Announcements

- Second assignment solutions available on Thursday.
- Third assignment available tomorrow: start with tutorial lessons 1, 3, 6, 7.
- Project progress meetings.
- Thursday: animal skin. *CIS*.
- Tuesday: glare effects and silver tarnish. *CIS*.
- *No incompletes*.

I hope you had a good break!

The solutions to the assignment will be made available on Thursday. The graded assignments will be handed out next Tuesday, the latest.

The third assignment, due on Thursday of next week, is the easiest of the three, in terms of intellectual effort, but just as time consuming. It consists of several modeling tasks on `SoftImage`, and a few simple written questions. While the assignment is not on the WWW yet, all materials needed to start working on it are already available: see the class WWW pages to set up your accounts for `SoftImage`, and do lessons 1, 3, 6 and 7 of the tutorial. Remember to save the models you create during these lessons, since you will submit them electronically as part of your assignment.

Project progress meetings are taking place this week. Remember to come fully prepared to these meetings.

Are the Thursday speakers, Komal Sethi and Sandeep Jain, and the Tuesday speakers, Helios Tsoi and Gary Herman, here?

All color figures from most of the Thursday and Tuesday papers are on the WWW. I brought all the papers to class to see the pictures.

As a final reminder, keep in mind that no incompletes will be given for this class. The reason is that, unlike faculty, I am only teaching for this one quarter.

Today

Rotations:

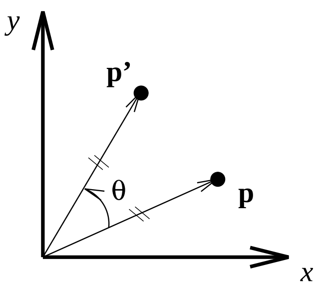
- Euler angles.
- Rotation matrices.
- Axis and angle.
- Quaternions.

3D Rotations

Problem: representing transformation rotating points about origin of *fixed* coordinate system.

Same problem as representing orientation of object: represent orientation as rotation relative to arbitrary initial placement.

In 2D, use one angle θ.



Other authors define operators which rotate the coordinate system axes, while the object remains fixed.

The most common sign convention is to associate positive θ with counterclockwise rotations.

The orientation of a solid is as important as its position for computer animation. And just as

- position is represented as the displacement (vector) of the arbitrarily chosen object center from the origin, so is
- orientation presented as the rotation of the arbitrarily chosen object axes (local coordinate system) relative to the axes of the (global) coordinate system.

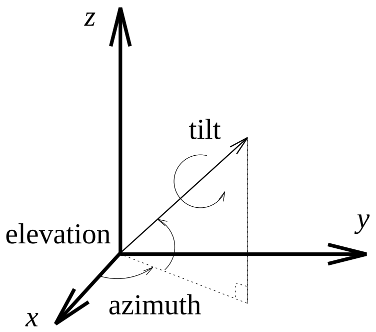
Outside the domain of graphics, mathematical physics investigates the representation of solid orientation, under the general topic of classical mechanics and rigid body kinematics (movement analysis). If you want to learn more about today's topic, that's the literature you should consult.

3D Rotations: D.O.F.

3D rotations have three degrees of freedom (d.o.f.), e.g.

for camera orientation,

- two d.o.f. for direction of gaze, and
- one d.o.f. for tilt angle.



The orientation of an object can be described by these three numbers: *azimuth*, *elevation*, *tilt*. In the case of a camera, the azimuth and elevation are (almost) the angles of spherical coordinates, and specify the direction the camera lens is facing; the tilt angle specifies the amount by which the camera is rotated about its own axis.

Different disciplines, and different tasks, use different representations. Azimuth, elevation, tilt, *roll*, *pitch*, and *yaw* are all terms related to rotation representations. In this presentation, we will concentrate on representations used in graphics.

Euler Angles

Represent rotation by *Euler angles* angles θ, χ, ϕ. Transform point by rotating counterclockwise

- about z axis by θ, then
- about y axis by χ, and finally
- about x axis by ϕ.

As expected, it takes three angles to fully confine the three d.o.f. of a rotation.

Euler Angles: General

Another representation: transform point by rotating counterclockwise

- about z axis by θ_1, then
- about y axis by χ, and finally
- about z axis by θ_2.

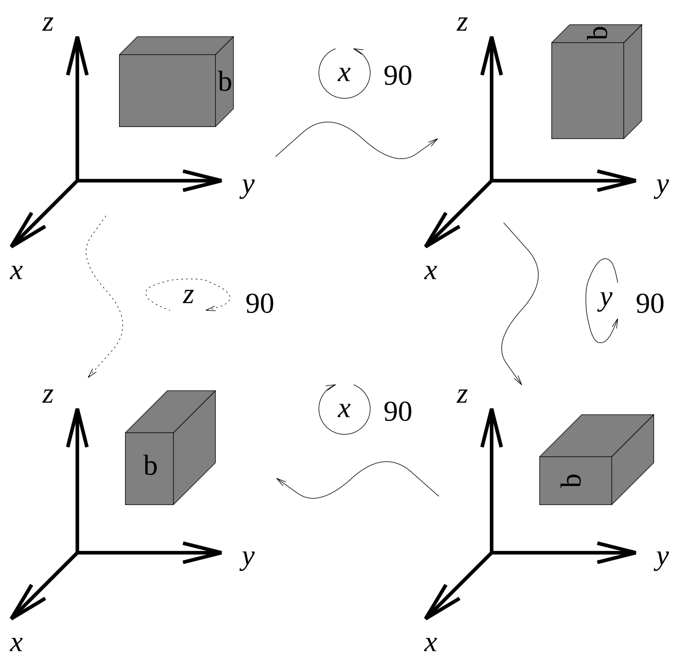
θ_1, χ, θ_2 also called *Euler angles*.

