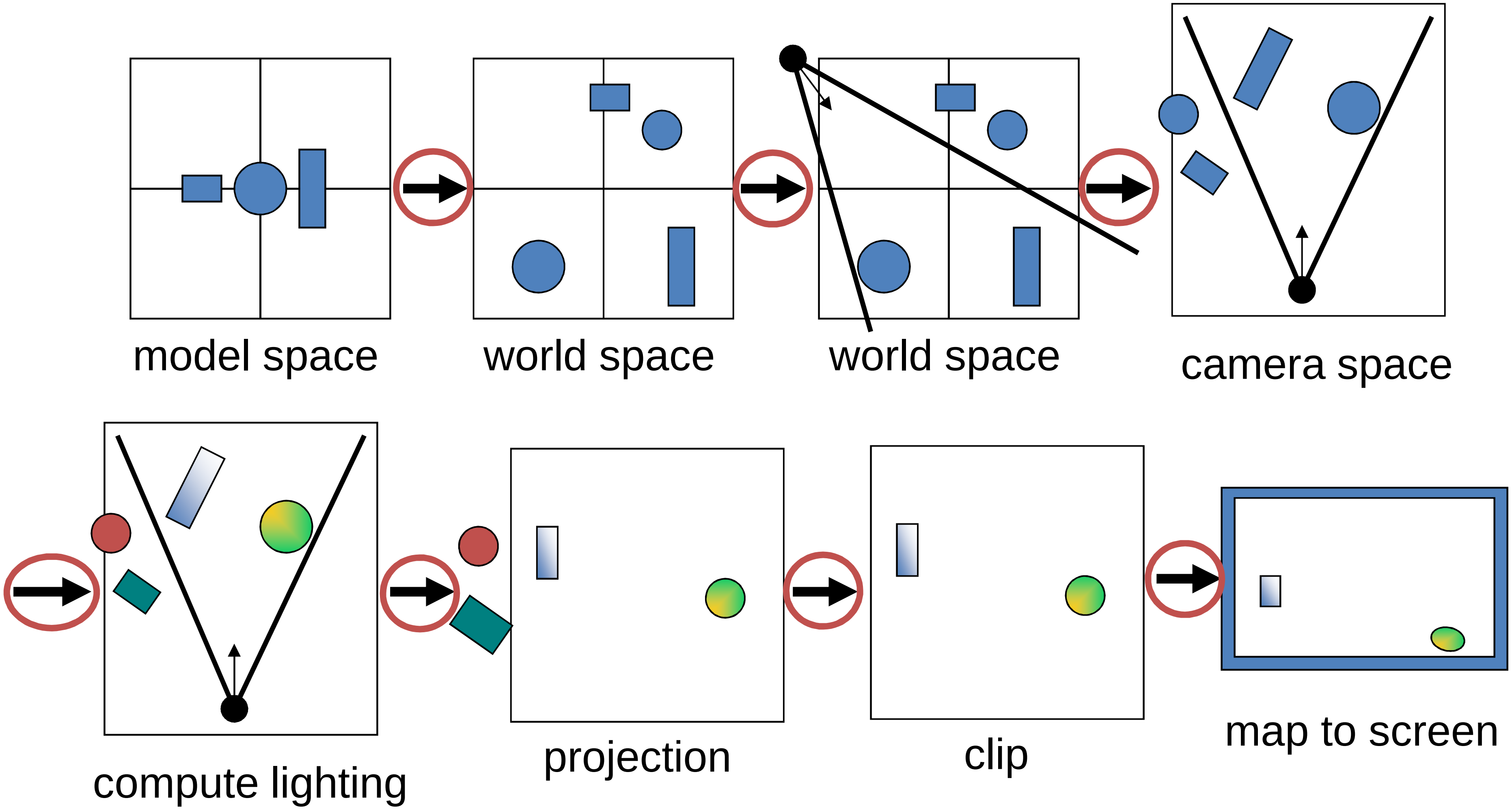
Transformations

Why transforms?

- Animate objects and camera
- Implementing rendering pipeline
- Because we can

Remember: Geometry Stage



transformations

Rendering: Transformations

- So far, discussion has been in *screen space*
- But model is stored in *model space*
  (a.k.a. object space)
- Three sets of geometric transformations:
  - *Modeling transforms*
  - *Viewing transforms*
  - *Projection transforms*

Rendering: Transformations

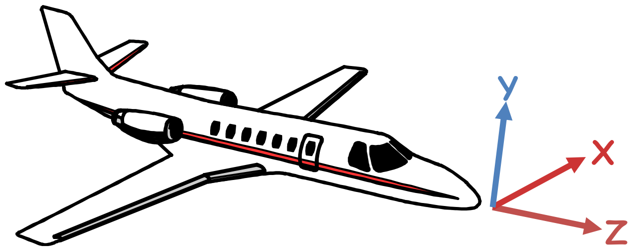
- All these transformations involve shifting coordinate systems (i.e., basis sets)
- That's what matrices do…
- Represent coordinates as vectors, transforms as matrices

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

- Multiply matrices = concatenate transforms!

