CS 402:
Computer Graphics

Lecture Notes 06:

Ch. 5: Geometric Transformations

Prof. Dr. Mostafa Gadalhaqq

Professor of Computer Science

Faculty of Computer and Artificial Intelligence

Misr University for Science and Technology

CS402: Computer Graphics

Geometric Transformations

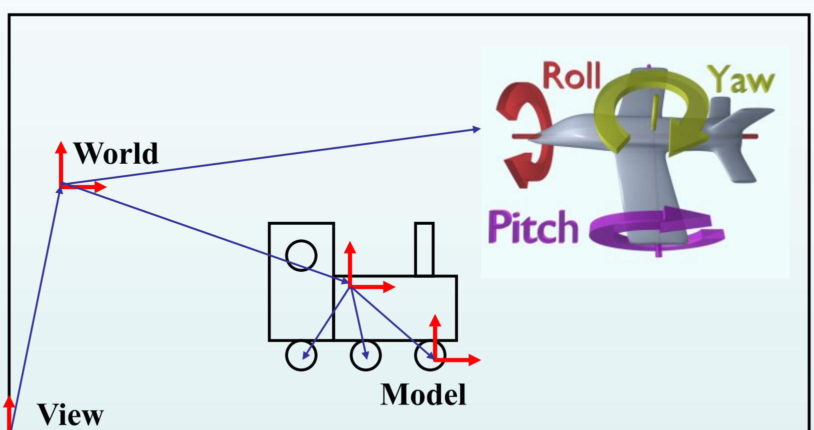
- What is Geometric Transformations?
- Why Geometric Transformations?
- Rigid Geometric Transformations:
  - Translation, Scaling , Rotation.
- Combining Transformations
  - Homogenous Coordinates.
- Non-rigid Transformations

Readings (H&B): Geometric Transformations (Chap. 5).

What is Geometric Transformation?

- **Rigid-body Transformations:** No deformations resulted in the body due to transformations, e.g.:
  - Translation
  - Scaling
  - Rotation
- **Non-rigid Transformations:** The object is deformed resulted due to transformations, e.g.:
  - Sheer
  - Affine Transformations

Why Geometric Transformation?



Why Geometric Transformation?

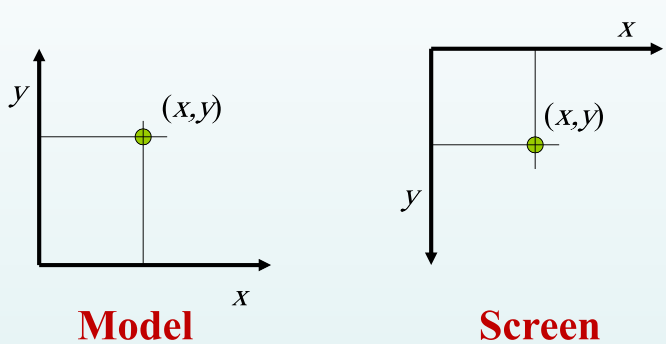
- **Model of objects**
  - world coordinates: km, mm, etc.
  - Hierarchical models:

    human = torso + arm + arm + head + leg + leg

    arm = upper-arm + lower-arm + hand …
- **Viewing**

  zoom in, zoom out, translate, etc.
- **Animation**

Model vs Screen Coord.



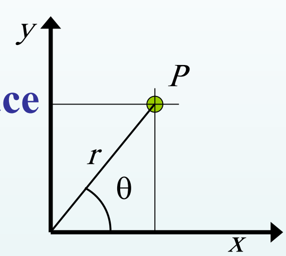
Remember: (0,0) of the Screen is the top-left corner

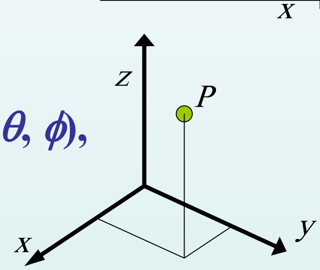
Basic Geometry

A Quick Review

Geometry Review: Points

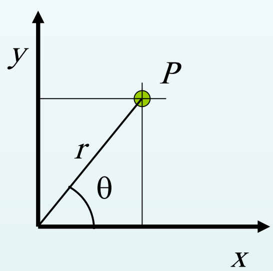
- A **Point** is a position in *n*D space
- Notation: P
- **2D-P**: (x, y) , (r, θ)
- **3D-P**: (x, y, z) , (r, θ, z), (r, θ, ϕ),





Geometry Review: Coordinates

2D Polar coordinates:



Coordinate transformation

from (r,θ) to (x,y) space :

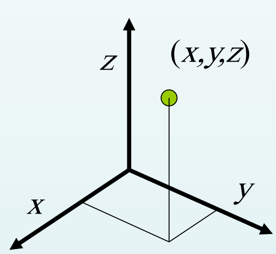
$x = r\cos\theta$

$y = r\sin\theta$

Angle θ : in radians

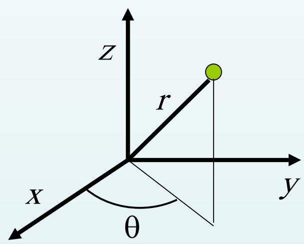
Geometry: Coordinates

3D Cartesian coordinates:



Geometry: Coordinates

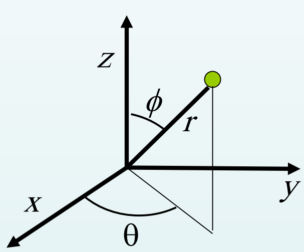
3D Cylindrical coordinates:



$$x = r\cos\theta$$
$$y = r\sin\theta$$
$$z = z$$

Geometry: Coordinates

3D Spherical coordinates:



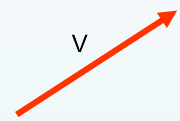
$$x = r\cos\theta\sin\phi$$
$$y = r\sin\theta\sin\phi$$
$$z = r\cos\phi$$

θ : azimuth or longitude

ϕ : elevation or latitude

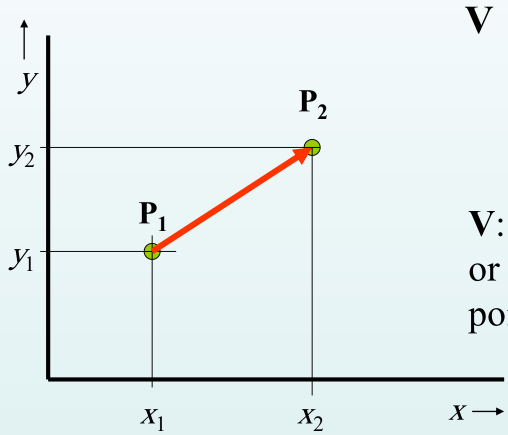
Geometry: Vectors

- **Vector:**
  - "arrow"



  - Represents: displacement, velocity, force, etc.
  - has a **magnitude** and **direction**
  - has *no* position
- **Notation:** V= (v_x, v_y, v_z) , or (x, y, z)

Geometry: Vectors



$$\begin{aligned}\mathbf{V} &= \mathbf{P_2} - \mathbf{P}_1 \\ &= (x_2 - x_1, y_2 - y_1) \\ &= (V_x, V_y)\end{aligned}$$

V: directed line segment, or difference between two points

Geometry: Vectors

Length of a vector:

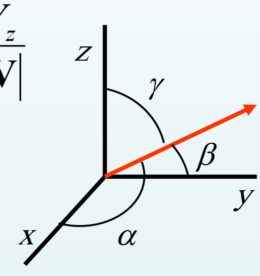
$$|\mathbf{V}| = \sqrt{V_x^2 + V_y^2} \quad \text{(2D : Pythagoras)}$$

$$|\mathbf{V}| = \sqrt{V_x^2 + V_y^2 + V_z^2} \quad \text{(3D)}$$

Geometry: Vectors

- **Direction of a vector: Direction angles.**

$$\cos\alpha = \frac{V_x}{|\mathbf{V}|}, \cos\beta = \frac{V_y}{|\mathbf{V}|}, \cos\gamma = \frac{V_z}{|\mathbf{V}|}$$



- **Unit vector $\overline{V}$:**

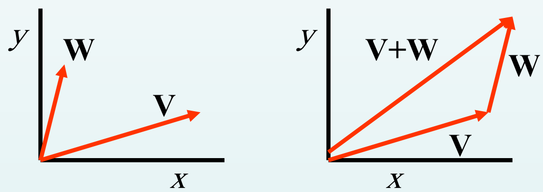
$$\overline{\mathbf{V}} = \frac{\mathbf{V}}{|\mathbf{V}|}$$

- **Magnitude info is removed, direction is kept.**

Geometry: Vectors

Vector addition:

Add components, put vector head to tail

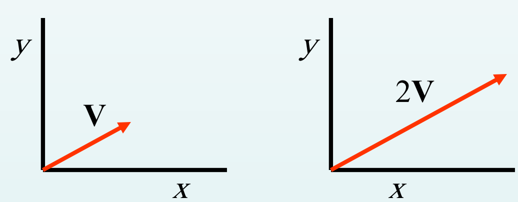


$$\mathbf{V} + \mathbf{W} = (V_x + W_x, V_y + W_y, V_z + W_z)$$

Geometry: Vectors

Vector multiplication with scalar *s:*

Multiplicate each components with s

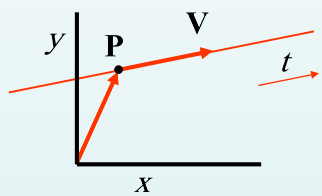


$$s\mathbf{V} = (sV_x, sV_y, sV_z)$$

Geometry: Line

Infinite line through P with direction V:

$$L(t) = P + Vt, \quad t \in \Re$$

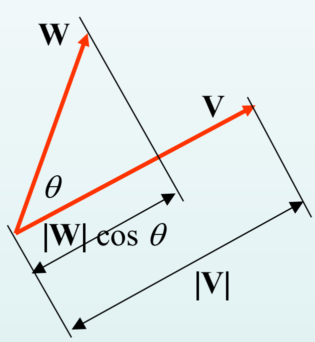


t : parameter along line

$t \in [a, b]$: line segment

Geometry: Vectors

Scalar/Dot product:



$$\mathbf{V} \cdot \mathbf{W} = |\mathbf{V}||\mathbf{W}|\cos\theta$$

$$= V_x W_x + V_y W_y + V_z W_z$$

Geometry: Vectors

Properties of Scalar product:

$$\mathbf{V}\cdot\mathbf{W} = |\mathbf{V}||\mathbf{W}|\cos\theta$$
$$= V_x W_x + V_y W_y + V_z W_z$$