Total of *twelve* conventions in use!

In general *Euler angles* are angles of some combination of three successive rotations about axes, which can effect arbitrary 3D rotation.

A rotation cannot be represented by some angles about *any* three axes. For example, rotating three times about the z axis (by any amounts θ_1, θ_2, and θ_3) cannot effect arbitrary 3D rotations. In this case, of course, θ_1, θ_2, and θ_3 are not called "Euler angles."

Euler Angles: Example



It might seem at first hard to believe that one can specify an arbitrary rotation using only rotations about two axes. But it is indeed possible. The example shown on this slide illustrates how a rotation about the z axis by 90 degrees can be effected using 90-degree rotations about the x and y axes only. In this diagram, the letter "b" is inscribed on the box, and is used to disambiguate the box's configuration. Also, the rotations are performed as if the origin was located at the center of the box; the axes are drawn on the side of the box, rather than penetrating the box, for the sake of clarity.

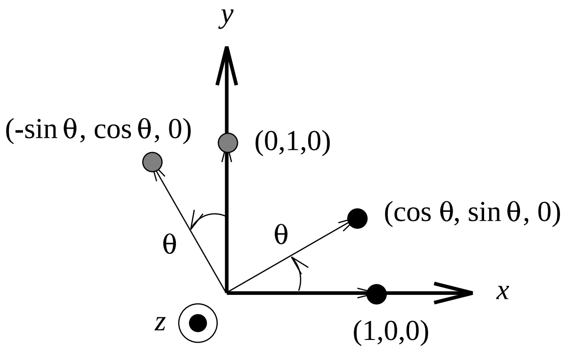
Rotation Matrices: About Axes

In graphics, we use matrix $\mathbf{R}$ such that

$$\mathbf{p}' = \mathbf{R}\mathbf{p}.$$

Rotation about z axis by θ is

$$\mathbf{R}_z = \begin{pmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix}.$$



The derivation of $\mathbf{R}_z$ becomes clear if we consider the effect of this matrix on the unit vectors along the x and y axes:

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

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

$$\begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} = \begin{pmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}.$$

This is exactly the desired effect, as the diagram suggests.

Rotation Matrices: About Axes

To get matrix $\mathbf{R}_x$, for rotation about x axis by ϕ, permute axes

$$z \Rightarrow x \quad x \Rightarrow y \quad y \Rightarrow z$$

and obtain

$$\mathbf{R}_x = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi & \cos\phi \end{pmatrix}$$

by

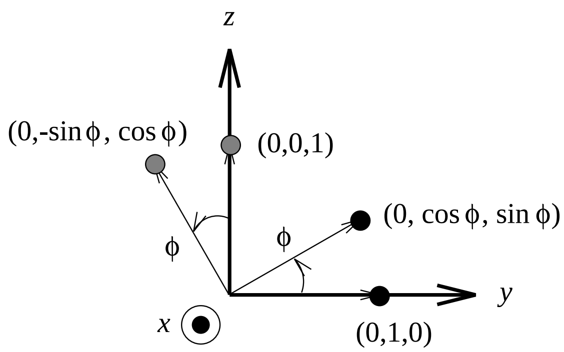
1. permuting columns,

2. permuting rows,

3. and changing θ to ϕ.

Similarly, we find $\mathbf{R}_y$, for rotation about y axis by χ.

Rotation Matrices: About Axes



We perform the axes permutation in two steps:

1. First, we permute the columns of
$$\mathbf{R}_z = \begin{pmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix}$$
to match the rearrangement of the elements of $\mathbf{p}$; we get
$$\begin{pmatrix} 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \\ 1 & 0 & 0 \end{pmatrix}.$$

2. Then, we permute the rows to match the rearrangement of the elements of $\mathbf{p}'$, and change θ to ϕ; this yields $\mathbf{R}_x$, as shown on the previous slide.

Rotation Matrices: General

A general rotation matrix is
$$\mathbf{R} = \begin{pmatrix} R_{00} & R_{01} & R_{02} \\ R_{10} & R_{11} & R_{12} \\ R_{20} & R_{21} & R_{22} \end{pmatrix}$$
such that unit vectors along axes map to mutually perpendicular, unit vectors, i.e.
$$\vec{r}_i \cdot \vec{r}_j = \begin{cases} 1 & \text{for } i = j \\ 0 & \text{otherwise.} \end{cases}$$
where $\vec{r}_i$ is column $i+1$ of $\mathbf{R}$.

To ensure right-handedness, require
$$\det \mathbf{R} > 0.$$

We need six constraints to allow only three d.o.f. to the elements of $\mathbf{R}$. The reason that $\mathbf{R}$ should map unit vectors to mutually perpendicular vectors is that rotations do not shear objects. As rotations do not scale objects either, the unit vectors should also maintain their unit length.

We express these constraints by using the fact that the unit vectors along the axes map to the columns of $\mathbf{R}$, e.g.
$$\vec{r}_0 = \begin{pmatrix} R_{00} \\ R_{10} \\ R_{20} \end{pmatrix} = \begin{pmatrix} R_{00} & R_{01} & R_{02} \\ R_{10} & R_{11} & R_{12} \\ R_{20} & R_{21} & R_{22} \end{pmatrix} \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}.$$
We use the dot product operation, denoted by "$\cdot$", to express both the unit length requirement
$$\vec{r}_i \cdot \vec{r}_i = |\vec{r}_i| = 1,$$
and the perpendicularity constraint
$$\vec{r}_i \cdot \vec{r}_j = 0,$$
when $i \neq j$.