Rendering: Transformations

- Modeling transforms
  - Size, place, scale, and rotate objects parts of the model w.r.t. each other
  - Object coordinates → world coordinates

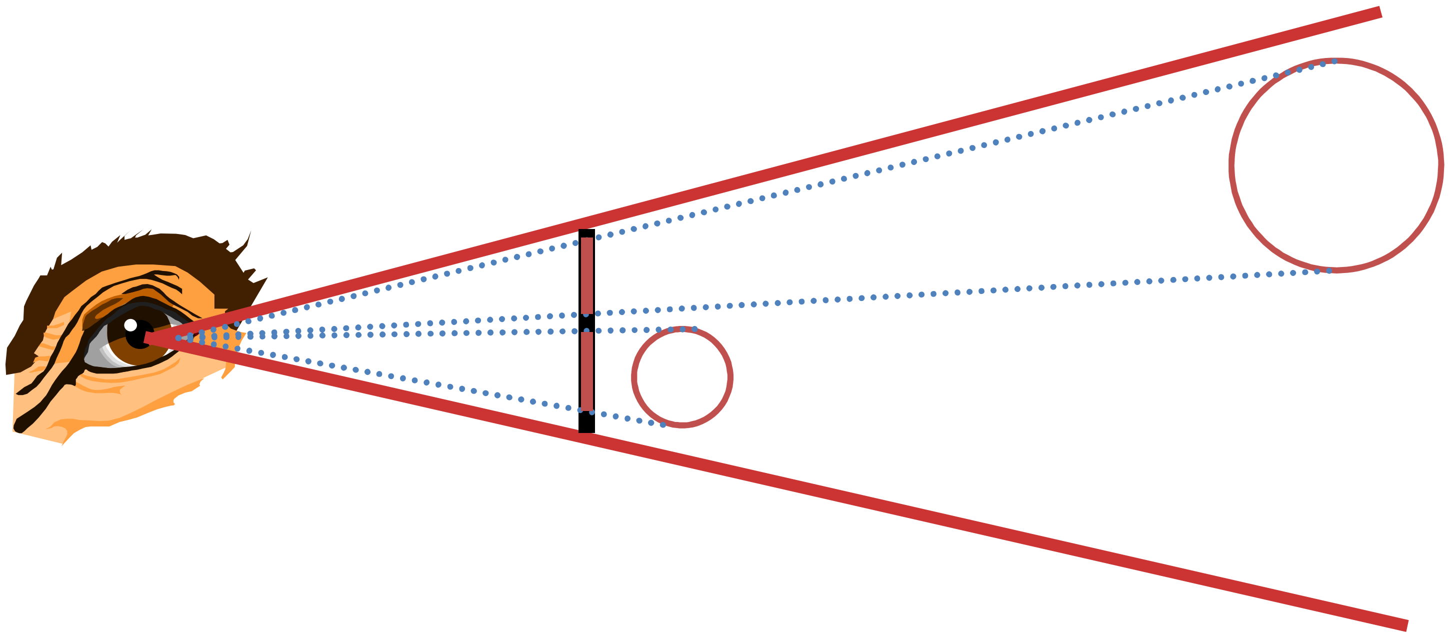


Rendering: Transformations

- Viewing transform
  - Rotate & translate the world to lie directly in front of the camera
    - Typically place camera at origin
    - Typically looking along -Z axis
  - World coordinates → view coordinates

