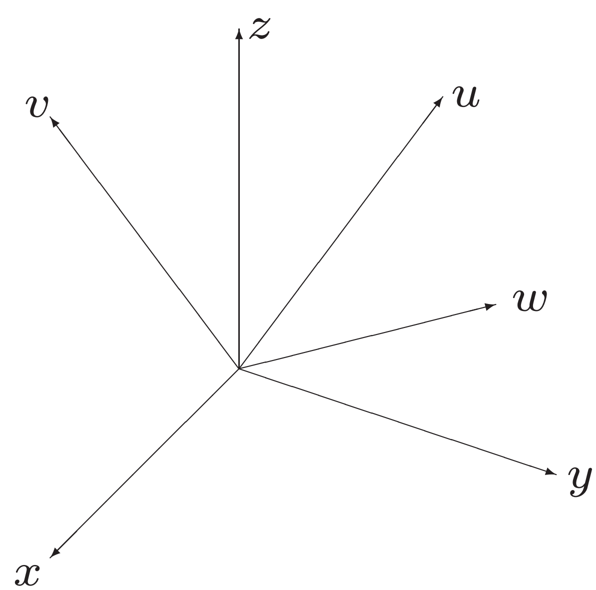
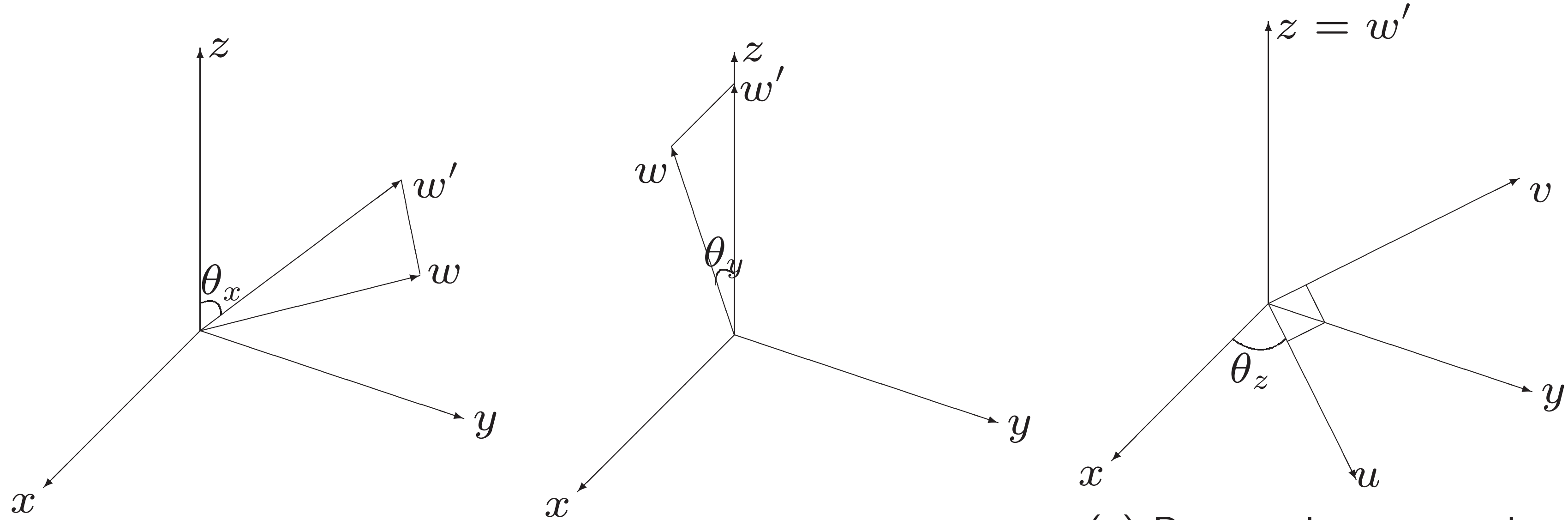
Viewing Systems II: Orientations and Quaternions