The constraints guarantee that $\det \mathbf{R} = \pm 1$, due to the orthogonality of the columns. If $\det \mathbf{R} < 0$, $\mathbf{R}$ effects *reflection*, as well as rotation; for example, the mirror image of an object is found by flipping the sign of the x coordinate, when the mirror is on the plane $x = 0$. In order to ascertain rotation, we must ensure that the axes-aligned unit vectors map to a right-handed configuration, that it,

$$\vec{r}_0 \times \vec{r}_1 = \vec{r}_2,$$

where "$\times$" denotes the vector (cross) product operation. This is the purpose of the last constraint, namely

$$\det \mathbf{R} > 0.$$

It seems that we have defined seven constraints on the elements of $\mathbf{R}$, not six as we claimed. However, this last constraint is an inequality and thus does not reduce the d.o.f. in selecting $\mathbf{R}$'s elements. This becomes clearer if we consider the following two similar, but simpler, situations:

- Asking a person to select a random number that is positive rather than an arbitrary one. In this case, the set of possible options is reduced by half, from $\Re$ to $\Re_+$, but, in both cases, there is one d.o.f. in making the choice.
- Moving on a half-plane versus moving on a plane. In both cases, we have two d.o.f. of movement, even though, in the former one, our position is restricted by the inequality defining the half-plane.

Summary

Criterion	Euler	Matrix
Storage	+	-
Composition	-	+
Drifting	+	-
Transformation	-	+
Intuition	-	-
Interpolation	+	-

This chart compares the representation of rotations by Euler angles and rotation matrices along several dimensions.

As a rotation has three d.o.f., we need at least three numbers to represent it. Euler angles are triplets of real numbers, and therefore require the least amount of space possible to represent rotations. However, matrices require nine numbers and are thus highly inefficient.

If two rotations are applied back to back to the same point, how can we express the resulting transformation as a single rotation? Using matrices, we simply multiply the individual rotation matrices in order. For example, the rotation matrix for a rotation about z by θ, followed by a rotation about y by χ is

$$\mathbf{R} = \mathbf{R}_y \mathbf{R}_z.$$

By contrast, when we use Euler angles to represent rotations, direct composition is not as easy; usually, a conversion to another representation is needed.

Drifting arises in the context of successive compositions, which arise during animations. When rotation matrices are multiplied, the result should be, theoretically, a new rotation matrix, obeying all the constraints. In practice, however, roundoff errors accumulate in the composition calculations, and, eventually, we get matrices that fail to obey the constraints. While there are algorithms that discover the "nearest rotation matrix" to a non-rotation matrix, they are complex and introduce jerks in the animation when applied; worse, they are frequently needed as matrix multiplication exhibits drifting quite fast. Euler angles do not have to obey any constraints, and thus drifting is not an issue: roundoff errors do occur, but the rotation representation never has to be adjusted to meet any constraints inherent in the representation.

To apply a rotation to a point, we multiply a rotation matrix by the point's position vector. Euler angles are applied onto points after converting the rotation they represent to some other form; they cannot be applied directly.

For both representations, it is hard to figure out how they transform a solid object without doing any calculations. Some people find it easy to mentally perform the successive rotations by the Euler angles, while others can visualize the mapping of the axes by looking at the columns of a rotation matrix. Both tasks are time consuming, however.

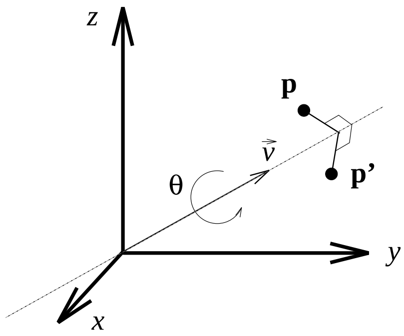
The interpolation problem is related to computer animation. If the animator specifies two *keyframes*, i.e. the initial and ending configuration of an object, how can we automatically interpolate between the keyframes to render the intermediate frames of the animation? The in-betweening problem for orientation is usually solved by interpolating between the parameters representing orientation (the Euler angles or the matrix elements). For Euler angles, this works well. But interpolating matrix elements is very hard since there is no guarantee that the elements of the interpolated matrix will obey the six constraints.

There are many more criteria relative to which we may compare rotation representations, and no representation is perfect on all counts. That's why there are so many representations in use: each is tailored to fit the task on which it's applied. Quaternions, the representation we are about to present, is overall a very good one, but it's not perfect either. We will discuss how it fares after we present the quaternion theory.

Axis & Angle

Represent rotation by

- unit vector $\vec{v}$ about which points revolve, and
- θ, angle of revolution.

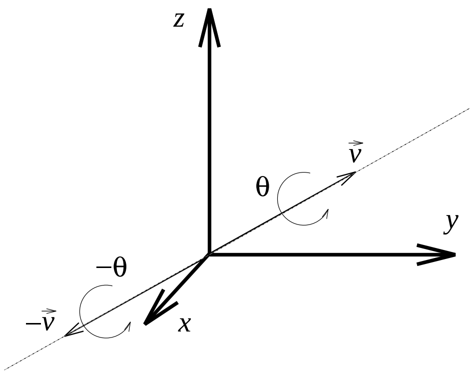


θ is positive when the right-hand thumb lies along $\vec{v}$, and the index points towards the direction of revolution.

The axis & angle representation is both intuitive and compact.

