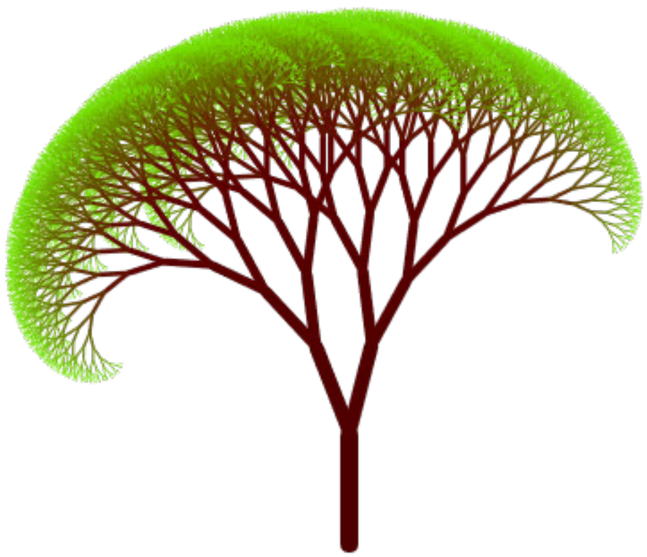
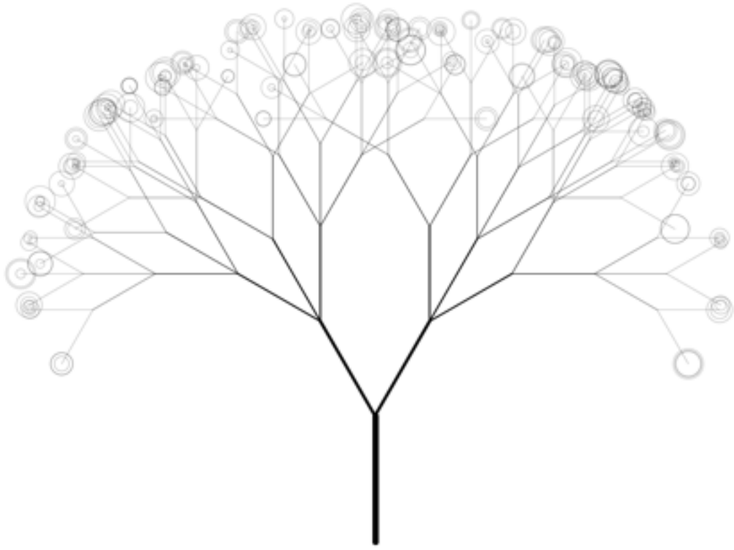
3D Transformations and Complex Representations

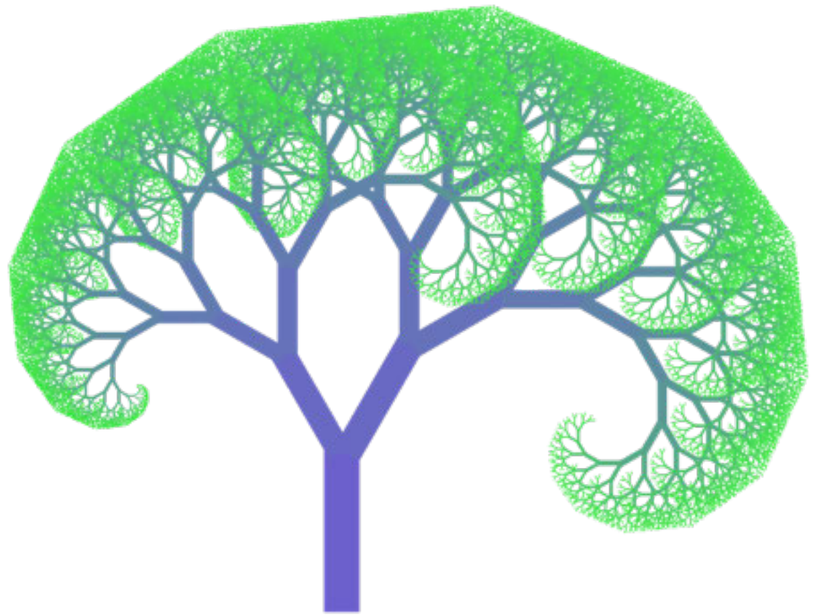
Computer Graphics
CMU 15-462/15-662, Fall 2016

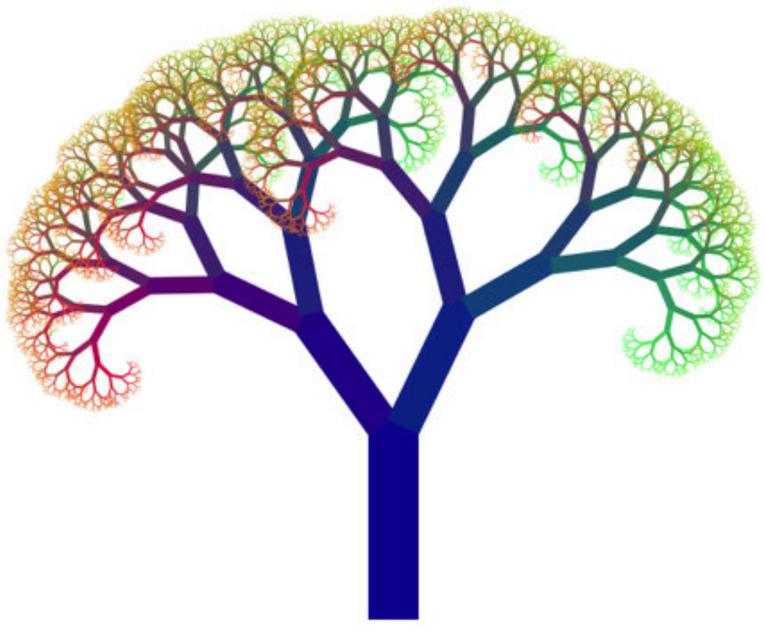
Quiz 4: Trees and Transformations

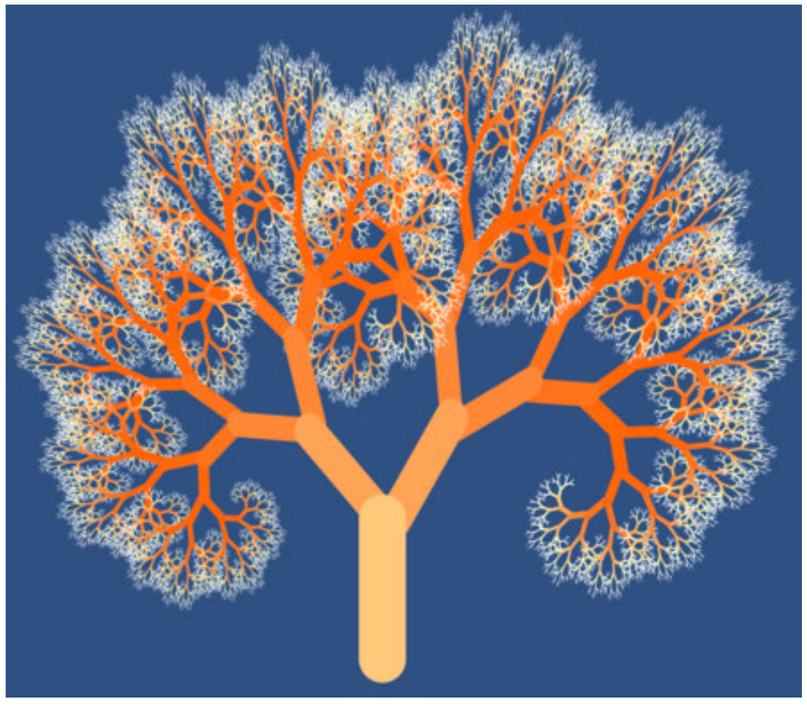
Student solutions (beautiful!):

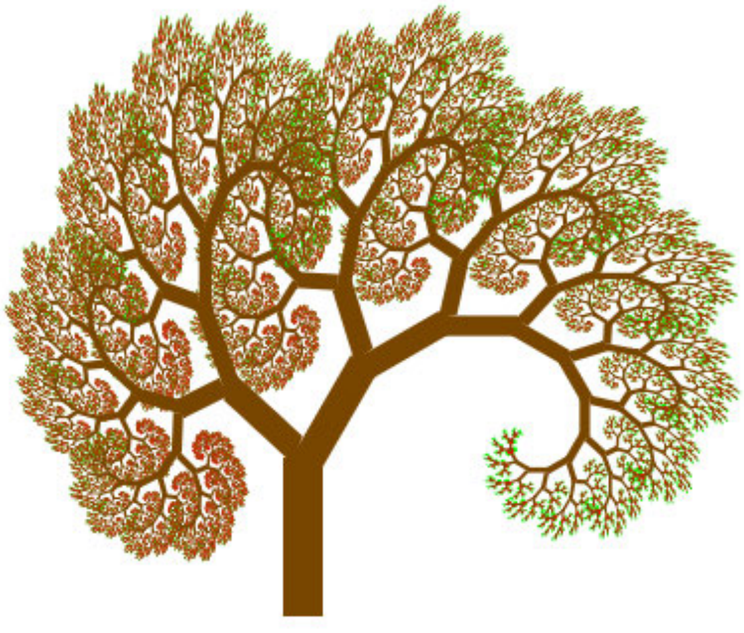






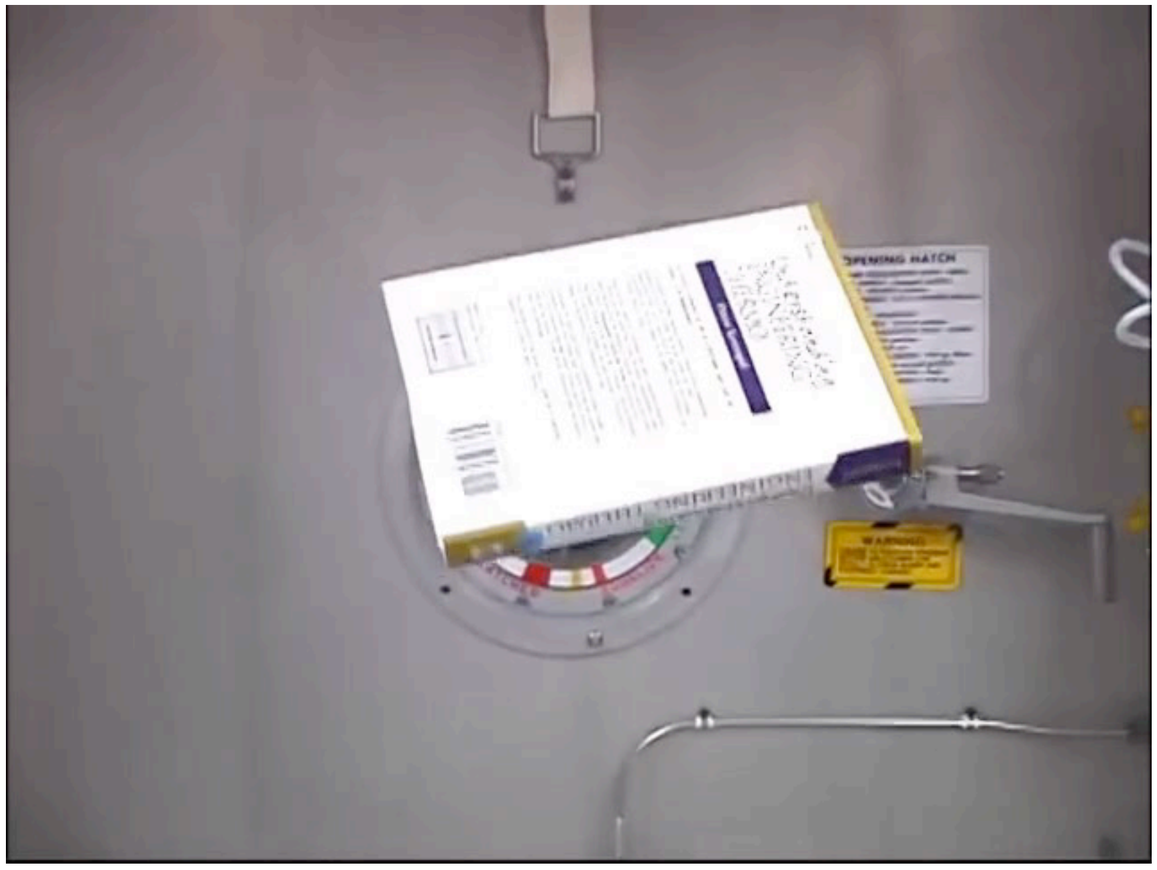






Rotations in 3D

- What is a rotation, intuitively?
- *How do you know a rotation when you see it?*
  - length/distance is preserved (no stretching/shearing)
  - lines get mapped to lines (linear)
  - orientation is preserved (e.g., text remains readable)



