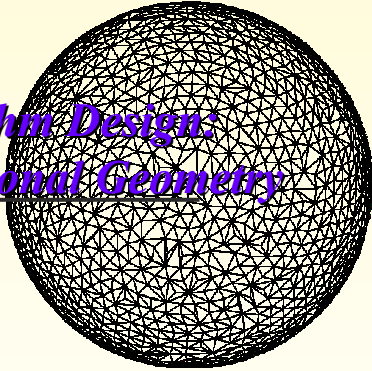


## Algorithm Design: Computational Geometry

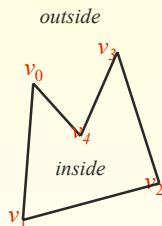


### Coming up...

- One important **problem** and a couple of **solutions**
- The fascinating area of *Computational Geometry*
- Areas of **application**

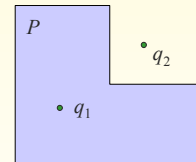
### Polygons

- A *polygon* is a 2D shape whose boundary consists of straight-line segments
- Representation:
  - The “corners” of the polygon:
    - $v_0, v_1, v_2, v_3, v_4$
  - The “sides” of the polygon:
    - $v_0v_1, v_1v_2, v_2v_3, v_3v_4, v_4v_0$
- A *polygon* divides space into an *inside* and an *outside*

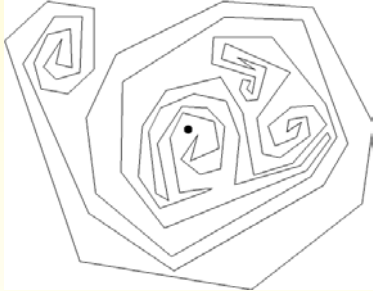


### Point-in-Polygon Problem

- For a given a polygon  $P$  and a point  $q$ , does  $q$  lie inside  $P$ ?



## Point-in-Polygon Problem



## Point-in-Polygon Problem

- ANY IDEAS????



## Ray Crossing



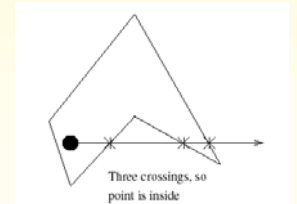
Hoping fences analogy

## Ray Crossing

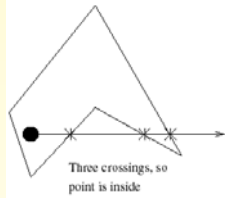


Hoping fences analogy

1. Shoot a ray  $r$  out from  $q$
2. If the number of times the ray  $r$  crosses  $P$  is **odd**, then  $q$  is inside. If it is **even**, then  $q$  is outside.



## Ray Crossing

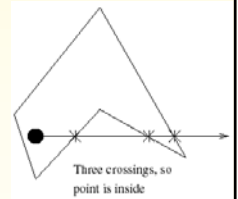


1. Let  $num\_crossings = 0$
2. Consider each side  $s$  of the polygon  $P$   
If a ray shot **Eastward** from  $q$  intersects  $s$ , then increase  $num\_crossings$  by 1
3. When all the sides of the polygon  $P$  have been considered, if  $num\_crossings$  is even then  $q$  is outside of  $P$  and inside if  $num\_crossings$  is odd

## Defining “Intersects”

A ray shot Eastward intersects a side  $s$  if

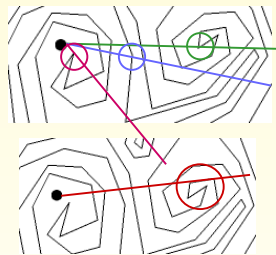
1. the ray passes “above” one endpoint of  $s$  and “below” another endpoint of  $s$
2. the intersection of the ray with  $s$  is East (right) of  $q$



## Degeneracies

- What if the ray hits a *corner* of  $P$  or is collinear with one of the *sides*  $s$  of  $P$ ?

- The ray crosses  $s$  if one of  $s$ 's endpoints is strictly “above” the ray and the other endpoint is on or “below” the ray



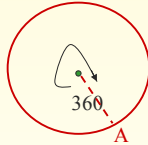
## Ray Crossing

If one of  $s$ 's endpoints is strictly above the ray and the other endpoint on or below it, and if the point of intersection of the ray and  $s$  is to the right of  $q$

1. Let  $num\_crossings = 0$
2. Consider each side  $s$  of the polygon  $P$   
If a ray shot **Eastward** from  $q$  intersects  $s$ , then increase  $num\_crossings$  by 1
3. When all the sides of the polygon  $P$  have been considered, if  $num\_crossings$  is even then  $q$  is outside of  $P$  and inside if  $num\_crossings$  is odd

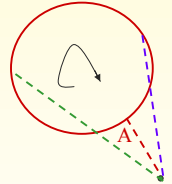
## Going around a circle

- Suppose you started walking on the boundary of a circle at a point A. Consider holding a string attached **inside** the circle. At position A, the angle you have traveled is 0
- As you go around the circle, the angle with the red line increases
- By the time you get back to A, it is 360 degrees