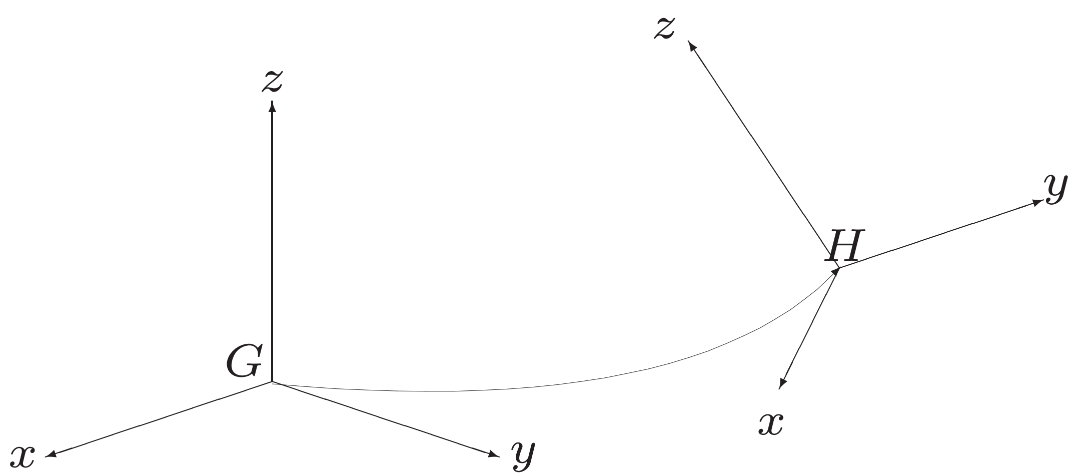


(a) Rotate about x so that w lies in xz-plane.

(b) Rotate about y so that w coincides with z.

(c) Rotate about z so that u and v coincide with x and y.

Rotating a frame to coincide with the standard frame



Smooth Interpolation of Frames

It is possible to perform any change of orientation about an arbitrary axis with three rotations, one about each of the coordinate axes, by a triple of three angles, $(\theta_x, \theta_y, \theta_z)$. These define a general rotation matrix, by composing the three basic rotations:

$$\mathbf{R}(\theta_x, \theta_y, \theta_z) = \mathbf{R}_z(\theta_z)\mathbf{R}_y(\theta_y)\mathbf{R}_x(\theta_x).$$

These three angles are called the **Euler angles** for the rotation. Thus, we can parameterize any rotation in 3-space as triple of numbers, each in the range $\alpha \in [0, 2\pi]$.

With $c_a = \cos(\theta_a)$ and $s_a = \sin(\theta_a)$,

$$R(\theta_x, \theta_y, \theta_z) = \begin{pmatrix} c_y c_z & c_y s_z & -s_y & 0 \\ s_x s_y c_z - c_x s_z & s_x s_y s_z + c_x c_z & s_x c_y & 0 \\ c_x s_y c_z + s_x s_z & c_x s_y s_z - s_x c_z & c_x c_y & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$
$$= R_z(\theta_z) R_y(\theta_y) R_x(\theta_x),$$

where $R_x(\theta_x)$, $R_y(\theta_y)$ and $R_z(\theta_z)$ are the standard rotation matrices.

Given a point P represented as a homogeneous row vector, the rotation of P is given by $P' = PR(\theta_x, \theta_y, \theta_z)$. Animation between two rotations involves interpolating independently the three angles θ_x, θ_y and θ_z.

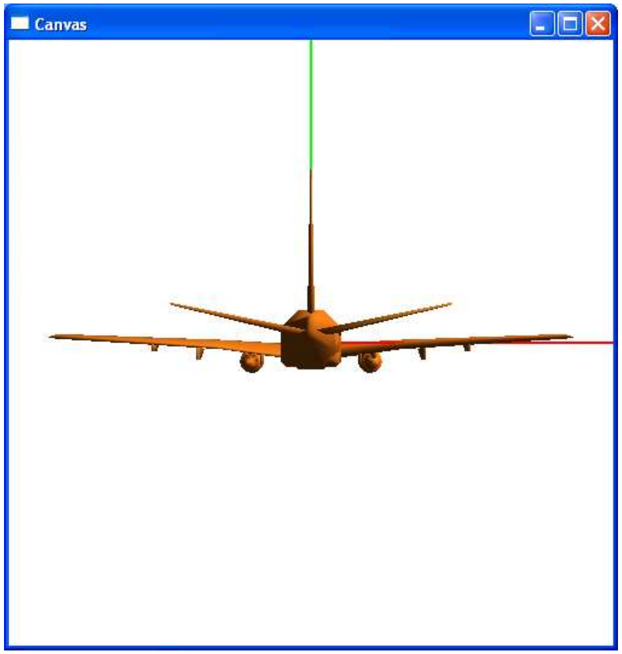
The standard rotation matrices are given by