3D Rotations—Degrees of Freedom

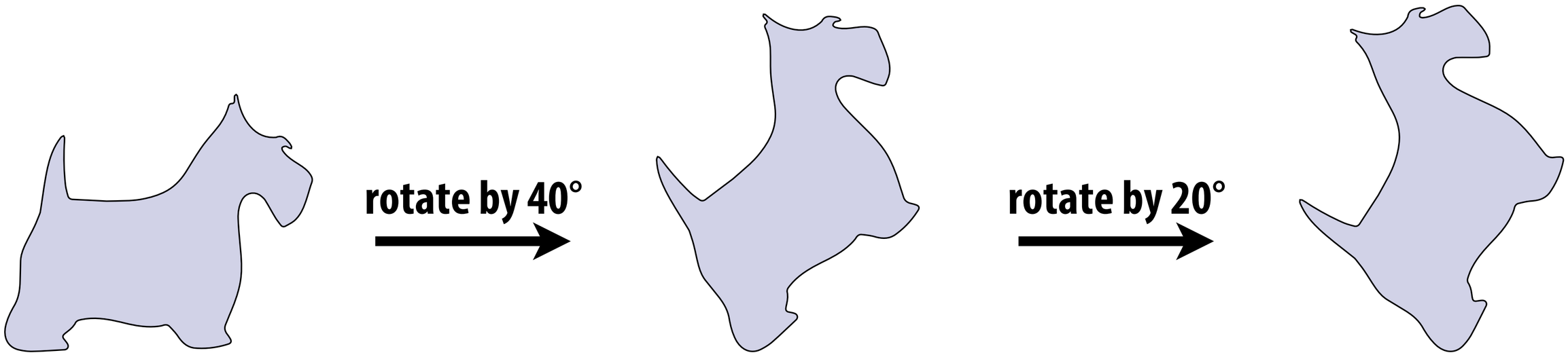
- How many numbers do we need to specify a rotation in 3D?
- For instance, we could use rotations around X, Y, Z. But do we *need* all three?
- Well, to rotate Pittsburgh to another city (say, São Paulo), we have to specify two numbers: latitude & longitude:
- Do we really need both latitude and longitude? Or will one suffice?
- Is that the *only* rotation from Pittsburgh to São Paulo? (How many more numbers do we need?)

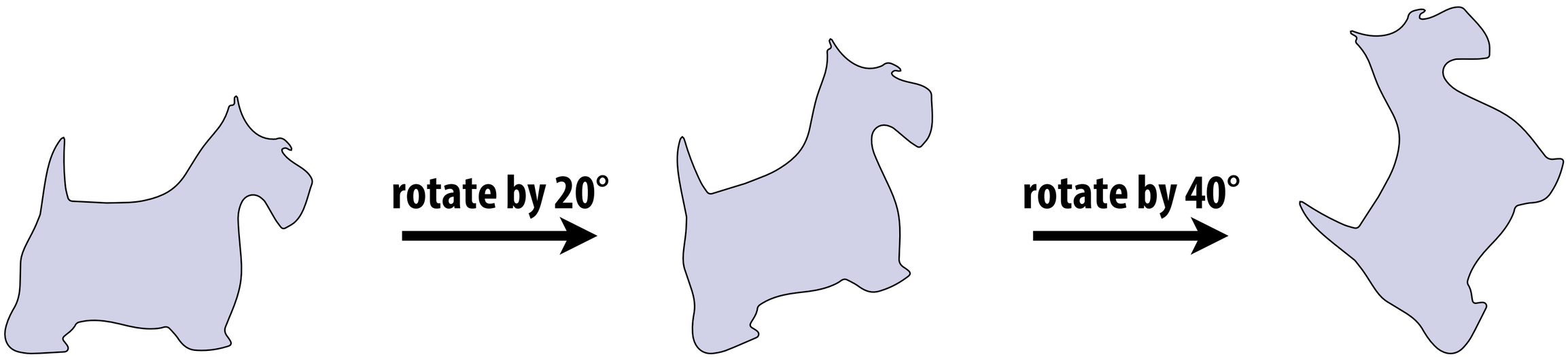
NO: We can keep São Paulo fixed as we rotate the globe.

Hence, we MUST have three degrees of freedom.

Commutativity of Rotations—2D

- In 2D, order of rotations doesn't matter:

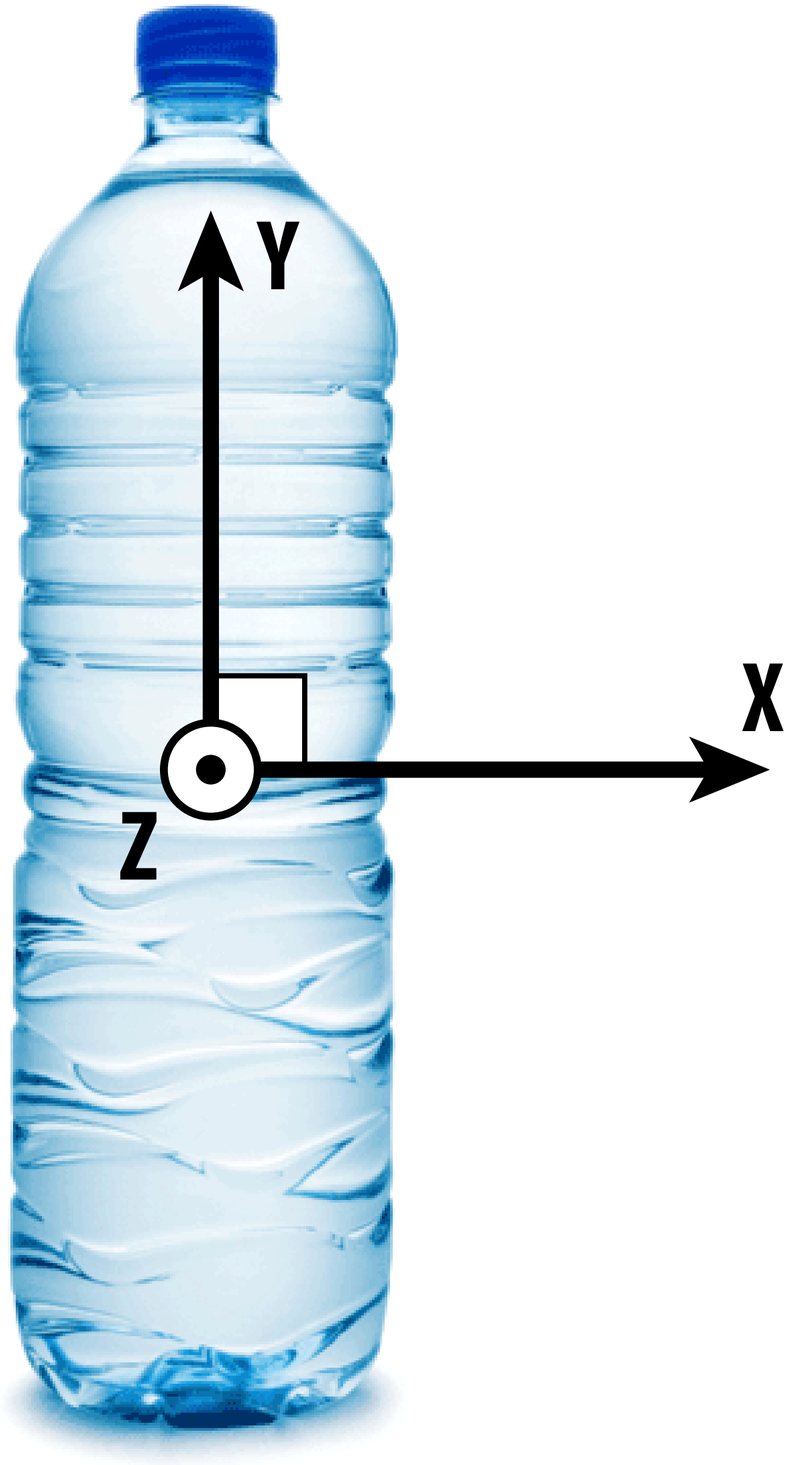




Why not?

Commutativity of Rotations—3D

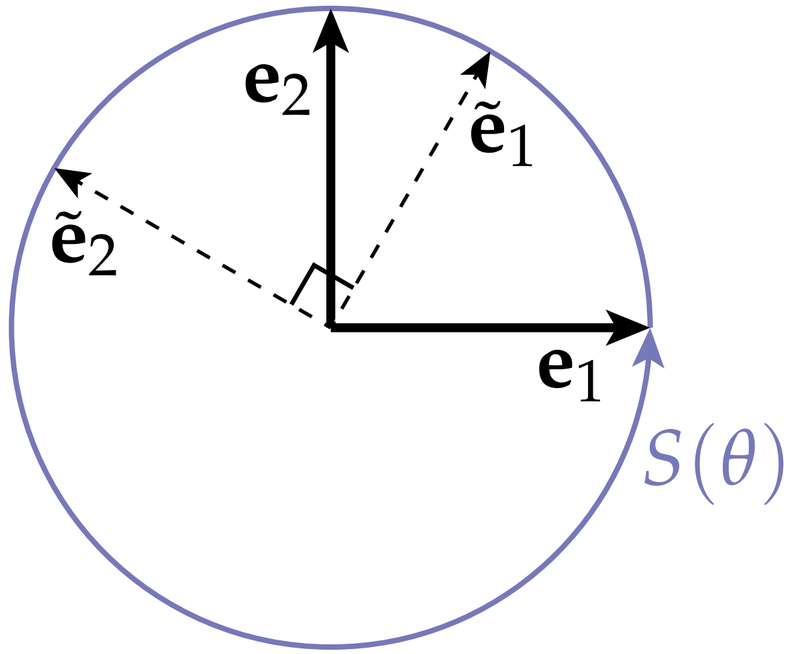
- What about in 3D?
- IN-CLASS ACTIVITY:
  - Rotate 90° around Y, then 90° around Z, then 90° around X
  - Rotate 90° around Z, then 90° around Y, then 90° around X
  - (Was there any difference?)



