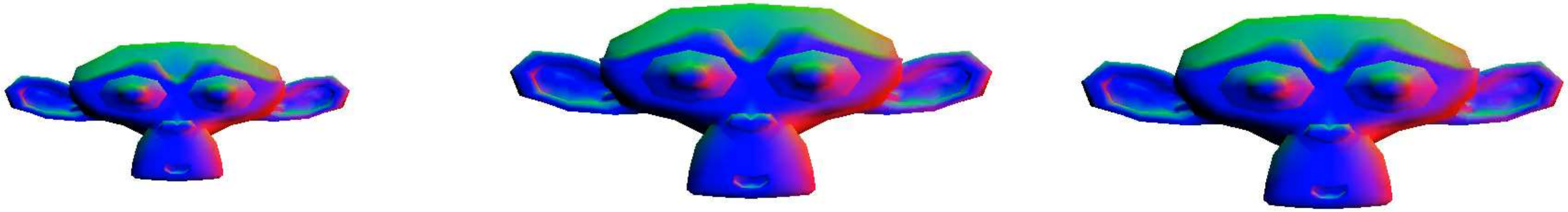
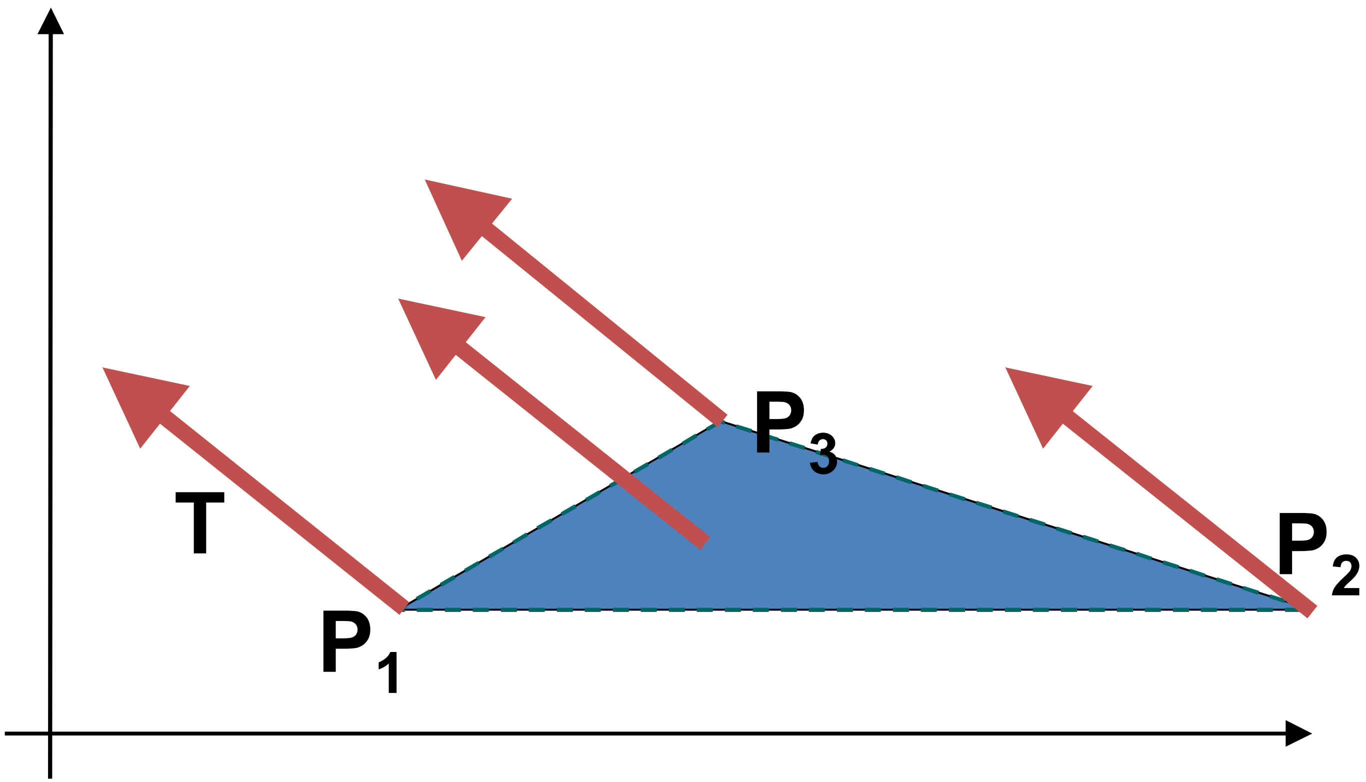
Transformations



