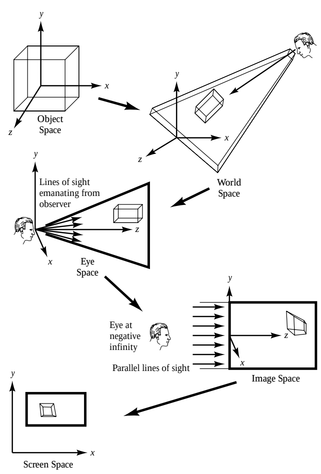
CENG 732
Computer Animation

Spring 2006-2007
Week 2
Technical Preliminaries and
Introduction to Keyframing

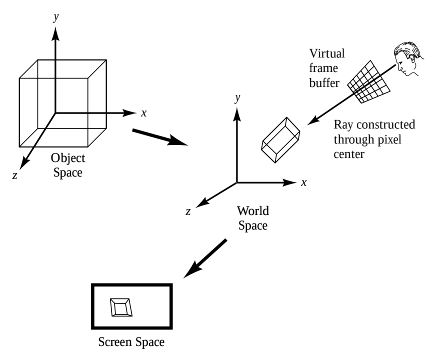
This week

- Recap from CEng 477
  - The Display Pipeline
  - Basic Transformations / Composite Transformations
- Round-off Error Considerations
- Orientation representations
- Basic Orientation Interpolation Example

The Display Pipeline



Ray Casting Display Pipeline



Animation

- Animation is typically produced by the following:
  - Modifying the position and orientation of objects in world space over time; modifying the shape of objects over time; modifying display attributes of objects over time; transforming the observer position and orientation in world space over time; or some combination of these transformations

Applying Transformations to Points

- Points are represented in homogenous coordinates and the transformation matrix is left multiplied by the column vector that represents the point

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} a & b & c & d \\ e & f & g & h \\ i & j & k & m \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Composite Transformations

- A series of transformations can be multiplied together to produce a compound (or composite) transformation.

$$P' = M_1 M_2 M_3 M_4 M_5 M_6 P$$
$$M = M_1 M_2 M_3 M_4 M_5 M_6$$
$$P' = MP$$

Basic Transformations

- Translation
- Scaling
- Rotations around major axes

Translation

$$\begin{bmatrix} x + t_x \\ y + t_y \\ z + t_z \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Scaling

$$\begin{bmatrix} S_x \cdot x \\ S_y \cdot y \\ S_z \cdot z \\ 1 \end{bmatrix} = \begin{bmatrix} S_x & 0 & 0 & 0 \\ 0 & S_y & 0 & 0 \\ 0 & 0 & S_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Rotation around x-axis

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Rotation around y-axis

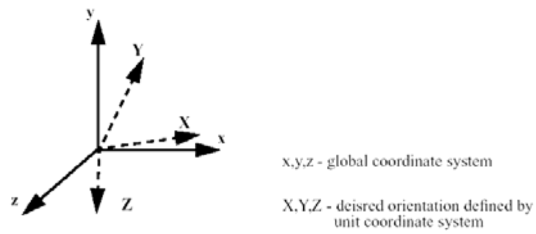
$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & 0 & \sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Rotation around z-axis

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Rotations: an alternative method

- The desired rotation defines a unit coordinate system

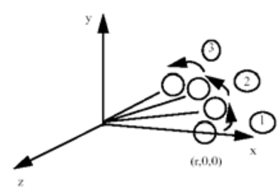


Extracting Transformations from a Matrix

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} A_{11} & A_{12} & A_{13} & A_{14} \\ A_{21} & A_{22} & A_{23} & A_{24} \\ A_{31} & A_{32} & A_{33} & A_{34} \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Round-off Errors

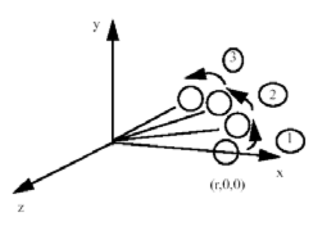
- Assume you want to rotate a sphere around the origin.
- How would you do that?



