Transformations

Lecture 4

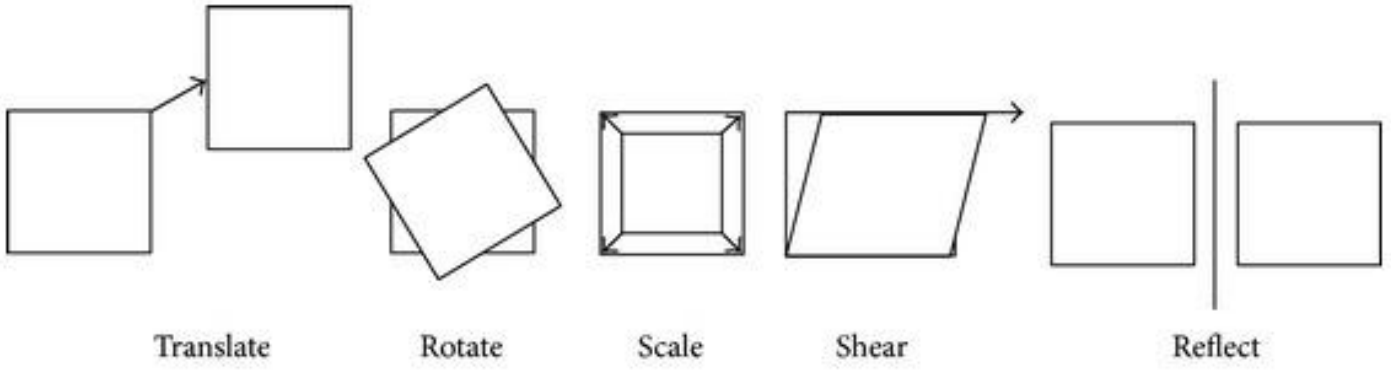
Yoonsang Lee and Taesoo Kwon
Spring 2019

Topics Covered

- 2D Affine Transformation
  - What is Transformation?
  - Linear Transformation, Translation
  - Affine Transformation
- Composing Transformations & Homogeneous Coordinates
  - Composing Transformations
  - Homogeneous Coordinates
- 3D Affine Transformations
  - 3D Cartesian Coordinate System
  - Transformations in 3D

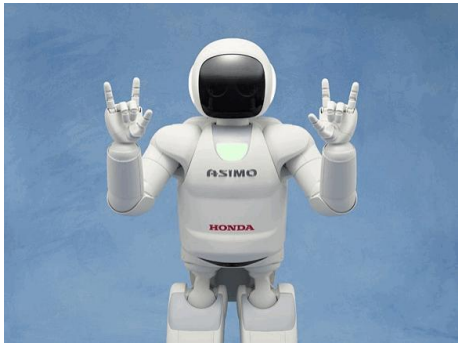
What is Transformation？

- Geometric **Transformation**
  - One-to-one mapping (function) of a set having some geometric structure to itself or another such set.
  - *“moving a set of points”*
- Examples:

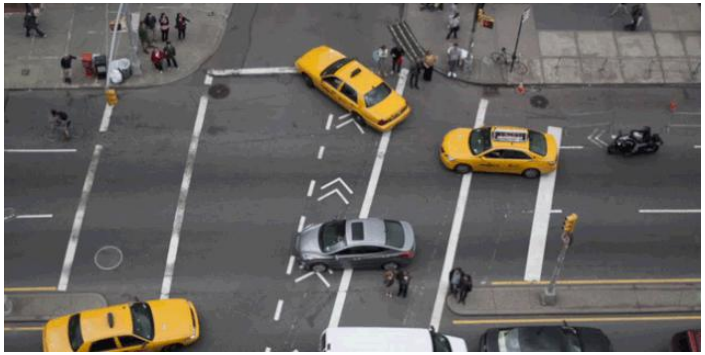


Why transformations are important in CG?