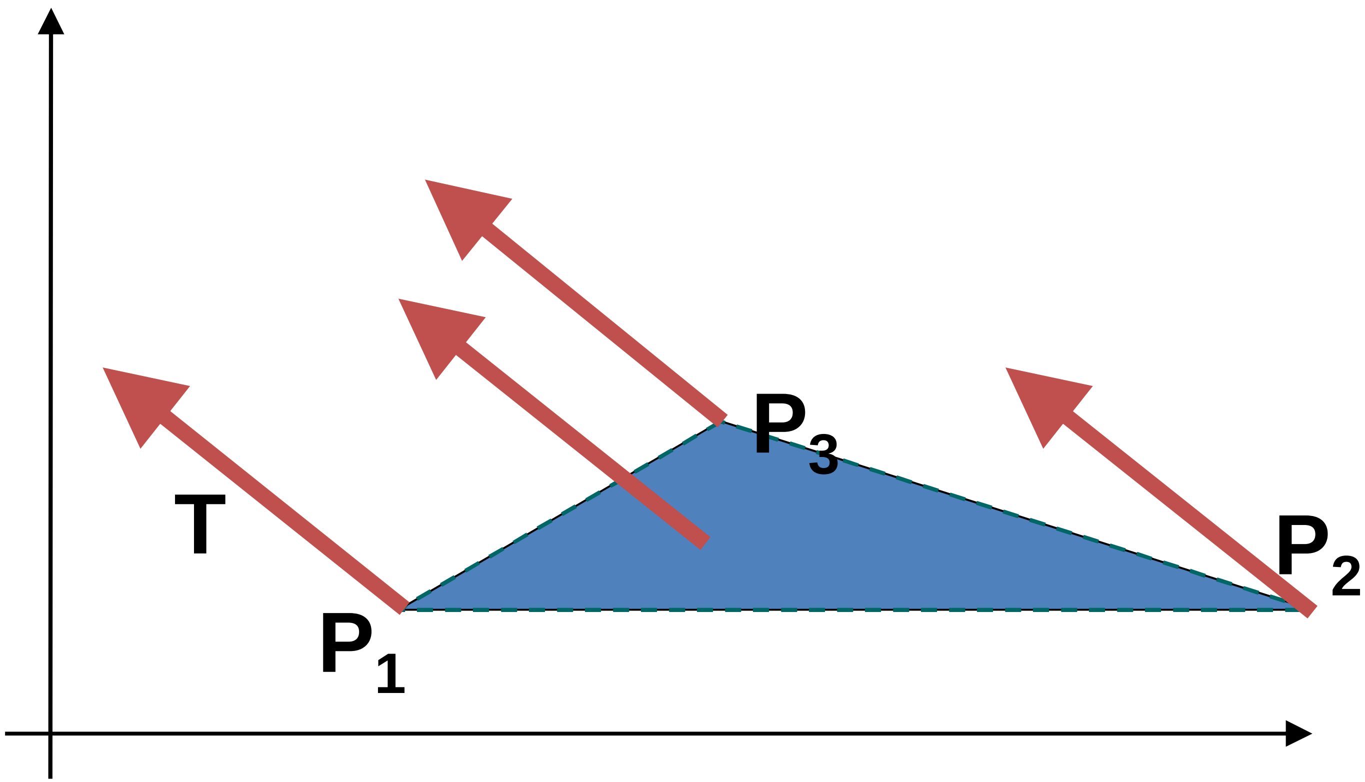
Rendering: Transformations

- Projection transform
  - Apply perspective foreshortening
    - Distant = small: the *pinhole camera* model
  - View coordinates → screen coordinates



Rigid body transformation

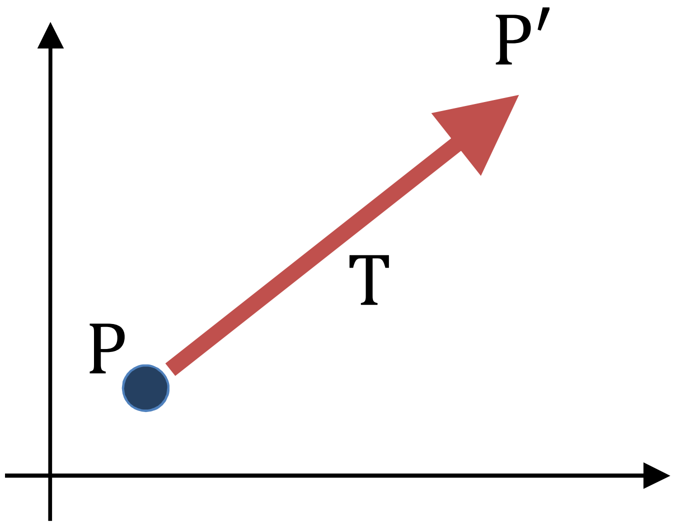
- Object transformed by transforming boundary points



Basic Transformations

Translation

- Translating a point from position P to position P' with translation vector T



$$\mathrm{P} = \begin{pmatrix} x \\ y \end{pmatrix} \quad \mathrm{P}' = \begin{pmatrix} x' \\ y' \end{pmatrix} \quad \mathrm{T} = \begin{pmatrix} t_x \\ t_y \end{pmatrix}$$

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

$$\mathrm{P}' = \mathrm{P} + \mathrm{T}$$

Translation 3D

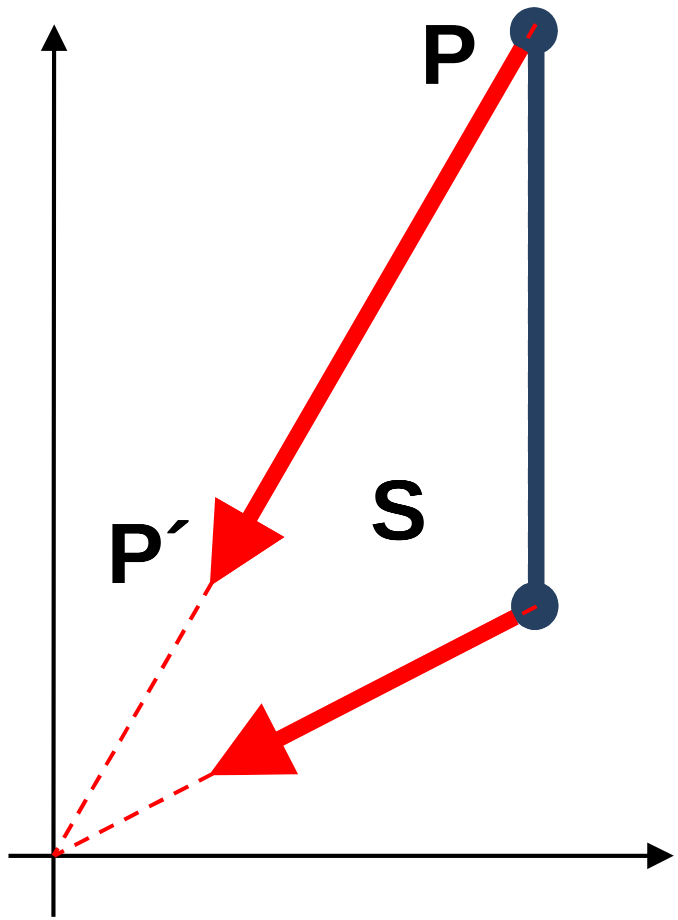
- For convenience we usually describe objects in relation to their own coordinate system
- We can *translate* or move points to a new position by adding offsets to their coordinates:

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} x' \\ y' \\ z' \end{pmatrix} + \begin{pmatrix} t_x \\ t_y \\ t_z \end{pmatrix}$$

  - Note that this translates all points uniformly

Scaling

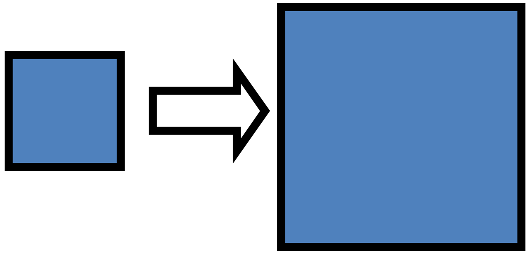
$$x' = x \cdot s_x, \qquad y' = y \cdot s_y$$