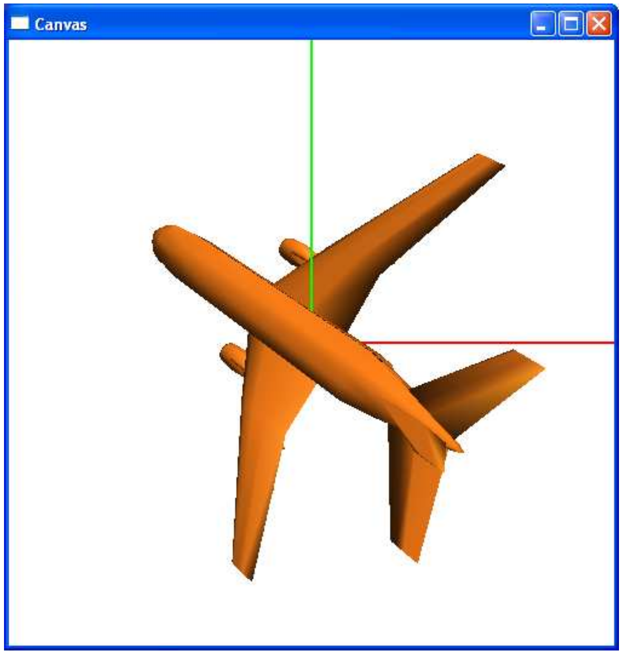
$$R_x(\theta_x) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & c_x & -s_x & 0 \\ 0 & s_x & c_x & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

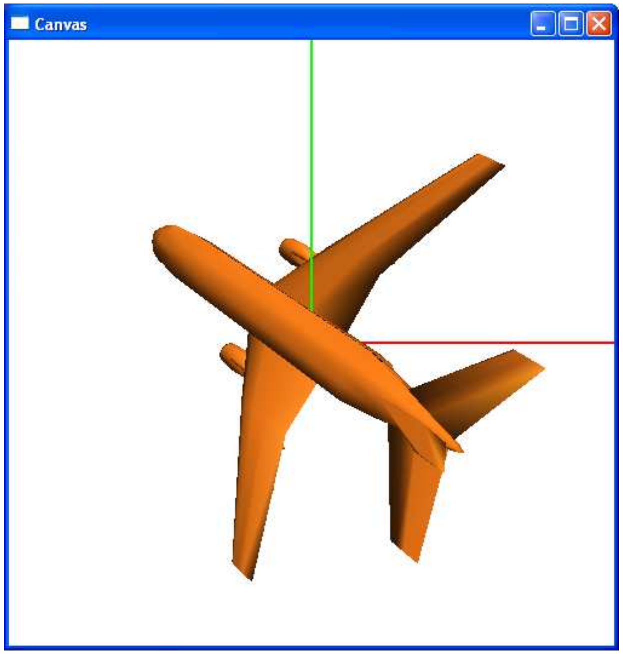
$$R_y(\theta_y) = \begin{pmatrix} c_y & 0 & s_y & 0 \\ 0 & 1 & 0 & 0 \\ -s_y & 0 & c_y & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$R_z(\theta_z) = \begin{pmatrix} c_z & -s_z & 0 & 0 \\ s_z & c_z & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Specifying Orientation using Quaternions are easier than Euler angles








```
glMatrixMode (GL_MODELVIEW);
glRotatef(45, 1, 0, 0);
glRotatef(45, 0, 1, 0);
glRotatef(45, 0, 0, 1);
glutAirPlane(1);
```

```
Quaternion = -0.46, -0.21, -0.41, 0.75
Rotation Matrix =
   0.55  0.82  0.06  0.00
  -0.43  0.22  0.87  0.00
0.71 -0.51  0.48  0.00
   0.00  0.00  0.00  1.00
```