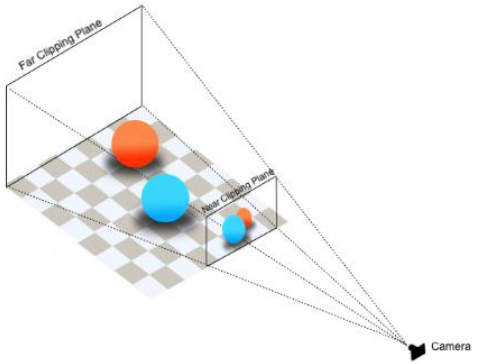
- Movement



- Image/object manipulation



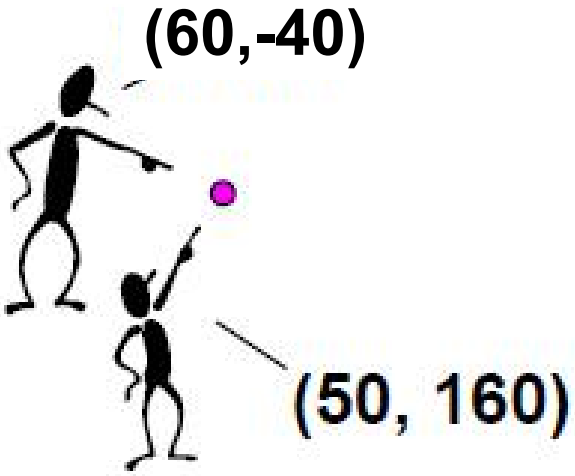
- Viewing transform



Geometry

- **A part of mathematics concerned with questions of size, shape, and relative positions of figures**
- **Coordinates are used to represent points and vectors**
  - **a naming scheme**
  - **The same point can be described by different coordinates**

points



vectors

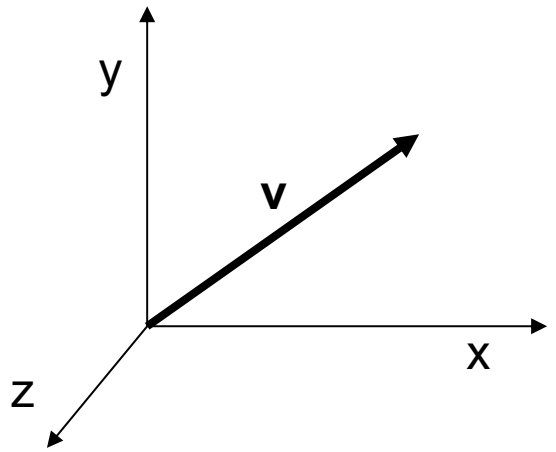
Points

- **Conceptually, points and vectors are very different**
  - **A point $\dot{p}$ is a place in space**
  - **A vector $\vec{v}$ describes a direction independent of position (pay attentions notations)**