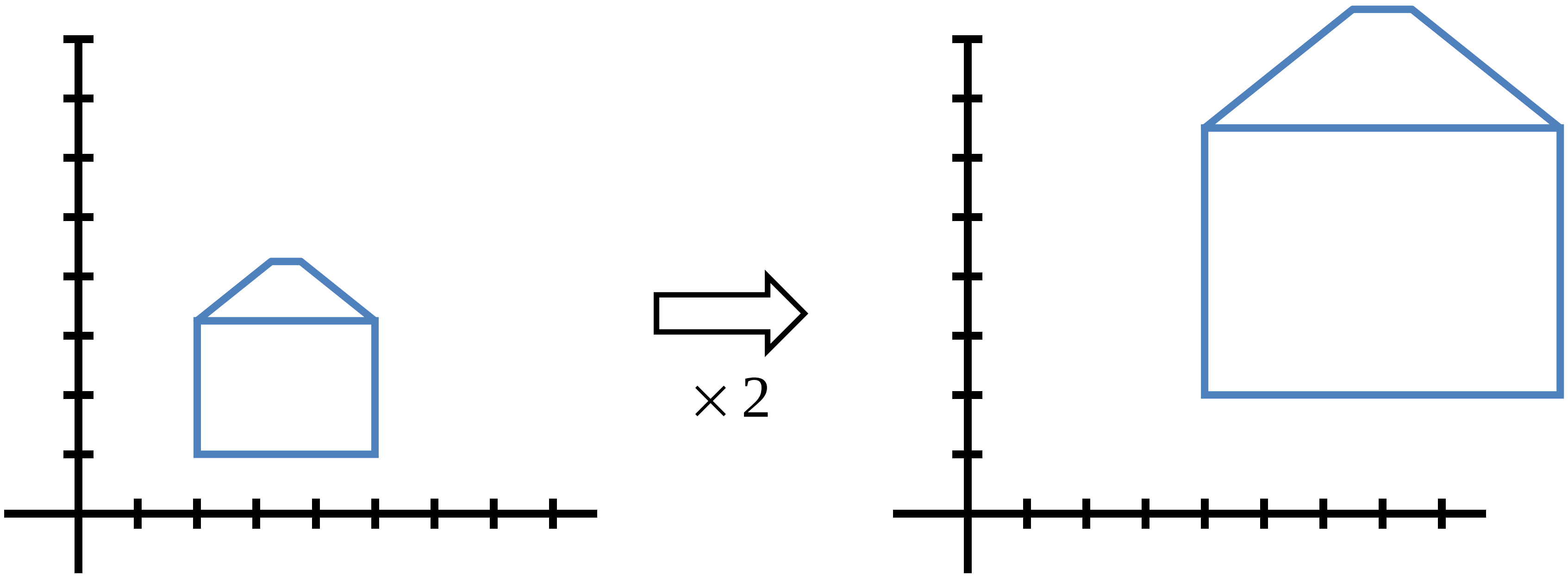


example: a line scaled using $s_x = s_y = 0.33$ is reduced in size and moved closer to the coordinate origin

