

# Affine Transformations

## Shear Transformation

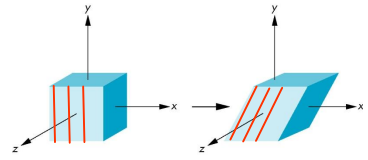
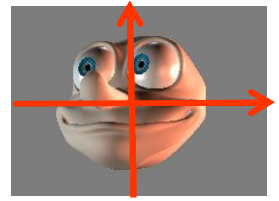
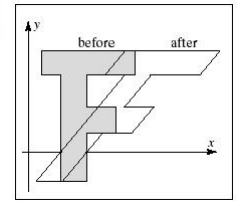
- ❖ Along x-direction
- ❖ 2D matrix form

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

- ❖ 3D

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

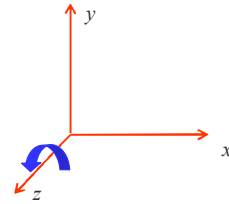
- ❖ it is an Affine map



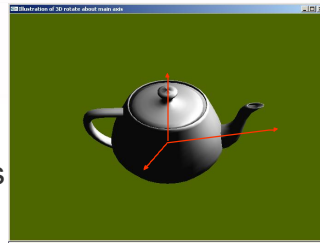
## Rotation in 3D

- ❖ Rotation about a coordinate axis
- ❖ z-axis
- ❖ Simple extension of planner rotation
- ❖ Matrix form

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} c & -s & 0 & 0 \\ s & c & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$



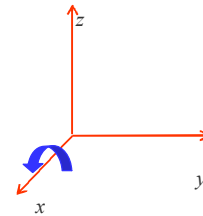
- ❖ where  $c$  denotes  $\cos \theta$  and  $s$  denotes  $\sin \theta$
- ❖  $P' = R_z(\theta).P$



## Rotation about x-axis and y-axis

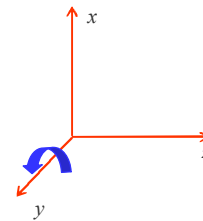
$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & c & -s & 0 \\ 0 & s & c & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

$$P' = R_x(\theta).P$$



$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} c & 0 & s & 0 \\ 0 & 1 & 0 & 0 \\ -s & 0 & c & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

$$P' = R_y(\theta).P$$



## Composing Affine Transformations

- ❖ Composing or concatenating the transformation
- ❖ Concatenation of two Affine transformations is also Affine
- ❖ Functional form and matrix form

$$P'' = T_2(P') = T_2(T_1(P)) = (T_2T_1)(P) = T(P)$$

$$P'' = M_2P' = M_2M_1P = MP$$

$$M = M_2M_1$$

- ❖ Matrix multiplication is enough
- ❖ Reverse order
- ❖ Example

## 2D Rotating About an Arbitrary Point

- ❖ Pivot point:  $V = (V_x, V_y)$
- ❖ New trigonometric formula or a combined transformation?
- ❖ Steps
  1. Translate through  $(-V_x, -V_y)$
  2. Rotate about the origin through angle  $\theta$
  3. Translated back through  $(V_x, V_y)$
- ❖ The matrix form:

$$\begin{bmatrix} 1 & 0 & V_x \\ 0 & 1 & V_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos \theta & -\sin \theta & 0 \\ \sin \theta & \cos \theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & -V_x \\ 0 & 1 & -V_y \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \cos \theta & -\sin \theta & dx \\ \sin \theta & \cos \theta & dy \\ 0 & 0 & 1 \end{bmatrix}$$

where

$$\begin{aligned} dx &= -\cos \theta V_x + \sin \theta V_y + V_x \\ dy &= -\sin \theta V_x - \cos \theta V_y + V_y \end{aligned}$$

## 3D Example

- ❖ Similar steps for scaling and shearing about a pivot point
- ❖ 3D example: scaling about arbitrary pivot point
  - Scaling factors:  $S_x, S_y, S_z$
  - Pivot point:  $(V_x, V_y, V_z)$
  - Steps:
    1. Translate through  $(-V_x, -V_y, -V_z) : M_1$
    2. Scale about the origin,  $M_2$
    3. Translate back through  $(V_x, V_y, V_z) : M_3$

$$M = M_3 M_2 M_1$$

## 3D Rotation About a Fixed Point

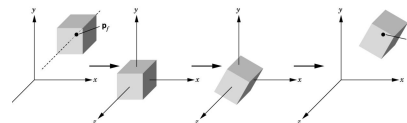
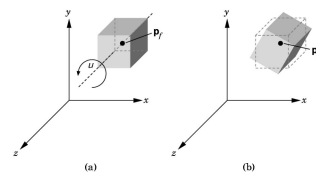
- ❖ Fixed point,  $P_f$
- ❖ Steps: (for z-axis)
  - Translate through  $-P_f, M_1$
  - Rotate about the z-axis,  $M_2$
  - Translate back through  $P_f, M_3$

$$M = M_3 M_2 M_1$$

$$\begin{bmatrix} \cos \theta & -\sin \theta & 0 & dx \\ \sin \theta & \cos \theta & 0 & dy \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

