Geometry with complex numbers

Bruce Merry

University of Cape Town

10 May 2012

Outline

Outline

Definition

Complex numbers have the form $a + b\mathbf{i}$

- $(a + b\mathbf{i}) + (c + d\mathbf{i}) = (a + c) + (b + d)\mathbf{i}$
- $-(a + b\mathbf{i}) = (-a) + (-b)\mathbf{i}$
- $s(a + b\mathbf{i}) = (sa) + (sb)\mathbf{i}$

Complex numbers form a 2D vector space

Multiplication

By definition, $\mathbf{i}^2 = -1$:

- $(a + b\mathbf{i})(c + d\mathbf{i}) = (ac - bd) + (ad + bc)\mathbf{i}$
- $(a + b\mathbf{i})^{-1} = \frac{a}{n} + \frac{b}{n}\mathbf{i}$ where $n = a^2 + b^2$

Complex numbers form a field

Polar Form

$$\cos x = \frac{x^0}{0!} - \frac{x^2}{2!} + \frac{x^4}{4!} - \cdots$$
$$\sin x = \frac{x^1}{1!} - \frac{x^3}{3!} + \frac{x^5}{5!} - \cdots$$
$$e^x = \frac{x^0}{0!} + \frac{x^1}{1!} + \frac{x^2}{2!} + \cdots$$

Polar Form

$$\cos x = \frac{x^0}{0!} - \frac{x^2}{2!} + \frac{x^4}{4!} - \cdots$$

$$\sin x = \frac{x^1}{1!} - \frac{x^3}{3!} + \frac{x^5}{5!} - \cdots$$

$$e^x = \frac{x^0}{0!} + \frac{x^1}{1!} + \frac{x^2}{2!} + \cdots$$

$$e^{b\mathbf{i}} = \frac{b^0}{0!} + \frac{b^1}{1!}\mathbf{i} - \frac{b^2}{2!} - \frac{b^3}{3!}\mathbf{i} + \frac{b^4}{4!} + \frac{b^5}{5!}\mathbf{i} - \cdots$$

Polar Form

$$\begin{aligned}
\cos x &= \frac{x^0}{0!} - \frac{x^2}{2!} + \frac{x^4}{4!} - \cdots \\
\sin x &= \frac{x^1}{1!} - \frac{x^3}{3!} + \frac{x^5}{5!} - \cdots \\
e^x &= \frac{x^0}{0!} + \frac{x^1}{1!} + \frac{x^2}{2!} + \cdots \\
e^{b\mathbf{i}} &= \frac{b^0}{0!} + \frac{b^1}{1!}\mathbf{i} - \frac{b^2}{2!} - \frac{b^3}{3!}\mathbf{i} + \frac{b^4}{4!} + \frac{b^5}{5!}\mathbf{i} - \cdots \\
&= \cos b + (\sin b)\mathbf{i}
\end{aligned}$$

Polar Form

$$\cos x = \frac{x^0}{0!} - \frac{x^2}{2!} + \frac{x^4}{4!} - \cdots$$

$$\sin x = \frac{x^1}{1!} - \frac{x^3}{3!} + \frac{x^5}{5!} - \cdots$$

$$e^x = \frac{x^0}{0!} + \frac{x^1}{1!} + \frac{x^2}{2!} + \cdots$$

$$e^{b\mathbf{i}} = \frac{b^0}{0!} + \frac{b^1}{1!}\mathbf{i} - \frac{b^2}{2!} - \frac{b^3}{3!}\mathbf{i} + \frac{b^4}{4!} + \frac{b^5}{5!}\mathbf{i} - \cdots$$

$$= \cos b + (\sin b)\mathbf{i}$$

$$e^{a+b\mathbf{i}} = e^a \cos b + (e^a \sin b)\mathbf{i}$$

Polar Form

$$\cos x = \frac{x^0}{0!} - \frac{x^2}{2!} + \frac{x^4}{4!} - \cdots$$

$$\sin x = \frac{x^1}{1!} - \frac{x^3}{3!} + \frac{x^5}{5!} - \cdots$$

$$e^x = \frac{x^0}{0!} + \frac{x^1}{1!} + \frac{x^2}{2!} + \cdots$$

$$e^{b\mathbf{i}} = \frac{b^0}{0!} + \frac{b^1}{1!}\mathbf{i} - \frac{b^2}{2!} - \frac{b^3}{3!}\mathbf{i} + \frac{b^4}{4!} + \frac{b^5}{5!}\mathbf{i} - \cdots$$

