Computer Graphics

(Affine Transformations)

Thilo Kielmann
Fall 2003
Vrije Universiteit, Amsterdam
kielmann@cs.vu.nl

http://www.cs.vu.nl/~graphics/

Reminder: Homogeneous Coordinates

$$P = [\alpha_1 \ \alpha_2 \ \alpha_3 \ 1] \begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ P_0 \end{bmatrix} \qquad p = \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \\ 1 \end{bmatrix}$$

$$w = [\delta_1 \ \delta_2 \ \delta_3 \ 0] \begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ P_0 \end{bmatrix} \qquad a = \begin{bmatrix} \delta_1 \\ \delta_2 \\ \delta_3 \\ 0 \end{bmatrix}$$

Reminder: Matrix Representation

Two frames: (v_1, v_2, v_3, P_0) and (u_1, u_2, u_3, Q_0)

$$\begin{aligned} u_1 &= \gamma_{1,1}v_1 + \gamma_{1,2}v_2 + \gamma_{1,3}v_3 \\ u_2 &= \gamma_{2,1}v_1 + \gamma_{2,2}v_2 + \gamma_{2,3}v_3 \\ u_3 &= \gamma_{3,1}v_1 + \gamma_{3,2}v_2 + \gamma_{3,3}v_3 \\ Q_0 &= \gamma_{4,1}v_1 + \gamma_{4,2}v_2 + \gamma_{4,3}v_3 + P_0 \end{aligned}$$

$$M = \begin{bmatrix} \gamma_{1,1} & \gamma_{1,2} & \gamma_{1,3} & 0 \\ \gamma_{2,1} & \gamma_{2,2} & \gamma_{2,3} & 0 \\ \gamma_{3,1} & \gamma_{3,2} & \gamma_{3,3} & 0 \\ \gamma_{4,1} & \gamma_{4,2} & \gamma_{4,3} & 1 \end{bmatrix}$$

Change Between Matrix Representations

$$\begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ Q_0 \end{bmatrix} = M \begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ P_0 \end{bmatrix}$$

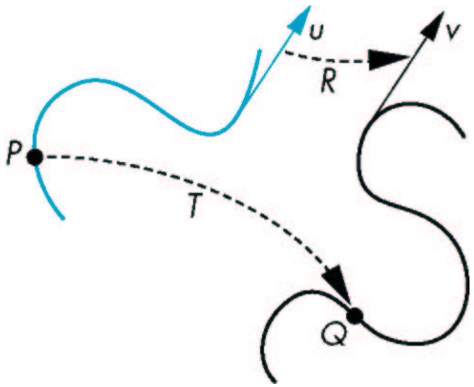
$$a = M^T b \qquad b = Aa = (M^T)^{-1} a$$

Goal: rotation/scaling/translation
by changing homogeneous coordinate systems!

Outline for today

- **Affine** Transformations
- Rotation, Translation, Scaling
- . . . in Homogeneous Coordinates
- Concatenation of Transformations
- OpenGL Transformation Matrices

Affine Transformations



$Q = T(P) \qquad v = R(u)$

homogeneous coordinates: $Q = f(P) \qquad v = f(u)$

Linear Functions

$f()$ is a *linear function iff:* $f(\alpha p + \beta q) = \alpha f(p) + \beta f(q)$

E.g., a linear function maps a line to a line:

$f(P(\alpha)) = f(P_0 + \alpha d) = f(P_0) + \alpha f(d)$

For non-singular (invertible) matrices A:
$Ap(\alpha) = Ap_0 + \alpha Ad \qquad \Rightarrow$ they define linear functions!