Three different ways

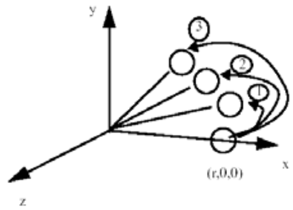
- Apply a delta y-axis rotation to the points on the sphere each frame
- Apply a delta y-axis rotation to the transformation matrix and then apply it to the points
- Add a delta value to an angle variable and construct the transformation matrix from scratch each frame

Approach 1



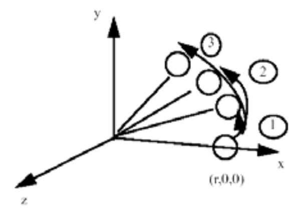

```
for each point P of the moon {
    P' = P
}
Rdy= y-axis rotation of 5 degrees
repeat until (done) {
   for each point P' of the moon {
       P' = Rdy*P'
   }
   record a frame of the animation
}
```

Approach 2




```
R = identity matrix
Rdy= y-axis rotation of 5 degrees
repeat until (done) {
   for each point P of the moon {
        P' = R*P
   }
   record a frame of the animation
   R = R*Rdy
}
```

Approach 3




```
y = 0
repeat until (done) {
   R = y-axis rotation matrix of 'y' degrees
   for each point P of the moon {
        P' = R*P
   }
   record a frame of the animation
   y = y+5
}
```

Orientation Representation

- How do we represent the arbitrary orientation of an object in 3D space?
- Does that representation allow for interpolation if we want to interpolate the in-between frames of two given key-frames (key-orientations) of the object?

Orientation Representation

- Transformation Matrix Representation
- Fixed Angle Representation
- Euler Angle Representation
- Axis-Angle Representation
  - Example on Axis-Angle Representation
- Quaternion Representation

Transformation Matrix Representation

$$\begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ -1 & 0 & 0 \end{bmatrix}$$

a) Positive 90 degree y-axis rotation

$$\begin{bmatrix} 0 & 0 & -1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix}$$

b) Negative 90 degree y-axis rotation

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

c) Half way between orientation representations

Fixed Angle Representation

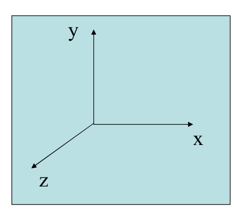
Rotate about global axes in a fixed order

Rotating about global axes is what the rotation matrices do

Can use any triple of axes

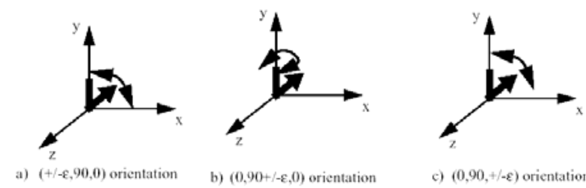
Rotate about x, then y, then z

(10, 90, -45)



Fixed Angle Representation

Gimbal Lock

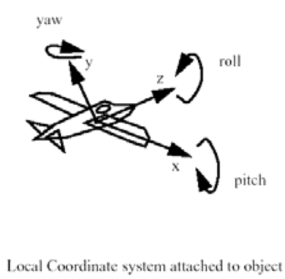


Euler Angle Representation

Rotate about local axes of object

Roll, Pitch, Yaw

(10, 90, -45)



Equivalence of Fixed angles and Euler angles

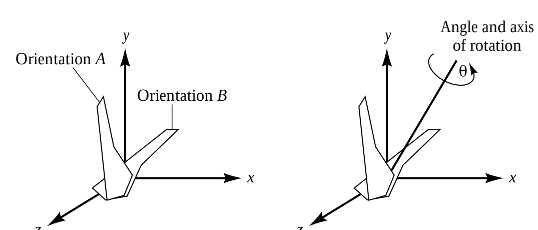
$$R_y'(\beta)R_x(\alpha) = R_x(\alpha)R_y(\beta)R_x(\alpha)R_x(-\alpha) = R_x(\alpha)R_y(\beta)$$

$$R_z''(\gamma)R_y'(\beta)R_x(\alpha) = R_x(\alpha)R_y(\beta)R_z(\gamma)R_y(-\alpha)R_x(-\beta)R_y(\beta)R_x(\alpha) = R_x(\alpha)R_y(\beta)R_z(\gamma)$$

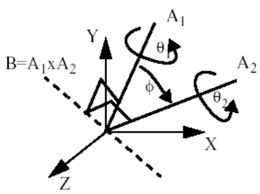
Angle and Axis Representation

- Euler's rotation theorem
  - One orientation can be derived from another by a single rotation about an axis
- So, we can use an axis and a single angle to represent an orientation (with respect to the object's initial orientation)
- We can implement interpolation in this representations