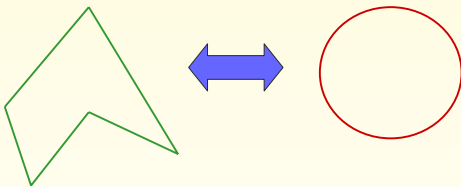
## Going around a circle

- Suppose you started walking on the boundary of a circle at a point A. Consider holding a string attached **outside** the circle. At position A, the angle you have traveled is 0
- As you go around the circle, the total angle traveled increases, then decreases and then increases again
- By the time you get back to A, the cumulative angle is 0 degrees



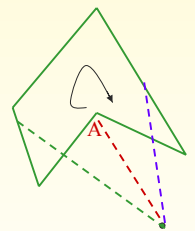
## Circles and Polygons

- A polygon is a **\*distorted\*** circle



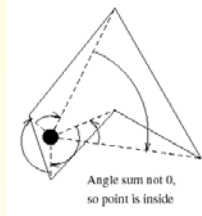
## Winding Number

- Suppose you started walking on the boundary of a polygon at a point A. Consider holding a string attached at the point  $q$ . At position A, the angle you have traveled is 0
- As you go around the polygon, you keep track of the cumulative angle you have traveled
- By the time you get back to A, if the cumulative angle is **360**, then  $q$  is **inside**; if the cumulative angle is 0, then  $q$  is **outside**



## Winding Number

1. Sum the signed angles formed at the point  $q$  by each side's endpoints
2. If the total is 360, then  $q$  is inside, if it is 0, then  $q$  is outside



Note that if you choose as a convention to record angles as positive that are clockwise and then process the sides of the polygon though a bug was crawling along them clockwise, then all but 1 angle is positive in the above picture and the total is +360.

## Which is better?

### Ray Crossing

1. Let  $num\_crossings = 0$
2. Consider each side  $s$  of the polygon  $P$   
If a ray shot Eastward from  $q$  intersects  $s$ , then increase  $num\_crossings$  by 1
3. When all the sides of the polygon  $P$  have been considered, if  $num\_crossings$  is even then  $q$  is outside of  $P$  and inside if  $num\_crossings$  is odd

### Winding Number

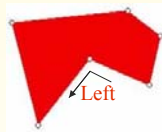
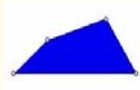
1. Sum the signed angles formed at the point  $q$  by each side's endpoints
2. If the total is 360, then  $q$  is inside, if it is 0, then  $q$  is outside

**Winding Number** uses inverse trigonometric functions and generally very sensitive to round-off error

**Ray Crossing** much faster in practice (up to 63 times faster for polygons with up to 100 sides)

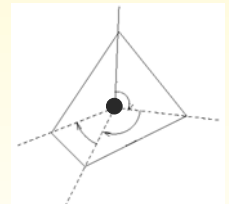
## Types of Polygons

- **Convex** When traversing the boundary in clockwise order, all the turns made are *right*
- **Non-convex** At least one of the turns is a left turn



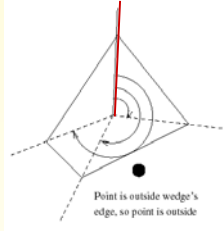
## Convex Polygons

- When considering general polygons, need to look at each one of the sides of the polygon
- In convex polygons,
  - The angles for the winding number computation are never negative!
- Goal: a faster algorithm for convex polygons



## Convex Polygons - Extra

1. Add a central point to polygon
2. Divide into wedges
3. Choose an anchor line
4. Compute and save the angles from the anchor line to each wedge's line, as well as each wedge's polygon side
5. Compute the angle from the anchor line to  $q$ .
6. Do **binary search** to determine which wedge  $q$  is in
7. If  $q$  is left of the wedge's polygon edge,  $q$  is inside  $P$ ; otherwise, it is outside  $P$



## Computational Geometry

- Discipline concerned with the study of *geometric* algorithms
- Many real problems involve geometry:
  - Going to the nearest post office
  - Moving a ladder in tight quarters
  - Studying maps
- Michael Shamos, PhD thesis, "Computational Geometry," 1978

## Computational Geometry

- Generally concerned with objects that are represented as points, line segments, polygons, polyhedra, etc.

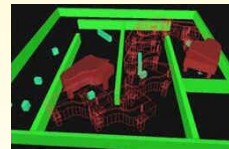


## Areas of Application

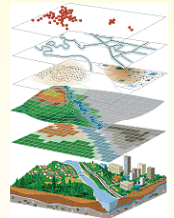
- Computer graphics



- Robotics



- Geographic Information Systems



## Areas of Application

- Computer Aided Design



- Pattern Recognition