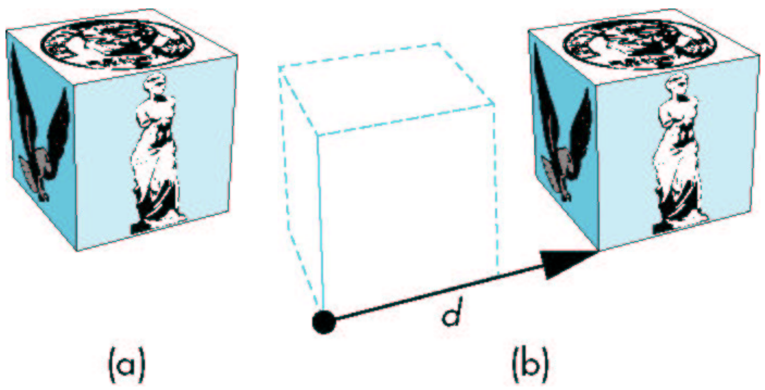
Linear Functions (2)

Two views:

1. change in the underlying frame
   yields a new representation of vertices

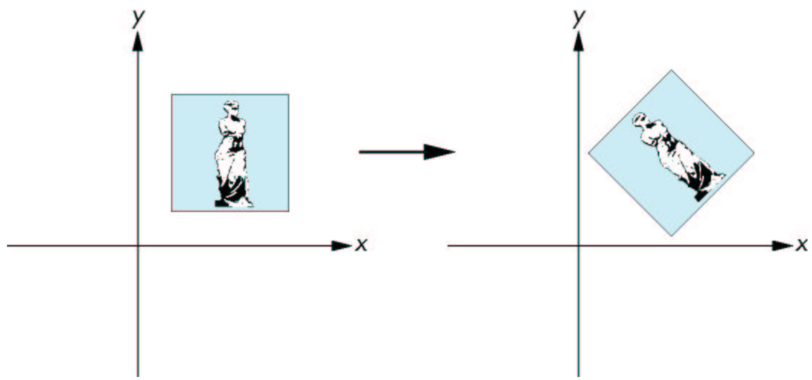
2. transformation of the vertices within the same frame

Translation



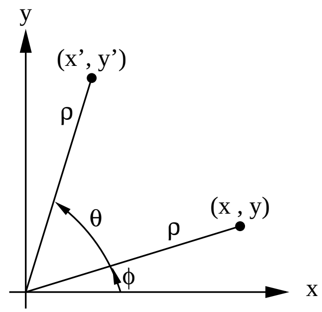
$$P' = P + d$$

Rotation



rotation (in 2D) about a fixed point

2D Rotation about (0,0)



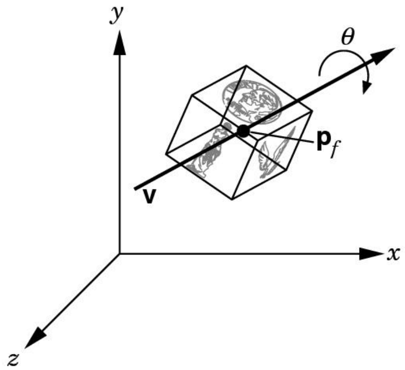
$$x = \rho \cos \phi$$
$$y = \rho \sin \phi$$
$$x' = \rho \cos(\phi + \theta)$$
$$y' = \rho \sin(\phi + \theta)$$

$$x' = \rho \cos \phi \cos \theta - \rho \sin \phi \sin \theta = x \cos \theta - y \sin \theta$$
$$y' = \rho \cos \phi \sin \theta + \rho \sin \phi \cos \theta = x \sin \theta + y \cos \theta$$

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

$\longrightarrow$

3D Rotation

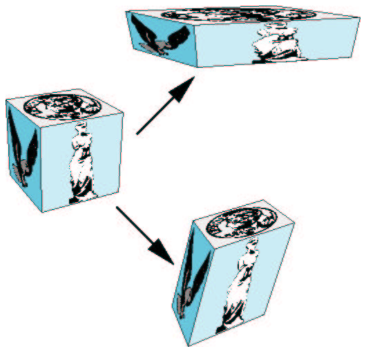


3D rotation requires vector, fixed point, and angle

2D rotation about $(0, 0)$ is 3D rotation about z axis

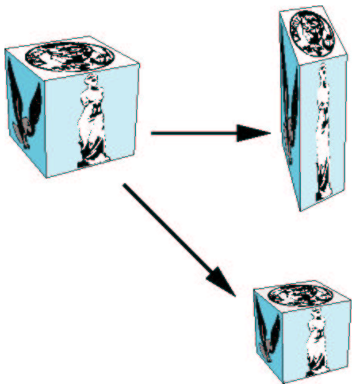
Rigid-body Transformations

Translation and rotation keep the shape of an object:
They are rigid-body transformations.