Quaternions:

$$i^2 = j^2 = k^2 = -1 \quad ij = k, \quad jk = i, \quad ki = j.$$

Combining these, it follows that $ji = -k$, $kj = -i$ and $ik = -j$. A quaternion is defined to be a generalized complex number of the form

$$q = q_0 + q_1 i + q_2 j + q_3 k.$$

We will see that quaternions bear a striking resemblance to our notation for angular displacement. In particular, we can rewrite the quaternion notation in terms of a scalar and vector as

$$q = (s, \vec{u}) = s + u_x i + u_y j + u_z k.$$

Furthermore define the product of quaternions to be

$$q_1 q_2 = (s_1 s_2 - (\vec{u}_1 \cdot \vec{u}_2), \quad s_1 \vec{u}_2 + s_2 \vec{u}_1 + \vec{u}_1 \times \vec{u}_2).$$

Define the conjugate of a quaternion $q = (s, \vec{u})$ to be $\bar{q} = (s, -\vec{u})$. Define the magnitude of a quaternion to be the square root of this product:

$$|q|^2 = q\bar{q} = s^2 + |\vec{u}|^2.$$

A unit quaternion is one of unit magnitude, $|q| = 1$. A pure quaternion is one with a 0 scalar component

$$p = (0, \vec{v}).$$

Any quaternion of nonzero magnitude has a multiplicative inverse, which is

$$q^{-1} = \frac{1}{|q|^2}\bar{q}.$$

Quaternion and Rotation:

Define the rotation operator

$$R_q(p) = qpq^{-1}.$$

$$R_q(p) = (0, (s^2 - (\vec{u}\cdot\vec{u}))\vec{v} + 2\vec{u}(\vec{u}\cdot\vec{v}) + 2s(\vec{u}\times\vec{v})).$$

Unit quaternions can be shown to be isomorphic to orientations and given by

$$q = (\cos\theta, (\sin\theta)\vec{u}), \qquad \text{where } |\vec{u}| = 1.$$

This is equivalent to a rotation by an angle 2θ around the axis $\vec{u}$.

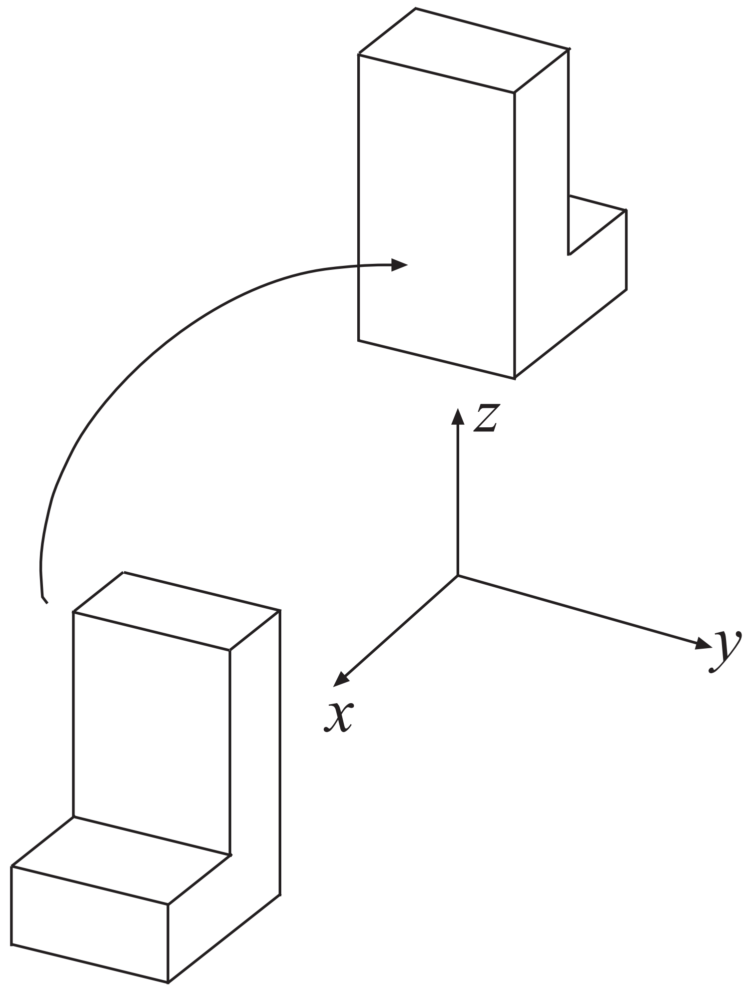
Thus, in summary, we encode points in 3-space as pure quaternions