Scaling

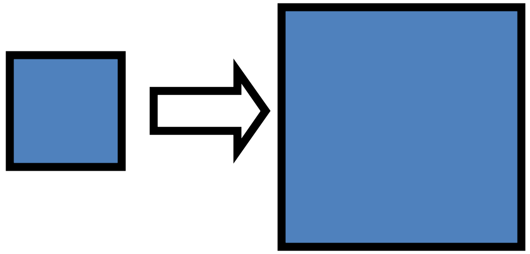
- Uniform scaling: $S_x = S_y$



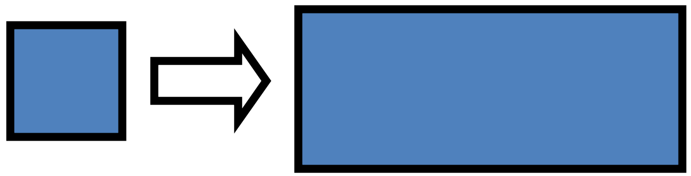


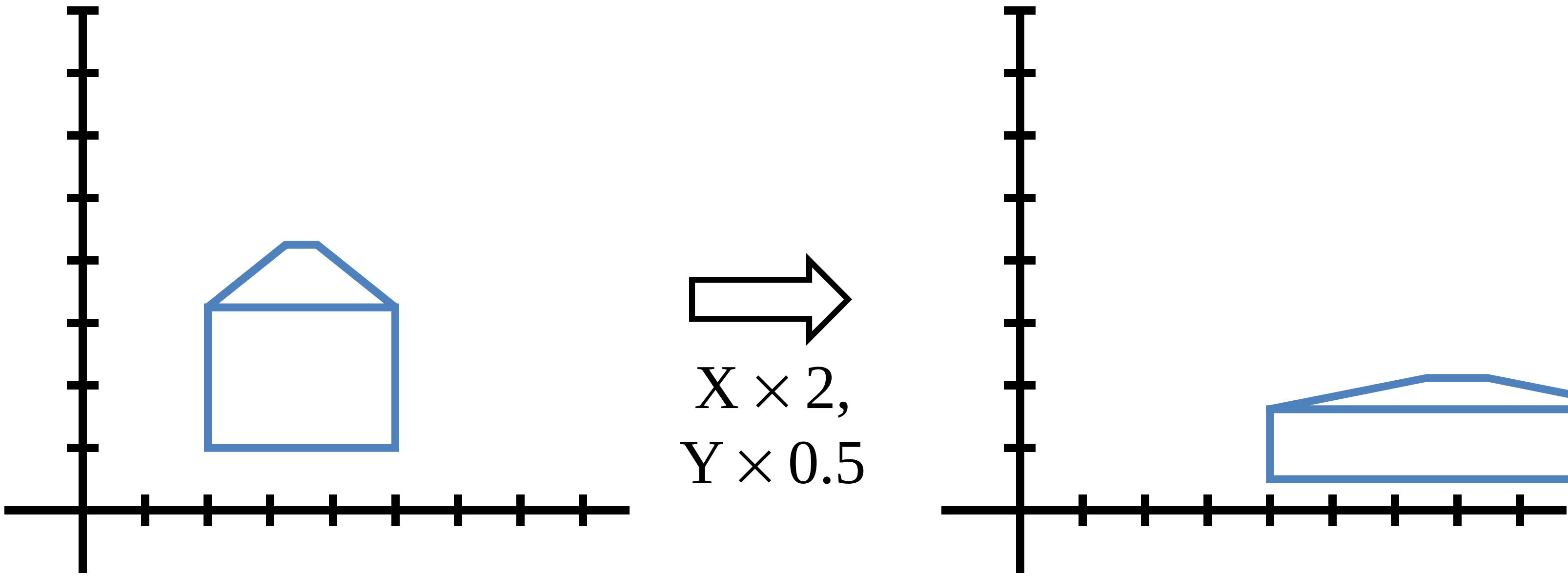
Scaling

- Uniform scaling: $S_{\mathbf{x}} = S_{\mathbf{y}}$



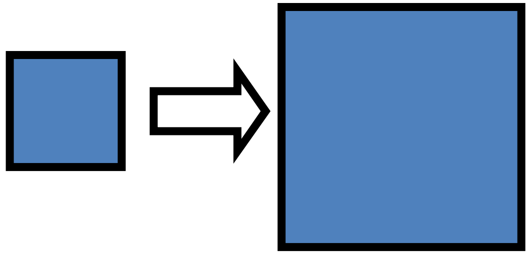
- Differential scaling: $S_{\mathbf{x}} \neq S_{\mathbf{y}}$



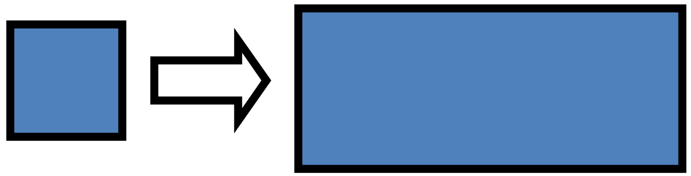


Scaling

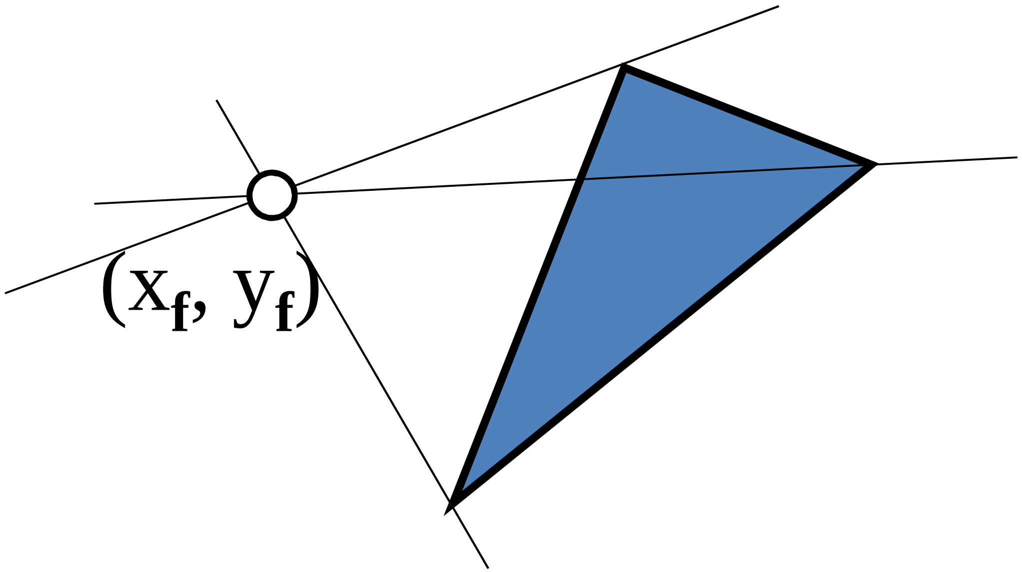
- uniform scaling: $S_{\mathbf{x}} = S_{\mathbf{y}}$



- differential scaling: $S_{\mathbf{x}} \neq S_{\mathbf{y}}$



- fixed point:



Scaling 3D

- Scaling operation

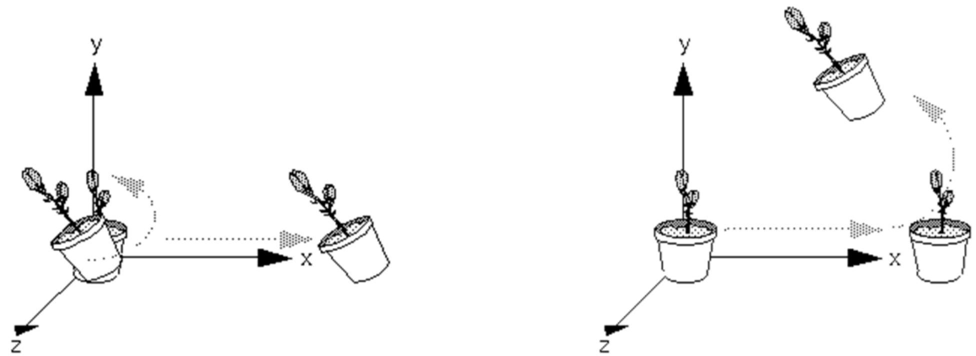
$$\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix} = \begin{pmatrix} s_x x \\ s_y y \\ s_z z \end{pmatrix}$$