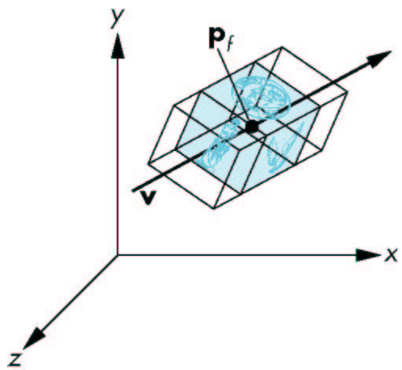
(examples for non-rigid-body transformations)

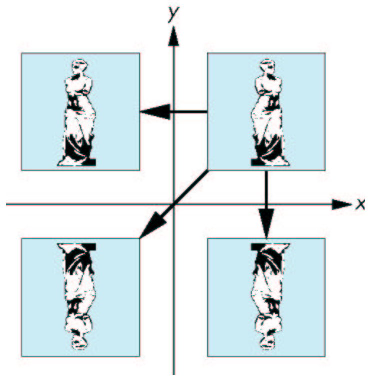
Scaling



scaling is an affine, non-rigid-body transformation,
necessary for modeling and viewing

Scaling (2)





scaling requires vector, fixed point, and scaling factor α
$\alpha > 1$: bigger, $0 \le \alpha < 1$: smaller, $\alpha < 0$: reflection

Translation in Homogeneous Coordinates

$$p' = p + d$$
$$p = [x \;\; y \;\; z \;\; 1]^T \quad p' = [x' \;\; y' \;\; z' \;\; 1]^T \quad d = [\alpha_x \;\; \alpha_y \;\; \alpha_z \;\; 0]^T$$

$$x' = x + \alpha_x$$
$$y' = y + \alpha_y$$
$$z' = z + \alpha_z$$

$$p' = Tp, \qquad T = \begin{bmatrix} 1 & 0 & 0 & \alpha_x \\ 0 & 1 & 0 & \alpha_y \\ 0 & 0 & 1 & \alpha_z \\ 0 & 0 & 0 & 1 \end{bmatrix} = T(\alpha_x, \alpha_y, \alpha_z)$$

Inverting Translation

$$T^{-1}(\alpha_x, \alpha_y, \alpha_z) = T(-\alpha_x, -\alpha_y, -\alpha_z) = \begin{bmatrix} 1 & 0 & 0 & -\alpha_x \\ 0 & 1 & 0 & -\alpha_y \\ 0 & 0 & 1 & -\alpha_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Scaling in Homogeneous Coordinates

Fixed point at the origin $(0, 0, 0)$:
(Later, we will use concatenation for scaling at arbitrary points.)

$x' = \beta_x x$
$y' = \beta_y y$
$z' = \beta_z z$

$$p' = Sp, \qquad S = \begin{bmatrix} \beta_x & 0 & 0 & 0 \\ 0 & \beta_y & 0 & 0 \\ 0 & 0 & \beta_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = S(\beta_x, \beta_y, \beta_z)$$

Inverting Scaling

$$S^{-1}(\beta_x, \beta_y, \beta_z) = S(\tfrac{1}{\beta_x}, \tfrac{1}{\beta_y}, \tfrac{1}{\beta_z})$$

Rotation in Homogeneous Coordinates

- Final goal: rotate around arbitrary vector and fixed point
- Let's start with rotation around an axis and the origin. . .