Commutative : $\mathbf{V}\cdot\mathbf{W} = \mathbf{W}\cdot\mathbf{V}$

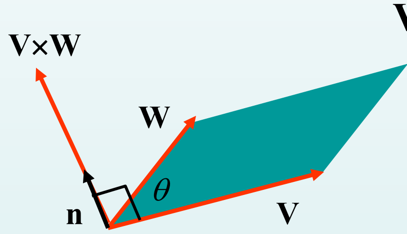
Distributive : $\mathbf{V}\cdot(\mathbf{W}+\mathbf{Z}) = \mathbf{V}\cdot\mathbf{W} + \mathbf{V}\cdot\mathbf{Z}$

Associative : $\mathbf{V}\cdot(\mathbf{W}\cdot\mathbf{Z}) = (\mathbf{V}\cdot\mathbf{W})\cdot\mathbf{Z}$??

Geometry: Vectors

Vector/Cross product:

gives a vector (in 3D) n perpendicular to V and W



$$\mathbf{V}\times\mathbf{W} = \mathbf{n}|\mathbf{V}||\mathbf{W}|\sin\theta$$
$$= (V_y W_z - V_z W_y,$$
$$V_z W_x - V_x W_z,$$
$$V_x W_y - V_y W_x)$$

Geometry: Vectors

Properties of Vector product:

$$\text{Anticommutative}: \mathbf{V} \times \mathbf{W} = -\mathbf{W} \times \mathbf{V}$$

$$\text{Not associative}: \quad \mathbf{V} \times (\mathbf{W} \times \mathbf{Z}) \neq (\mathbf{V} \times \mathbf{W}) \times \mathbf{Z}$$

$$\text{Distributive}: \quad \mathbf{V} \times (\mathbf{W} + \mathbf{Z}) = \mathbf{V} \times \mathbf{W} + \mathbf{V} \times \mathbf{Z}$$

Scalar vs Vector Product

Scalar product:

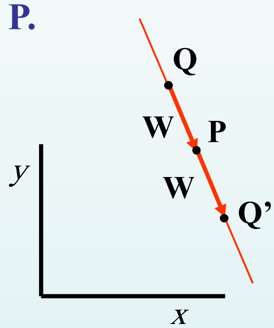
- **Result is scalar**
- **Test if vectors are perpendicular**
- **cos**
- **project,…**

Vector product:

- **Result is vector**
- **Get a vector perpendicular to two given vectors**
- **sine**
- **surface area,…**

Example

- **Given a point P. Reflect a point Q with respect to P.**



$$W = P - Q$$

$$Q' = Q + 2W$$

$$= 2P - Q$$

or: $= P + (P - Q)$

We don't need coordinates!

Back to

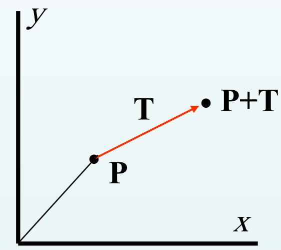
Geometric Transformations

Geometric Trans.: Translation

- Translate with T= (t_x, t_y)

$$x' = x + t_x$$
$$y' = y + t_y$$

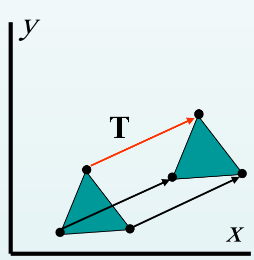
$$\boxed{\mathbf{P' = P + T}}$$



$$\mathbf{P}' = \begin{pmatrix} x' \\ y' \end{pmatrix}, \mathbf{P} = \begin{pmatrix} x \\ y \end{pmatrix} \text{ and } \mathbf{T} = \begin{pmatrix} t_x \\ t_y \end{pmatrix}$$

Geometric Trans.: Translation

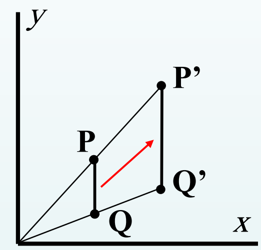
- Translate a polygon:
  - Apply the same operation on all points.



Geometric Trans.: Scaling

Scale with factors s_x and s_y:

$$x' = s_x x$$
$$y' = s_y y$$



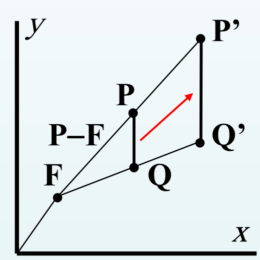
$$\boxed{\mathbf{P' = S\,P}}$$

$$\mathbf{P}' = \begin{pmatrix} x' \\ y' \end{pmatrix}, \mathbf{S} = \begin{pmatrix} s_x & 0 \\ 0 & s_y \end{pmatrix}, \text{and } \mathbf{P} = \begin{pmatrix} x \\ y \end{pmatrix}$$