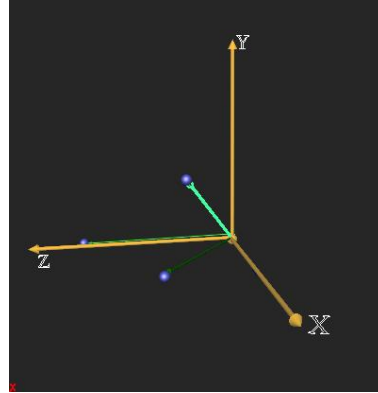
where

$$\begin{aligned} dx &= x_f - x_f \cos \theta + y_f \sin \theta \\ dy &= y_f - x_f \sin \theta - y_f \cos \theta \end{aligned}$$



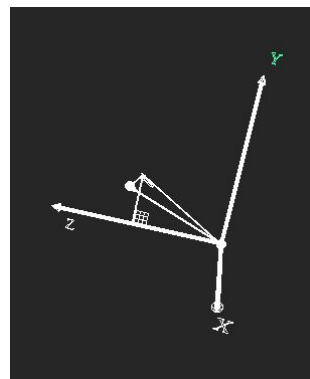
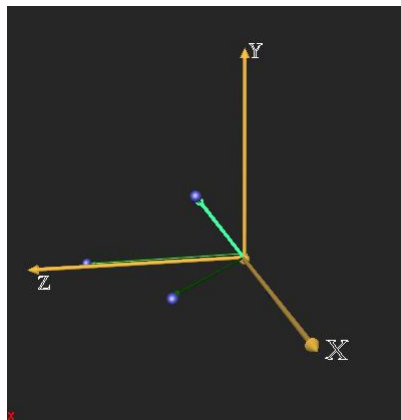
## 3D rotation about arbitrary axis

- ❖ Input:
  - An axis: a vector  $u$  and a fixed point (e.g. The origin)
  - An angle:
- ❖ Rotation is a rigid body transformation
- ❖ output



$$R_u(\theta) = R_x(-\theta_x)R_y(-\theta_y)R_z(\theta)R_y(\theta_y)R_x(\theta_x)$$

## Projection



Example :  $u = \left(\frac{1}{\sqrt{3}}, \frac{1}{\sqrt{3}}, \frac{1}{\sqrt{3}}\right)$  ,  $\theta = 60^\circ$  , Find  $R_u(\theta)$

Solution:  $u_x = u_y = u_z = \frac{1}{\sqrt{3}}$  and  $d = \sqrt{u_x^2 + u_y^2} = \sqrt{\frac{2}{3}}$

$\cos(\theta_y) = d = \sqrt{\frac{2}{3}}$  ,  $\sin(\theta_y) = u_x = \frac{1}{\sqrt{3}}$

$\cos(\theta_x) = \frac{u_z}{d} = \frac{1}{\sqrt{2}}$  ,  $\sin(\theta_x) = \frac{u_y}{d} = \frac{1}{\sqrt{2}}$

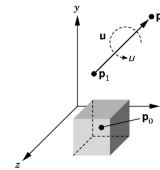
$$R_x(\theta_x) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & 0 \\ 0 & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} , R_y(\theta_y) = \begin{bmatrix} \sqrt{\frac{2}{3}} & 0 & -\frac{1}{\sqrt{3}} & 0 \\ 0 & 1 & 0 & 0 \\ \frac{1}{\sqrt{3}} & 0 & \sqrt{\frac{2}{3}} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$R_z(\theta) = R_z(60^\circ) = \begin{bmatrix} \frac{1}{\sqrt{2}} & -\frac{\sqrt{3}}{2} & 0 & 0 \\ \frac{\sqrt{3}}{2} & \frac{1}{2} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$R_u(\theta) = R_x(-\theta_x) R_y(-\theta_y) R_z(\theta) R_y(\theta_y) R_x(\theta_x)$$

