Computer Graphics

Sources

- Fundamentals of Computer Graphics Fourth Edition. Tiger Book
- An Integrated Introduction to Computer Graphics and Geometric Modeling

Mass-Points

- mass-point: a point P in affine space together with a nonzero mass m.
- denote a mass-point by the pair (mP, m). Here m is the mass and $P = mP = m$ is the point; mP by itself has no meaning.
- Vectors reside in this space as objects with zero mass, we write $(v, 0)$.
- Sum of two points: sum of mass located at center of mass(!not 3 vals).

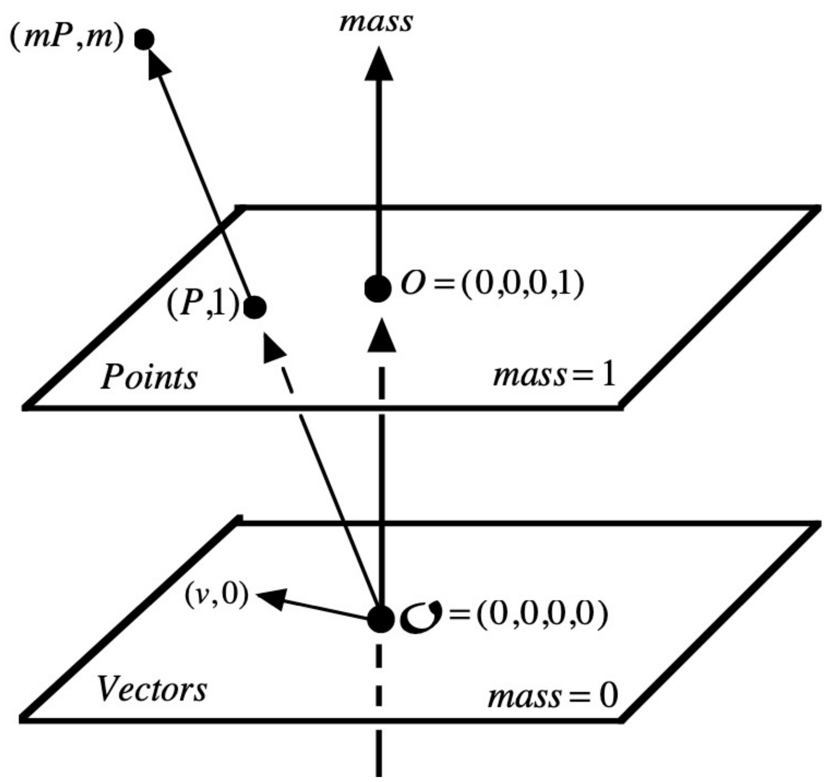
$$(m_1P_1, m_1) + (m_2P_2, m_2) = (m_1P_1 + m_2P_2, m_1 + m_2)$$

Quaternions

- Division algebras: 1, 2, 4. (8)
  -> line(real), plane(complex), space of mass-points(quaternions)

- denote this special mass-point (identity for quaternion multiplication) by *O,* identify this mass-point with the point located at the origin, in the 3-dimensional affine space of affine points. Vectors lie in the quaternion subspace orthogonal to *O*.

$$(mP, m) = mO + v\,.$$



Interpreted in terms of coordinates, Equation (3.1) means that every quaternion q can be written as $q=(q_1,q_2,q_3,q_4)$ or equivalently

$$q=q_4O+q_1i+q_2j+q_3k\,.$$

In many texts, the letter O is dropped, and quaternions are written as

$$q=q_4+q_1i+q_2j+q_3k\,.$$

Quaternion multiplication is an extension of complex multiplication. To extend complex multiplication to the quaternions, we need only extend complex multiplication to the 4-dimensional basis O,i,j,k. The point O is the identity for quaternion multiplication, so

$$O^2=O. \qquad Oi=i=iO \qquad Oj=j=jO \qquad Ok=k=kO \tag{3.4}$$

Multiplication on the basis vectors i,j,k is defined by the following rules:

$$i^2=j^2=k^2=-O \quad \text{and} \quad ij=k=-ji,\ \ jk=i=-kj,\ \ ki=j=-ik\,. \tag{3.5}$$

$$
\begin{aligned}
uv &= (u_1 i + u_2 j + u_3 k)(v_1 i + v_2 j + v_3 k) \\
&= u_1 i(v_1 i + v_2 j + v_3 k) + u_2 j(v_1 i + v_2 j + v_3 k) + u_3 k(v_1 i + v_2 j + v_3 k) \\
&= -(u_1 v_2 + u_2 v_2 + u_3 v_3)O + (u_1 v_2 - u_2 v_1)k + (u_3 v_1 - u_1 v_3)j + (u_2 v_3 - u_3 v_3)i
\end{aligned}
$$