$$= \cos b + (\sin b)\mathbf{i}$$

$$e^{a+b\mathbf{i}} = e^a \cos b + (e^a \sin b)\mathbf{i}$$

If $z = re^{\mathbf{i}\theta} = r\cos\theta + (r\sin\theta)\mathbf{i}$ then $|z| = r, \arg z = \theta$.

Complex Conjugate

The conjugate of z is written $\overline{z}$:

- $\overline{a + b\mathbf{i}} = a - b\mathbf{i}$

Complex Conjugate

The conjugate of z is written $\overline{z}$:

- $\overline{a + b\mathbf{i}} = a - b\mathbf{i}$
- $\overline{re^{\theta\mathbf{i}}} = re^{-\theta\mathbf{i}}$

Complex Conjugate

The conjugate of z is written $\overline{z}$:

- $\overline{a + b\mathbf{i}} = a - b\mathbf{i}$
- $\overline{re^{\theta\mathbf{i}}} = re^{-\theta\mathbf{i}}$
- $\overline{z + w} = \overline{z} + \overline{w}$
- $\overline{zw} = \overline{z}\,\overline{w}$
- $\overline{z^{-1}} = \overline{z}^{-1}$

Complex Conjugate

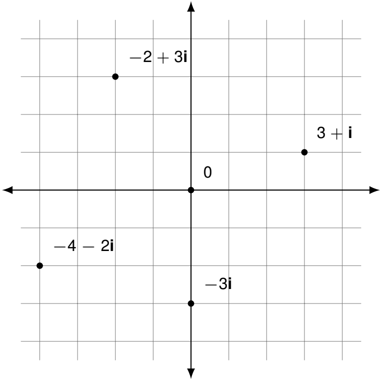
The conjugate of z is written $\overline{z}$:

- $\overline{a + b\mathbf{i}} = a - b\mathbf{i}$
- $\overline{re^{\theta\mathbf{i}}} = re^{-\theta\mathbf{i}}$
- $\overline{z + w} = \overline{z} + \overline{w}$
- $\overline{zw} = \overline{z}\,\overline{w}$
- $\overline{z^{-1}} = \overline{z}^{-1}$
- $\overline{z} + z = 2\Re(z)$
- $\overline{z}\,z = |z|^2$

Outline

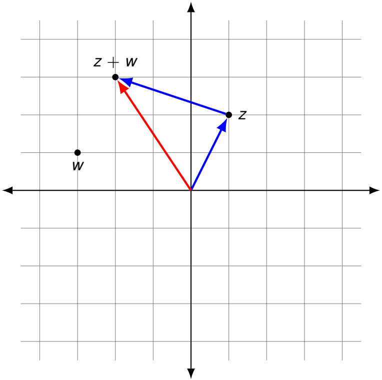
The Argand Plane

Treat $a + b\mathbf{i}$ as coordinates (a, b)



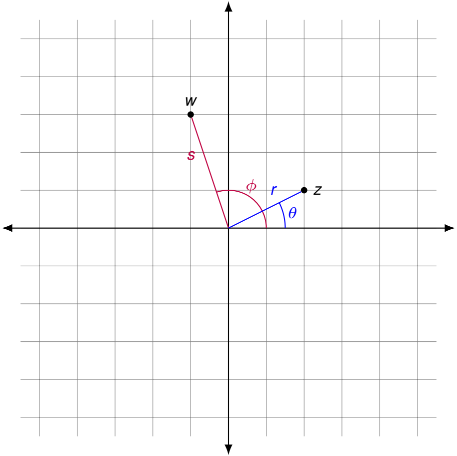
Addition

Addition works just like for vectors:



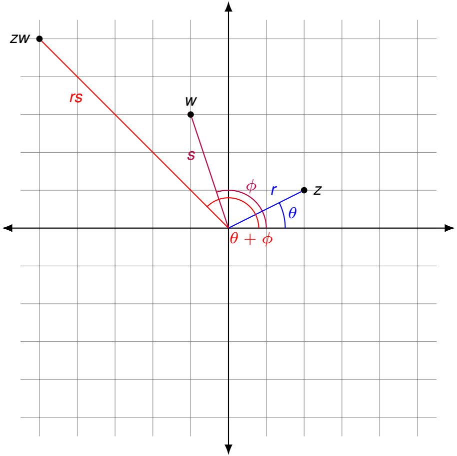
Multiplication

Multiplication rotates and scales:



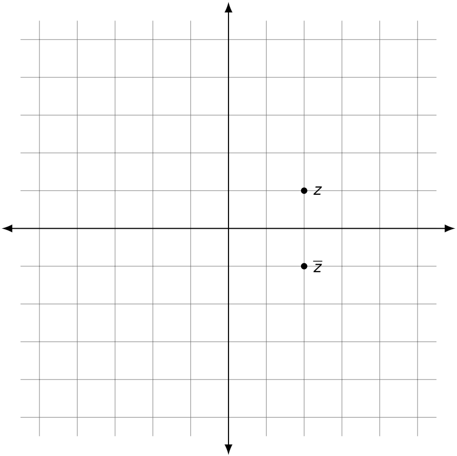
Multiplication

Multiplication rotates and scales: $re^{\mathbf{i}\theta} \cdot se^{\mathbf{i}\phi} = rse^{\mathbf{i}(\theta+\phi)}$



Complex Conjugate

$\overline{z}$ is the reflection of z in the real axis:



Outline

1. Complex numbers
   - Definition
   - Geometric interpretation
2. Geometry
   - Introduction
   - Algorithms
   - Problems

Why Use Complex Numbers?

- C++ provides a `complex` template class

Why Use Complex Numbers?

- C++ provides a `complex` template class
- All the benefits of vectors

Why Use Complex Numbers?

- C++ provides a `complex` template class
- All the benefits of vectors
- Can manipulate angles without trigonometry

Why Use Complex Numbers?

- C++ provides a `complex` template class
- All the benefits of vectors
- Can manipulate angles without trigonometry
  - Represent θ using $re^{\mathbf{i}\theta}$ (r arbitrary)

Why Use Complex Numbers?

- C++ provides a `complex` template class
- All the benefits of vectors
- Can manipulate angles without trigonometry
  - Represent θ using $re^{\mathbf{i}\theta}$ (r arbitrary)
  - Multiply to add angles

Why Use Complex Numbers?

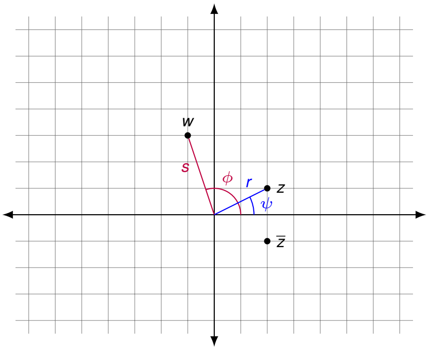
- C++ provides a `complex` template class
- All the benefits of vectors
- Can manipulate angles without trigonometry
  - Represent θ using $re^{\mathbf{i}\theta}$ (r arbitrary)
  - Multiply to add angles
  - Conjugate to negate an angle

Warnings

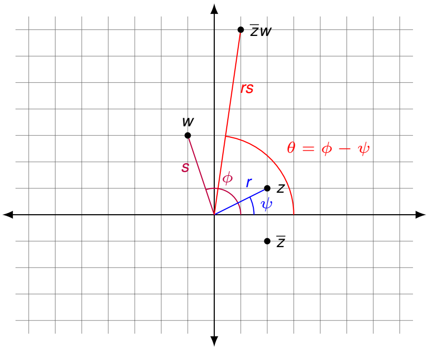
- Avoid floating-point if at all possible
- Always think about the corner cases
- Be careful of overflows