Geometric Trans.: Scaling

Scaling w.r.t. the origin:

$$x' = s_x x$$
$$y' = s_y y$$



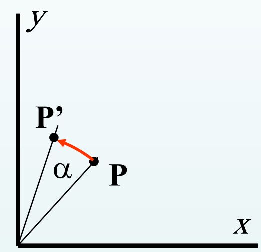
Scaling w.r.t. a Fixed-point F:

$$x' = F_x + s_x(x - F_x)$$
$$y' = F_y + s_y(y - F_y)$$

Geometric Trans.: Rotation

- **Rotation with an angle α**

$$x' = x\cos\alpha - y\sin\alpha$$
$$y' = x\sin\alpha + y\cos\alpha$$



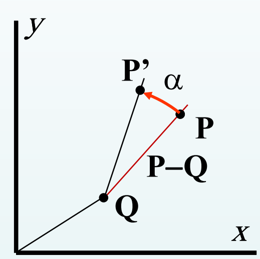
P’ = R P

$$\mathbf{P}' = \begin{pmatrix} x' \\ y' \end{pmatrix}, \mathbf{R} = \begin{pmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{pmatrix}, \text{and } \mathbf{P} = \begin{pmatrix} x \\ y \end{pmatrix}$$

Geometric Trans.: Rotation

- **Rotation around origin**

$$P_x' = P_x\cos\alpha - P_y\sin\alpha$$
$$P_y' = P_x\sin\alpha + P_y\cos\alpha$$



- **Rotation around a pivot-point Q**

$$P_x' = q_x + (P_x - q_x)\cos\alpha - (P_y - q_y)\sin\alpha$$
$$P_y' = q_y + (P_x - q_x)\sin\alpha + (P_y - q_y)\cos\alpha$$

Homogeneous Coordinates

- Why Homogeneous Coordinates:
  - Transformations in regular coord. are Messy!
  - Transformations with respect to points: even more messy!
  - Homogeneous coord. is a nice trick to combine transformations?

Homogeneous Coordinates

- Homogeneous coord. produce:
  - uniform representation of translation, rotation, scaling
  - uniform representation of points and vectors
  - compact representation of sequence of transformations

Homogeneous Coordinates

- Add extra coordinate to the points:

$$\mathbf{P} = (p_x, p_y, p_h)$$

or

$$\mathbf{P} = (x, y, h)$$

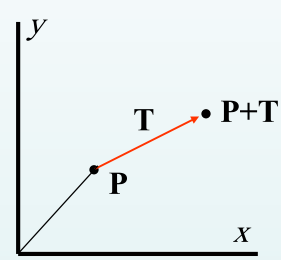
- OpenGL Representations:
  - Points: $h = 1$ → (x, y, 1)
  - Vectors: $h = 0$ → (x, y, 0)

Translation Matrix in H.C.

- Translation Matrix, T:

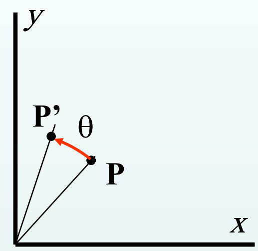
$$\boxed{\mathbf{P}' = \mathbf{T}(t_x, t_y)\mathbf{P}}$$

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$



Rotation Matrix in H.C.

- **Rotation Matrix, R:**

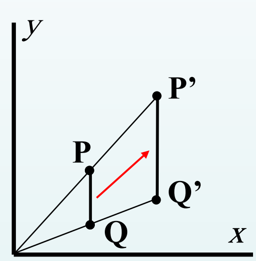


$$\boxed{\mathbf{P}' = \mathbf{R}(\theta)\mathbf{P}}$$

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

Scaling Matrix in H.C.

- **Scaling Matrix, S:**



$$\boxed{\mathbf{P}' = \mathbf{S}(s_x, s_y)\mathbf{P}}$$

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

Inverse transformations

- **Inverse Translation:**

$$\mathbf{T}^{-1}(t_x, t_y) = \mathbf{T}(-t_x, -t_y)$$

- **Inverse Scaling:**

$$\mathbf{S}^{-1}(s_x, s_y) = \mathbf{S}(\frac{1}{s_x}, \frac{1}{s_y})$$

- **Inverse Rotation:**

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

Combined Transformations

$$\mathbf{P}' = \mathbf{M_1 P} \qquad \text{first transformation...}$$

$$\mathbf{P}'' = \mathbf{M_2 P}' \qquad \text{second transformation...}$$

Combined :

$$\begin{aligned} \mathbf{P}'' &= \mathbf{M_2 (M_1 P)} \\ &= \mathbf{M_2 M_1 P} \\ &= \mathbf{MP} \quad \text{with } \mathbf{M} = \mathbf{M_2 M_1} \end{aligned}$$

Combined Transformations

$$\mathbf{P}' = \mathbf{T}(t_{1x}, t_{1y})\mathbf{P} \qquad \text{first translation}$$

$$\mathbf{P}'' = \mathbf{T}(t_{2x}, t_{2y})\mathbf{P}' \qquad \text{second translation}$$

Combined :

$$\mathbf{P}'' = \mathbf{T}(t_{2x}, t_{2y})\mathbf{T}(t_{1x}, t_{1y})\mathbf{P}$$

$$= \begin{pmatrix} 1 & 0 & t_{2x} \\ 0 & 1 & t_{2y} \\ 0 & 0 & 1 \end{pmatrix}\begin{pmatrix} 1 & 0 & t_{1x} \\ 0 & 1 & t_{1y} \\ 0 & 0 & 1 \end{pmatrix}\mathbf{P} = \begin{pmatrix} 1 & 0 & t_{1x}+t_{2x} \\ 0 & 1 & t_{1y}+t_{2y} \\ 0 & 0 & 1 \end{pmatrix}\mathbf{P}$$

$$= \mathbf{T}(t_{1x}+t_{2x}, t_{1x}+t_{2y})\mathbf{P}$$

Combined Transformations

- **Composite Translation:**

$$\boxed{\mathbf{T}(t_{2x}, t_{2y})\mathbf{T}(t_{1x}, t_{1y}) = \mathbf{T}(t_{1x}+t_{2x}, t_{1x}+t_{2y})}$$