$$p = (0, \vec{v}),$$

and we encode a rotation by angle θ about a unit vector $u \rightarrow$ as a unit quaternion

$$q = (\cos(\theta/2), \sin(\theta/2)\vec{u}),$$

then the image of the point under this rotation is given by the vector part of the result of the quaternion rotation operator $R_q(p) = qpq^{-1}$.



Rotation example.

Composing Rotations:

Given two unit quaternions q and q', a rotation by q followed by a rotation by q' is equivalent to a single rotation by the product $q'' = q'q$. That is,

$$R_{q'}R_q = R_{q''} \qquad \text{where } q'' = q'q.$$

This follows from the associativity of quaternion multiplication, and the fact that $(qq')^{-1} = q^{-1}q'^{-1}$, as shown below.

$$\begin{aligned} R_{q'}(R_q(p)) &= q'(qpq^{-1})q'^{-1} \\ &= (q'q)p(q^{-1}q'^{-1}) \\ &= (q'q)p(qq')^{-1} \\ &= q''pq''^{-1} \\ &= R_{q''}(p). \end{aligned}$$

Matrices and Quaternions:

Given a unit quaternion

$$q = (\cos(\theta/2), \sin(\theta/2)\vec{u}) = (w, (x, y, z))$$

what is the corresponding affine transformation (expressed as a rotation matrix). By simply expanding the definition of $R_q(p)$, it is not hard to show that the following (homogeneous) matrix is equivalent

$$\begin{pmatrix} 1-2y^2-2z^2 & 2xy-2wz & 2xz+2wy & 0 \\ 2xy+2wz & 1-2x^2-2z^2 & 2yz-2wx & 0 \\ 2xz-2wy & 2yz+2wx & 1-2x^2-2y^2 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

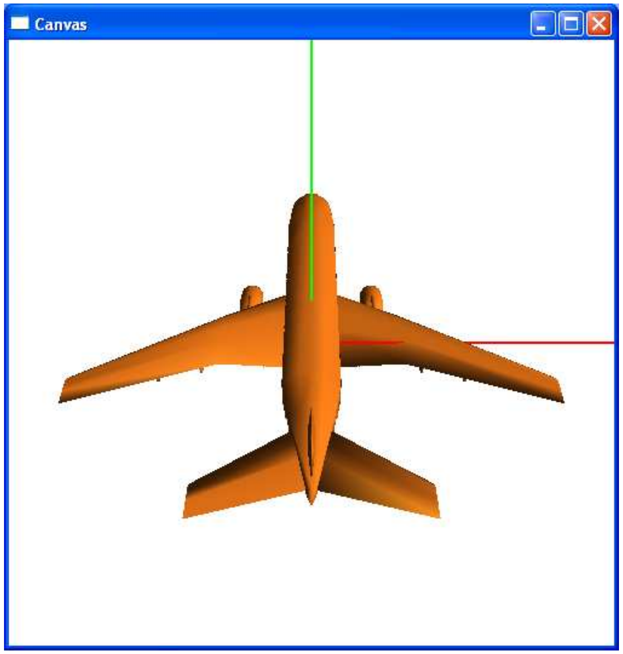
To convert from an orthogonal rotation matrix to a unit quaternion, we observe that if $M = [m_{i,j}]$ is the affine transformation in homogeneous form,