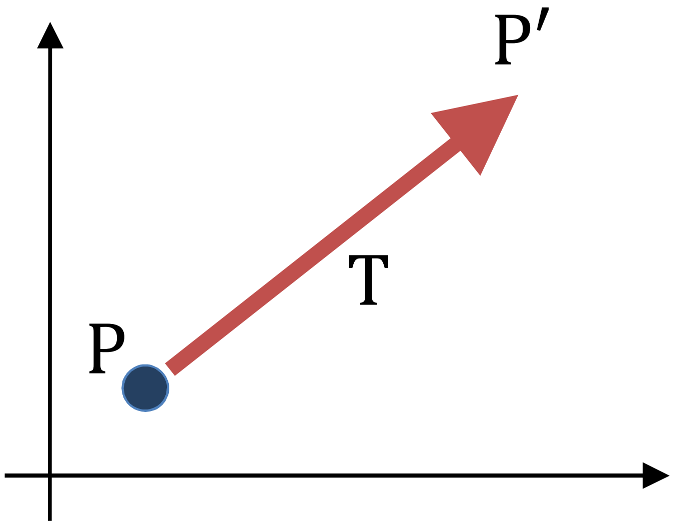
Rigid body transformation

- Object transformed by transforming boundary points



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

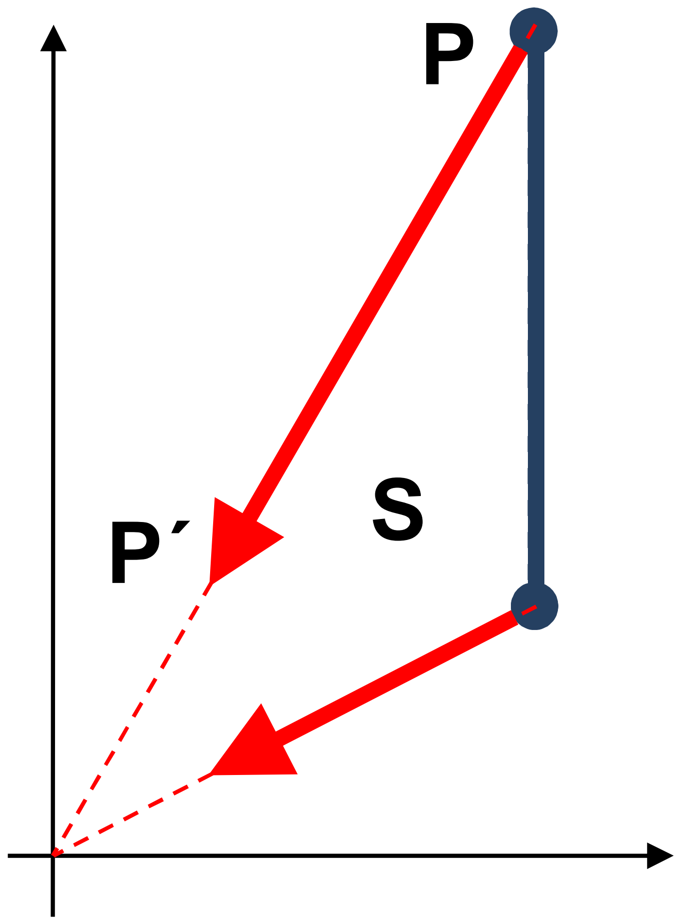
- For convenience we usually describe objects in relation to their own coordinate system
- We can *translate* or move points to a new position by adding offsets to their coordinates:

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

  - Note that this translates all points uniformly

Scaling

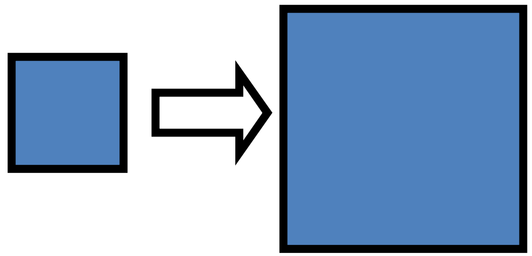
$$x' = x \cdot s_x, \qquad y' = y \cdot s_y$$