- Matrix form

$$\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix} = \underbrace{\begin{pmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & s_z \end{pmatrix}}_{\textit{scaling matrix}} \begin{pmatrix} x \\ y \\ z \end{pmatrix}$$

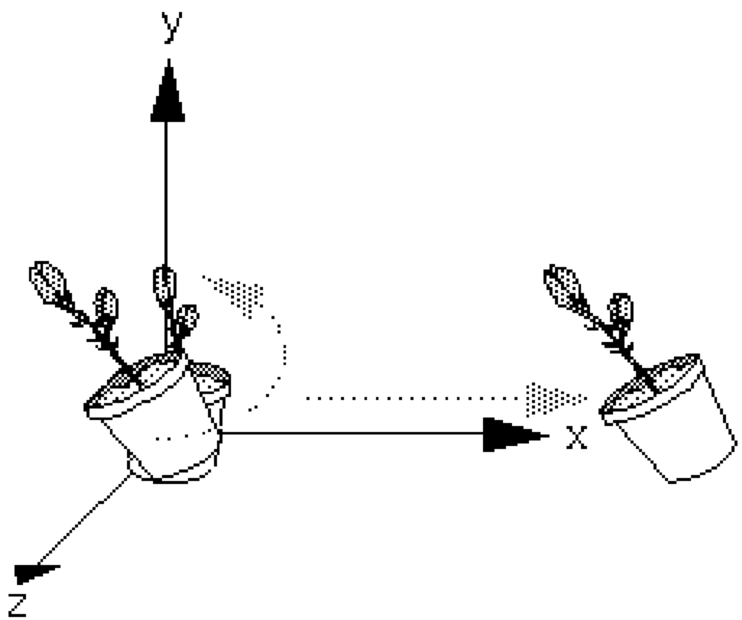
Why Matrices?

- All transformations representable by a matrix multiplication
  - Uniform way of representing transformations
  - Matrix multiplications are **associative**
    - $(M_1 \cdot M_2) \cdot M_3 = M_1 \cdot (M_2 \cdot M_3)$
    - Composite Transformations can be premultiplied
  - Not **commutative** $M_1 \cdot M_2 \neq M_2 \cdot M_1$ which is also true for transformations