- **Composite Scaling:**

$$\boxed{\mathbf{S}(s_{2x}, s_{2y})\mathbf{S}(s_{1x}, s_{1y}) = \mathbf{S}(s_{1x}s_{2x}, s_{1y}s_{2y})}$$

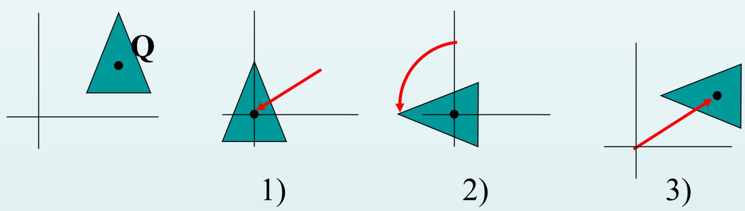
- **Composite Rotation:**

$$\boxed{\mathbf{R}(\theta_2)R(\theta_1) = \mathbf{R}(\theta_1+\theta_2)}$$

Rotation Around a Pivot-Point

Rotate over angle θ around a pivot-point Q:

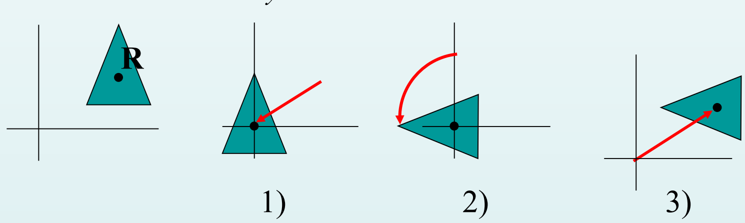
1) Translate such that Q coincides with origin;
2) Rotate over angle θ around origin;
3) Translate back.



Rotation Around a Pivot-Point

Rotate over angle θ around a pivot-point Q:

1) $\mathbf{P}' = \mathbf{T}(-q_x, -q_y)\mathbf{P}$
2) $\mathbf{P}'' = \mathbf{R}(\theta)\mathbf{P}'$
3) $\mathbf{P}''' = \mathbf{T}(q_x, q_y)\mathbf{P}''$



Rotation Around a Pivot-Point

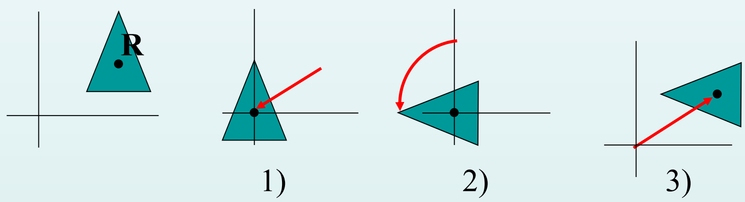
1) $\mathbf{P}' = \mathbf{T}(-q_x, -q_y)\mathbf{P}$

2) $\mathbf{P}'' = \mathbf{R}(\theta)\mathbf{P}' = \mathbf{R}(\theta)\mathbf{T}(-q_x, -q_y)\mathbf{P}$

3) $\mathbf{P}''' = \mathbf{T}(q_x, q_y)\mathbf{P}''$

$$= \mathbf{T}(q_x, q_y)\mathbf{R}(\theta)\mathbf{P}'$$

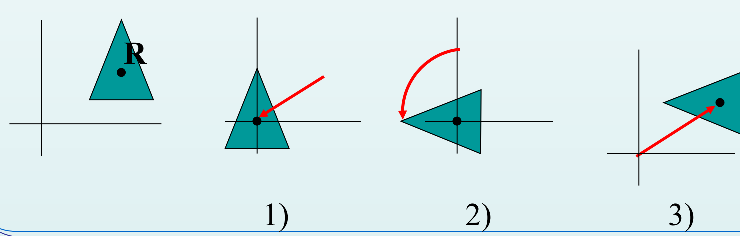
$$= \mathbf{T}(q_x, q_y)\mathbf{R}(\theta)\mathbf{T}(-q_x, -q_y)\mathbf{P}$$



Rotation Around a Pivot-Point

$$\boxed{\mathbf{P}''' = \mathbf{T}(R_x, R_y)\mathbf{R}(\theta)\mathbf{T}(-R_x, -R_y)\mathbf{P}}$$

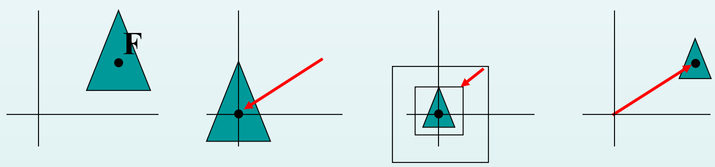
$$\mathbf{P}''' = \begin{pmatrix} \cos\theta & -\sin\theta & R_x(1-\cos\theta) + R_y \sin\theta \\ \sin\theta & \cos\theta & R_y(1-\cos\theta) - R_x \sin\theta \\ 0 & 0 & 1 \end{pmatrix}\mathbf{P}$$



Scaling w.r.t. a Fixed Point

Scale with factors s_x and s_x w.r.t. a fixed point **F** :

1) Translate such that **F** coincides with origin;
2) Schale w.r.t. origin;
3) Translate back again.