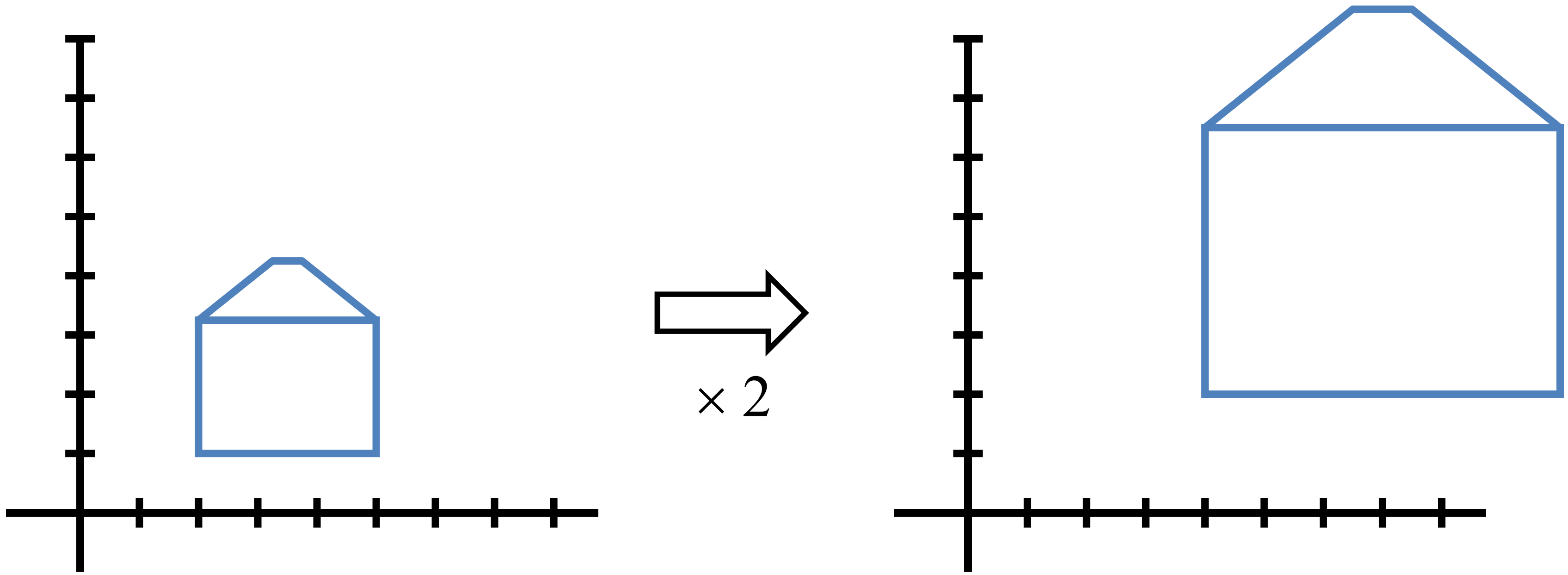


example: a line scaled using s_x=s_y=0.33 is reduced in size and moved closer to the coordinate origin

Scaling

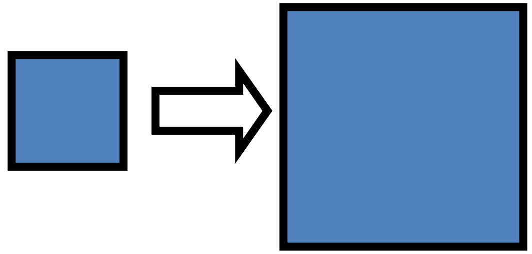
- Uniform scaling: $S_{\mathbf{x}} = S_{\mathbf{y}}$