CONCLUSION: bad things can happen if we're not careful about the order in which we apply rotations!

Representing Rotations—2D

- First things first: how do we get a rotation matrix in 2D? (Don't just regurgitate the formula!)
- Suppose I have a function S(θ) that for a given angle θ gives me the point (x,y) around a circle (CCW).
  - Right now, I *do not care how this function is expressed!**
- What's e1 rotated by θ? $\tilde{\mathbf{e}}_1 = S(\theta)$
- What's e2 rotated by θ? $\tilde{\mathbf{e}}_2 = S(\theta + \pi/2)$
- How about $\mathbf{u} := a\mathbf{e}_1 + b\mathbf{e}_2$?



$$\mathbf{u} := aS(\theta) + bS(\theta + \pi/2)$$

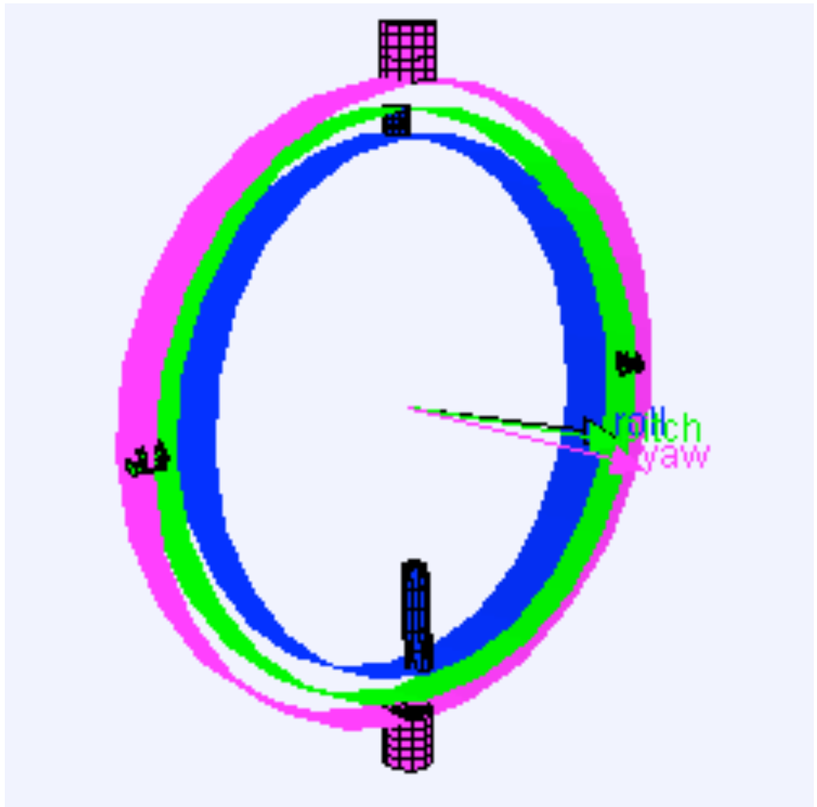
What then must the matrix look like?

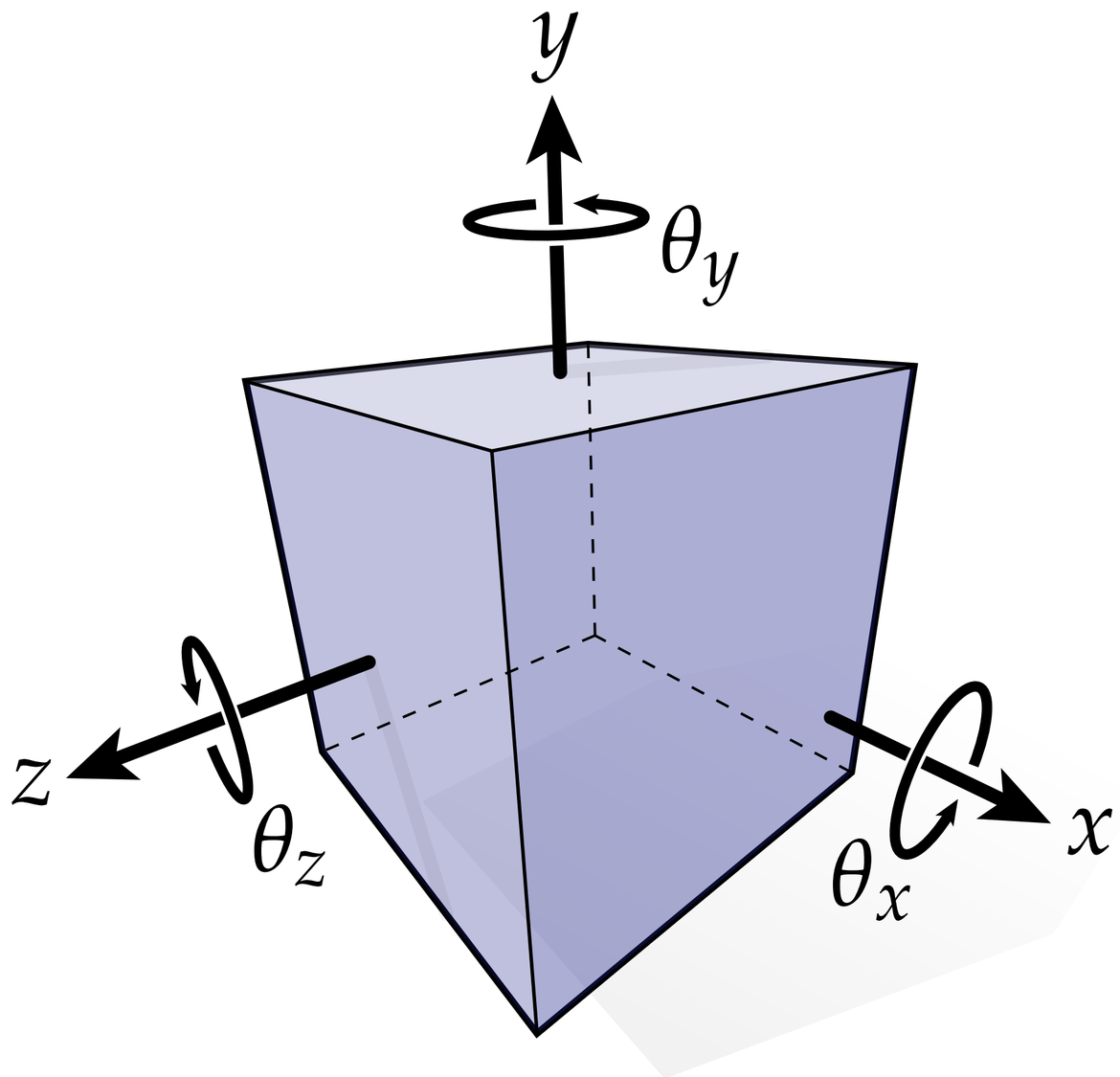
$$\begin{bmatrix} S(\theta) & S(\theta+\pi/2) \end{bmatrix} = \begin{bmatrix} \cos(\theta) & \cos(\theta+\pi/2) \\ \sin(\theta) & \sin(\theta+\pi/2) \end{bmatrix} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix}$$

*I.e., during most of life, NOTHING is gained by thinking in terms of sin(θ) and cos(θ).

Representing Rotations in 3D—Euler Angles

- How do we express rotations in 3D?
- One idea: we know how to do 2D rotations.
- Why not simply apply rotations around the three axes? (X,Y,Z)
- Scheme is called *Euler angles*
- PROBLEM: "Gimbal Lock"





Rotation from Axis/Angle

- **Alternatively, there is a general expression for a matrix that performs a rotation around a given axis u by a given angle θ:**

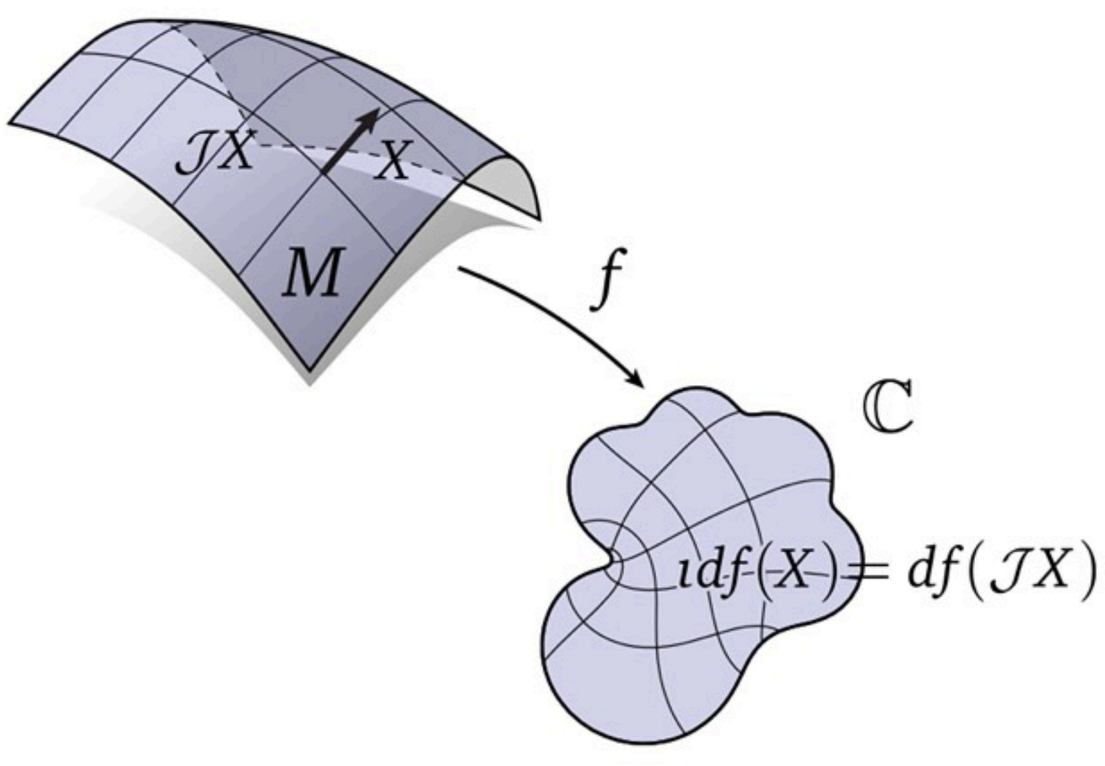
$$\begin{bmatrix} \cos\theta + u_x^2(1-\cos\theta) & u_x u_y(1-\cos\theta) - u_z\sin\theta & u_x u_z(1-\cos\theta) + u_y\sin\theta \\ u_y u_x(1-\cos\theta) + u_z\sin\theta & \cos\theta + u_y^2(1-\cos\theta) & u_y u_z(1-\cos\theta) - u_x\sin\theta \\ u_z u_x(1-\cos\theta) - u_y\sin\theta & u_z u_y(1-\cos\theta) + u_x\sin\theta & \cos\theta + u_z^2(1-\cos\theta) \end{bmatrix}$$