Rotation around z axis ("2D rotation")

$$x' = x\cos\theta - y\sin\theta$$
$$y' = x\sin\theta + y\cos\theta$$
$$z' = z$$

$\leftarrow$

$$p' = R_z p, \qquad R_z = R_z(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Rotation around axes (2)

$$R_z = R_z(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$R_x = R_x(\theta) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$R_y = R_y(\theta) = \begin{bmatrix} \cos\theta & 0 & \sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

(Signs are consistent with positive rotation in right-handed system.) →x →y

Inverting Rotation

$R^{-1}(\theta) = R(-\theta)$ (for x, y, and, z)

$\cos(-\theta) = \cos(\theta) \qquad \sin(-\theta) = -\sin(\theta)$

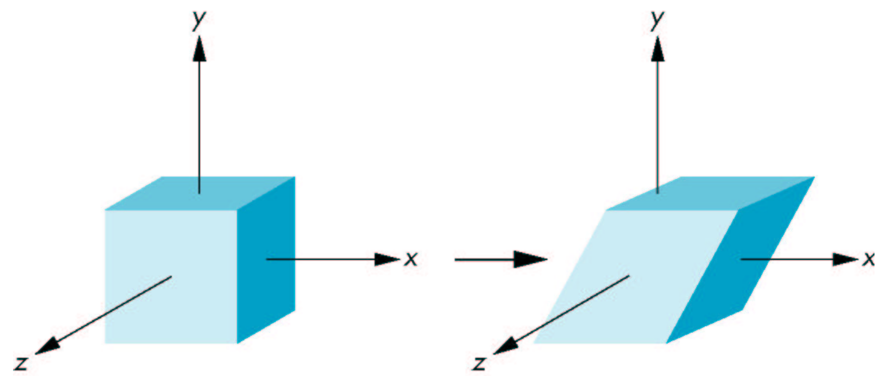
$$\Longrightarrow R^{-1}(\theta) = R^T(\theta)$$

composed rotation:

$R(\theta) = R_z(\alpha)R_y(\beta)R_x(\gamma)$

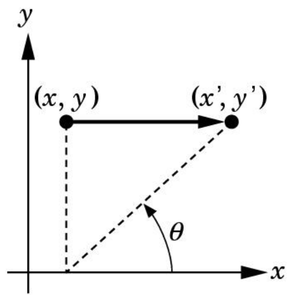
finding α, β, γ can be tricky. . .

Excursion: Shear



"Pull the top to the right and the bottom to the left."

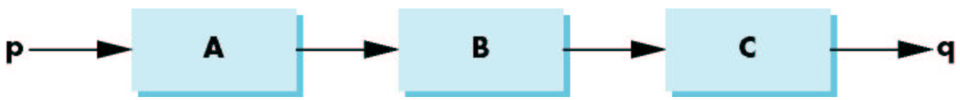
The Shear Matrix



$$x' = x + y\cot\theta$$
$$y' = y$$
$$z' = z$$

$$H_x(\theta) = \begin{bmatrix} 1 & \cot\theta & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad H_x^{-1}(\theta) = H_x(-\theta)$$

Concatenation of Transformations

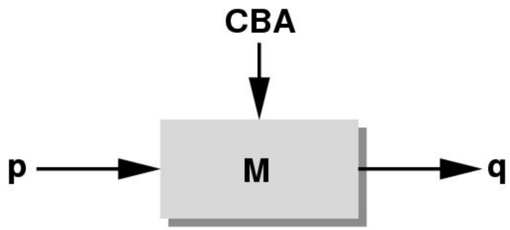


$q = CBAp$

for a single point: $q = (C(B(Ap)))$

this always multiplies a column matrix with a square matrix

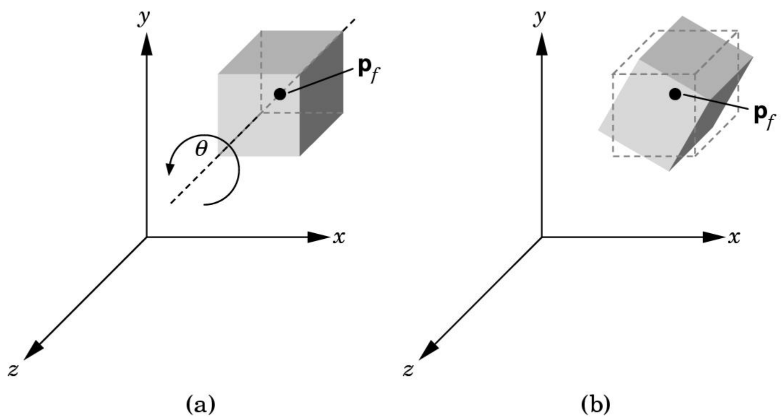
Concatenation of Transformations (2)