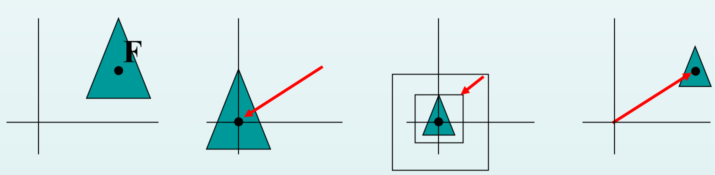


1) 2) 3)

Scaling w.r.t. a Fixed Point

Schale w.r.t. a fixed point **F** :

1) $\mathbf{P}' = \mathbf{T}(-F_x, -F_y)\mathbf{P}$
2) $\mathbf{P}'' = \mathbf{S}(s_x, s_y)\mathbf{P}'$
3) $\mathbf{P}''' = \mathbf{T}(F_x, F_y)\mathbf{P}''$

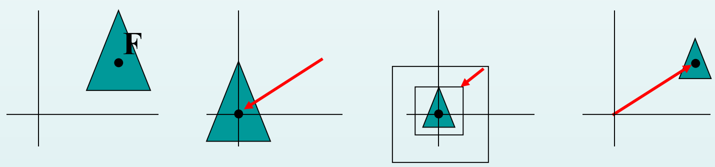


1) 2) 3)

Scaling w.r.t. a Fixed Point

$$\mathbf{P}''' = \mathbf{T}(F_x, F_y)\mathbf{S}(s_x, s_y)\mathbf{T}(-F_x, -F_y)\mathbf{P}$$

$$\mathbf{P}''' = \begin{pmatrix} s_x & 0 & F_x(1-s_x) \\ 0 & s_y & F_y(1-s_y) \\ 0 & 0 & 1 \end{pmatrix}\mathbf{P}$$



1) 2) 3)

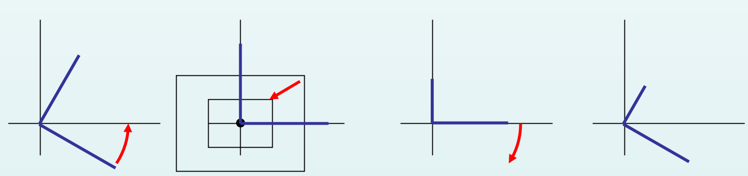
Scale in other directions

Scale with factors s_1 and s_2 w.r.t. rotated frame:

1) Rotate such that frame coincides with standard xy-frame;

2) Scale w.r.t. origin;

3) Rotate back again.



1) 2) 3)

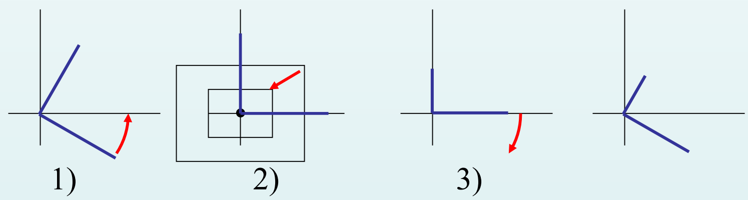
Scale in other directions

Scale in other direction:

1) $\mathbf{P}' = \mathbf{R}(\theta)\mathbf{P}$

2) $\mathbf{P}'' = \mathbf{S}(s_1, s_2)\mathbf{P}'$

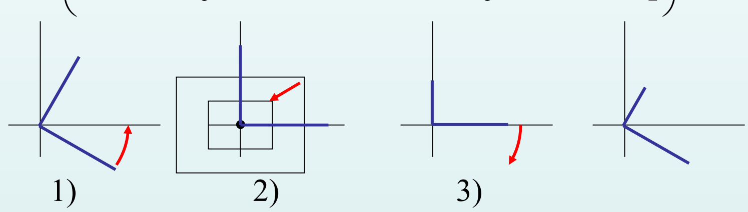
3) $\mathbf{P}''' = \mathbf{R}(-\theta)\mathbf{P}''$



Scale in other directions

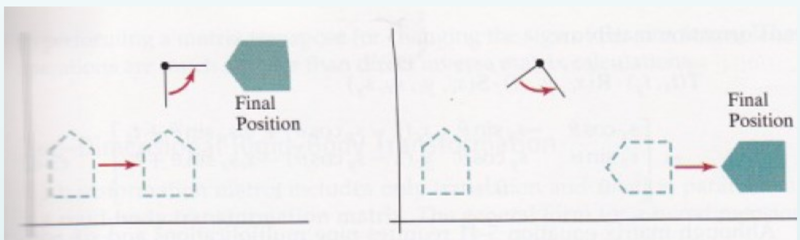
$$\boxed{\mathbf{P}''' = \mathbf{R}(-\theta)\mathbf{S}(s_1, s_2)\mathbf{R}(\theta)\mathbf{P}}$$

$$\mathbf{P}''' = \begin{pmatrix} s_1\cos^2\theta + s_2\sin^2\theta & (s_2 - s_1)\cos\theta\sin\theta & 0 \\ (s_2 - s_1)\cos\theta\sin\theta & s_1\sin^2\theta + s_2\cos^2\theta & 0 \\ 0 & 0 & 1 \end{pmatrix}\mathbf{P}$$



Order of Transformations

Matrix multiplication does not commute.