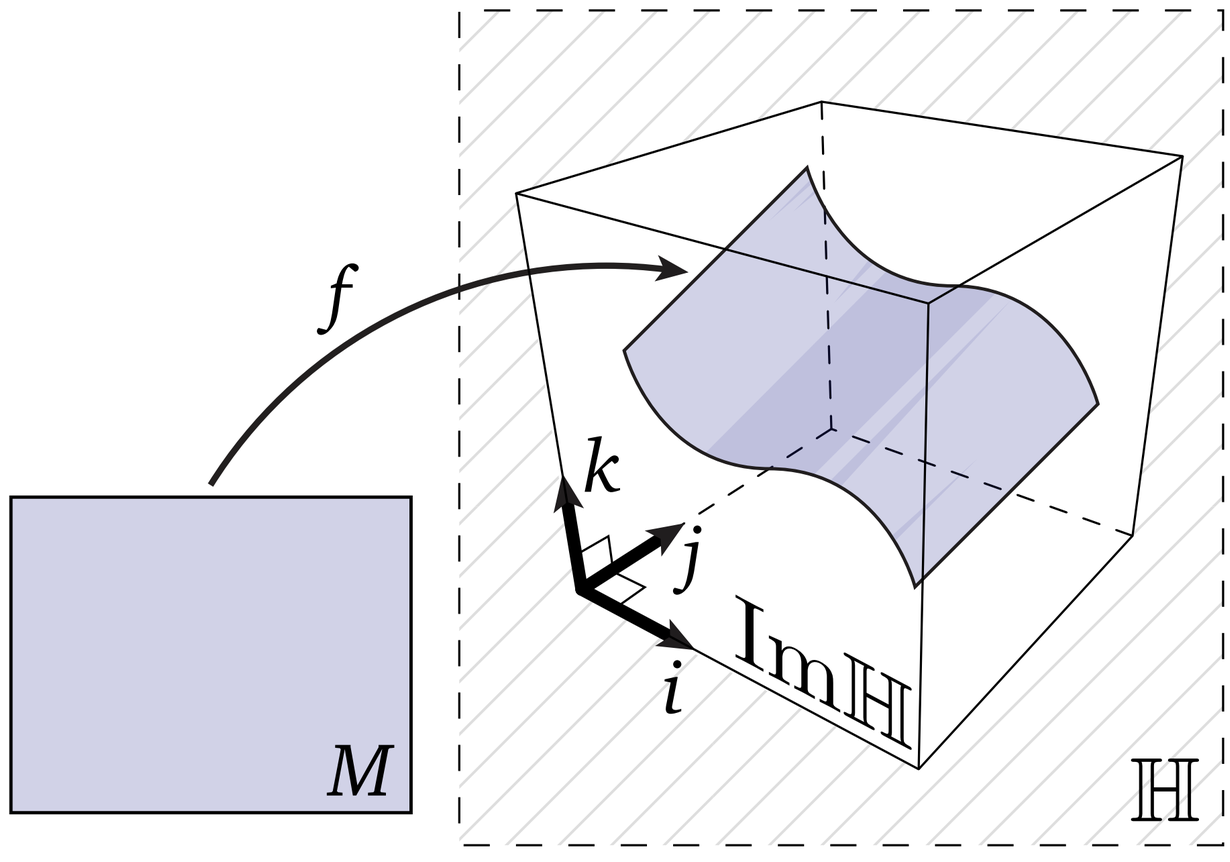
Just memorize this matrix! :-)

…we'll see a different way, later on.

Complex Analysis—Motivation

- Natural way to encode geometric transformations in 2D, 3D
- Simplifies notation / thinking / debugging
- *Moderate* reduction in computational cost/bandwidth/storage
- Fluency with complex analysis can lead into deeper/novel solutions to problems…





DON'T: Think of these numbers as "complex."

DO: Imagine we're simply defining additional operations (like dot and cross).

*A bit of an oversimplification, but go with it for now!

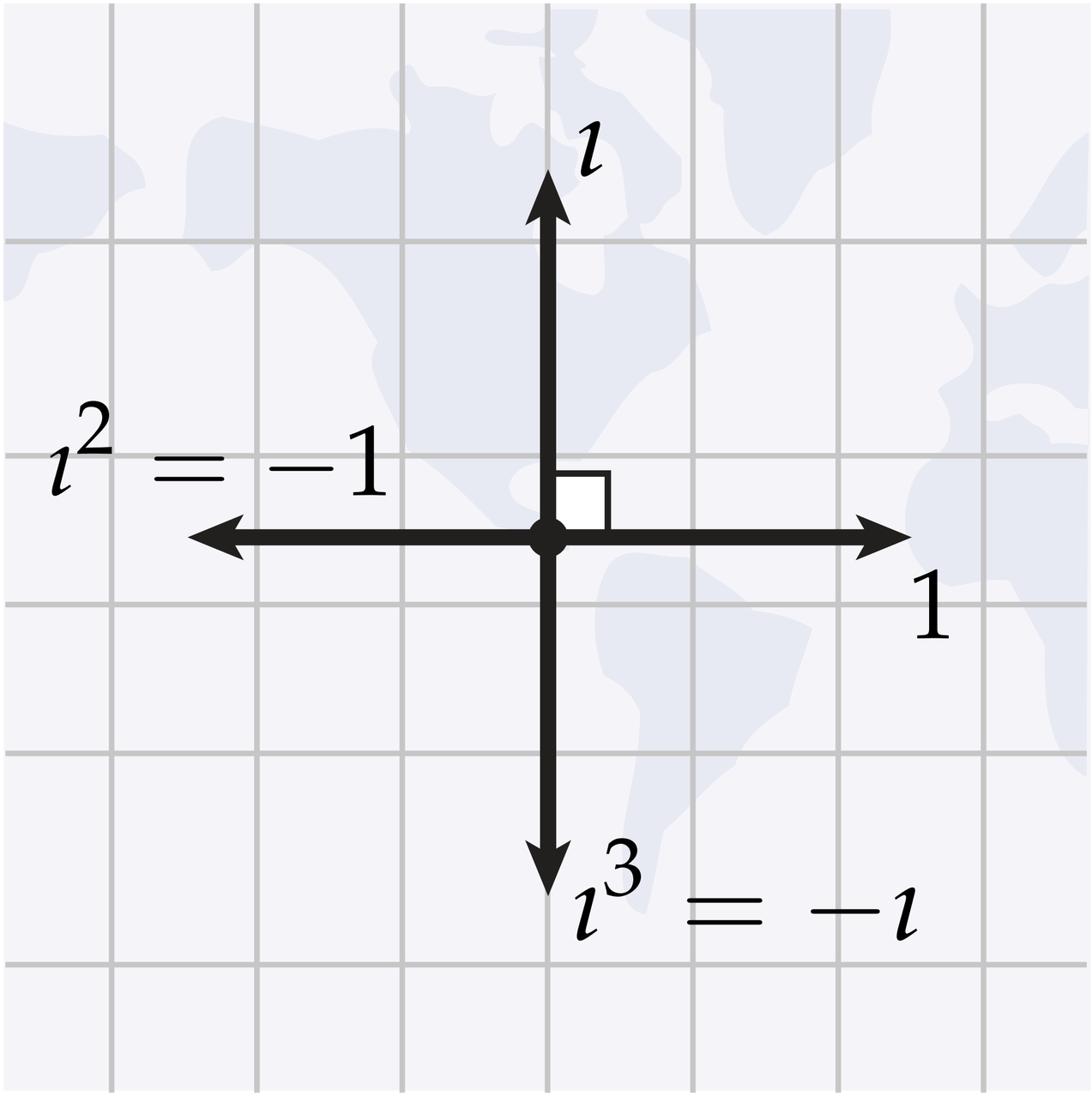
Imaginary Unit

$$i := \sqrt{-1}$$

nonsense!

More importantly: obscures geometric meaning.

Imaginary Unit—Geometric Description



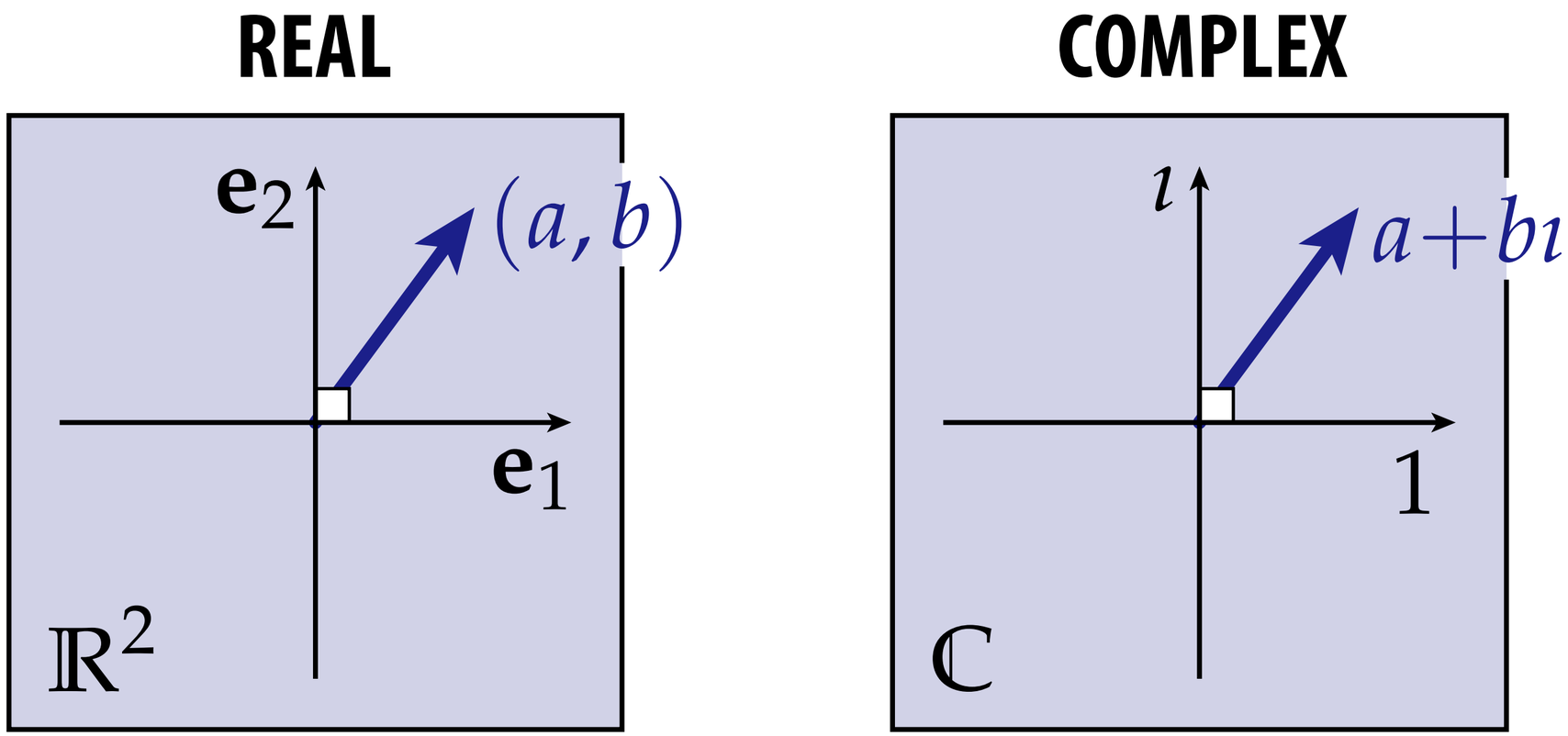
"iota*"

Symbol ɪ denotes quarter-turn in the counter-clockwise direction.