Axis & Angle: Redundancy

- Rotating by θ about $\vec{v}$ is same as
- rotating by $-\theta$ about $-\vec{v}$.



Of course, there is always some redundancy in all representations using angles; that is, if we add any integer multiple of 2π to any angle, we get the same rotation. Here, however, we are concerned about redundancies beyond this trivial one.

Axis & Angle: Derivations

Derived representations from $\vec{v}$, θ:

- $\vec{u} = \theta\vec{v}$ is non-redundant. Also,

$$\begin{aligned} \theta &= |\vec{u}| \\ \vec{v} &= \frac{\vec{u}}{|\vec{u}|}. \end{aligned}$$

- *Quaternion* $\dot{q} = (a, b, c, d)$ such that

$$\begin{aligned} a &= \cos\frac{\theta}{2} \\ (b, c, d) &= \vec{v}\sin\frac{\theta}{2}. \end{aligned}$$

  Redundant: $\pm\dot{q}$ represent same rotation.

All these representations are usually considered equivalent to each other since converting from one to the other is trivial. It is not uncommon to treat them as the same representation, implicitly assuming that the trivial conversions take place as appropriate.

The $\vec{u}$ representation removes the redundancy of the axis & angle one; that is, every rotation is represented by a single $\vec{u}$. This is evident from the fact that the formulas yielding the axis & angle representation given the $\vec{u}$ representation force θ to be positive. Alternatively, we may observe that $\vec{u} = \theta\vec{v} = (-\theta)(-\vec{v})$; that is, both axis & angle representations of the same rotation yield the same $\vec{u}$ representation.

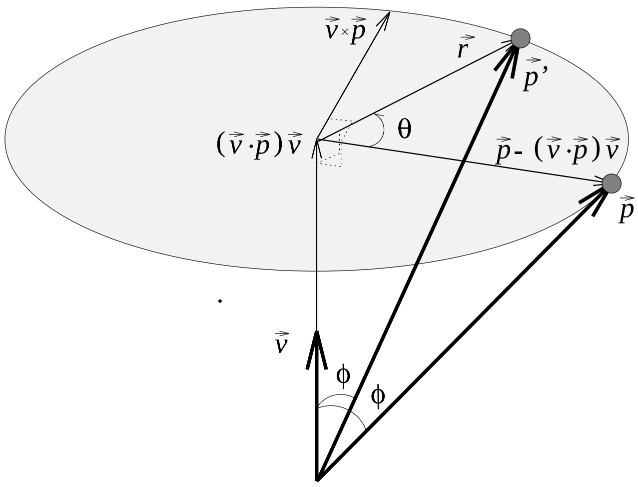
The quaternion representation, however is still redundant. The reason is that the axis $-\vec{v}$ and the rotation $2\pi - \theta$ (which is the same as $-\theta$, in practice) yield

$$\begin{aligned} a &= \cos\left(\pi - \frac{\theta}{2}\right) \\ &= -\cos\frac{\theta}{2} \\ \begin{pmatrix} b \\ c \\ d \end{pmatrix} &= -\vec{v}\sin\left(\pi - \frac{\theta}{2}\right) \\ &= -\vec{v}\sin\frac{\theta}{2}, \end{aligned}$$

which is the quaternion $-\dot{q}$.

Axis & Angle: Transformation

Given position vector $\vec{p}$ of $\mathbf{p}$, find $\vec{p'}$, position vector of $\mathbf{p}'$.



Axis & Angle: Transformation

$$\vec{p'} = \vec{p}\cos\theta + (\vec{v}\times\vec{p})\sin\theta + (1-\cos\theta)(\vec{v}\cdot\vec{p})\vec{v}.$$

Derivation steps:

1. Decompose $\vec{p}$ into components along and perpendicular to $\vec{v}$:
$$(\vec{v}\cdot\vec{p})\vec{v} \qquad \vec{p}-(\vec{v}\cdot\vec{p})\vec{v}.$$
2. Shaded area is 3D circle with radius $|\vec{p}|\sin\phi$.
3. Find $\vec{r}$ by adding orthogonal components
$$(\vec{v}\times\vec{p})\sin\theta \qquad [\vec{p}-(\vec{v}\cdot\vec{p})\vec{v}]\cos\theta.$$
4. Find $\vec{p'}$ by
$$\vec{r}+(\vec{v}\cdot\vec{p})\vec{v}.$$

Here is the derivation in more detail. First identify $\vec{p}$ and $\vec{v}$. Decompose $\vec{p}$ into two components, one along $\vec{v}$ and one perpendicular to it:

- The component of $\vec{p}$ along $\vec{v}$ (a unit vector) has magnitude $\vec{v}\cdot\vec{p}$ and direction along $\vec{v}$; this yields the vector
$$(\vec{v}\cdot\vec{p})\vec{v},$$
which appears on the diagram.
- Subtracting this vector from $\vec{p}$ produces the perpendicular component
$$\vec{p}-(\vec{v}\cdot\vec{p})\vec{v},$$
which also appears on the diagram.

The length of the perpendicular component is, by simple trigonometry, $|\vec{p}|\sin\phi$. This is the same as the length of the vector $\vec{v}\times\vec{p}$. Hence, the shaded ellipse is really a circle viewed in perspective. It follows that $\mathbf{p}'$ lies on that circle, at an angle θ away from $\mathbf{p}$. Relative to the origin, $\vec{p'}$ is given by
$$(\vec{v}\cdot\vec{p})\vec{v}+\vec{r}$$
where the first summand takes us form the origin to the center of the circle and the second takes us from the center to $\mathbf{p}'$, on the circumference. Our next step, naturally, is to find $\vec{r}$.