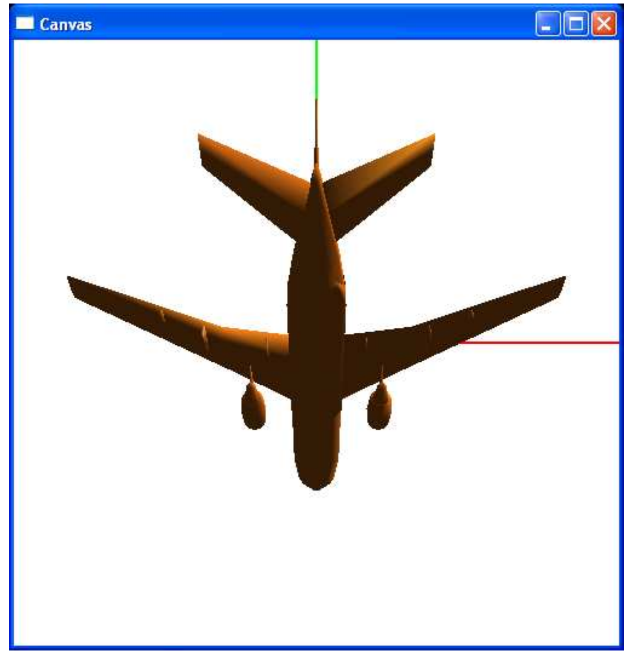
$$\text{trace}(M) = 4 - 4(x^2 + y^2 + z^2) = 4w^2.$$

Once we have w, we can find the order quantities by cancelling symmetric terms:

$$x = \frac{m_{32} - m_{23}}{4w},$$
$$y = \frac{m_{13} - m_{31}}{4w},$$
$$z = \frac{m_{21} - m_{12}}{4w}$$

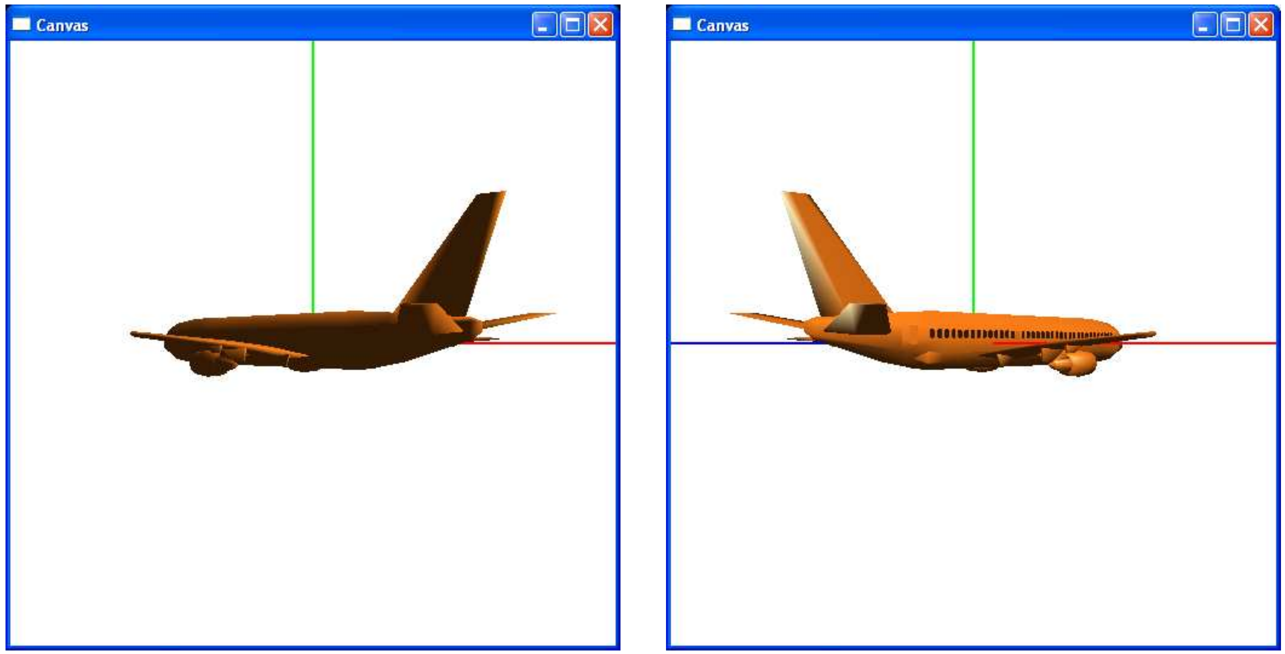
Pitch





Rotation Axis = 1, 0, 0, 0
Rotation Angle = $\pm\pi/4$
Quaternion Vector = ±0.382683 0.000000 0.000000 0.923880