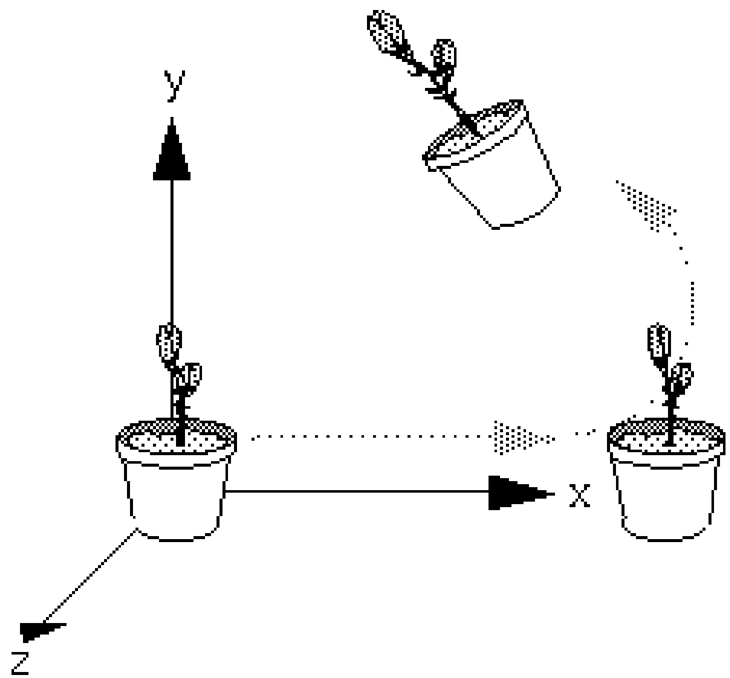


Transformation Order

- Order matters



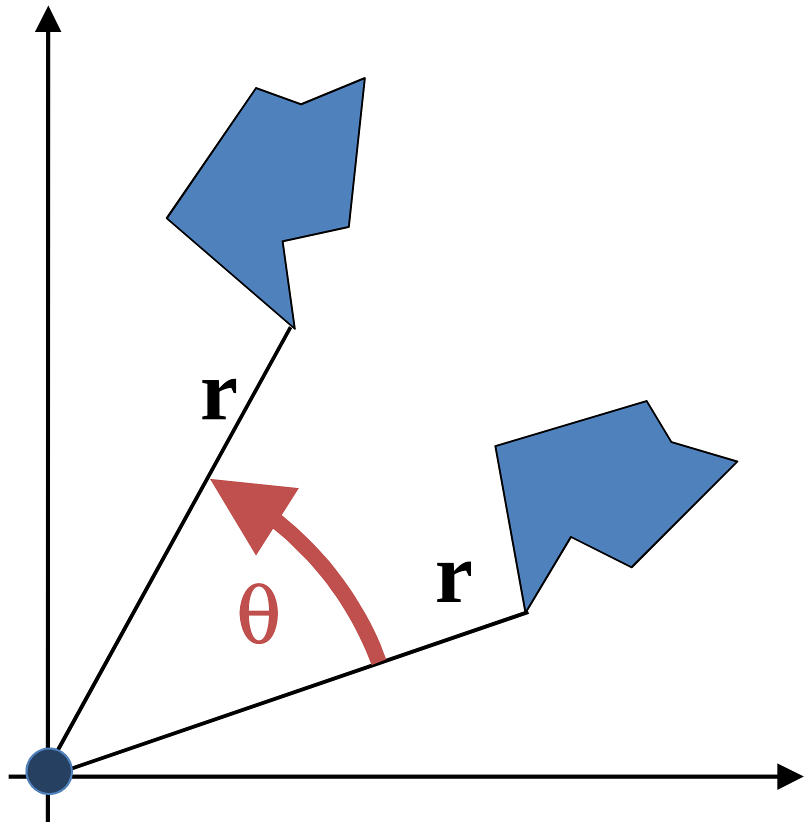
Trans * Rot * v



Rot * Trans * v

Rotation

- Rotation of an object by an angle θ around the origin



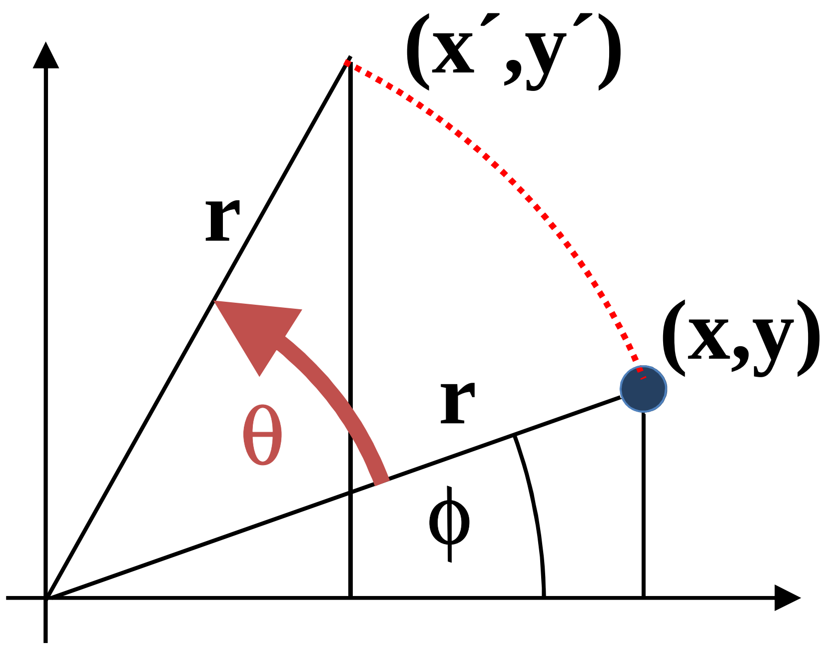
Rotation

- Positive angle $\Rightarrow$ ccw rotation

$x = r \cdot \cos\phi \qquad y = r \cdot \sin\phi$

$$
\begin{aligned}
x' &= r \cdot \cos(\phi + \theta) \\
&= \underline{r \cdot \cos\phi} \cdot \cos\theta - \underline{r \cdot \sin\phi} \cdot \sin\theta \\
&= \quad x \quad \cdot \cos\theta - \quad y \quad \cdot \sin\theta
\end{aligned}
$$

$$
\begin{aligned}
y' &= r \cdot \sin(\phi + \theta) \\
&= r \cdot \cos\phi \cdot \sin\theta + r \cdot \sin\phi \cdot \cos\theta
\end{aligned}
$$



$$
\begin{aligned}
x' &= x \cdot \cos\theta - y \cdot \sin\theta \\
y' &= x \cdot \sin\theta + y \cdot \cos\theta
\end{aligned}
$$

2-D Rotation

- This is easy to capture in matrix form

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

- 3-D is more complicated
  - Need to specify an
  - Simple cases: rotation about X, Y, Z axes

3-D Rotation

- *What does the 3-D rotation matrix look like for a rotation about the Z-axis?*
  - Build it coordinate-by-coordinate

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

3-D Rotation

- *What does the 3-D rotation matrix look like for a rotation about the Y-axis?*
  - Build it coordinate-by-coordinate

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

3-D Rotation

- *What does the 3-D rotation matrix look like for a rotation about the X-axis?*
  - Build it coordinate-by-coordinate

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

Rotation Matrices

- Remember basis!
- What does a z-rotation by 0°, 90° do to the basis?
  (draw transformation of basis vectors)

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

Rotation Matrices

- Rotation matrix is **orthogonal**
  - Columns/rows linearly independent
  - Columns/rows sum to 1
- The inverse of an orthogonal matrix is just its transpose:

$$\begin{bmatrix} a & b & c \\ d & e & f \\ h & i & j \end{bmatrix}^{-1} = \begin{bmatrix} a & b & c \\ d & e & f \\ h & i & j \end{bmatrix}^{T} = \begin{bmatrix} a & d & h \\ b & e & i \\ c & f & j \end{bmatrix}$$

Translation Matrices?