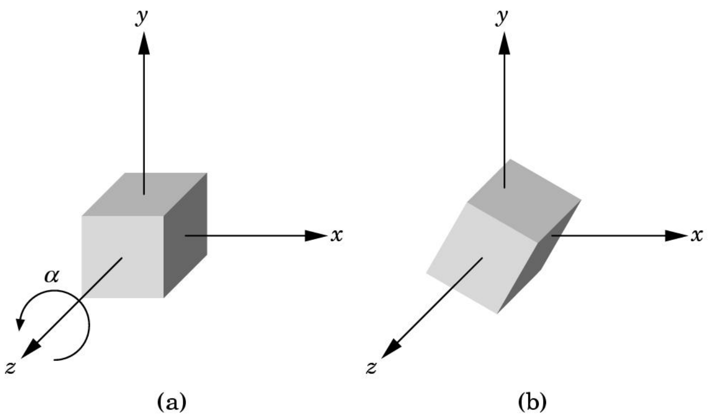
for many points: $M = CBA \qquad q_i = Mp_i$

efficient execution in pipeline!

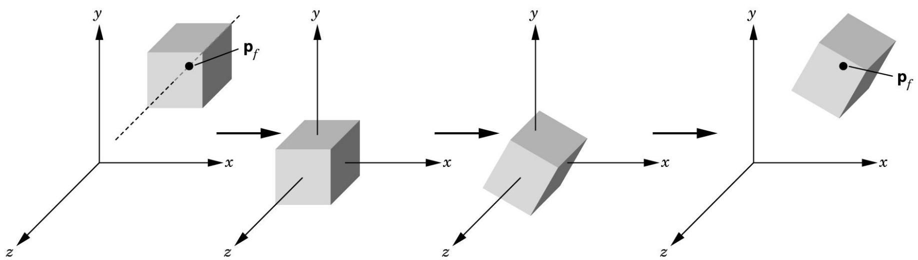
Rotation about a Fixed Point



. . . we already know this

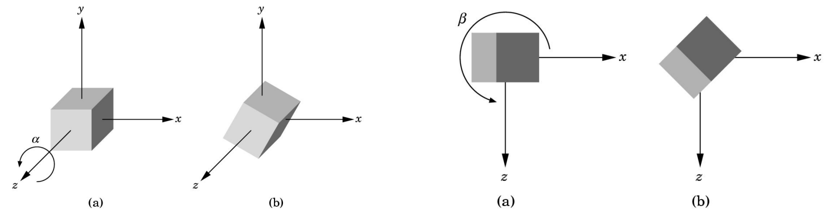


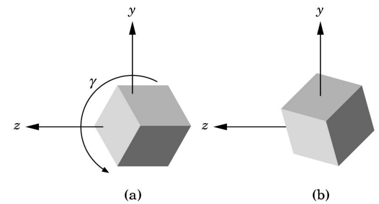
Do it in three steps



$$M = T(p_f)R_z(\theta)T(-p_f)$$

Composing a Rotation

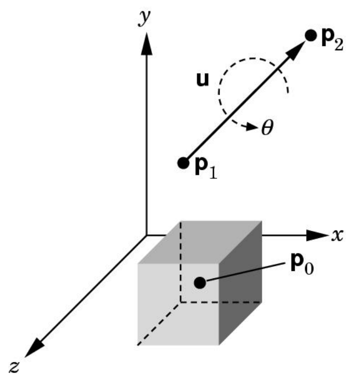




$$R(\theta) = R_x(\alpha)R_y(\beta)R_z(\gamma)$$

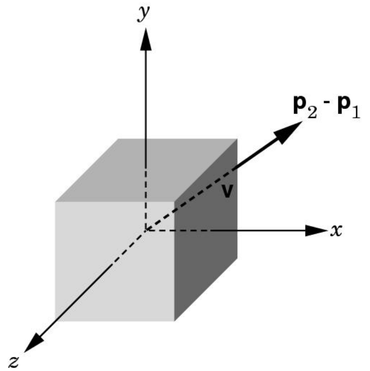
actually finding α, β, and γ isn't trivial