Notation

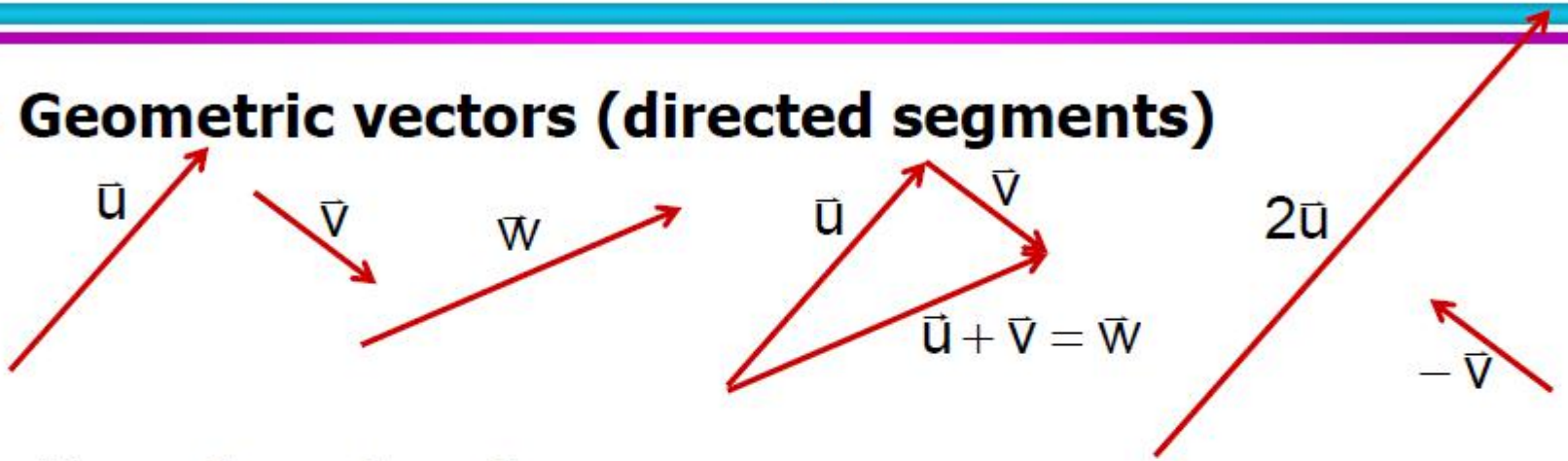
- 3D Euclidean vector

$$\vec{v} = \begin{bmatrix} a \\ b \\ c \end{bmatrix} = \begin{bmatrix} a & b & c \end{bmatrix}^t$$