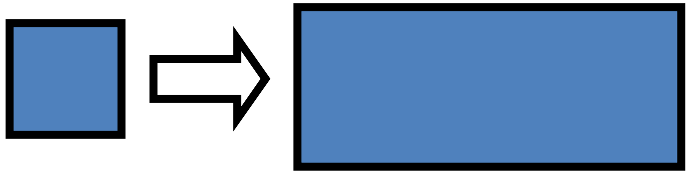


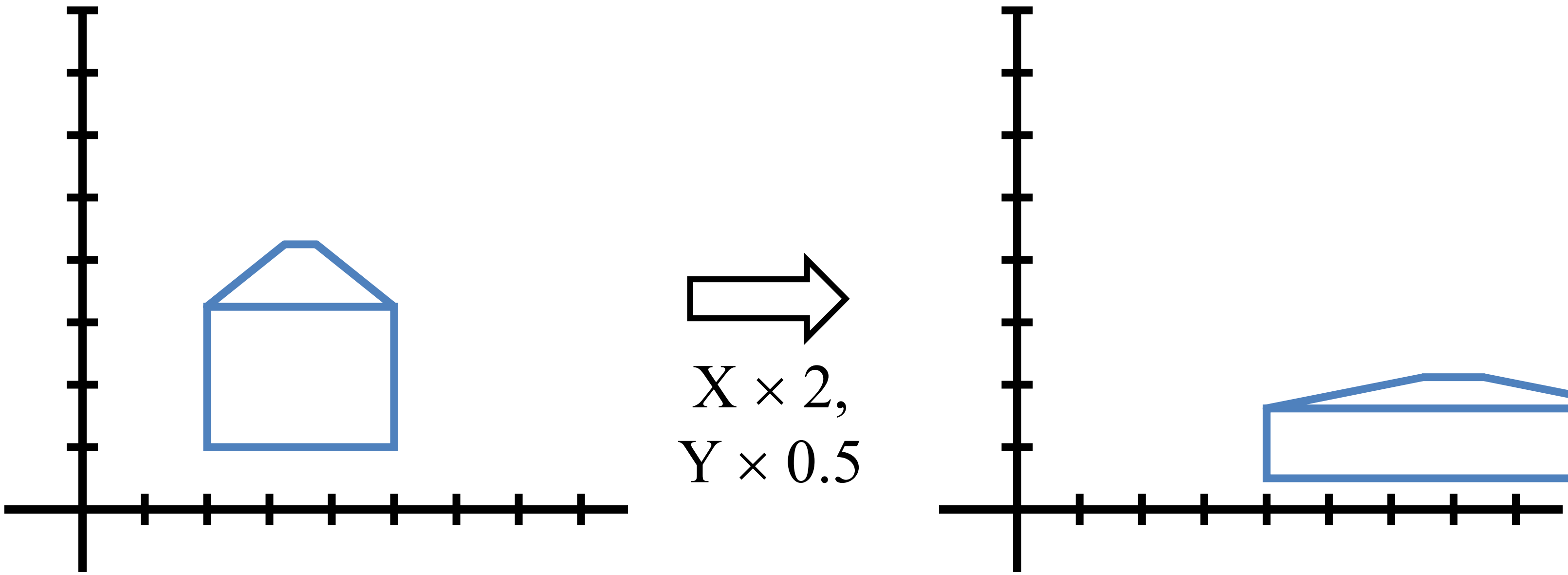
Scaling

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



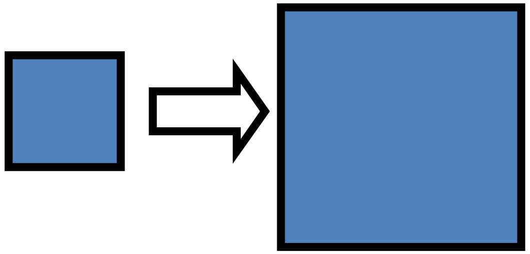
- Differential scaling: $S_{\mathbf{x}} \neq S_{\mathbf{y}}$