$\vec{r}$ can be found by observing that it lies on the plane of the circle, and that the mutually perpendicular vectors
$$(\vec{v}\times\vec{p})$$
and
$$[\vec{p}-(\vec{v}\cdot\vec{p})\vec{v}],$$
both of which lie on the circle, form a local coordinate system for the circle's plane. It follows, that, by scaling these vectors by the sine and cosine of θ, respectively, and adding them up, we obtain $\vec{r}$.

Quaternions

General quaternion is arbitrary list of 4 numbers a, b, c, d, written as

- (a,b,c,d), or
- $a+bI+cJ+dK$, where I, J, K are placeholders, or
- $q+\vec{q}$, where
  - $q=a$ is *real part*, and
  - $\vec{q}=(b,c,d)$ is *imaginary part*.

Quaternions are extension of complex numbers.

A *placeholder* is a notational device that allows us to distinguish different elements of the quaternion quadruplet, by tagging them with a special symbol.

It is because quaternions are an extension of complex numbers that we use terms like “real part” and “imaginary part.” Also, the placeholder representation is very similar to the complex number notation $a + \imath b$. And just as we write a as a shorthand for $a + \imath 0$, so we may consider real numbers a special case of quaternions with zero imaginary part; thus, we write just a for the quaternion $(a, 0, 0, 0)$, and similarly we write (b, c, d) for $(0, b, c, d)$. Finally, there is also an exponential-like notation for quaternions, resembling the exponential form $re^{\imath\theta}$ of complex numbers.

This multiplicity of notations seems to be quite redundant at first, but it comes in handy when quaternion operations and properties are written down, as we will soon see.

Quaternions: Operations

Like 4D vectors, e.g. dot product of

$$\begin{aligned} \dot{q} &= (q_0, q_x, q_y, q_z) = q + \vec{q} \\ \dot{p} &= (p_0, p_x, p_y, p_z) = p + \vec{p} \end{aligned}$$

is

$$\dot{q} \cdot \dot{p} = q_0 p_0 + q_x p_x + q_y p_y + q_z p_z = qp + \vec{q} \cdot \vec{p}.$$

Special operations:

Conjugate: $\overline{\dot{q}} = (q_0, -q_x, -q_y, -q_z) = q - \vec{q}.$

Multiplication:

$$\dot{q}\dot{p} = (qp - \vec{q} \cdot \vec{p}) + (q\vec{p} + p\vec{q} + \vec{q} \times \vec{p}).$$

Conjugation: $C_{\dot{q}}(\dot{p}) = (\dot{q}\dot{p})\overline{\dot{q}}.$

In the definition of the multiplication operation, the first parenthesized sum yields the real part of the result (since all summands are scalars), while the second one yields the imaginary part of the result (since all summands are vectors).

The conjugation operation is also called *conjugate product*.

Quaternions: Transformation

Let $\dot{q}$ be quaternion representing rotation:

$$\dot{q} = \cos\frac{\theta}{2} + \vec{v}\sin\frac{\theta}{2}.$$

Then

$$\begin{aligned} |\dot{q}| &= \sqrt{\cos^2\frac{\theta}{2} + \sin^2\frac{\theta}{2}|\vec{v}|^2} \\ &= 1. \end{aligned}$$

So rotations represented by *unit* quaternions.

To apply rotation by unit quaternion $\dot{q}$,

1. turn $\vec{p}$ into purely imaginary quaternion: $\dot{p} = 0 + \vec{p}$,

2. apply conjugation by $\dot{q}$: $C_{\dot{q}}(\vec{p})$.

Here, we have taken advantage of the many notations for quaternions to express $\dot{q}$ as a single expression in terms of the axis $\vec{v}$ (a unit vector) and angle of rotation θ. Also, we implicitly defined the quaternion norm, calculated by treating the quaternion as a 4D vector.

The reasoning shown on the slide shows that quaternions representing rotations have unit norm. It turns out that the reverse is true, as well: all unit quaternions represent rotations. We will see later how the rotation axis and angle are derived from the quaternion. For now, we will argue on the basis of the d.o.f. of rotations: in the case of matrices, the nine elements of $\mathbf{R}$ had to be constrained by six equations to allow three d.o.f. in their specification. Similarly, the four components of a quaternion $\dot{q}$ have be subjected to one constraint (that of unit norm) to allow three d.o.f. in specifying $\dot{q}$.

A purely imaginary quaternion is one with zero real part.

From now on, we will frequently talk about "quaternions representing rotations," without explicitly stating that we are assuming unit quaternions. In other words, in the context of rotations, when we refer to "quaternions," read "unit quaternions".

Quaternions: Transformation

Recall

$$\dot{s}\dot{r} = (sr - \vec{s}\cdot\vec{r}) + (s\vec{r} + r\vec{s} + \vec{s}\times\vec{r})$$

$$q = \cos\frac{\theta}{2} \qquad \vec{q} = \vec{v}\sin\frac{\theta}{2}.$$

Rotating $\vec{p}$ using quaternion $\dot{q}$:

$$\begin{aligned}
C_{\dot{q}}(\vec{p}) &= \dot{q}\vec{p}\bar{\dot{q}} \\
&= [(-\vec{q}\cdot\vec{p}) + (q\vec{p} + \vec{q}\times\vec{p})][(q) + (-\vec{q})] \\
&= [(-\vec{q}\cdot\vec{p})q - (q\vec{p} + \vec{q}\times\vec{p})\cdot(-\vec{q})] + \\
&\quad [(-\vec{q}\cdot\vec{p})(-\vec{q}) + q(q\vec{p} + \vec{q}\times\vec{p}) + \\
&\quad (q\vec{p} + \vec{q}\times\vec{p})\times(-\vec{q})] \\
&= (\vec{q}\cdot\vec{p})\vec{q} + q^2\vec{p} + 2q(\vec{q}\times\vec{p}) - \\
&\quad [(\vec{q}\cdot\vec{q})\vec{p} - (\vec{q}\cdot\vec{p})\vec{q}] \\
&= 2(\vec{q}\cdot\vec{p})\vec{q} + 2q(\vec{q}\times\vec{p}) + (q^2 - |\vec{q}|^2)\vec{p} \\
&= 2\sin^2\frac{\theta}{2}(\vec{v}\cdot\vec{p})\vec{v} + 2\cos\frac{\theta}{2}\sin\frac{\theta}{2}(\vec{v}\times\vec{p}) + \\
&\quad \left(\cos^2\frac{\theta}{2} - \sin^2\frac{\theta}{2}\right)\vec{p} \\
&= (1-\cos\theta)(\vec{v}\cdot\vec{p})\vec{v} + \sin\theta(\vec{v}\times\vec{p}) + \cos\theta\vec{p} \\
&= \vec{p'}.
\end{aligned}$$

Several identities from vector analysis and trigonometry were used in this derivation. In particular, the first two steps were derived by applying the definition of quaternion multiplication, and using the fact that $\vec{p}$ is a purely imaginary quaternion. In the next step, we used several identities, the notable ones being

$$\begin{aligned}
\vec{q}\times\vec{p} &= -(\vec{p}\times\vec{q}) \\
(\vec{q}\times\vec{p})\cdot\vec{q} &= 0 \\
(\vec{q}\times\vec{p})\times\vec{q} &= (\vec{q}\cdot\vec{q})\vec{p} - (\vec{q}\cdot\vec{p})\vec{q};
\end{aligned}$$

this last identity is illustrated on the next slide.

In the next step we collected terms in preparation for the substitution of $\dot{q}$ by $\vec{v}$ and θ, which took place in the following step. Subsequently, we used the trigonometric identities

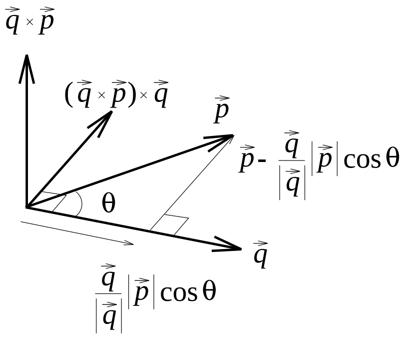
$$\begin{aligned}
\cos\theta &= \cos^2\frac{\theta}{2} - \sin^2\frac{\theta}{2} \\
&= 1 - 2\sin^2\frac{\theta}{2} \\
\sin\theta &= 2\cos\frac{\theta}{2}\sin\frac{\theta}{2}.
\end{aligned}$$

The final step followed from the expression rotating a point about the axis $\vec{v}$ by an angle θ:

$$\vec{p'} = \vec{p}\cos\theta + (\vec{v}\times\vec{p})\sin\theta + (1-\cos\theta)(\vec{v}\cdot\vec{p})\vec{v}.$$

Quaternions: Transformation

$$(\vec{q}\times\vec{p})\times\vec{q} = (\vec{q}\cdot\vec{q})\vec{p} - (\vec{q}\cdot\vec{p})\vec{q}.$$



From the diagram, it is clear that the vectors

$$(\vec{q} \times \vec{p}) \times \vec{q}$$

and

$$\vec{p} - \frac{\vec{q}}{|\vec{q}|} |\vec{p}| \cos\theta$$

have the same direction. The magnitude of the former vector is $|\vec{q}|^2 |\vec{p}| \sin\theta$, while the later has magnitude $|\vec{p}| \sin\theta$ (see the diagram). It follows that

$$(\vec{q} \times \vec{p}) \times \vec{q} = |\vec{q}|^2 \left(\vec{p} - \frac{\vec{q}}{|\vec{q}|} |\vec{p}| \cos\theta \right).$$

Since $\vec{q} \cdot \vec{q} = |\vec{q}|^2$ and $\vec{q} \cdot \vec{p} = |\vec{q}||\vec{p}| \cos\theta$, it follows that

$$|\vec{q}|^2 \left(\vec{p} - \frac{\vec{q}}{|\vec{q}|} |\vec{p}| \cos\theta \right) = (\vec{q} \cdot \vec{q})\vec{p} - (\vec{q} \cdot \vec{p})\vec{q}.$$

Quaternions: Multiplication

$$\begin{aligned}
\dot{q}\dot{p} &= (qp - \vec{q} \cdot \vec{p}) + (q\vec{p} + p\vec{q} + \vec{q} \times \vec{p}) \\
&= \begin{pmatrix} q_0 p_0 - q_x p_x - q_y p_y - q_z p_z \\ q_0 p_x + q_x p_0 + q_y p_z - q_z p_y \\ q_0 p_y - q_x p_z + q_y p_0 + q_z p_x \\ q_0 p_z + q_x p_y - q_y p_x + q_z p_0 \end{pmatrix} \\
&= Q_+(\dot{q})\dot{p} \\
&= Q_-(\dot{p})\dot{q},
\end{aligned}$$

where

$$\begin{aligned}
Q_+(\dot{q}) &= \begin{pmatrix} q_0 & -q_x & -q_y & -q_z \\ q_x & q_0 & -q_z & q_y \\ q_y & q_z & q_0 & -q_x \\ q_z & -q_y & q_x & q_0 \end{pmatrix} \\
Q_-(\dot{p}) &= \begin{pmatrix} p_0 & -p_x & -p_y & -p_z \\ p_x & p_0 & p_z & -p_y \\ p_y & -p_z & p_0 & p_x \\ p_z & p_y & -p_x & p_0 \end{pmatrix}.
\end{aligned}$$

A quaternion is multiplied by a matrix by writing the four numbers comprising the quaternion as a column vector.