or equivalently

$$uv = -(u \cdot v)O + u \times v \tag{3.6}$$

More generally for two arbitrary quaternions $aO + u$ and $bO + v$

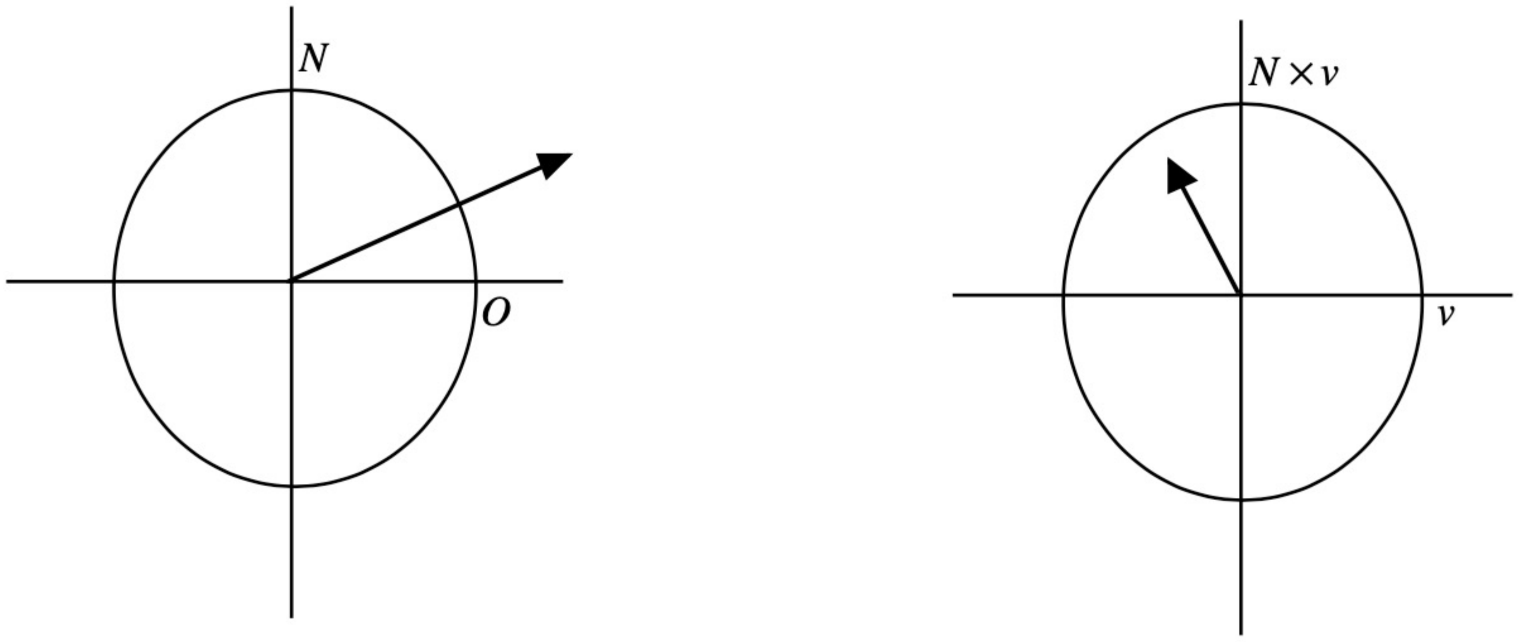
: $$(aO + u)(bO + v) = aO(bO + v) + u(bO + v) = abO + av + bu + uv$$