*Use ɪ instead of i to avoid confusion w/ indices i. (\imath in LaTeX)

Complex Numbers

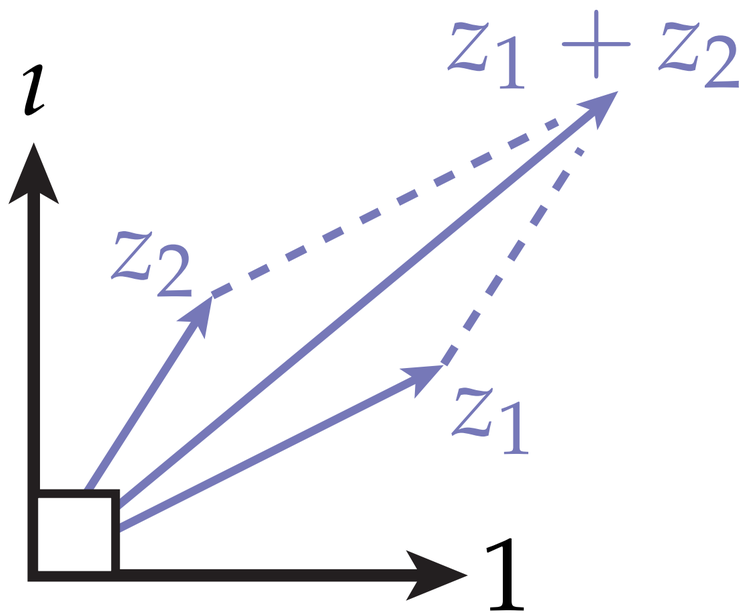
- Complex numbers are then just 2-vectors
- Instead of e_1,e_1, use "1" and "ι" to denote the two bases
- Otherwise, behaves exactly like a real 2-dimensional space



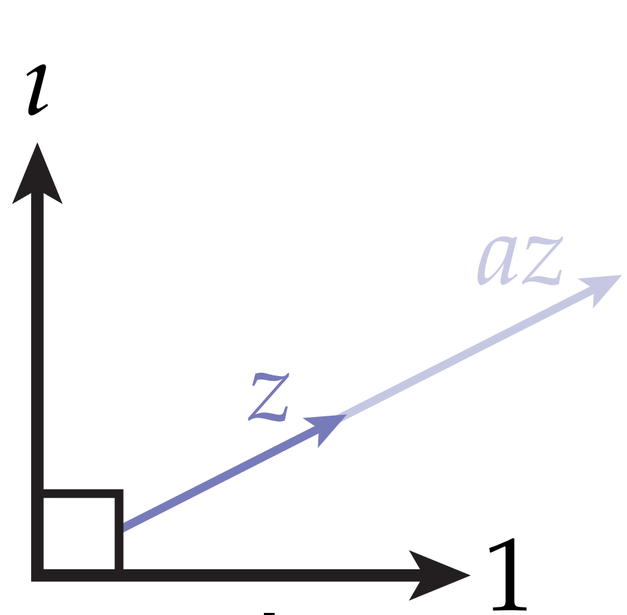
- …except that we're going to define a useful new notion of the product between two vectors.

Complex Arithmetic

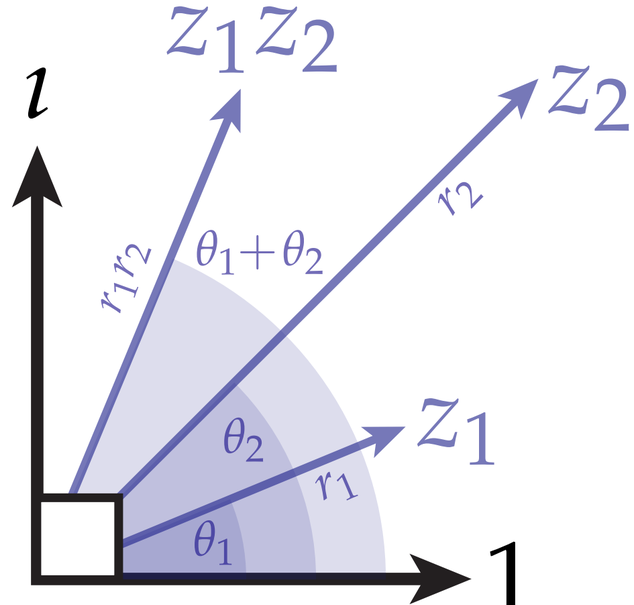
- Same operations as before, plus one more:



vector addition



scalar multiplication



complex multiplication

- Complex multiplication:
  - angles *add*
  - magnitudes *multiply*

"POLAR FORM"*:

$$z_1 := (r_1, \theta_1)$$
$$z_2 := (r_2, \theta_2)$$
$$z_1 z_2 = (r_1 r_2, \theta_1 + \theta_2)$$

have to be more careful here!

*Not *really* now it works, but useful geometric intuition.

Complex Product—Rectangular Form

- **Complex product in "rectangular" coordinates $(1, \imath)$:**

$$z_1 = (a + b\imath)$$

$$z_2 = (c + d\imath)$$

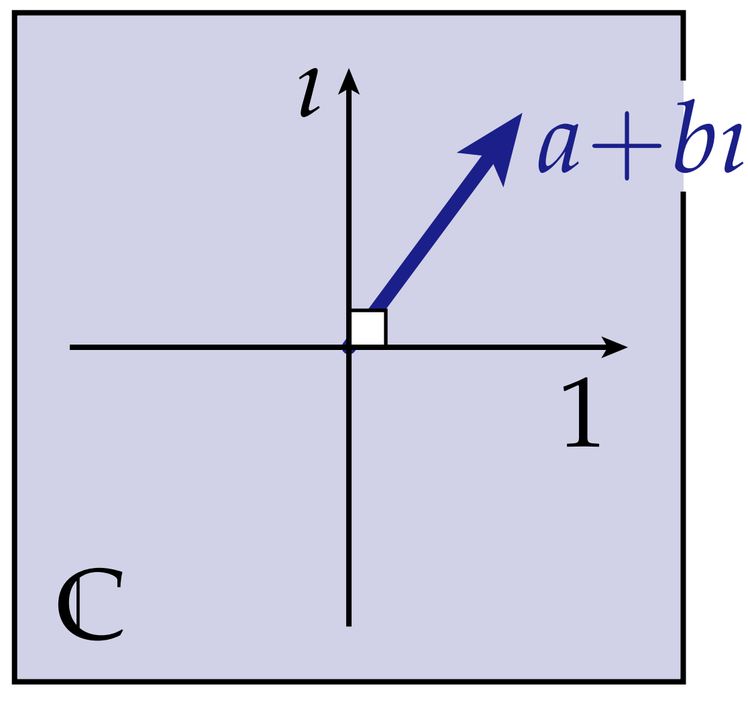
$$z_1 z_2 = ac + ad\imath + bc\imath + bd\imath^2 =$$

two quarter turns—same as -1

$$\boxed{(ac - bd) + (ad + bc)\imath.}$$

"real part" $\mathrm{Re}(z_1 z_2)$

"imaginary part" $\mathrm{Im}(z_1 z_2)$



- **We used a lot of "rules" here. Can you justify them geometrically?**
- **Does this product agree with our geometric description (last slide)?**

Complex Product—Polar Form

- **Perhaps most beautiful identity in math:**

$$e^{\imath\pi} + 1 = 0$$

- **Specialization of *Euler's formula*:**

$$e^{\imath\theta} = \cos(\theta) + \imath \sin(\theta)$$

- **Can use to "implement" complex product:**

$$z_1 = ae^{\imath\theta}, \quad z_2 = be^{\imath\phi}$$

$$z_1 z_2 = abe^{\imath(\theta+\phi)}$$

(as with real exponentiation, exponents *add)*

Leonhard Euler (1707–1783)

- **Most prolific mathematician of all time**
- **Opera Omnia—1 vol./yr. starting 1911**
- **Still going! Now ~75 vols., 25k pages**
- **228 papers posthumously**
- **Many later works while blind**
- **(Work was also *good*...)**

[source: William Dunham]

Q: How does this operation differ from our earlier, "fake" polar multiplication?

2D Rotations: Matrices vs. Complex

- **Suppose we want to rotate a vector u by an angle θ, then by an angle ϕ.**

REAL / RECTANGULAR

$$\mathbf{u} = (x, y)$$

$$\mathbf{A} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$$

$$\mathbf{B} = \begin{bmatrix} \cos\phi & -\sin\phi \\ \sin\phi & \cos\phi \end{bmatrix}$$

$$\mathbf{Au} = \begin{bmatrix} x\cos\theta - y\sin\theta \\ x\sin\theta + y\cos\theta \end{bmatrix}$$

$$\mathbf{BAu} = \begin{bmatrix} (x\cos\theta - y\sin\theta)\cos\phi - (x\sin\theta + y\cos\theta)\sin\phi \\ (x\cos\theta - y\sin\theta)\sin\phi + (x\sin\theta + y\cos\theta)\cos\phi \end{bmatrix}$$

$$= \cdots \text{some trigonometry} \cdots =$$

$$\mathbf{BAu} = \begin{bmatrix} x\cos(\theta+\phi) - y\sin(\theta+\phi) \\ x\sin(\theta+\phi) + y\cos(\theta+\phi) \end{bmatrix}.$$

(…and simplification is not always this obvious.)

COMPLEX / POLAR

$$u = re^{\imath\alpha}$$

$$a = e^{\imath\theta}$$

$$b = e^{\imath\phi}$$

$$abu = re^{\imath(\alpha+\theta+\phi)}.$$

Or if we want rectangular coords:

$$= r\begin{bmatrix} \cos(\alpha+\theta+\phi) \\ \sin(\alpha+\theta+\phi) \end{bmatrix}$$

Pervasive theme in graphics:

Sure, there are often many "equivalent" representations.

