Programming Challenges

Week 9 - Geometry

Claus Aranha
caranha@cs.tsukuba.ac.jp

College of Information Sciences

2015-06-22

Last updated June 20, 2015

Introduction

Geometry

Problems related to points, lines, angles and circles.
Usually there will be more than one way to calculate.

Geometrical Constructs

Lines, Segmenst, Planes, Circles, Convex Polygons, Concave Polygons, etc...

Geometrical Computing

Our main concern, are degeneracies and instability

Degeneracies

Numerical Instability

$$\arcsin\left(\sin\left(\pi/4\right)\right) \neq \pi/4$$

Don't forget that operations with real numbers are not guaranteed to be precise;

Degeneracies

Special cases for geometric calculations. Normally caused by divisions by zero. But sometimes have other sources.

$$\tan\left(\pi/2\right) = \frac{\sin\left(\pi/2\right)}{\cos\left(\pi/2\right)} = \frac{1}{0}$$

Line (1)

Characteristics

- Infinite;
- Divide a plan into two;
- Segment – a limited line

Line (2)

Representation – Two Points

- A line can be described by two points;
- $(x_0, y_0), (x_1, y_1)$

Problems with this representation

- Not unique: We can have two identical lines represented by different points
- Calculating extra points requires interpolation;

Line (3)

Representation – Point and angle

$$y = mx + b$$

- m (slope): $\frac{y_1 - y_0}{x_1 - x_0}$
- b (y-intercept): the point where $x = 0$;

Problem: When the line is vertical, we have a degeneration (division by zero on the slope)

Line (4)

Representation – Point and angle 2

$$ay + bx + c = 0$$

or

$$x = c$$

When the line is vertical

```
p2l(double[] p1, double[] p2):
    if (p1[0] == p2[0]): // vertical line
        l.a = 1;
        l.b = 0;
        l.c = -p1[0];
    else:
        l.b = 1;
        l.a = -(p1[1]-p2[1])/(p1[0]-p2[0]);
        l.c = -(l.a*p1[0])-(l.b*p1[1]);
```

Line Intersection (1)

Line Intersection

We can calculate if two lines are parallel quickly, by checking if their inclination is the same. Note the Epsilon!

```
parallelQ(line l1, line l2):
   return ((abs(l1.a-l2.a) <= EPSILON)&&
           (abs(l1.b-l2.b) <= EPSILON))
```

Line Intersection (2)

Line Intersection Point

If the lines are not parallel, they have one intersection point.

$$x = \frac{b_2 - b_1}{m_1 - m_2}, y = m_1 \frac{b_2 - b_1}{m_1 - m_2} + b_1$$

```
intersection_point(line l1, line l2):
  if (!(parallelQ(l1,l2))):
    p[0] = (l2.b*l1.c - l1.b*l2.c)/
           (l2.a*l1.b - l1.a*l2.b);
    if (abs(l1.b) > EPSILON): // Vertical?
      p[1] = - (l1.a*(p[X])+l1.c)/l1.b;
    else:
      p[1] = - (l2.a*(p[X])+l2.c)/l2.b;
```

Line Intersection (3)

Angle between two lines

Two non parallel lines will always intersect at a given angle. If the lines are in the $ax + by + c = 0$ format, we can calculate their angles as follows:

```
intersection_angle(line l1, line l2):
    num = l1.a*l2.b - l2.a*l1.b
    den = l1.a*l2.a * l1.b*l2.b
    return(tan(num/den))
```

Line Intersection (4)

Closest Point

- The closest point p_l in a line l to point p, is the point where the line (p, p_l) intersects l.
- Closest point can be used to find the distance between a line and a point ($d(p, p_l)$);

- Degenerate/Easy cases: p is in l, l is vertical, l is horizontal;
- The slope m of the line (p, p_l) is $\frac{1}{l.a}$;
- Calculate the intersection between (p, p_l) and l;

Line Segments (1)

Line segments are lines delimited by start and end points;

```
typedef struct {
    point p1,p2
} segment;
```

Line Segments (2)

Degenerative Cases

- Are the Segments in the same line? (test for same points)
- Are the Segments parallel? (no intersection)

- Calculate the intersecting point between the lines.
- Test if this point is whithin a rectangle defined by each line segment.