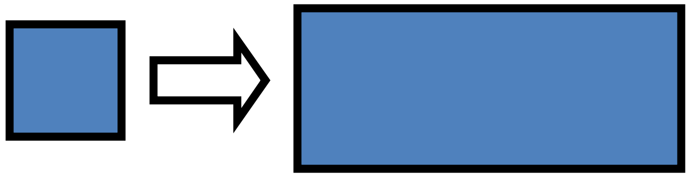


Scaling

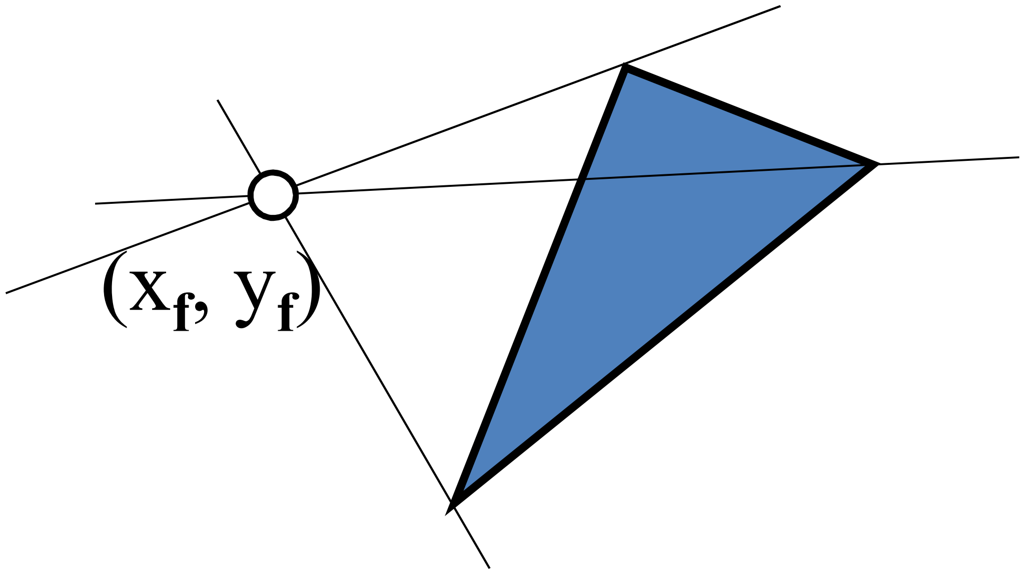
- uniform scaling: $S_x = S_y$



- differential scaling: $S_x \neq S_y$

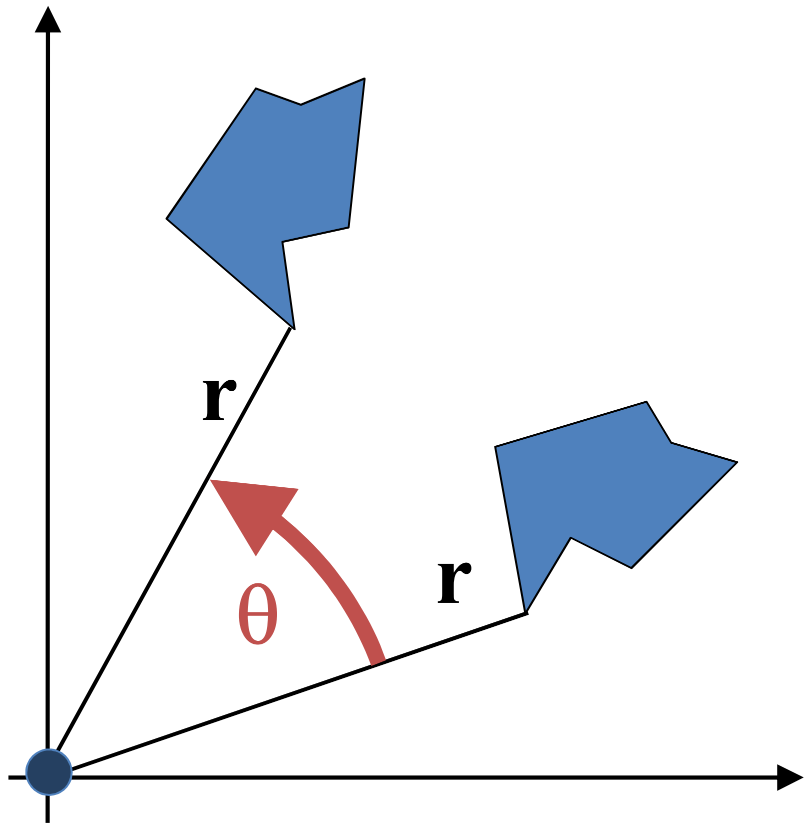


- fixed point:



Rotation

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



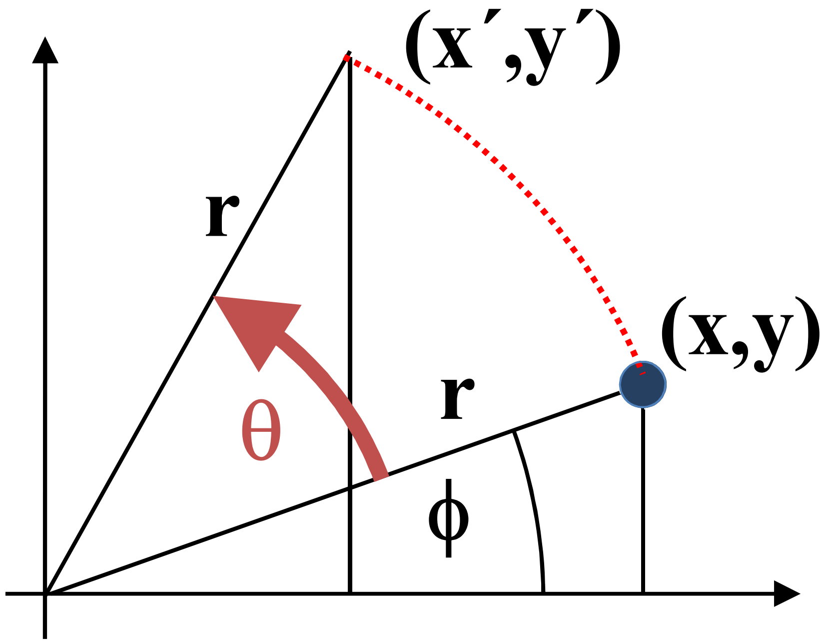
Rotation

- Positive angle $\Rightarrow$ ccw rotation

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

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

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

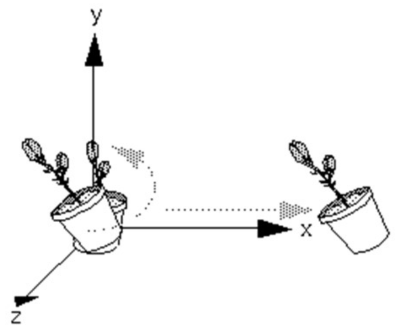


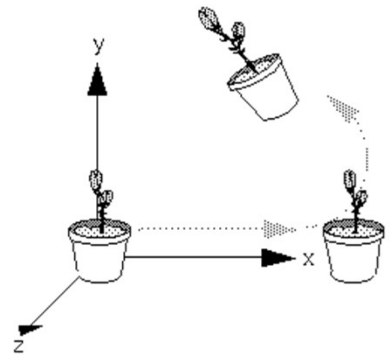
$$x' = x \cdot \cos\theta - y \cdot \sin\theta$$
$$y' = x \cdot \sin\theta + y \cdot \cos\theta$$

Transformation Matrices

Why Matrices?

