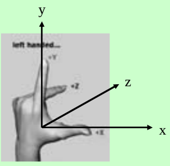
3D Transformations

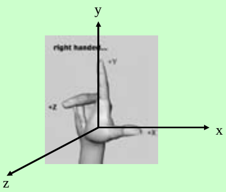
- Coordinate system
- 3D Vectors
- Basic 3D Transformations

from 2D to 3D

Coordinate Systems



Left-handed coordinate system



Right-handed coordinate system

3D-Vectors

- Have length and direction

  $V = \begin{bmatrix} v_x \\ v_y \\ v_z \end{bmatrix}$ or $[v_x \ v_y \ v_z]^T$

- Length is given by the Euclidean Norm

  $|V| = \sqrt{(v_x^2+v_y^2+v_z^2)}$

- Dot Product

  $V.U = [v_x, v_y, v_z].[u_x, u_y, u_z] = v_x u_x + v_y u_y + v_z u_z$

  $= |V||U| \cos\beta$

3D Vectors – Cross Product

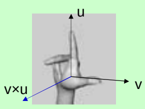
- Cross Product

  $V\times U = [v_y u_z - v_z u_y, -(v_x u_z - v_z u_x), v_x u_y - v_y u_x]$

  $V\times U = -(U\times V)$

  $|V\times U| = |V| |U| \sin\beta$ = ? can you write it using dot product?

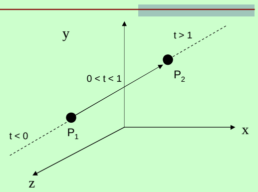
  $X\times Y = Z$ $Y\times Z = X$ $X\times Z = -Y$



Parametric Definition of Line

- $L = P_1 + tV$

  $= tP_2+(1 - t)P_1$



Plane

- Parametric:

$t_2P_3+(1-t_2)(t_1P_2+(1-t_1)P_1)$

- General Equation:

  $Ax + By + Cz + D = 0$

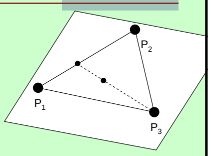
- Normalized Form:

  $A'x + B'y + C'z + D' = 0$

  Where:

  $A'=A/d$, $B'=B/d$, $C'=C/d$, $D'=D/d$

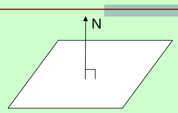
  $d = \sqrt{(A^2+B^2+C^2)}$



Plane (2)

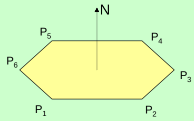
- Normal Vector:

  $N = [\,A\;\; B\;\; C\,]^T$



In computer graphics, N determines the face of a polygon which is usually the visible face

e.g.: in right-handed system:



Basic 3D Transformations - General

if:

$x_2 = \alpha_{11}x_1 + \alpha_{12}y_1 + \alpha_{13}z_1$

$y_2 = \alpha_{21}x_1 + \alpha_{22}y_1 + \alpha_{23}z_1$

$z_2 = \alpha_{31}x_1 + \alpha_{32}y_1 + \alpha_{33}z_1$

$$\begin{bmatrix} x_2 \\ y_2 \\ z_2 \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{33} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \\ z_1 \end{bmatrix}$$

Homogeneous 3D coordinate system:

$x_2 = \alpha_{11}x_1 + \alpha_{12}y_1 + \alpha_{13}z_1 + c_1$

$y_2 = \alpha_{21}x_1 + \alpha_{22}y_1 + \alpha_{23}z_1 + c_2$

$z_2 = \alpha_{31}x_1 + \alpha_{32}y_1 + \alpha_{33}z_1 + c_3$

$$\begin{bmatrix} x_2 \\ y_2 \\ z_2 \\ 1 \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} & c_1 \\ a_{21} & a_{22} & a_{23} & c_2 \\ a_{31} & a_{32} & a_{33} & c_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \\ z_1 \\ 1 \end{bmatrix}$$

Basic 3D Transformations - Translate

$$\begin{bmatrix} x_2 \\ y_2 \\ z_2 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \\ z_1 \\ 1 \end{bmatrix}$$

Basic 3D Transformations- Scale

$$\begin{bmatrix} x_2 \\ y_2 \\ z_2 \\ 1 \end{bmatrix} = \begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \\ z_1 \\ 1 \end{bmatrix}$$