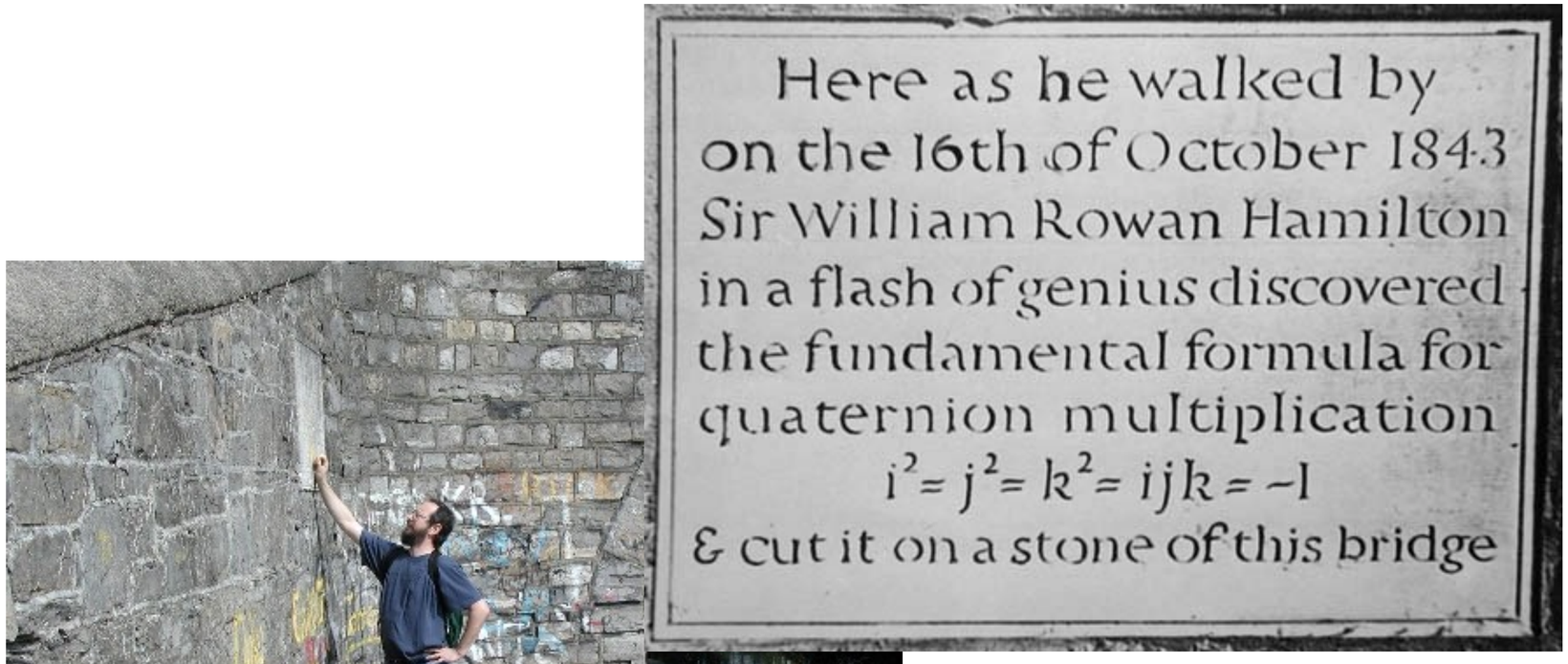
…But why not choose the one that makes life easiest*?

*Or most efficient, or most accurate…

Quaternions

- TLDR: Kind of like complex numbers but for 3D rotations
- **Weird situation:** can't do 3D rotations w/ only 3 components!

William Rowan Hamilton
(1805-1865)



(Not Hamilton)

Quaternions in Coordinates

- **Hamilton's insight: in order to do 3D rotations in a way that mimics complex numbers for 2D, actually need FOUR coords.**
- **One real, *three* imaginary:**

$$\mathbb{H} := \text{span}(\{1, \imath, \jmath, k\})$$

"H" is for *Hamilton!*

$$q = a + b\imath + c\jmath + dk \in \mathbb{H}$$

- **Quaternion product determined by**

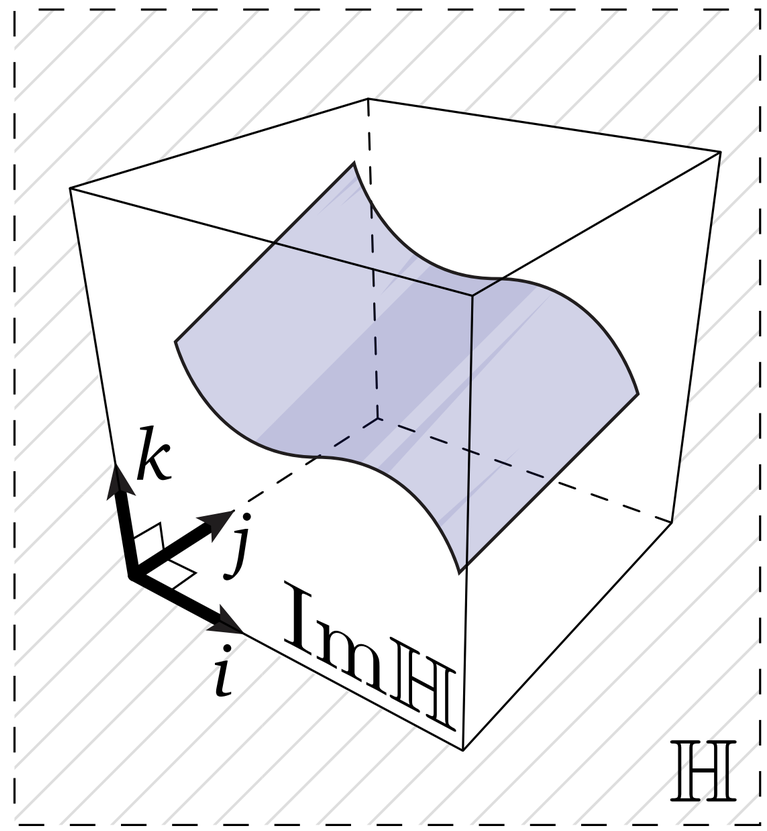
$$\imath^2 = \jmath^2 = k^2 = \imath\jmath k = -1$$

together w/ "natural" rules (distributivity, associativity, etc.)

- **WARNING: product no longer commutes!**

$$\text{For } q, p \in \mathbb{H}, \quad qp \neq pq$$

(Will understand this *a lot* better when we study transformations.)



Quaternion Product in Components

- **Given two quaternions**

$$q = a_1 + b_1 i + c_1 j + d_1 k$$
$$p = a_2 + b_2 i + c_2 j + d_2 k$$

- **Can express their product as**

$$\begin{aligned} qp = {} & a_1 a_2 - b_1 b_2 - c_1 c_2 - d_1 d_2 \\ & + (a_1 b_2 + b_1 a_2 + c_1 d_2 - d_1 c_2) i \\ & + (a_1 c_2 - b_1 d_2 + c_1 a_2 + d_1 b_2) j \\ & + (a_1 d_2 + b_1 c_2 - c_1 b_2 + d_1 a_2) k \end{aligned}$$

…fortunately there is a (much) nicer expression.

Quaternions—Scalar + Vector Form

- If we have *four* components, how do we talk about pts in 3D?
- Natural idea: we have three imaginary parts—why not use these to encode 3D vectors?

$$(x, y, z) \mapsto 0 + x\imath + yj + zk$$

- Alternatively, can think of a quaternion as a pair

$$\begin{array}{ccc} (\text{ scalar}, & \text{vector }) & \in \mathbb{H} \\ \cap & \cap & \\ \mathbb{R} & \mathbb{R}^3 & \end{array}$$

- Quaternion product then has simple(r) form:

$$(a, \mathbf{u})(b, \mathbf{v}) = (ab - \mathbf{u} \cdot \mathbf{v}, a\mathbf{u} + b\mathbf{v} + \mathbf{u} \times \mathbf{v})$$

- For vectors in R3, gets even simpler:

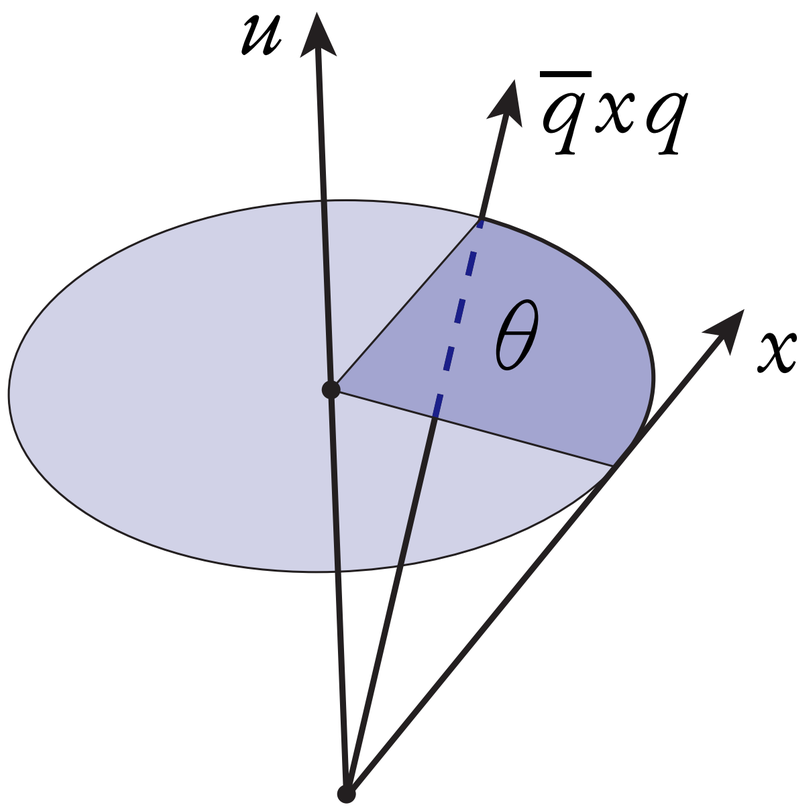
$$\mathbf{uv} = \mathbf{u} \times \mathbf{v} - \mathbf{u} \cdot \mathbf{v}$$

3D Transformations via Quaternions

- **Main use for quaternions in graphics? *Rotations.***
- **Consider vector x ("pure imaginary") and *unit* quaternion q:**