## Rotation About an Arbitrary axis

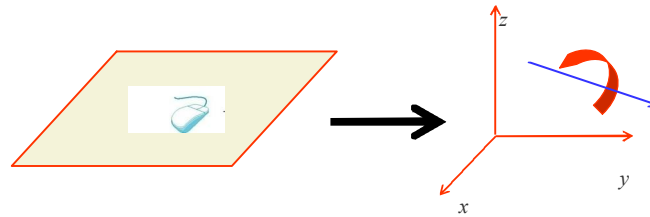
- ❖ A fixed point:  $P_0$
- ❖ An arbitrary axis:  $u = P_2 - P_1$
- ❖ The angle of rotation:  $\theta$
- ❖ Normalized version of  $u$  :  $V = (\alpha_x, \alpha_y, \alpha_z)$
- ❖ Steps:
  - o Translate through  $-P_0$
  - o Rotate  $\theta$  about  $u$
  - o Translate through  $P_0$



$$M = T(P_0)R_u(\theta)T(-P_0)$$

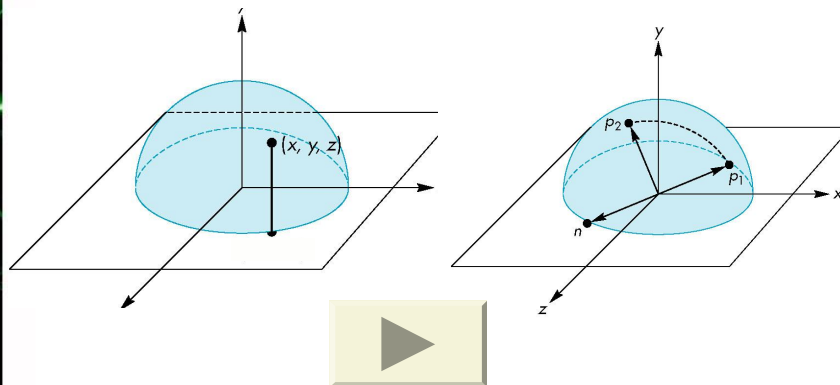
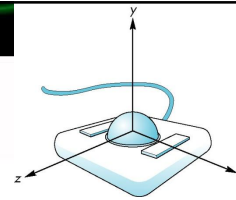
## A better interface for 3d rotation

- ❖ How can we extract a 3d vector and the rotation angle from a 2D mouse movement?



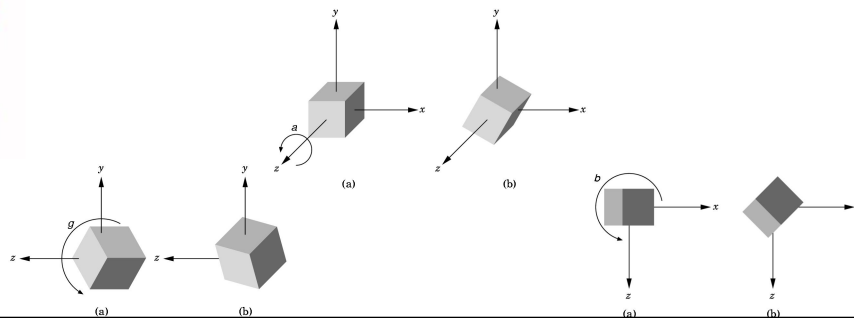
## A Virtual trackball

- ❖ Unproject 2D points onto the unit sphere



## General Rotation

- ❖ Three successive rotations about three axis
- ❖ The order is not unique
- ❖ The resulting rotation matrix is unique!
- ❖  $R = R_x R_y R_z$

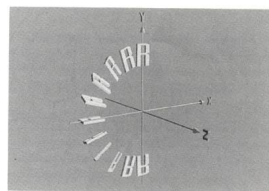
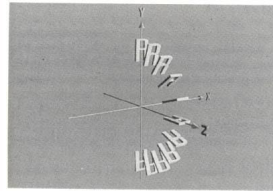
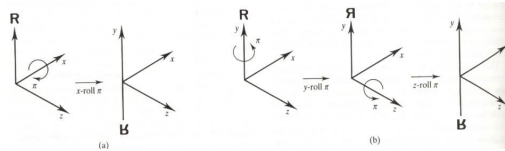


## Euler Angles

- ❖ Result of Euler theorem:  
Any 3d can be obtained by several rolls about x,y and z
- ❖ Euler angles
- ❖ Problem of steady rotations in 3D