Yaw

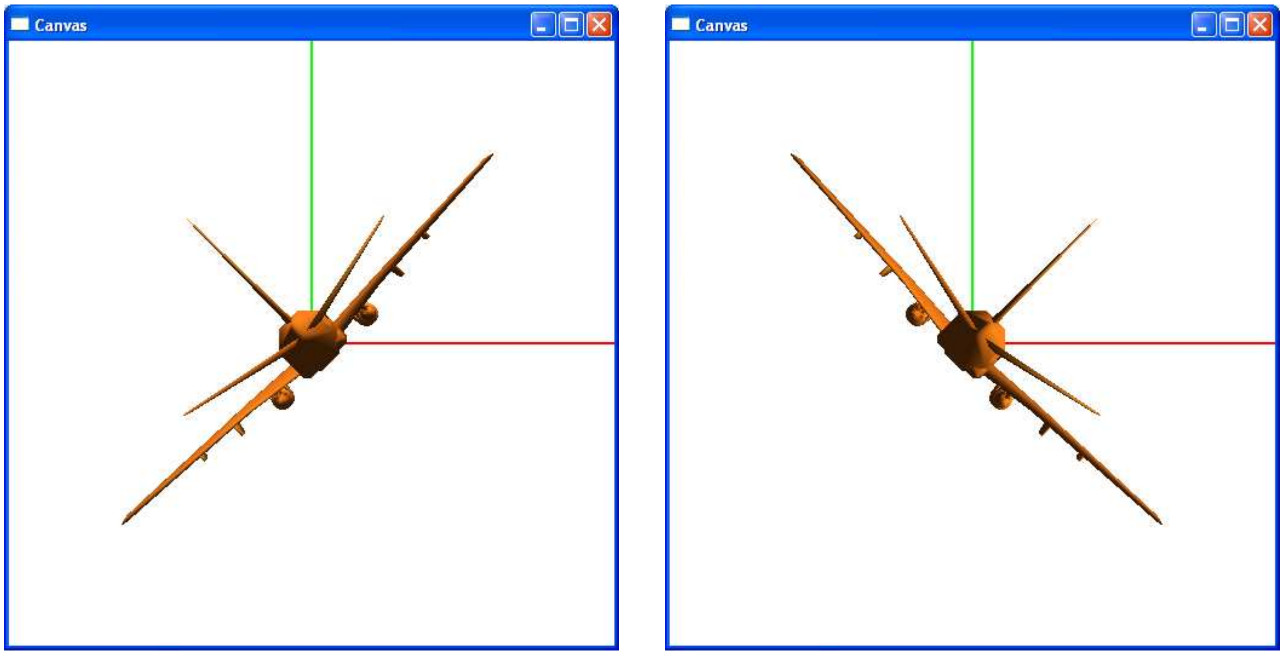


Rotation Axis = 0, 1, 0, 0

Rotation Angle = $\pm\pi/4$

Quaternion Vector = 0.000000 $\pm$0.382683 0.000000 0.923880

Roll



Rotation Axis = 0, 0, 1, 0
Rotation Angle = $\pm\pi/4$
Quaternion Vector = 0.000000 0.000000 $\pm$0.382683 0.923880

Additonal Examples

The following web-page contains a good quaternion intro and C++ source codes:
http://www.lboro.ac.uk/departments/ma/gallery/quat/intro.html

Reading Assignment and News

Before the next class please review Chapter 3 and its practice exercises, of the recommended text.

(Recommended Text: Interactive Computer Graphics, by Edward Angel, Dave Shreiner, 6th edition, Addison-Wesley)

Please track Blackboard for the most recent Announcements and Project postings related to this course.

(http://www.cs.utexas.edu/users/bajaj/graphics2012/cs354/)