ACM/ICPC: Competitive Programming Notebook

Lucas Mattioli, Marlon Mendes, Simião Carvalho

Contents

Points

Comparing floating point values

Returns true if double values a and b are equal

```
const double EPS { 1e-9 };
bool equals(double a, double b)
{
    return fabs(a - b) < EPS;
}
```

Listing 1: equals

Lines

General equation of a line

Non-normalized form: ax + by + c = 0

```
class Line {
public:
    double a;
    double b;
    double c;

    Line(double av, double bv, double cv) : a(av), b(bv), c(cv) {}

    Line(const Point& p, const Point& q)
    {
        a = p.y - q.y;
        b = q.x - p.x;
        c = p.x * q.y - p.y * q.x;
    }
};
```

Listing 2: General equation of a line

General equation of a line normalized

```
class Line {
public:
    double a;
    double b;
    double c;

    Line(double av, double bv, double cv) : a(av), b(bv), c(cv) {}

    Line(const Point& p, const Point& q)
    {
        a = p.y - q.y;
        b = q.x - p.x;
        c = p.x * q.y - p.y * q.x;

        auto k = a ? a : b;

        a /= k;
        b /= k;
```

```
19          c /= k;
    }
21  };
```

Listing 3: General equation of a line

Point on a line

Is the given point located on the given Line?

```
1   template<typename T>
    struct Line {
3       bool contains(const Point<T>& P) const
        {
5           return equals(a*P.x + b*P.y + c, 0);
        }
7   };
```

Listing 4: Point on line

Vectors

Angle between vector and X-axis

Returns an angle in radians in the interval $[-\pi, +\pi]$. A positive angle means in the COUNTER-clockwise direction. A negative angle is measured in the clockwise direction. Note that the atan2 swaped the parameters.

```
1   inline double angle(double x, double y) {
        return atan2(y, x);
3   }
```

Listing 5: angle between X-axis and vectorx, y

Translation

```
1   Point translate(const Point& P, double dx, double dy)
    {
3       return Point { P.x + dx, P.y + dy };
    }
```

Listing 6: Translate point

Rotation around origin

```
    Point rotate(const Point& P, double angle)
2   {
        auto x = cos(angle) * P.x - sin(angle) * P.y;
4       auto y = sin(angle) * P.x + cos(angle) * P.y;

6       return Point { x, y };
    }
```

Rotation around another point

```
Point rotate(const Point& P, double angle, const Point& C)
{
    auto Q = translate(P, -C.x, -C.y);
    Q = rotate(Q, angle);
    Q = translate(Q, C.x, C.y);

    return Q;
}
```

Rotation around origin 3D

$$R_x = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{bmatrix}, \quad R_y = \begin{bmatrix} \cos\theta & 0 & -\sin\theta \\ 0 & 1 & 0 \\ \sin\theta & 0 & \cos\theta \end{bmatrix}$$

$$R_z = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$