so substituting the right hand side of Equation (3.6) for uv , we find that

$$(aO + u)(bO + v) = (ab - u \cdot v)O + (av + bu + u \times v) \tag{3.7}$$

(3.7) general form, most useful!
Neither commutative nor anticommutative

Mutually Orthogonal Planes in 4-D

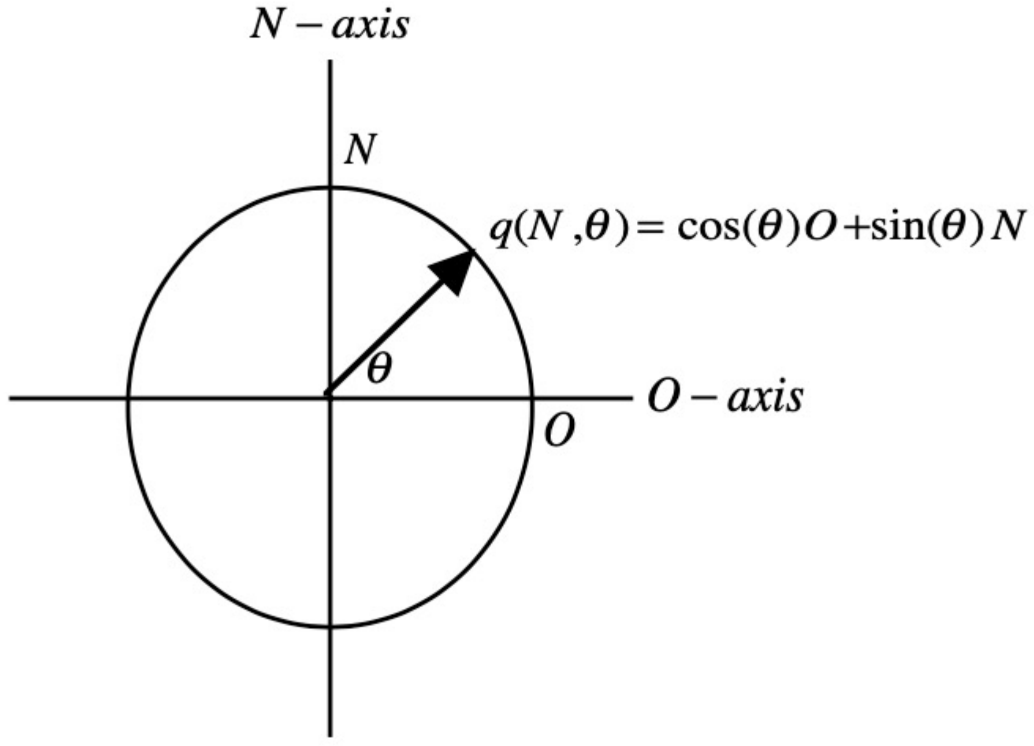


(a) the plane of O, N *(b) the plane perpendicular to O, N*

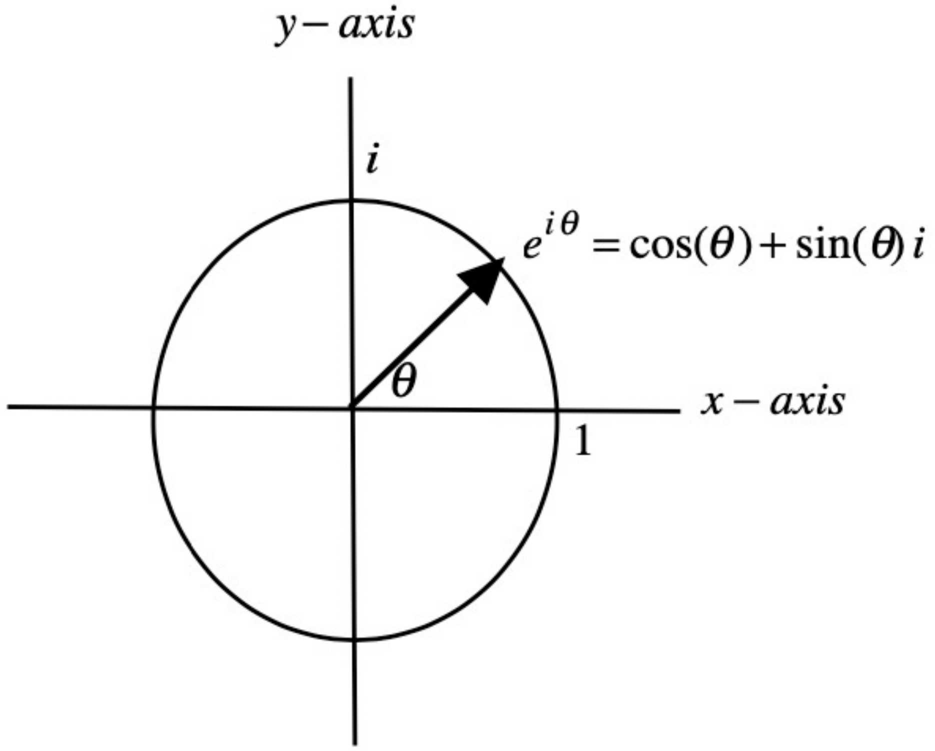
Figure 3: A mass-point represented by its projections (arrows) into two mutually orthogonal planes in 4-dimensions.