Rotation About an Arbitrary (Rot.) Axis



Rotation is specified by the vector $u = P_2 - P_1$, the fixed point P_0, and the angle θ.

Arbitrary Rotation (2)



$v = \frac{u}{|u|} = \begin{bmatrix} \alpha x \\ \alpha y \\ \alpha z \end{bmatrix}$ ("**normalized**," unit-length vector)

First translate to the origin: $M = T(p_0)RT(-p_0)$

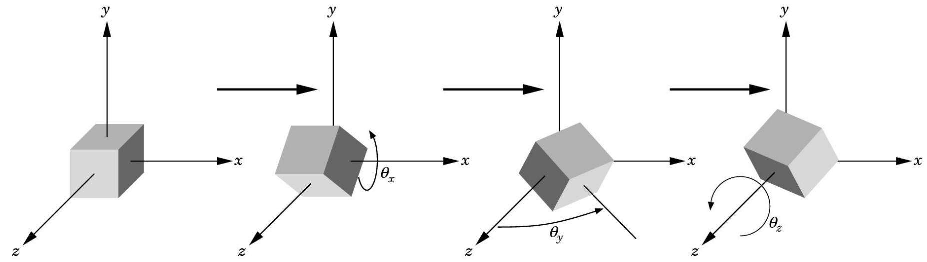
Arbitrary Rotation (3)

We can compose an arbitrary rotation from rotations about the three axes.
Problem: finding the three angles from θ

Strategy: align the rotation axis v with the z axis (two rotations). Then, rotate by θ around z, and undo the two alignment rotations.

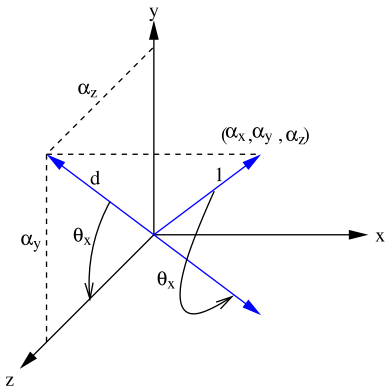
$$R = R_x(-\theta_x)R_y(-\theta_y)R_z(\theta)R_y(\theta_y)R_x(\theta_x)$$