Example Vector Spaces

- **Geometric vectors (directed segments)**



- **N-tuples of scalars**

$$\bar{u} = (1,3,7)^t \qquad \bar{u} + \bar{v} = (3,5,4)^t = \bar{w}$$

$$\bar{v} = (2,2,-3)^t \qquad 2\bar{u} = (2,6,14)^t$$

$$\bar{w} = (3,5,4)^t \qquad -\bar{v} = (-2,-2,3)^t$$

- **We can use N-tuples to represent vectors**

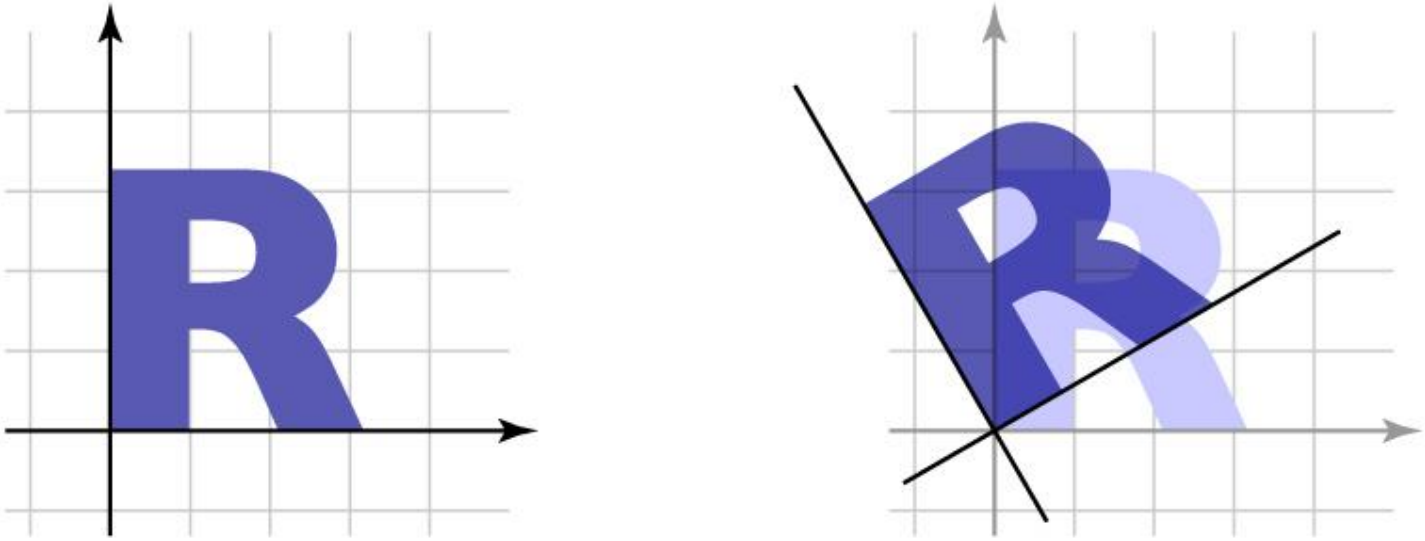
KAIST

Transforming geometry

- "Moving a set of points/vectors"

  Transformation T maps any input vector v in the vector space S to T(v).

$$S \rightarrow \{T(\mathbf{v}) \,|\, \mathbf{v} \in S\}$$



Linear transformations

- One way to define a transformation is by matrix multiplication:

$$T(\mathbf{v}) = M\mathbf{v}$$

- Such transformations are *linear*, which is to say:

$$T(a\mathbf{u} + \mathbf{v}) = aT(\mathbf{u}) + T(\mathbf{v})$$

(and in fact all linear transformations can be written this way)

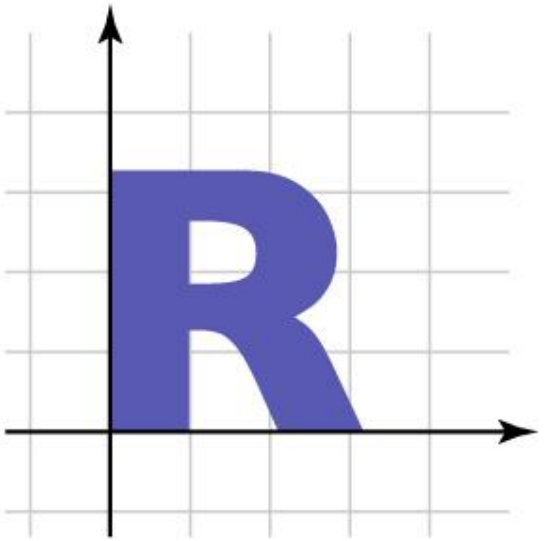
Geometry of 2D linear trans.

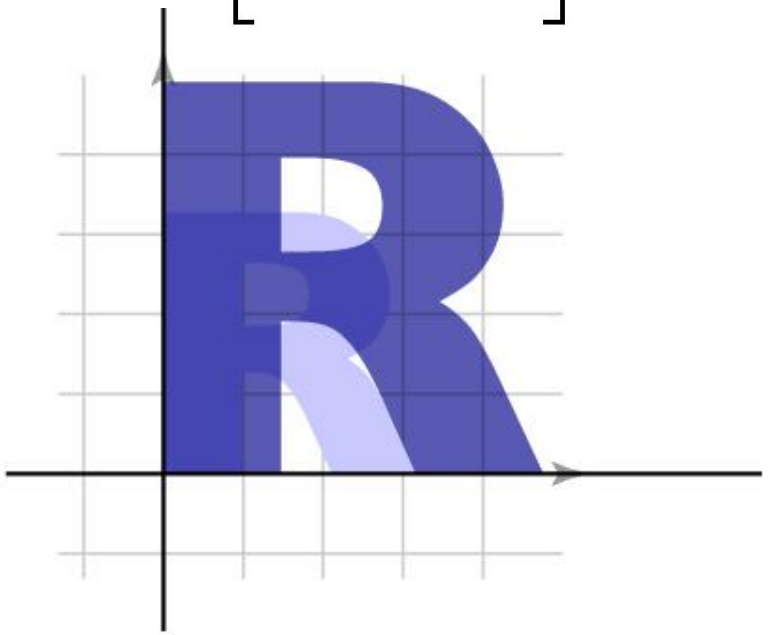
- 2x2 matrices have simple geometric interpretations
  - uniform scale
  - non-uniform scale
  - rotation
  - shear
  - reflection
- Reading off the matrix