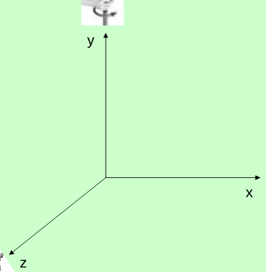
Basic 3D Transformations- Rotate

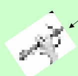
Rotation along axis:

x: from y to z

y: from z to x

z: from x to y

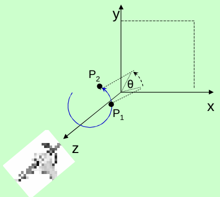




Basic 3D Transformations – Rotate (Z)

- Rotation around z-axis:
- $z_2 = z_1$ (will not change)

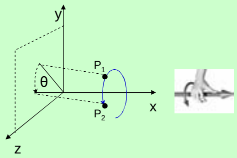
$$\begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



Basic 3D Transformations – Rotate (X)

- Rotation around x-axis:
- $x_2 = x_1$ (will not change)

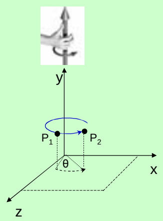
$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



Basic 3D Transformations – Rotate (Y)

- Rotation around y-axis:
- $y_2 = y_1$ (will not change)

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



Basic 3D Transformations – Shear

Along x-axis $SH_{yz} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ sh_y & 1 & 0 & 0 \\ sh_z & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$

Along y-axis $SH_{xz} = \begin{bmatrix} 1 & sh_x & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & sh_z & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$

Along z-axis $SH_{xy} = \begin{bmatrix} 1 & 0 & sh_x & 0 \\ 0 & 1 & sh_y & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$

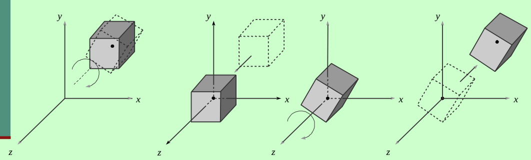
Inverse Transforms

- Inverse Translate: negate t_x, t_y, t_z
- Inverse Scale: from s_x to $1/s_x$
- Inverse Rotate: negate the angle
- Shear: negate the shear ratio

Composition of Transformations

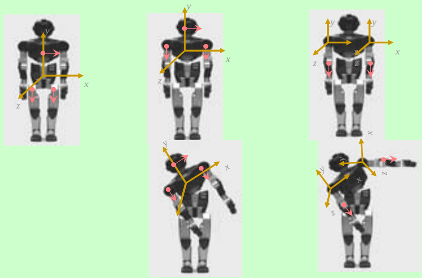
e.g.

Rotation around an arbitrary point



$M = T(p_x, p_y, p_z) \cdot R_z(-\theta) \cdot T(-p_x, -p_y, -p_z)$

Transformation hierarchy

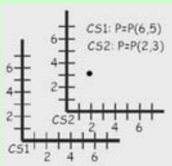


Transform of Coordinate System

- So far, transforming points within the same coordinate system.
- Sometimes necessary to change the coordinate system
- e.g. having many objects each with its own coordinate system (robot arm)

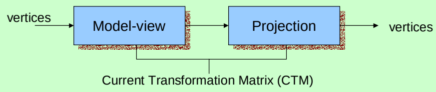
Transform of Coordinate System (2)

- P_i = Point in coordinate system 1 (CS_i)
- $M_{1\text{->}2}$ converts representation of representation of point in CS_1 to representation of point in CS_2
- Alternative interpretation: $M_{1\text{->}2}$ transforms axes of (2) into axes of (1)
- Example: $M_{1\text{->}2}$ = T(-4, -2)



OpenGL Transformation Matrices

- Model-View Matrix
- Projection Matrix



- Manipulated separately

```
glMatrixMode (GL_MODELVIEW);
glMatrixMode (GL_PROJECTION);
```


OpenGL Transformation Matrices (2)

- Load or post multiply

```
glLoadIdentity();
glLoadMatrixf(*m);
glMultMatrixf(*m);
```

- Library functions to compute matrices

```
glTranslatef(dx, dy, dz);
glRotatef(angle, vx, vy, vz);
glScalef(sx, sy, sz);
```

- Recall: Last transformation is applied first