Translation → Rotation … Rotation → Translation

General Transformations

$$x' = \boxed{rs_{xx}x + rs_{xy}y} + \boxed{trsx}$$
$$y' = \boxed{rsy_x x + rs_{yy}y} + \boxed{trsy}$$

$$P' = M\,P$$

rotation and scaling

translation

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} rs_{xx} & rs_{xy} & trs_x \\ rs_{yx} & rs_{yy} & trs_y \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Other 2D transformations

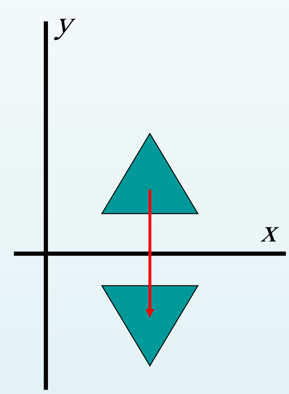
- Reflection
- Shear
- Affine transformation

Reflection over axis

Reflect over *x*-axis:

$$x' = x$$
$$y' = -y$$

$$\mathbf{P}' = \begin{pmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \mathbf{P}$$

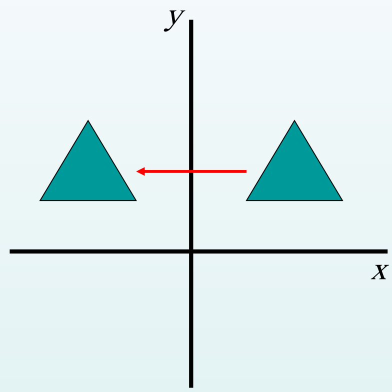


Reflection over axis

Reflect over *y*-axis:

$$x' = -x$$
$$y' = y$$

$$\mathbf{P'} = \begin{pmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \mathbf{P}$$

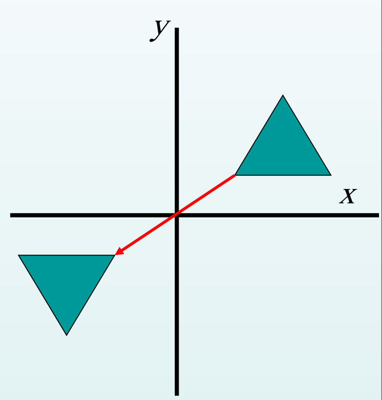


Reflect over origin

Reflect over origin:

$$x' = -x$$
$$y' = -y$$

$$\mathbf{P'} = \begin{pmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \mathbf{P}$$



Same as $\mathbf{P'} = \mathbf{R}(180)\mathbf{P}$

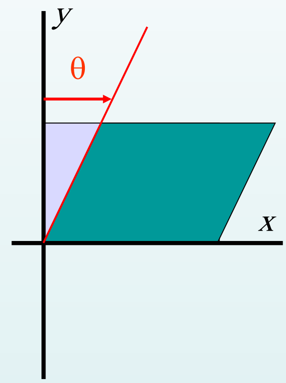
Shear

Shear the along the *x*-axis:

$$x' = x + sh_x y$$
$$y' = y$$

$$\mathbf{P'} = \begin{pmatrix} 1 & sh_x & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \mathbf{P}$$

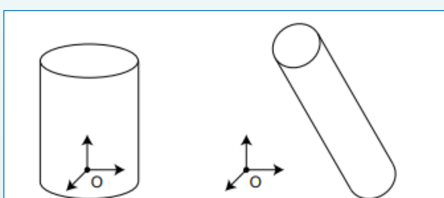
with $sh_x = \tan\theta$



Affine Transformation

- **Affine transformation is a general non-rigid transformation. That is, transformed object is deformed.**

$$\mathbf{P}' = \begin{pmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{pmatrix} \mathbf{P}$$



Original cylinder model

Transformed cylinder. It has been scaled, rotated, and translated

3D-Transformations

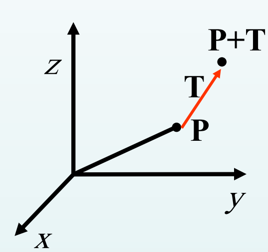
- For the time being, all 3D-transformations are done in the same way as we did with the 2D-Transformations but with adding the third coordinate (z).
- Review 3D-Transformations in chHearn&Baker Book.

Translation Matrix in 3D H.C.

- Translation Matrix, T:

$$\boxed{\mathbf{P}' = \mathbf{T}(t_x, t_y, t_z)\mathbf{P}}$$

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

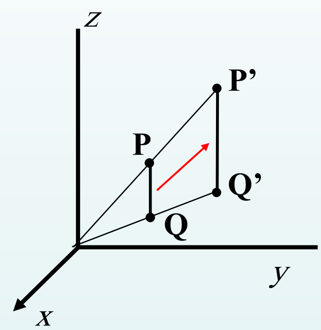


Scaling Matrix in 3D H.C.

- **Scaling Matrix, S:**

$$\mathbf{P'} = \mathbf{S}(s_x, s_y, s_z)\mathbf{P}$$

$$\begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix} = \begin{pmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

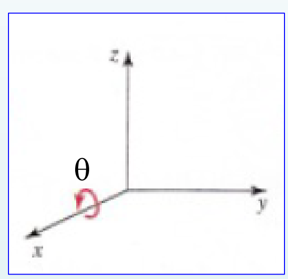


Rotation Matrix in 3D H.C.