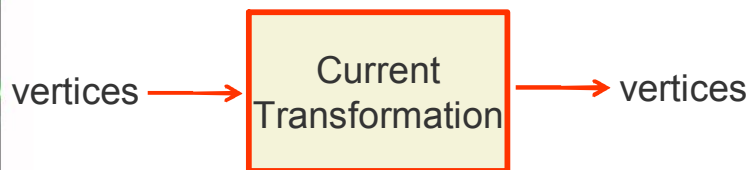
## Example

1. A single rotation:  $x$ -roll of  $\pi$
  2. First perform a  $y$ -roll of  $\pi$  then a  $z$ -roll of  $\pi$
  3. A lot of other possibilities
- ❖ Intermediate frames for 1 and 2



## OpenGL Transformation Matrices

- ❖ Current Transformation Matrix (CTM)
- ❖ Initially is set to the 4x4 identity matrix





## OpenGL pre-defined Transformations

- ❖ Translation: `glTranslate{fd}(dx,dy,dz)`
- ❖ Scaling: `glScale{fd}(sx,sy,sz)`
- ❖ Rotation: about arbitrary vector but with the origin as fixed point
- ❖ `glRotate{fd}(angle, dx,dy,dz)`
- ❖ No other specific functions
- ❖ `glLoadMatrix{fd}(m)`: setting transformation matrix directly

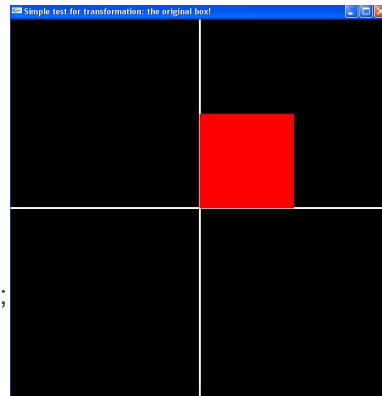


## Concatenation in OpenGL

- ❖ OpenGL's rule:  
The transformation specified last is the one applied first

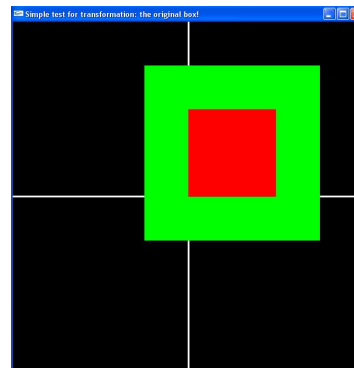
## A Simple OpenGL experiment

```
void draw_box(){
    glBegin(GL_POLYGON);
        glVertex2f(0.0,0.0);
        glVertex2f(0.0, 0.5);
        glVertex2f(0.5, 0.5);
        glVertex2f(0.5, 0.0);
    glEnd();
}
void display(void){
    glClear(GL_COLOR_BUFFER_BIT);
    glLoadIdentity();
    glColor3f(1.0, 0.0, 0.0);
    draw_box();
    glFlush();
}
```



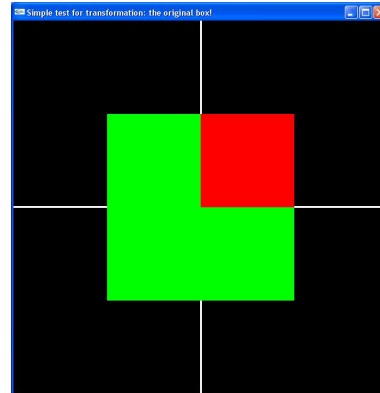
## What is the output?

```
void display(void){
    glClear(GL_COLOR_BUFFER_BIT);
    glLoadIdentity();
    glColor3f(1.0, 1.0, 1.0);
    glTranslatef(-0.25, -0.25, 0);
    glScalef(2.0, 2.0, 0.0);
    glColor3f(0.0, 1.0, 0.0);
    draw_box();
    glLoadIdentity();
    glColor3f(1.0, 0.0, 0.0);
    draw_box();
    glFlush();
}
```



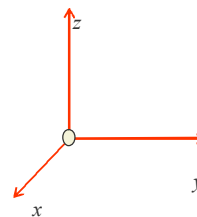
## And this?!

```
void display(void){  
    glClear(GL_COLOR_BUFFER_BIT);  
    glLoadIdentity();  
    glColor3f(1.0, 1.0, 1.0);  
    glScalef(2.0,2.0,0.0);  
    glTranslatef(-0.25, -0.25,0);  
    glColor3f(0.0, 1.0, 0.0);  
    draw_box();  
    glLoadIdentity();  
    glColor3f(1.0, 0.0, 0.0);  
    draw_box();  
    glFlush();  
}
```



## Coordinate system and frame

- ❖ Orthonormal basis for vectors
- ❖ Reference point for affine space (points)
- ❖ Vector basis + a reference point = frame
- ❖ Coordinate system (frame) changing
- ❖ The benefit!?



## Local and World coordinates