Sequence of Rotations

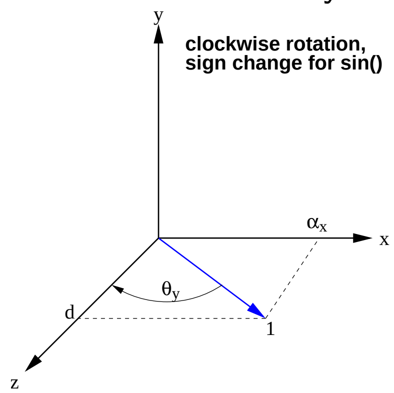


Alignment of Rotation Vector with z Axis

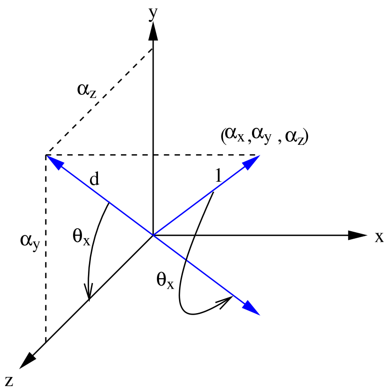
1. rotate about x

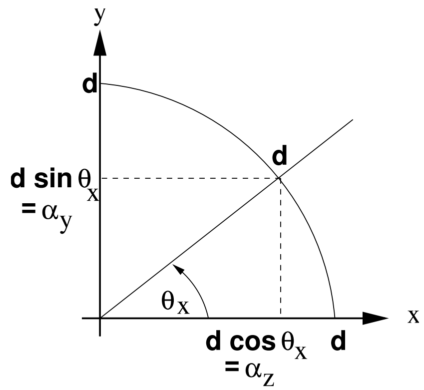


2. rotate about y



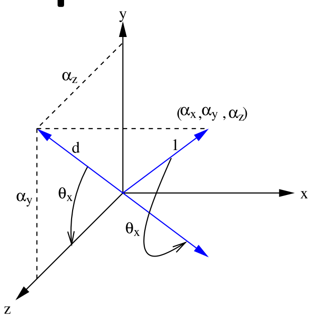
Computation of the x-Rotation





$$d = \sqrt{\alpha_y^2 + \alpha_z^2} \qquad \cos\theta_x = \alpha_z/d \qquad \sin\theta_x = \alpha_y/d$$

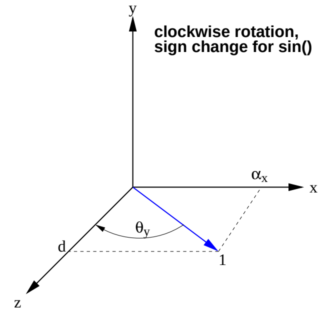
Computation of the x-Rotation (2)



$$\cos\theta_x = \alpha_z/d$$
$$\sin\theta_x = \alpha_y/d$$

$$R_x(\theta_x) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \alpha_z/d & -\alpha_y/d & 0 \\ 0 & \alpha_y/d & \alpha_z/d & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad \leftarrow$$

Computation of the y-Rotation



$$R_y(\theta_y) = \begin{bmatrix} d & 0 & -\alpha_x & 0 \\ 0 & 1 & 0 & 0 \\ \alpha_x & 0 & d & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad \leftarrow$$

. . . and finally:

$$M = T(p_0)R_x(-\theta_x)R_y(-\theta_y)R_z(\theta)R_y(\theta_y)R_x(\theta_x)T(-p_0)$$

- Once computed, M does the complex rotation with a single matrix multiplication per vertex.
- OpenGL gives us a simple function that computes M for us. (except for the translation)

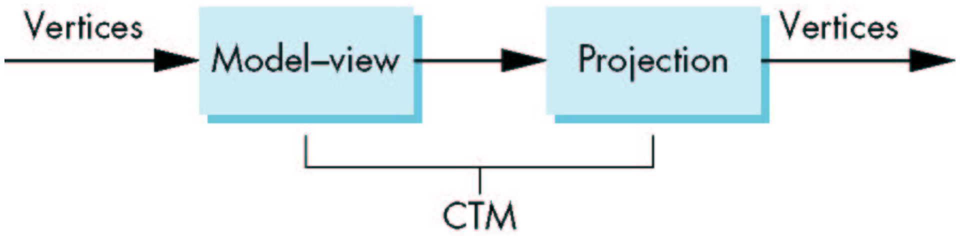
OpenGL Transformation Matrices

Current Transformation Matrix (CTM)

The CTM is a 4×4 matrix.

Operations:
$C \leftarrow I \quad C \leftarrow CT \quad C \leftarrow CS \quad C \leftarrow CR$
$C \leftarrow M \quad C \leftarrow CM$

Model-View and Projection Matrices



	`glMatrixMode(GL_MODELVIEW);`
$C \leftarrow I$	`glLoadIdentity();`
$C \leftarrow M$	`glLoadMatrixf(pointer_to_matrix);`
$C \leftarrow CM$	`glMultMatrixf(pointer_to_matrix);`

Direct Rotation/Translation/Scaling

$C \leftarrow CR$	`glRotatef( angle, vx, vy, vz );`
$C \leftarrow CT$	`glTranslatef( dx, dy, dz );`
$C \leftarrow CS$	`glScalef( sx, sy, sz );`

 (transformation tutor)

Composed Transformation

```
glMatrixMode(GL_MODELVIEW);
glLoadIdentity();
glTranslatef(4.0, 5.0, 6.0);
glRotatef(45.0, 1.0, 2.0, 3.0);
glTranslatef(-4.0, -5.0, -6.0);
```

Q: What kind of operation is this?