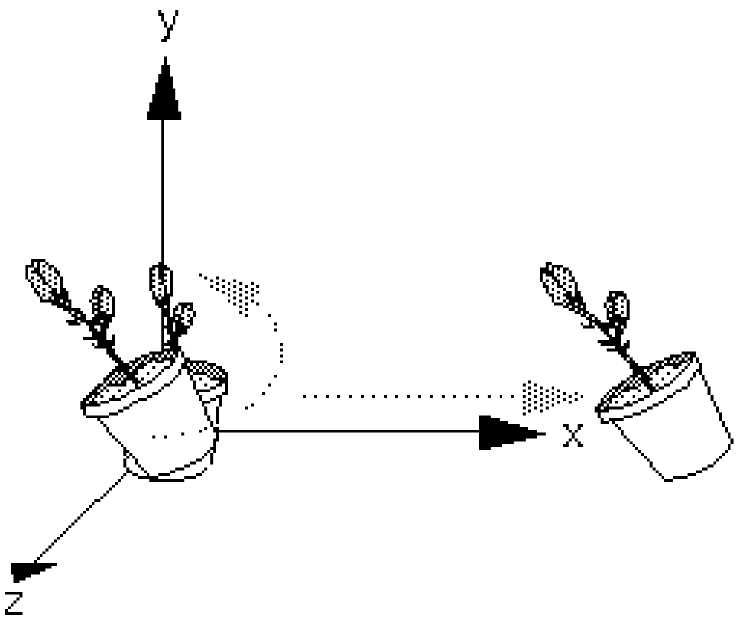
- All transformations representable by a matrix multiplication
  - Uniform way of representing transformations
  - Matrix multiplications are **associative**
    - $(M_1 \cdot M_2) \cdot M_3 = M_1 \cdot (M_2 \cdot M_3)$
    - Composite Transformations can be premultiplied
  - Not **commutative** $M_1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



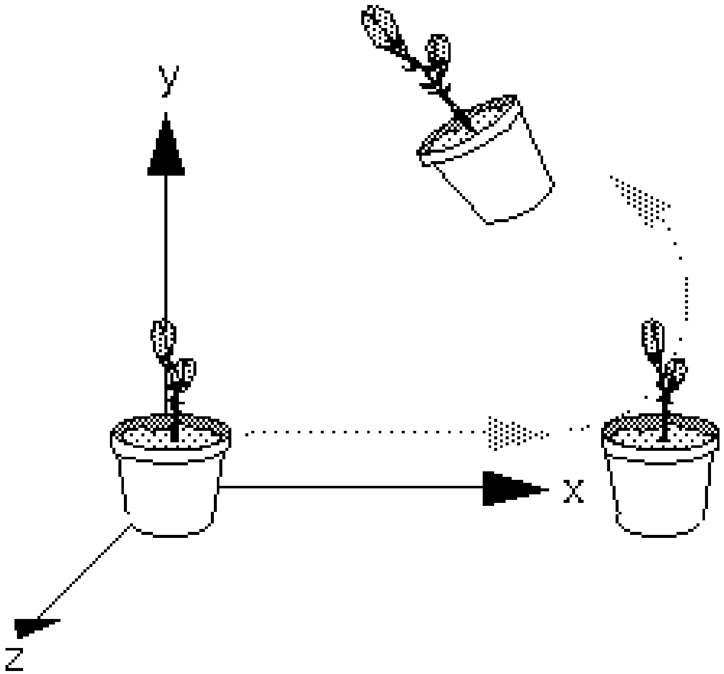


Transformation Order

- Order matters



Trans * Rot * v



Rot * Trans * v

Scaling Matrix

- Operation

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

- Matrix form

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

2-D Rotation

- Operation

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

- 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)