Triangles (1)

- Polygon defined by three line segments;
- Characterized by the relationship between its angles and the line segment sizes;
- Commonly used to represent more complex polygons;

Manipulating angles

- Angles can be represented by radians (0 to 2π) or degrees (0 to 360);
- Mixing the two of them is an easy way to insert bugs in your code;
- Make sure what is the usual input for your library's trigonometric functons;

Triangles (2)

Basic Triangle Facts

- Three angles, summing to a total of 180 degrees (π radians);
- Law of sines (A,B,C are angles; a,b,c are opposite edges):

$$\frac{a}{\sin A} = \frac{b}{\sin B} = \frac{c}{\sin C}$$

- Law of Cosines:

$$a^2 = b^2 + c^2 - 2bc\cos(A)$$

Triangles (3)

Right Triangles

A right triangle has one angle with 90 degrees ($\pi/2$ radians). It has many neat properties;

For α a non-right angle, with an *opposite* side and an *adjacent* side;

- $\cos(\alpha) = \frac{|adjacent|}{|hypotenuse|}$
- $\sin(\alpha) = \frac{|opposite|}{|hypotenuse|}$
- $\tan(\alpha) = \frac{|opposite|}{|adjacent|}$

Triangles (4)

Common problems with triangles

- Given two angles and a side, find the rest;
- Given two sides and an angle, find the rest;
- Given a side and a height, find the rest;
- Etc;

Triangles (5)

Area of a Triangle

a is the altitude, h is the base;

$$A(T) = (1/2)ah$$

Signed area

a,b,c are the points of a triangle.

$$(a_x b_y - a_y b_x + a_y c_x - a_x c_y + b_x c_y - c_x b_y)/2$$

- Negative signed area: a,b,c are clockwise;
- Positive signed area: a,b,c are counterclockwise;
- Zero signed area: a,b,c are collinear;

Circle 1

Representation

- Center point and radius;
- Three boundary points;

Measures

- Area: πr^2
- Circumference: $2\pi r$

Circle 2

Intersection between line and circle

Radius r and distance between center and line d;

- $d > r$ – no intersection;
- $d == r$ – tangent, one intersection;
- $d < r$ – two intersection points;

Intersection between two circles

- Two circles will intersect if the distance to their centers $\leq r_1 + r_2$
- The points of intersection form triangles with determined sides. Angles and coordinates can be calculated as needed.

Polygons

Definition

Let's define a polygon as a closed chain of non-intersecting line segments. We can represent polygons by listing the *n* vertices in order around its boundary.

```
typedef struct {
   int n;
   point p[MAXPOLY]
} polygon
```

- We can represent the “last” segment by (p[(n-1)%n],p[n%n])

Convex Polygons

Definition

A polygon P is convex if any line segment defined by two points within P are contained in P

- All internal angles in a convex polygon must be $< \pi$ radians;
- The sum of all angles in a convex polygon is 2π;
- We can test a polygon by convexity by checking that all its angles turn to the same side. (ccw a,b,c)

The Convex Hull

The convex hull is a basic algorithm often used to organize unstructured data.

The Graham Scan

Simple algorithm to create a convex hull

- select leftmost and lowest point as starting points;
- sort points by angle direction from the starting points;
- add the first point to the hull, and repeat.

How to avoid degeneracy? (Wrap around, collinear points, repeated points)

Area of a polygon

Convex Polygon

Add all signed triangular areas: The negative areas will compensate the positives.

Concave Polygon

Picasso Algorithm: Remove “ears” (triangles) from the polygon, adding to the total area.

Testing if a point is inside a polygon

Problems

- Dog and Gopher
- Rope Crisis in Ropeland
- Herding Frosh
- Chainsaw Massacre