Just think of *O, i, j, k*, (a) complex plane. line in 3d (b) orthogonal plane. vec in 3d (maybe actually just multiply by v.

Geometry of Quaternion Multiplication



(a) the quaternion plane of O, N



(b) the complex plane

Rotation in O, N

Let

- $N =$ *a unit vector*
- $p =$ *a quaternion in the plane of O, N*

Then

i. $L_{q(N,\theta)}(p) = q(N,\theta)\, p$ *rotates p by the angle* θ *in the plane of O, N*

ii. $R_{q(N,\theta)}(p) = p\, q(N,\theta)$ *rotates p by the angle* θ *in the plane of O, N*

iii. $R_{q^*(N,\theta)} p = p\, q^*(N,\theta)$ *rotates p by the angle* $-\theta$ *in the plane of O, N*

iv. $S_{q(N,\theta)}(p) = q(N,\, \theta)\; p\; q^*(N,\, \theta)$ *is the identity on p.*

Rotation in O, N perpendicular

Let

- $N = a\ unit\ vector$
- $v = a\ vector\ in\ the\ plane \perp\ O, N$

Then

i. $L_{q(N,\theta)}(v) = q(N,\theta)\, v$ *rotates v by the angle* θ *in the plane* $\perp$ *O, N*

ii. $R_{q(N,\theta)}(v) = v\, q(N,\theta)$ *rotates v by the angle* $-\theta$ *in the plane* $\perp$ *O, N*

iii. $R_{q^*(N,\theta)}(v) = v\, q^*(N,\theta)$ *rotates v by the angle* θ *in the plane* $\perp$ *O, N*

iv. $S_{q(N,\theta)}(v) = q(N,\theta)\, v\, q^*(N,\theta)$ *rotates v by the angle* 2θ *in the plane* $\perp$ *O, N.*

Rotation with Sandwich

Proposition 3.4: *Let $q(N,\phi) = \cos(\phi/2)O + \sin(\phi/2)N$. Then a vector v can be rotated around the axis N through the angle ϕ by sandwiching v between $q(N,\phi)$ and $q(N,\phi)^*$.*

Proof: Let $v = v_{||} + v_{\perp}$, where

- $v_{||} =$ component of v parallel to N
- $v_{\perp} =$ component of v perpendicular to N.

Then

- $S_{q(N,\theta/2)}(v_{||}) = v_{||}$ (Proposition 3.2)
- $S_{q(N\theta/2)}(v_{\perp})$ *rotates* $v_{\perp}$ *by the angle* θ *in the plane* $\perp$ *to* O, N. (Proposition 3.3)

Therefore $S_{q(N,\theta/2)}(v)$ has the same effect on the components of v as rotating v in 3-dimensions by the angle θ around the axis vector N (see Chapter 12, Section 12.3.2).

Mirror with Sandwich

Thus to rotate v by 180^0 around N, we can sandwich v between N and N^*. Since for vectors $N^* = -N$, it follows that mirror image of v in the plane perpendicular to N is given by

$$-S_N(v) = N v N .$$

Thus we are led to the following result.

Proposition 3.5: *A vector v can be mirrored in the plane perpendicular to the unit normal N by sandwiching v between two copies of N.*

Proof: Let $v = v_{||} + v_{\perp}$, where

- $v_{||} =$ component of v parallel to N
- $v_{\perp} =$ component of v perpendicular to N.

Since $N = \cos(\pi/2)O + \sin(\pi/2)N = q(N, \pi/2)$:

- $-S_N(v_{||}) = -S_{q(N,\pi/2)}(v_{||}) = -v_{||}$ (Proposition 3.2)
- $-S_N(v_{\perp}) = -S_{q(N,\pi/2)}(v_{\perp}) = v_{\perp}$ (Proposition 3.3)

Other conformal transformations

Proposition 3.6: *A vector v can be scaled by the factor c by sandwiching v between two copies of $\sqrt{c}\,O$.*