- **Matrix of 3D Rotation around x-axis, R_x:**

$$\mathbf{P'} = \mathbf{R}_x(\theta)\mathbf{P}$$

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

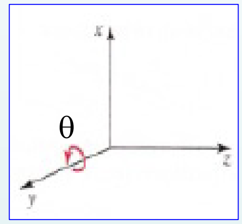


Rotation Matrix in 3D H.C.

- **Matrix of 3D Rotation around y-axis, R_y:**

$$\boxed{\mathbf{P}' = \mathbf{R}_y(\theta)\mathbf{P}}$$

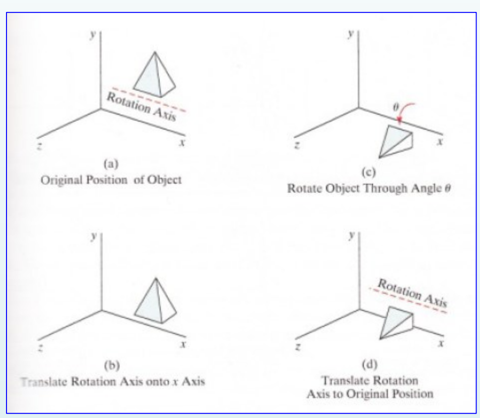
$$\begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix} = \begin{pmatrix} \cos\theta & 0 & -\sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ \sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$



General Rotation

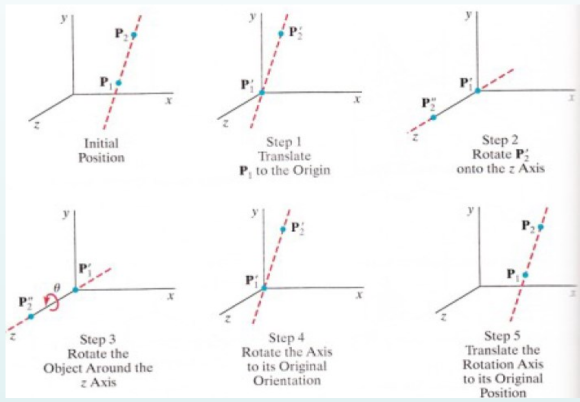
- **Rotation around an axis parallel to coord. axis:**

$$\boxed{\mathbf{R}(\theta) = \mathbf{T}^{-1} \cdot \mathbf{R}_x(\theta) \cdot \mathbf{T}}$$



General Rotation

- **Rotation around an arbitrary axis:**



$$\mathbf{R}(\theta) = \mathbf{T}^{-1} \cdot \mathbf{R}_x^{-1}(\alpha) \cdot \mathbf{R}_y^{-1}(\beta) \cdot \mathbf{R}_z(\theta) \cdot \mathbf{R}_y(\beta) \cdot \mathbf{R}_x(\alpha) \cdot \mathbf{T}$$

OpenGL Transformations

OpenGL maintains two matrices:

- `GL_PROJECTION`
- `GL_MODELVIEW`

Transformations are applied to the current matrix, to be selected with:

- `glMatrixMode(GL_PROJECTION)`

or

- `glMatrixMode(GL_MODELVIEW)`

OpenGL Transformations

Basic transformation functions: generate matrix and post-multiply this with current matrix.

- **Translate over [t_x, t_y, t_z]:**
  `glTranslate*(tx, ty, tz);`
- **Rotate over θ degrees, around axis [v_x, v_y, v_z]:**
  `glRotate*(theta, vx, vy, vz);`
- **Scale axes with factors s_x, s_y, s_z:**
  `glScale*(sx, sy, sz);`

OpenGL Transformations

- **OpenGL maintains stacks of transformation matrices.**
- **Two operations:**
  - `glPushMatrix():`
    - **Put copy of current matrix that on top of the stack;**
  - `glPopMatrix():`
    - **Remove top element of the stack.**
- **Handy for dealing with hierarchical models**
- **Handy for "undoing" transformations**

Transformations in OpenGL

Standard:

```
glRotate(10, 1, 2, 0);
glScale(2, 1, 0.5);
glTranslate(1, 2, 3);

glutWireCube(1);

glTranslate(-1, -2, -3);
glScale(0.5, 1, 2);
glRotate(-10, 1, 2, 0);
```

Using the stack:

```
glPushMatrix();
glRotate(10, 1, 2, 0);
glScale(2, 1, 0.5);
glTranslate(1, 2, 3);

glutWireCube(1);

glPopMatrix();
```

Undo transformation

Shorter, more robust

Summary of Geometric Transformations

- **Why Transformations?**
  - **Modelling, Viewing, Animation, …**
- **Several kinds of transformations:**
  - **Translation, Scaling, Rotation, Sheering, …**
- **Homogeneous coordinates:**
  - **Simplify transformations and easily Combine multiple transformations.**

Assignment 2

1. Draw the points $P_1 = (3,5)$ and $P2 = (5,7)$, then:
   - Write and apply the 2D rotation matrix R that rotate these points by 45^o clockwise around the origin.
   - Write and apply the 2D matrix R that rotate these points by 45^o anti-clockwise around the pivot-point (0,3).
   - Write and apply the 2D scaling matrix S that scale these points by (2, 3) w.r.t. the origin.
   - Write and apply the 2D scaling matrix S that scale these points by (2, 3) w.r.t. the fixed-point (4, 6).
2. Write an OpenGL program that read some points and apply the above transformations on such points.

Next Time

Primitives Drawing Algorithms