- But how to represent translation as a matrix?
- Answer: with **homogeneous coordinates**

Homogeneous Coordinates

Homogeneous Coordinates

- *Homogeneous coordinates:* represent coordinates in 3 dimensions with a 4-vector
- Points: Directions:

$$(x, y, z) = \begin{bmatrix} x/w \\ y/w \\ z/w \\ 1 \end{bmatrix} = \begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix}$$

$$(x, y, z) = \begin{bmatrix} x \\ y \\ z \\ 0 \end{bmatrix}$$

Homogeneous Coordinates

- *How can we represent translation as a 4x4 matrix?*
- A: Using the rightmost column:

$$\mathbf{T} = \begin{bmatrix} 1 & 0 & 0 & T_x \\ 0 & 1 & 0 & T_y \\ 0 & 0 & 1 & T_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Homogeneous Coordinates

- Homogeneous coordinates seem unintuitive, but they make graphics operations much easier
- Our transformation matrices are now 4x4:

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

Homogeneous Coordinates

- Homogeneous coordinates seem unintuitive, but they make graphics operations much easier
- Our transformation matrices are now 4x4:

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

Homogeneous Coordinates

- Homogeneous coordinates seem unintuitive, but they make graphics operations much easier
- Our transformation matrices are now 4x4:

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

Homogeneous Coordinates

- Homogeneous coordinates seem unintuitive, but they make graphics operations much easier
- Our transformation matrices are now 4x4:

$$\mathbf{S} = \begin{bmatrix} S_x & 0 & 0 & 0 \\ 0 & S_y & 0 & 0 \\ 0 & 0 & S_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Inverse Matrices

- translation

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

- rotation

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

- scaling

$$S^{-1}(s_x, s_y) = S(1/s_x, 1/s_y)$$

Composite Transformations (1)

n transformations are applied after each other on a point P, these transformations are represented by matrices $M_1, M_2, \ldots, M_n$.

$$P' = M_1 \times P$$

$$P'' = M_2 \times P'$$

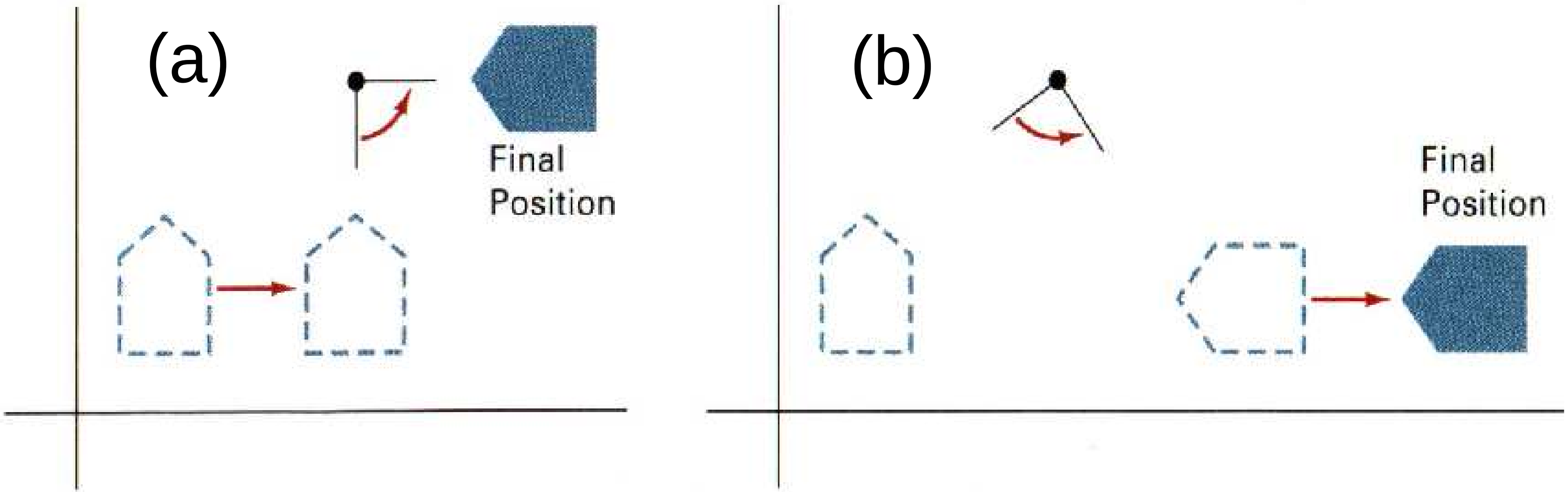
$$\ldots$$

$$P^{(n)} = M_n \times P^{(n-1)}$$

shorter: $P^{(n)} = (M_n \times \ldots (M_2 \times (M_1 \times P)) \ldots)$

Transformations are not commutative!

- Reversing the order in which a sequence of transformations is performed may affect the transformed position of an object.
- In (a), an object is first translated, then rotated. In (b), the object is rotated first, then translated.



Translation Matrices

- Now that we can represent translation as a matrix, we can composite it with other transformations
- Ex: rotate 90° about **X**, then 10 units down **Z**:

$$\begin{bmatrix} x' \\ y' \\ z' \\ w' \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 10 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(90°) & -\sin(90°) & 0 \\ 0 & \sin(90°) & \cos(90°) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix}$$

Translation Matrices

- Now that we can represent translation as a matrix, we can composite it with other transformations
- Ex: rotate 90° about **X**, then 10 units down **Z**:

$$\begin{bmatrix} x' \\ y' \\ z' \\ w' \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 10 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix}$$

Translation Matrices

- Now that we can represent translation as a matrix, we can composite it with other transformations
- Ex: rotate 90° about **X**, then 10 units down **Z**:

$$\begin{bmatrix} x' \\ y' \\ z' \\ w' \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 10 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix}$$

Translation Matrices

- Now that we can represent translation as a matrix, we can composite it with other transformations
- Ex: rotate 90° about **X**, then 10 units down **Z**:

$$\begin{bmatrix} x' \\ y' \\ z' \\ w' \end{bmatrix} = \begin{bmatrix} x \\ -z \\ y+10 \\ w \end{bmatrix}$$