These expressions are derived by tedious algebra, using the definitions of the dot and cross products. For example, the first row of the column vector on the second line is

$$q_0 p_0 - q_x p_x - q_y p_y - q_z p_z = qp - \vec{q} \cdot \vec{p},$$

which is the real part of the product. This is the same as the first element of the products

$$\begin{pmatrix} q_0 & -q_x & -q_y & -q_z \\ q_x & q_0 & -q_z & q_y \\ q_y & q_z & q_0 & -q_x \\ q_z & -q_y & q_x & q_0 \end{pmatrix} \begin{pmatrix} p_0 \\ p_x \\ p_y \\ p_z \end{pmatrix}$$

and

$$\begin{pmatrix} p_0 & -p_x & -p_y & -p_z \\ p_x & p_0 & p_z & -p_y \\ p_y & -p_z & p_0 & p_x \\ p_z & p_y & -p_x & p_0 \end{pmatrix} \begin{pmatrix} q_0 \\ q_x \\ q_y \\ q_z \end{pmatrix}.$$

If you would like to verify the full derivation, recall that

$$\begin{aligned}
\vec{q} \times \vec{p} &= \begin{vmatrix} \vec{\imath} & \vec{\jmath} & \vec{k} \\ q_x & q_y & q_z \\ p_x & p_y & p_z \end{vmatrix} \\
&= \begin{pmatrix} q_y p_z - q_z p_y \\ q_z p_x - q_x p_z \\ q_x p_y - q_y p_x \end{pmatrix}.
\end{aligned}$$

Quaternions: Rotation Matrix

Using matrix form of multiplication,

$$\begin{aligned}
C_{\dot{q}}(\dot{p}) &= (\dot{q}\dot{p})\overline{\dot{q}} \\
&= Q_-(\overline{\dot{q}})(\dot{q}\dot{p}) \\
&= Q_-(\overline{\dot{q}})[Q_+(\dot{q})\dot{p}] \\
&= [Q_-(\overline{\dot{q}})Q_+(\dot{q})]\dot{p}.
\end{aligned}$$

When

- $\dot{q}$ is unit quaternion, and
- $\dot{p} = \vec{p}$ is purely imaginary,

then applying rotation $\dot{q}$ to $\vec{p}$ is

$$\mathbf{R}\vec{p},$$

where

$$\mathbf{R} = Q_-(\overline{\dot{q}})Q_+(\dot{q})$$

is rotation matrix.

As a minor remark, the expressions $\vec{p'} = \mathbf{R}\vec{p}$ and $\mathbf{p}' = \mathbf{R}\mathbf{p}$ are equivalent when we are not using site notation, since points and their position vectors have identical coordinates.

In words, we have found a way to transform unit quaternions to rotation matrices. The reverse transformation follows...

Quaternions: Rotation Matrix

$\mathbf{R}$ is

$$\begin{pmatrix} a^2+b^2-c^2-d^2 & 2bc-2ad & 2bd+2ac \\ 2bc+2ad & a^2-b^2+c^2-d^2 & 2cd-2ab \\ 2bd-2ac & 2cd+2ab & a^2-b^2-c^2+d^2 \end{pmatrix},$$

where $\dot{q} = (a, b, c, d)$. Given

$$\mathbf{R} = \begin{pmatrix} R_{00} & R_{01} & R_{02} \\ R_{10} & R_{11} & R_{12} \\ R_{20} & R_{21} & R_{22} \end{pmatrix},$$

find $\dot{q}$ by equating matrix entries:

$$\begin{aligned} 1 + R_{00} + R_{11} + R_{22} &= 4a^2 \\ 1 + R_{00} - R_{11} - R_{22} &= 4b^2 \\ 1 - R_{00} + R_{11} - R_{22} &= 4c^2 \\ 1 - R_{00} - R_{11} + R_{22} &= 4d^2 \end{aligned}$$

$$\begin{array}{ll} R_{21} - R_{12} = 4ab & R_{02} - R_{20} = 4ac \\ R_{10} - R_{01} = 4ad & R_{10} + R_{01} = 4bc \\ R_{21} + R_{12} = 4cd & R_{02} + R_{20} = 4bd. \end{array}$$

The first equation is obtained by expanding the right-hand side of the identity

$$\mathbf{R} = Q_-(\overline{\dot{q}})Q_+(\dot{q}),$$

shown on the previous slide.

The group of four equations allows us to compute the magnitudes of a, b, c, and d; in constructing these equations, we used the fact that

$$a^2 + b^2 + c^2 + d^2 = |\dot{q}|^2 = 1.$$

The signs of $\dot{q}$'s components are found from the last six equations.