$$x \in \mathrm{Im}(\mathbb{H})$$

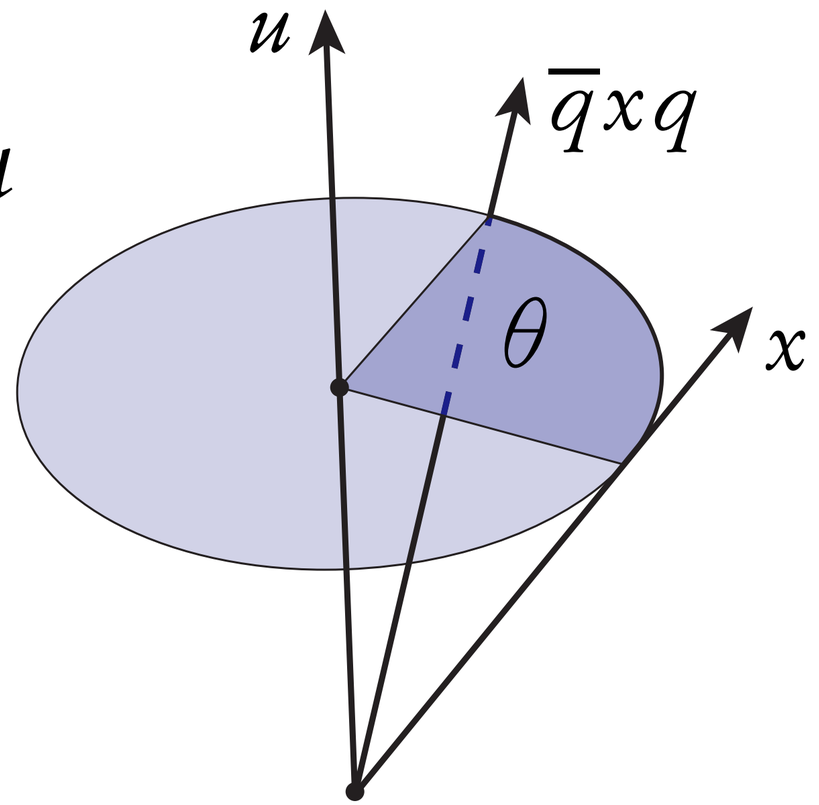
$$q \in \mathbb{H}, \quad |q|^2 = 1$$



Rotation from Axis/Angle, Revisited

- Given axis u, angle θ, quaternion q representing rotation is

$$q = \cos(\theta/2) + \sin(\theta/2)u$$

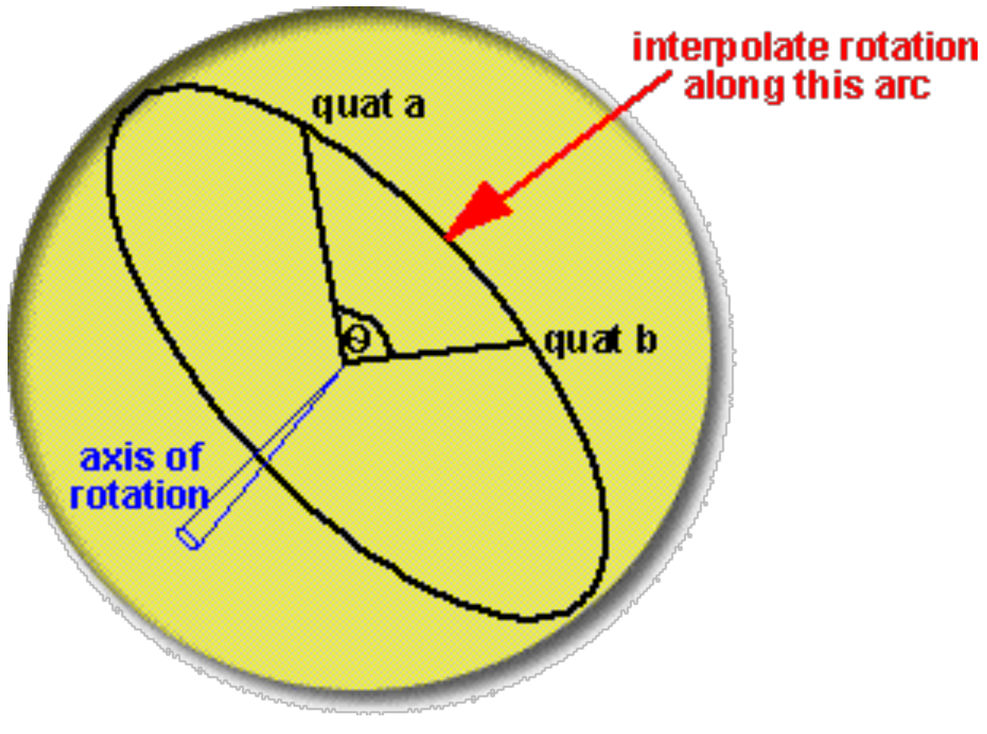


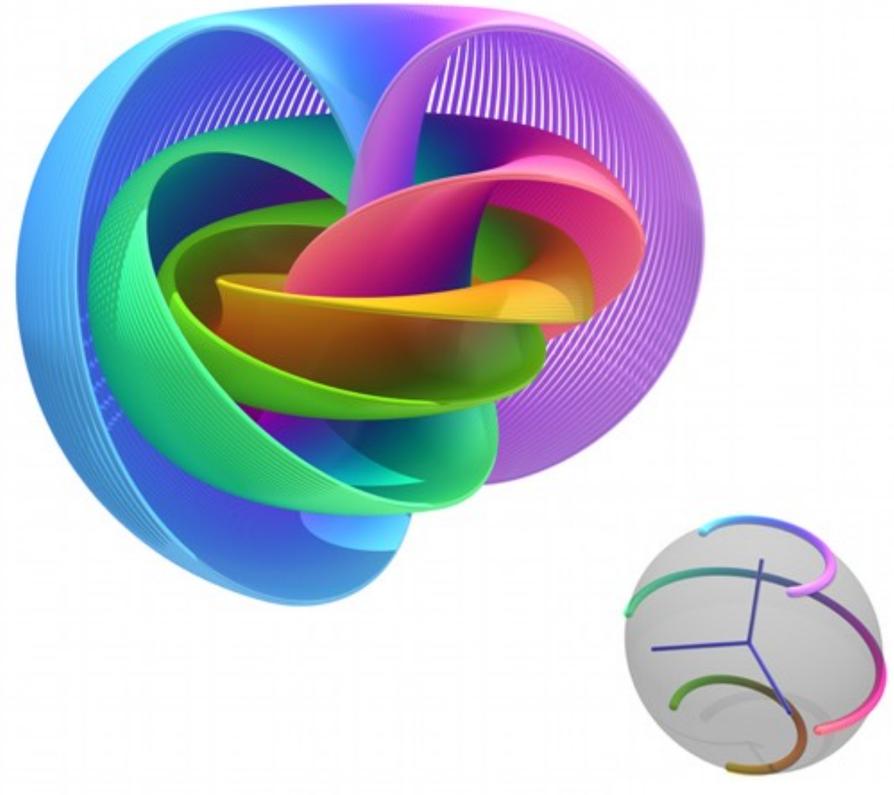
- Slightly easier to remember (and manipulate) than matrix:

$$\begin{bmatrix} \cos\theta + u_x^2(1-\cos\theta) & u_x u_y(1-\cos\theta) - u_z\sin\theta & u_x u_z(1-\cos\theta) + u_y\sin\theta \\ u_y u_x(1-\cos\theta) + u_z\sin\theta & \cos\theta + u_y^2(1-\cos\theta) & u_y u_z(1-\cos\theta) - u_x\sin\theta \\ u_z u_x(1-\cos\theta) - u_y\sin\theta & u_z u_y(1-\cos\theta) + u_x\sin\theta & \cos\theta + u_z^2(1-\cos\theta) \end{bmatrix}$$

More Quaternions and Rotation

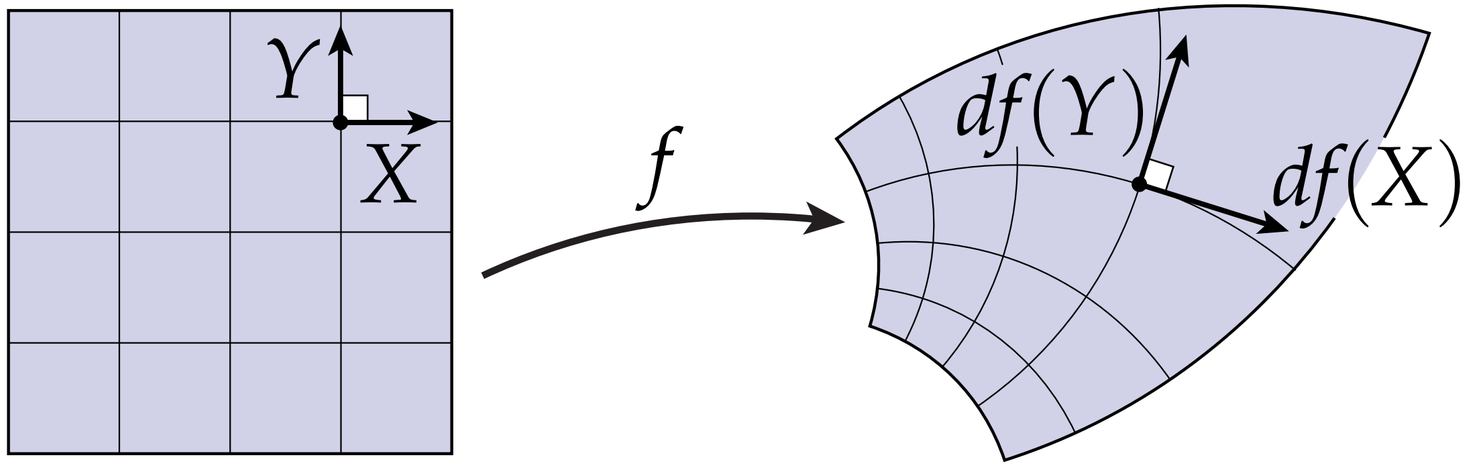
- Don't have time to cover everything, but…
- Quaternions provide some very nice utility/perspective when it comes to rotations:
  - Spherical linear interpolation ("slerp")
  - *Hopf fibration / "belt trick"*
  - …