A: Rotation about a fixed point

Q: Which point? (4.0, 5.0, 6.0) or (-4.0, -5.0, -6.0) ?

(Right→Left) Order of Transformations

```
glMatrixMode(GL_MODELVIEW);
glLoadIdentity();
glTranslatef(4.0, 5.0, 6.0);
glRotatef(45.0, 1.0, 2.0, 3.0);
glTranslatef(-4.0, -5.0, -6.0);
```

$C \leftarrow I$

$C \leftarrow CT(4.0, 5.0, 6.0)$

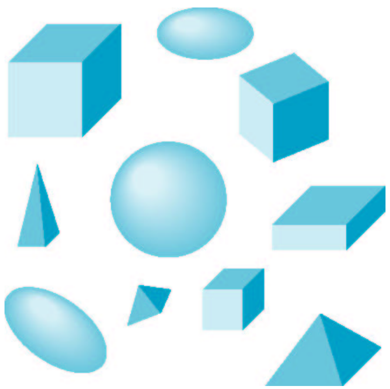
$C \leftarrow CR(45.0, 1.0, 2.0, 3.0)$

$C \leftarrow CT(-4.0, -5.0, -6.0)$

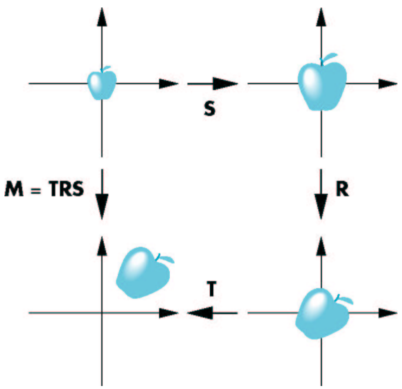
$C = IT(4.0, 5.0, 6.0)R(45.0, 1.0, 2.0, 3.0)T(-4.0, -5.0, -6.0)$

$q = Cp \quad \Rightarrow$ rotation about $(4.0, 5.0, 6.0)$

Excursion: Instance Transformation

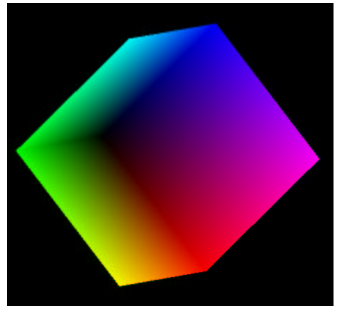


basic object "templates"



"instantiation"

Back to the Colored Cube



Spinning of the Cube

```
void display(void){
  glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
  glLoadIdentity();
  glRotatef(theta[0], 1.0, 0.0, 0.0);
  glRotatef(theta[1], 0.0, 1.0, 0.0);
  glRotatef(theta[2], 0.0, 0.0, 1.0);
  colorcube();
  glSwapBuffers();
}
```

Mouse Callback

```
void mouse(int btn, int state, int x, int y){
  if(btn==GLUT_LEFT_BUTTON && state == GLUT_DOWN)
     axis = 0;
  if(btn==GLUT_MIDDLE_BUTTON && state == GLUT_DOWN)
     axis = 1;
  if(btn==GLUT_RIGHT_BUTTON && state == GLUT_DOWN)
     axis = 2;
}
```


Idle Callback

```
void spinCube(){
  theta[axis] += 2.0;
  if ( theta[axis] > 360.0 ) theta[axis] -= 360.0;
  glutPostRedisplay();
}
```

 (cube)

Summary

- Rotation, Translation, Scaling
- Concatenation of Transformations
- OpenGL Transformation Matrices
- Next weeks:

  08/10/03 Object Hierarchies, Scene Graphs (Tom)
  15/10/03 Excursion to the CAVE!
  22/10/03 mid term break
  29/10/03 3D Viewing