Euler's Theorem



Interpolation Using Axis-Angle Representation



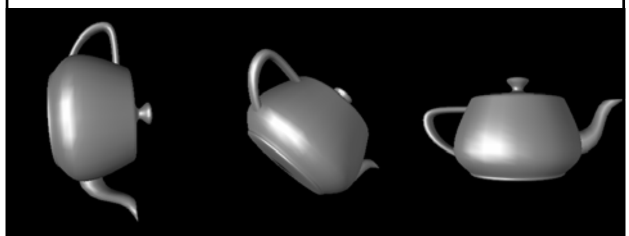
$$B = A_1 \times A_2$$

$$\phi = \cos^{-1}\left(\frac{A_1 \bullet A_2}{|A_1||A_2|}\right)$$

$$A_k = R_B(k \cdot \phi)A_1$$

$$\theta_k = (1-k) \cdot \theta_1 + k \cdot \theta_2$$

Example



Quaternions

- Similar to axis-angle representations quaternions can be used to represent orientation with four values (a scalar and a 3D vector)

[s,x,y,z] or [s,***v***]

Representing Rotations Using Quaternions

$$q = Rot_{\theta,(x,y,z)} = [\cos(\theta/2), \sin(\theta/2) \cdot (x,y,z)]$$

$$\begin{aligned} -q &= Rot_{-\theta,-(x,y,z)} \\ &= [\cos(-\theta/2), \sin((-\theta)/2) \cdot (-(x,y,z))] \\ &= [\cos(\theta/2), -\sin(\theta/2) \cdot (-(x,y,z))] \\ &= [\cos(\theta/2), \sin(\theta/2) \cdot x, y, z] \\ &= Rot_{\theta,(x,y,z)} \\ &= q \end{aligned}$$

Basic Quaternion Math

$[s_1,v_1]+[s_2,v_2] = [s_1+s_2,v_1+v_2]$

$$[s_1, v_1] \cdot [s_2, v_2] = [s_1 \cdot s_2 - v_1 \bullet v_2, s_1 \cdot v_2 + s_2 \cdot v_1 + v_1 \times v_2$$

$$[0, v_1] \cdot [0, v_2] = [0, v_1 \times v_2] \qquad \text{iff } v_1 \bullet v_2 = 0$$

$$q^{-1} = (1/\|q\|)^2 \cdot [s, -v]$$

$$where \qquad \|q\| = \sqrt{s^2 + x^2 + y^2 + z^2}$$

Unit-length Quaternion

$$q/(\|q\|)$$

Quaternions

- Quaternion representation both allow for interpolation between arbitrary orientations and for representation of a series of rotations

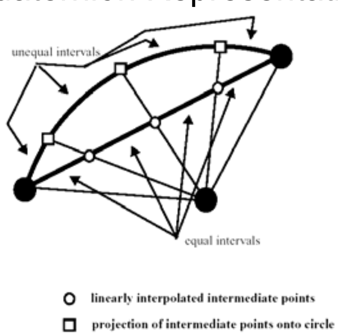
Rotating Vectors Using Quaternions

$$v' = Rot(v) = q^{-1} \cdot v \cdot q$$

$$Rot_q(Rot_p(v)) = q^{-1} \cdot (p^{-1} \cdot v \cdot p) \cdot q$$
$$= ((pq)^{-1} \cdot v \cdot (pq))$$
$$= Rot_{pq}(v)$$

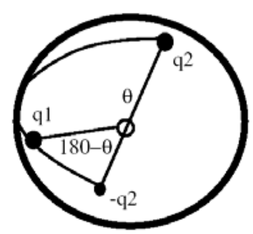
$$Rot^{-1}(Rot(v)) = q \cdot (q^{-1} \cdot v \cdot q) \cdot q^{-1} = v$$

Interpolation of Rotations using Quaternion Representation



Interpolation of Rotations using Quaternion Representation

$$\cos\theta = q1 \bullet q2 = s1 \cdot s2 + v1 \bullet v2$$



$$Slerp(q1, q2, u) = ((\sin((1-u) \cdot \theta))/(\sin\theta)) \cdot q1 + (\sin(u \cdot \theta))/(\sin\theta) \cdot q2$$