Outline

Dot And Cross Product

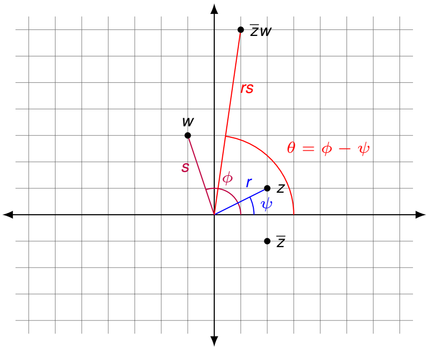


Dot And Cross Product



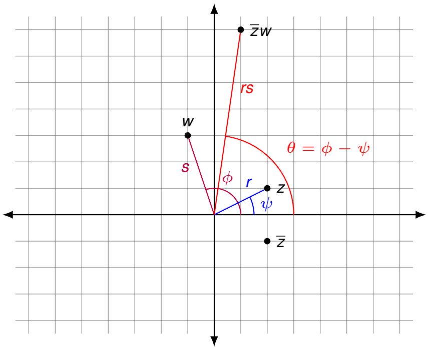
Dot And Cross Product

$$\bar{z}w = rs\cos\theta + (rs\sin\theta)\mathbf{i}$$



Dot And Cross Product

$$\bar{z}w = rs\cos\theta + (rs\sin\theta)\mathbf{i}$$




```
int dot(pnt z, pnt w) { return real(conj(z) * w); }
int cross(pnt z, pnt w) { return imag(conj(z) * w); }
```

Signed Triangle Area

For convenience, I usually define

```
int cross(pnt a, pnt b, pnt c) {
    return cross(b − a, c − a);
}
```

which is twice the signed area of $\triangle ABC$.

Line-Line Intersection Test

Start with a bounding box test

```
bool intersects(pnt a, pnt b, pnt p, pnt q) {
    if (max(a.real(), b.real()) < min(p.real(), q.real())) return false;
    if (min(a.real(), b.real()) > max(p.real(), q.real())) return false;
    if (max(a.imag(), b.imag()) < min(p.imag(), q.imag())) return false;
    if (min(a.imag(), b.imag()) > max(p.imag(), q.imag())) return false
```

Line-Line Intersection Test

Check if *PQ* lies entirely on one side of *AB*:

```
int cp = cross(a, b, p);
int cq = cross(a, b, q);
if (cp > 0 && cq > 0) return false;
if (cp < 0 && cq < 0) return false;
```

and check if *AB* lies entirely on one side of *PQ*

```
int ca = cross(p, q, a);
int cb = cross(p, q, b);
if (ca > 0 && cb > 0) return false;
if (ca < 0 && cb < 0) return false;
```

Passed all the tests, so the lines intersect

```
return true;
```

Outline

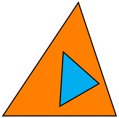
Scaled Triangles

Problem Statement

Google Code Jam 2008, EMEA semifinal, problem A

- Two triangles, T_1 and T_2 are given
- T_2 is T_1, scaled, rotated and translated
- Scale factor is strictly between 0 and 1

Find a fixed point of the transformation $T_1 \rightarrow T_2$



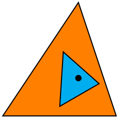
Scaled Triangles

Problem Statement

Google Code Jam 2008, EMEA semifinal, problem A

- Two triangles, T_1 and T_2 are given
- T_2 is T_1, scaled, rotated and translated
- Scale factor is strictly between 0 and 1

Find a fixed point of the transformation $T_1 \to T_2$



Scaled Triangles

Transformation

In complex numbers, this is an affine function

$$f(z) = vz + w$$

where v scales and rotates, w translates

Scaled Triangles

Finding The Transformation

$$\begin{aligned} B_2 - A_2 &= f(B_1) - f(A_1) \\ &= (vB_1 + w) - (vA_1 + w) \\ &= v(B_1 - A_1) \end{aligned}$$

Therefore

$$v = \frac{B_2 - A_2}{B_1 - A_1}.$$

Also, $A_2 = vA_1 + w$ gives

$$w = A_2 - vA_1$$

Scaled Triangles

Fixed Point

The fixed point z satisfies

$$\begin{aligned} z &= vz + w \\ (1 - v)z &= w \\ z &= \frac{w}{1 - v} \end{aligned}$$

