutive edges.



9

When the input polygon is convex.

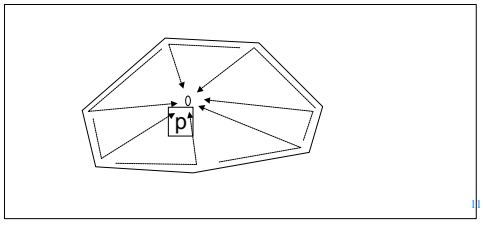
- The point-in-polygon procedure for simple polygons works in time O(n). This procedure works for convex polygons too.
- For convex polygons we can produce simpler code by exploiting some properties of convex polygons. Any suggestion?



10

Point-in-convex-polygon

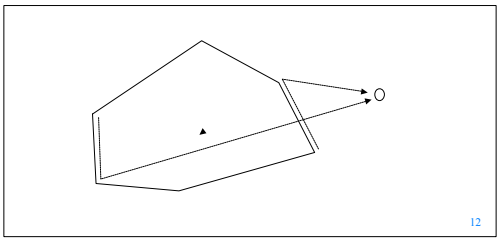
- We can exploit the fact that if p is inside a convex polygon C=[p0,...,pn] then it makes always the same type of turn with the edges of the polygon.



11

Point-in-convex-polygon, 2

- Instead if p is outside the convex polygon C, then we will always find two different turns while we visit the edges of C one by one in (clock-wise or counterclockwise) sequence.



12

Exercise

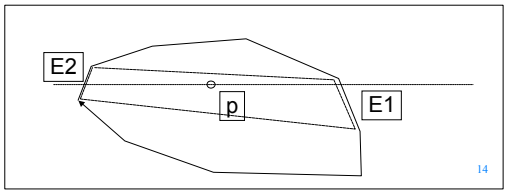
- Can you write the code to test is a point p is inside or outside the a convex polygon C =[p0,...,pn]?
- You should USE the procedures left(..), collinear(..) etc, that we introduced in the previous lecture.
- Can you handle the case when p in on the boundary of C?
- What is the time complexity of your procedure?

13

A fast point-in-convex-polygon test

(using binary search)

- Main idea: if p is in C then when we draw an horizontal line through p we hit two edges E1, E2. p is also inside the convex quadrilateral formed by the vertices of E1 and E2.



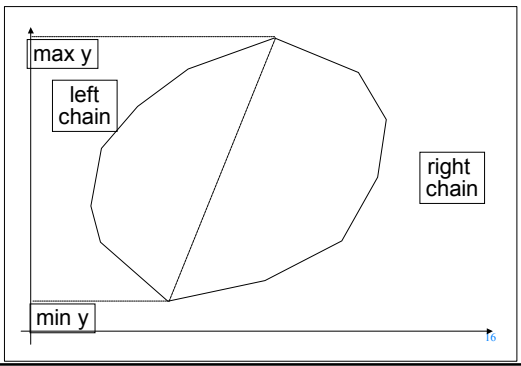
14

Fast point-in-convex-polygon test, 2

- Conversely, if p is outside Q(E1,E2) then it is outside C.
- The strategy is: find E1 and E2, then apply the previous method to the quadrilateral Q(E1,E2).
- To start with we find the points with the largest and smallest y-coordinate. We find the right chain and the left-chain. E1 is in the left chain, and E2 is in the right chain, so we need to describe only how we find one of them, the other is found using the same procedure.

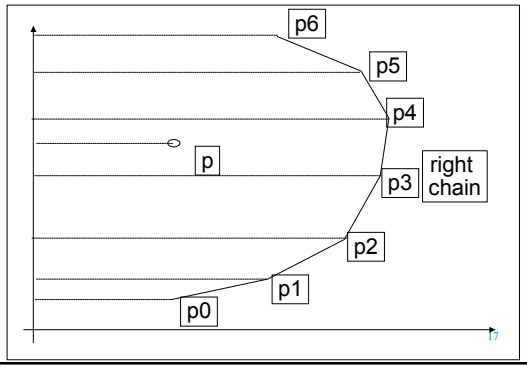
15

Left chain, right chain



16

Right chain only



17

Determining E1.

- We make a binary search with py on the array of numbers [p0y,p1y,p2y,p3y,p4y,p4y,p5y,p6y], which are the y coordinates of the vertices of the right chain. This array is sorted.
- We locate px in the array by binary search and we find the two neighbours: p3y and p4y. Then the horizontal line through p hits the edge (p3,p4) and this is the edge E1 we were looking for.
- Since we use binary search, to find E1 takes time O(log n), after the left chain and the right chain have been found.

18

Fast point-in-convex-polygon

(pseudocode)

- Preprocessing(C: convex polygon)
  Find top vertex,
  Find bottom vertex,
  Find left chain.
  Find right chain.
- Fast-in-convex-polygon(p: point, C: convex polygon)
  Use binary-search on left-chain with py to determine E1.
  Use binary search on right-chain with py to determine E2
  Find quadrilateral Q(E1,E2).
  RETURN(Slow-point-in-convex-polygon(p,Q)).

19

Observation and exercises

- Exercise, what is the running time of the prerpocessing on a convex polygon of n points?
- What is the running time of 1 in-polygon query?
- How much storage do we need? Can we do everything in place (i.e. using the original array of points of C)?

20

Conclusions

- We have seen how to test whether a query point is inside or outside a simple polygon
- We have seen a second easy method that works in linear time for convex polygons.
- Finally we have seen a fast method for testing whether a point is inside or outside a convex polygon that uses time O(log n) after linear preprocessing.
- Now we can click a mouse on an a drawing represented as a (simple, convex) polygon.

21

Homework

(due in 3 weeks)

- Write a program in C or Pascale to test whether a point is inside or outside a simple polygon.
- Test the code with a polygon with 20 vertices in such way to test all the special conditions discussed in class.
- Test at least 20 points for inside/outside.
- Make a drawing of the polygon and the points and compare the drawing with the output of your program.

22

Further readings

- http://www.ecse.rpi.edu/Homepages/wrf/Research/Short_Notes/pnpoly.html
- http://astronomy.swin.edu.au/~pbourke/geometry/insidepoly/
- http://softsurfer.com/Archive/algorithm_0103/algorithm_0103.htm

23