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

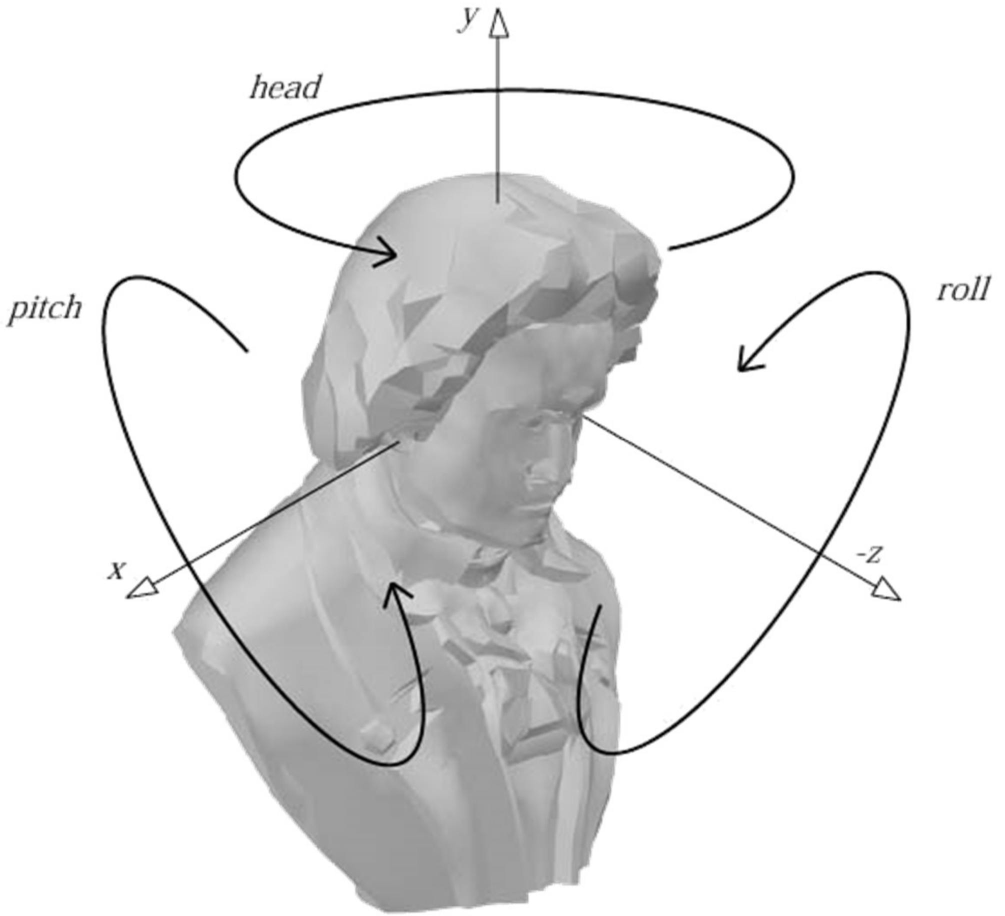
Rotation Matrices

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

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

The Euler Transform

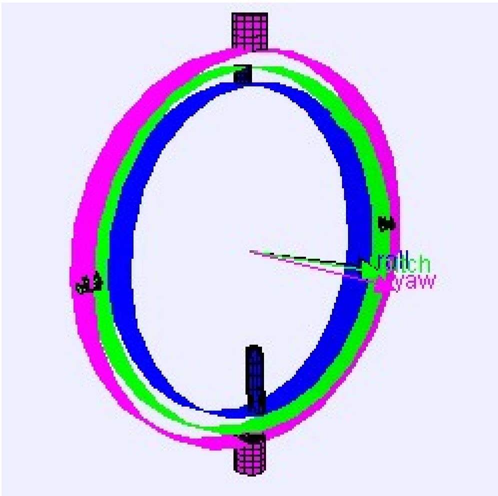
- $\mathrm{E}(h, p, r) = \mathbf{R}_z(r)\mathbf{R}_x(p)\mathbf{R}_y(h)$
- $h =$ head
- $p =$ pitch
- $r =$ roll
- Just 3 rotations



Gimbal Lock

- 3 rotation matrices allow for arbitrary oriantation in space

- Case were rank of $\mathrm{E}(h, p, r)$ drops below 3
- Example $h = 0,\ p = \frac{\pi}{2}$

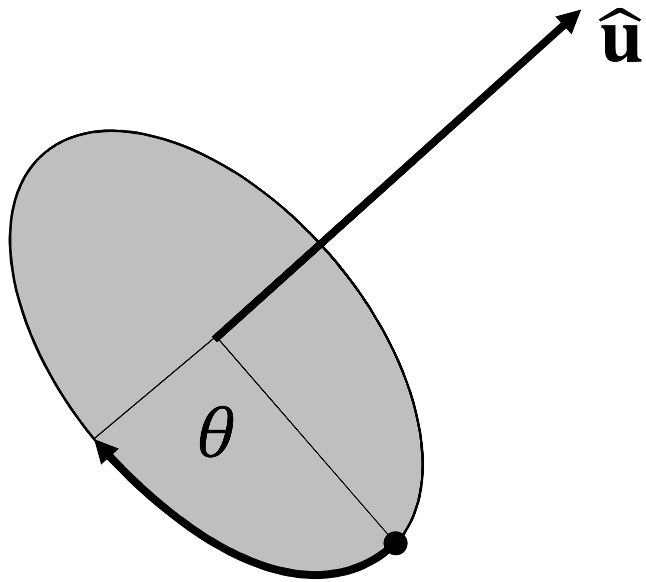


Quaternions

- $\mathbf{q} = \left(\mathrm{q}_x, \mathrm{q}_y, \mathrm{q}_z, \mathrm{q}_w\right) = (\mathbf{q}_v, \mathrm{q}_w)$
$$= \boldsymbol{i}\mathrm{q}_x + \boldsymbol{j}\mathrm{q}_y + \boldsymbol{k}\mathrm{q}_z + \mathrm{q}_w$$
- Extension of imaginary numbers
$\boldsymbol{ii} = \boldsymbol{jj} = \boldsymbol{kk} = -\mathbf{1}$
$\boldsymbol{ij} = -\boldsymbol{ji} = \boldsymbol{k}$
$\boldsymbol{jk} = -\boldsymbol{kj} = \boldsymbol{i}$
$\boldsymbol{ki} = -\boldsymbol{ik} = \boldsymbol{j}$
- Focus on unit quaternions
$$||\widehat{\mathbf{q}}|| = \mathrm{q}_x{}^2 + \mathrm{q}_y{}^2 + \mathrm{q}_z{}^2 + \mathrm{q}_w{}^2 = 1$$