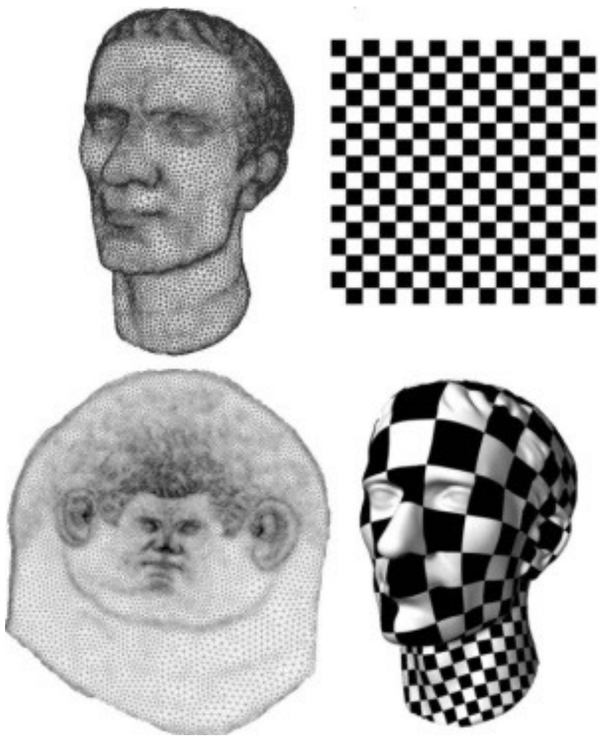


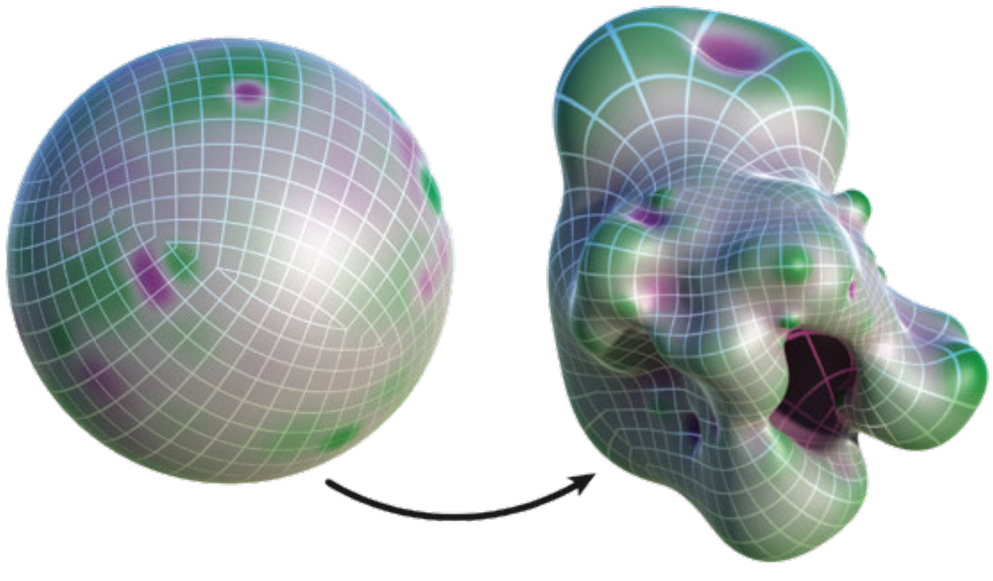


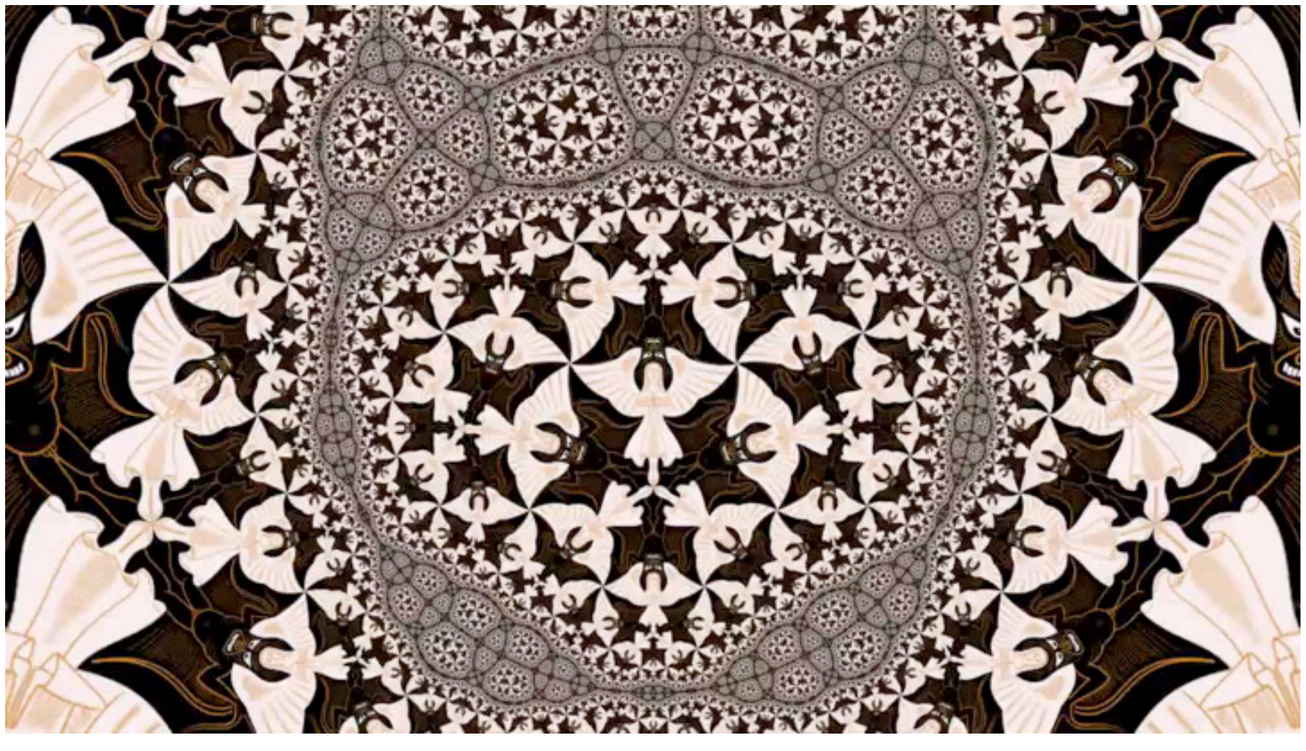
Where else are (hyper-)complex numbers useful in computer graphics?

Complex #s: Language of *Conformal Maps*



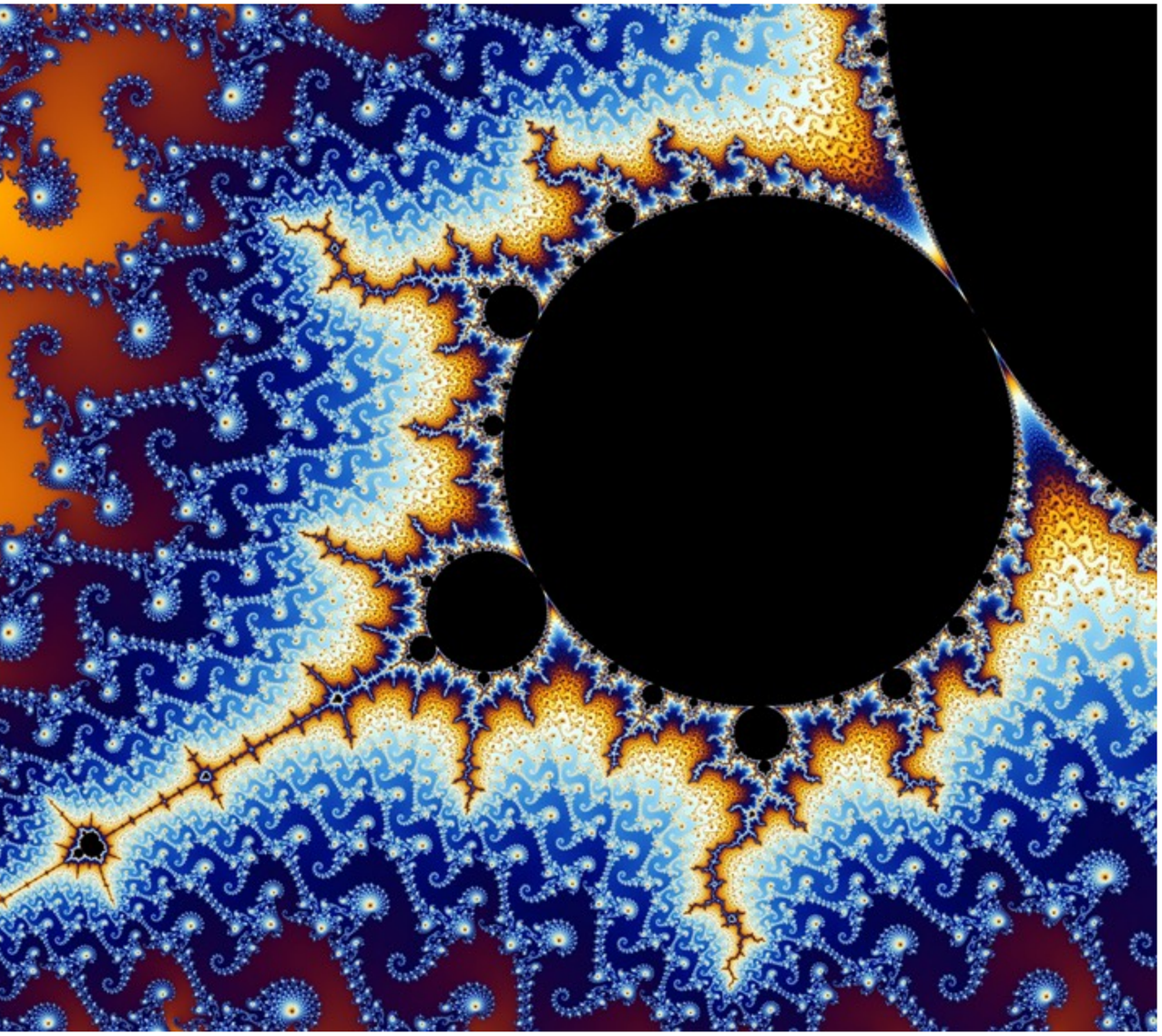


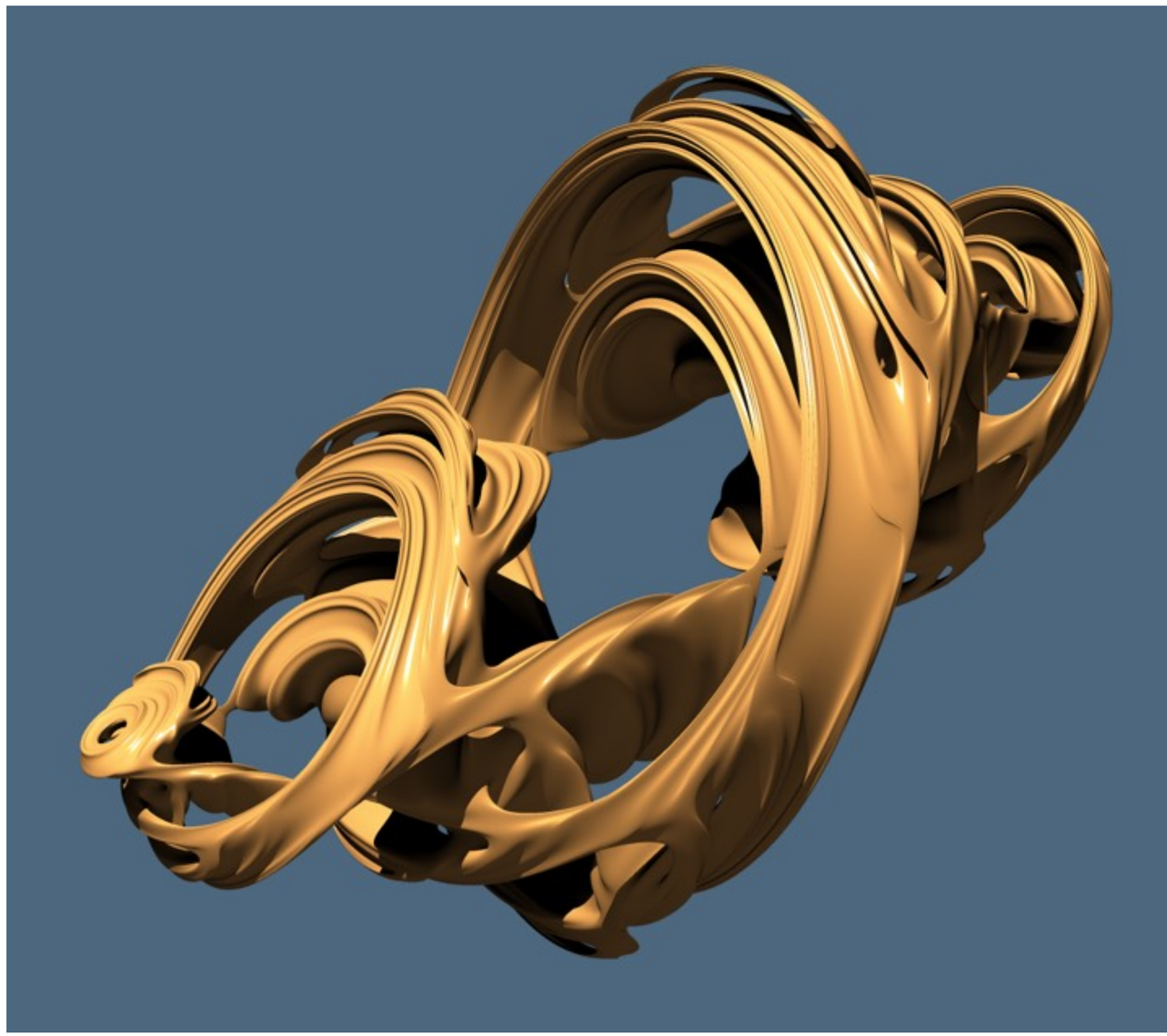




Useless-But-Beautiful Example: Fractals

- Defined in terms of iteration on (hyper)complex numbers:





(Will see exactly how this works later in class.)