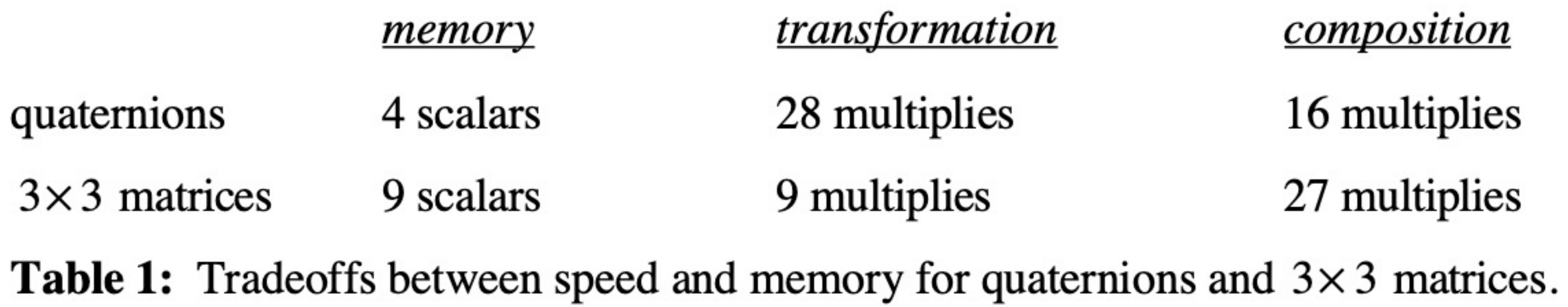
Proof: $S_{\sqrt{c}\,O}(v) = \sqrt{c}\,O\,v\,\sqrt{c}\,O = c\,v$.

Theorem 3.1: *Every conformal transformation of vectors in 3-dimensions can be modeled by sandwiching with a single quaternion.*

Proof: This result follows from Propositions 3.4-3.6, since by Equation (3.12) composition of sandwiching is equivalent to multiplication of quaternions.

transform with some appropriate quaternion (see Exercises 11 and 15). Recall, however, that rotation, mirror image, and uniform scaling, all have a fixed point Q of the transformation. Since $P = Q + (P - Q)$, we can compute conformal transformations on affine points P by using quaternions to transform the vectors $P - Q$ and then adding the resulting vectors to Q. In this way, quaternions can be applied to compute conformal transformations on points as well as on vectors.

Quaternion v.s. Matrix



	memory	*transformation*	*composition*
quaternions	4 scalars	28 multiplies	16 multiplies
3×3 matrices	9 scalars	9 multiplies	27 multiplies

Table 1: Tradeoffs between speed and memory for quaternions and 3×3 matrices.

Avoiding Distortion

To compose conformal transformations, we multiply the corresponding quaternions. Since every quaternion represents a conformal transformation, computing transformations by sandwiching with quaternions will never distort angles.

we can normalize the resulting quaternion q simply by dividing by its length $|q|$.

Key Frame Animation

- Can be used to interpolate in between frames.

$$q(t) = slerp(q_0, q_1, t) = \frac{\sin((1-t)\phi)}{\sin(\phi)} q_0 + \frac{\sin(t\phi)}{\sin(\phi)} q_1,$$

where ϕ is the angle between q_0 and q_1. Recall that SLERP guarantees that if q_0 and q_1 are unit quaternions, then $q(t)$ is also a unit quaternion, and if ϕ is the angle between q_0 and q_1, then $t\phi$ is the angle between q_0 and $q(t)$. Thus spherical linear interpolation applied to quaternions generates the appropriate intermediate rotations for key frame animation.

Conversion Formulas

$$Rot(N,\phi) = (\cos\phi) I + (1-\cos\phi)(N^T * N) + (\sin\phi)(N \times \_)$$

$$q(N,\phi/2) = \cos(\phi/2) O + \sin(\phi/2) N \,.$$

=> Skip derivation using trigonometry

$$q(N,\phi/2) = \frac{\sqrt{R_{1,1}+R_{2,2}+R_{3,3}+1}}{2} O + \frac{(R_{2,3}-R_{3,2})i + (R_{3,1}-R_{1,3})j + (R_{1,2}-R_{2,1})k}{2\sqrt{R_{1,1}+R_{2,2}+R_{3,3}+1}} .$$

$$Rot(N,\phi) = \begin{pmatrix} q_4^2+q_1^2-q_2^2-q_3^2 & 2q_1q_2+2q_3q_4 & 2q_1q_3-2q_2q_4 \\ 2q_1q_2-2q_3q_4 & q_4^2-q_1^2+q_2^2-q_3^2 & 2q_2q_3+2q_1q_4 \\ 2q_1q_3+2q_2q_4 & 2q_2q_3-2q_1q_4 & q_4^2-q_1^2-q_2^2+q_3^2 \end{pmatrix} .$$