Scaled Triangles

Sample code

```
typedef complex<double> pnt;
pnt verts[2][3];
for (int i = 0; i < 2; i++)
    for (int j = 0; j < 3; j++)
        cin >> verts[i][j].real() >> verts[i][j].imag();
pnt edge0 = verts[0][1] - verts[0][0];
pnt edge1 = verts[1][1] - verts[1][0];
pnt scale = edge1 / edge0;
pnt bias = verts[1][0] - scale * verts[0][0];
pnt z = bias / (1.0 - scale);
```

CropCircles

Problem Statement

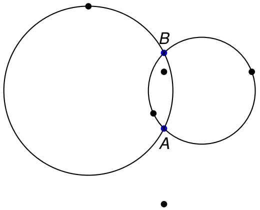
TopCoder SRM 539 Div 1, Level 2: Count the number of distinct circles that pass through at least three of the given points.

- Have to consider 3 points collinear (no circle)
- Have to consider 4+ points concyclic (shared circle)

CropCircles

Initial Idea

- Fix two points A, B, find all circles through A, B
- Count a circle only if A, B are the lowest-index points

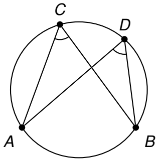


CropCircles

Circles And Angles

If A, B, C, D are concyclic, then

$$\angle ACB = \angle ADB$$

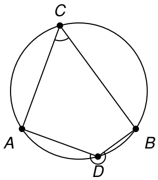


CropCircles

Circles And Angles

If A, B, C, D are concyclic, then

$$\angle ACB = \angle ADB \pmod{\pi}$$



CropCircles

Bucketing Angles

We need to bucket angles $\angle AXB$ for all X, but $re^{\mathbf{i}\theta}$ representation is not unique.

- Call arg(z)?

CropCircles

Bucketing Angles

We need to bucket angles $\angle AXB$ for all X, but $re^{\mathbf{i}\theta}$ representation is not unique.

- Call arg(z)? — no, involves trigonometry

CropCircles

Bucketing Angles

We need to bucket angles $\angle AXB$ for all X, but $re^{\mathbf{i}\theta}$ representation is not unique.

- Call arg(z)? — no, involves trigonometry
- Scale to unit length?

CropCircles

Bucketing Angles

We need to bucket angles $\angle AXB$ for all X, but $re^{\mathbf{i}\theta}$ representation is not unique.

- Call arg(z)? — no, involves trigonometry
- Scale to unit length? — no, involves floating-point

CropCircles

Bucketing Angles

We need to bucket angles $\angle AXB$ for all X, but $re^{\mathbf{i}\theta}$ representation is not unique.

- Call arg(z)? — no, involves trigonometry
- Scale to unit length? — no, involves floating-point
- Divide out by GCD

CropCircles

Bucketing Angles

We need to bucket angles $\angle AXB$ for all X, but $re^{\mathbf{i}\theta}$ representation is not unique.

- Call $\arg(z)$? — no, involves trigonometry
- Scale to unit length? — no, involves floating-point
- Divide out by GCD — all integer!

CropCircles

Bucketing Angles

We need to bucket angles $\angle AXB$ for all X, but $re^{\mathbf{i}\theta}$ representation is not unique.

- Call arg(z)? — no, involves trigonometry
- Scale to unit length? — no, involves floating-point
- Divide out by GCD — all integer!
- Can also negate to reduce mod π

CropCircles

Sample Code

```
for (int i = 0; i < N; i++)
    for (int j = i + 1; j < N; j++) {
        set<pnt, Compare> seen;
        pnt A = pnts[i], B = pnts[j];
        for (int k = N - 1; k >= 0; k--) {
            pnt C = pnts[k];
            pnt diff = conj(A - C) * (B - C);
            if (diff.imag() == 0) continue;
            diff /= gcd(diff.real(), diff.imag());
            if (diff.imag() < 0) diff = -diff;
            if (k > j) seen.insert(diff); else seen.erase(diff);
        }
        ans += seen.size();
    }
```

Questions

?