Quaternions and Rotations

- A rotation of an angle θ about an axis $\hat{\mathbf{u}} = (x, y, z)$ can be represented by $\hat{\mathbf{q}} = \left(\hat{\mathbf{u}} \sin\frac{\theta}{2}, \cos\frac{\theta}{2}\right)$
  - Compact
  - Avoids Gimbal Lock



- $\mathbf{q}^* \coloneqq (-\mathbf{q}_v, \mathrm{q}_w)$
- A (pure) vector $\mathbf{p} = \left(\mathrm{p}_x, \mathrm{p}_y, \mathrm{p}_z, 0\right) = \boldsymbol{i}\mathrm{p}_x + \boldsymbol{j}\mathrm{p}_y + \boldsymbol{k}\mathrm{p}_z$ can be rotated by $\hat{\mathbf{q}}\mathbf{p}\hat{\mathbf{q}}^*$

Translation Matrices?

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

Homogeneous Coordinates

Homogeneous Coordinates

- *Homogeneous coordinates:* represent coordinates with an additional dimension
- Points:

$$\begin{pmatrix} x \\ y \end{pmatrix} \hat{=} \begin{pmatrix} x/w \\ y/w \\ 1 \end{pmatrix} = \begin{pmatrix} x \\ y \\ w \end{pmatrix}$$

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} \hat{=} \begin{pmatrix} x/w \\ y/w \\ z/w \\ 1 \end{pmatrix} = \begin{pmatrix} x \\ y \\ z \\ w \end{pmatrix}$$

Directions:

$$\begin{pmatrix} x \\ y \end{pmatrix} \hat{=} \begin{pmatrix} x \\ y \\ 0 \end{pmatrix}$$

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} \hat{=} \begin{pmatrix} x \\ y \\ z \\ 0 \end{pmatrix}$$

Translation Matrix

- Operation

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

- Matrix form

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

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 $M_1, M_2, \ldots, M_n$.

$$P' = M_1 \times P$$

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

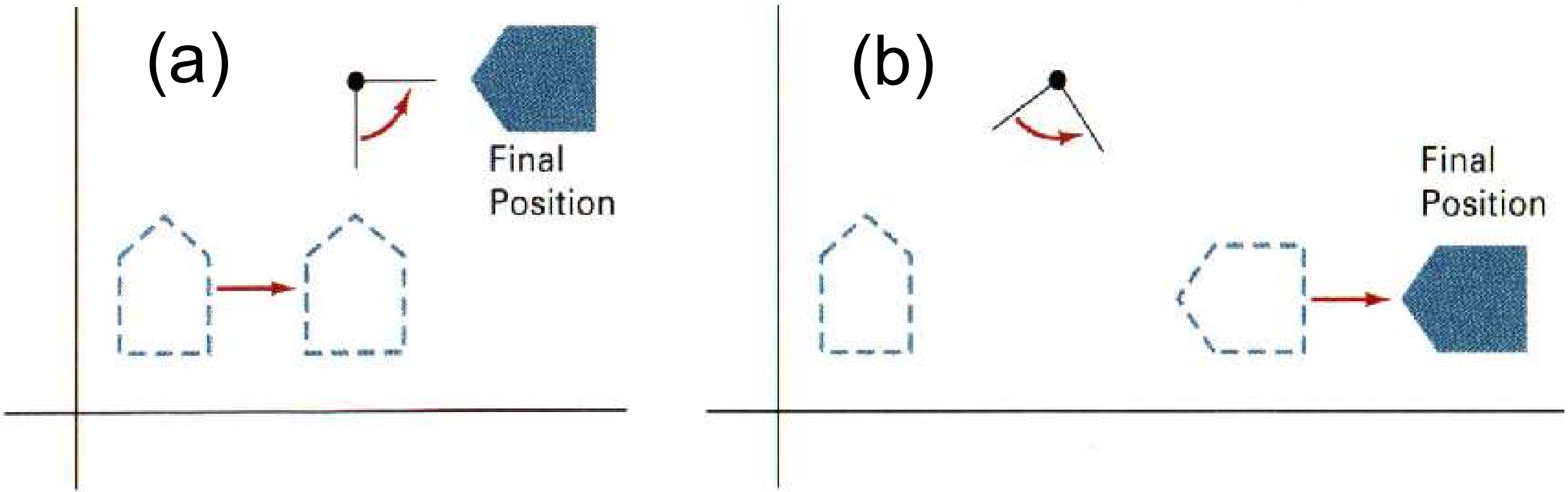
$$\ldots$$

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

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**:

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