Quaternions: Composition

Recall

$$\dot{q}\dot{r} = (qr - \vec{q}\cdot\vec{r}) + (q\vec{r} + r\vec{q} + \vec{q}\times\vec{r}).$$

Property of conjugation:

$$\begin{aligned} \overline{\dot{q}\dot{r}} &= (qr - \vec{q}\cdot\vec{r}) - (q\vec{r} + r\vec{q} + \vec{q}\times\vec{r}) \\ &= [qr - (-\vec{q})\cdot(-\vec{r})] + \\ &\quad [q(-\vec{r}) + r(-\vec{q}) + (-\vec{r})\times(-\vec{q})] \\ &= \overline{\dot{r}}\,\overline{\dot{q}}. \end{aligned}$$

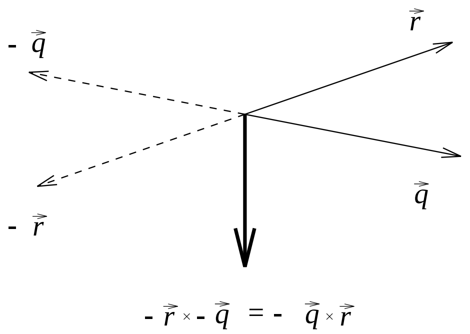
Hence

$$\begin{aligned} C_{\dot{q}}(C_{\dot{r}}(\dot{p})) &= C_{\dot{q}}(\dot{r}\dot{p}\overline{\dot{r}}) \\ &= (\dot{q}\dot{r})\dot{p}(\overline{\dot{r}}\,\overline{\dot{q}}) \\ &= (\dot{q}\dot{r})\dot{p}(\overline{\dot{q}\dot{r}}) \\ &= C_{\dot{q}\dot{r}}(\dot{p}). \end{aligned}$$

Rotation using $\dot{r}$, then $\dot{q}$, is same as rotation using $\dot{q}\dot{r}$.

Quaternions: Composition

$$-(\vec{q} \times \vec{r}) = (-\vec{r}) \times (-\vec{q}).$$



In our last derivation, we took advantage of the associativity of quaternion multiplication, a property which is easy, but tedious, to prove.

Of course, an alternate method to compose rotations represented by quaternions is to find the matrix representations of $\dot{q}$ and $\dot{r}$ and multiply the matrices; however, the result is not a quaternion. Although it is possible to convert a matrix to a quaternion, the overall computational cost becomes significantly higher than the algorithm shown on the previous slide, and is subject to more drifting.

Summary

Criterion	Euler	Matrix	Quaternions
Storage	+	-	+
Composition	-	+	+
Drifting	+	-	+
Transformation	-	+	+
Intuition	-	-	-
Interpolation	+	-	+

Axis & angle is intuitive, and easily convertible from/to quaternions. Hence, close to ideal for graphics.

This is the same table we saw earlier, augmented by a column for the quaternions.

In terms of storage, a quaternion representation of a rotation requires four numbers, just one more than the minimum of three. Also, the $\vec{u}$ representation we saw earlier, which is easily convertible from/to quaternions, is optimal.

Composition of quaternions is very simple as it is equivalent to quaternion multiplication. Moreover, it is more efficient than matrix multiplication, requiring many fewer operations to compute. Also, it is less prone to drifting, and, when drifting occurs, all it takes to meet the single constraint is to divide the quaternion by its norm; this is a simple operation, which rarely introduces noticeable jerks in animations.

To apply a rotation to a point, we perform conjugation, again a simple operation. Alternatively, we may derive the rotation matrix corresponding to the quaternion and use matrix multiplication; this is very efficient if special hardware performs the matrix multiplication.

It is easy, but non-trivial, to visualize the rotation effected by a quaternion $\dot{q}$. The axis of rotation $\vec{v}$ is easily found, as it is a scalar multiple of the imaginary part of $\dot{q}$:

$$\frac{\vec{q}}{|\vec{q}|}.$$

However, the rotation angle θ is encoded within the components of $\dot{q}$ in terms of sines and cosines, and thus it is hard to identify it without computation. Still, the necessary computations are simple, as θ must satisfy the following two equations:

$$\sin\frac{\theta}{2} = |\vec{q}| \quad \text{and} \quad \cos\frac{\theta}{2} = q,$$

which imply

$$\sin\theta = 2q|\vec{q}| \quad \text{and} \quad \cos\theta = q^2 - |\vec{q}|^2.$$

Finally, quaternions are easily interpolated. All quaternions representing rotations have unit norm, and so we can think of them as 4D points on the unit 4D hypersphere. Interpolating between two quaternions simply requires us to move along a great hyper-circle on that sphere, joining the two keyframe quaternions. The formulas for this interpolation are very simple, and are available in Ken Shoemake's 1985 SIGGRAPH paper, entitled *Animating Rotation with Quaternion Curves*.

References

- Guibas/Ramshaw (handout 12), section 1.
- Hart: pages 257–260 and `quatdemo`.
- Lerios.

All readings are in the course reader.

Section 2 of the Guibas/Ramshaw handout discusses quaternions in a more generalized setting. It takes a mathematician's point of view, viewing quaternions as an *algebra* over $\Re^4$, i.e. a set with some operations defined on it.

Hart takes a more practical point of view. He presents a method that engineers may use to visualize the effect of particular quaternions on objects. He also made available some demonstration software called `quatdemo`, which is available under the class WWW pages, as well as in the class directory on the Sweet Hall file system. Finally, Hart has an excellent list of references.

My piece comprises my lecture notes from 1993, when I took CS 348A. I summarize most formulas for the sake of quick reference. A more recent version of the document reproduced in the course reader is available from the class WWW pages.

For those of you who happen to own *Computer Graphics: Principles and Practice* by Foley, van Dam, Feiner and Hughes, there is a short discussion on quaternions in there, as well; consult the index.

Thank You...

for your patience during the first class I taught!