Linear transformation gallery

- Uniform scale $\begin{bmatrix} s & 0 \\ 0 & s \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} sx \\ sy \end{bmatrix}$

$$\begin{bmatrix} 1.5 & 0 \\ 0 & 1.5 \end{bmatrix}$$

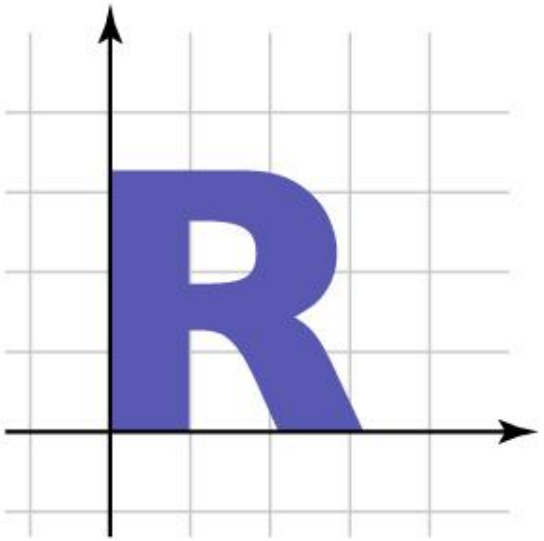


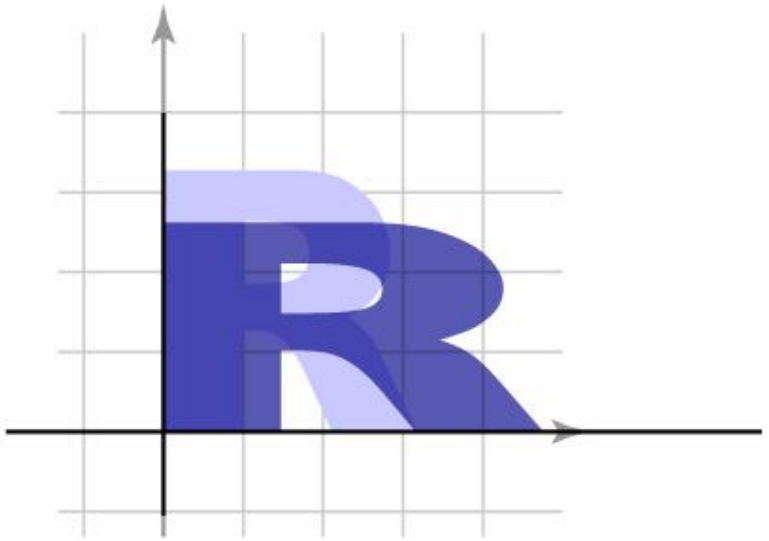


Linear transformation gallery

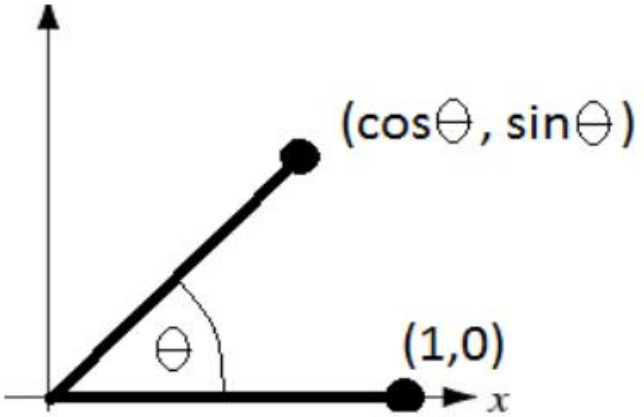
- Nonuniform scale $\begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} s_x x \\ s_y y \end{bmatrix}$

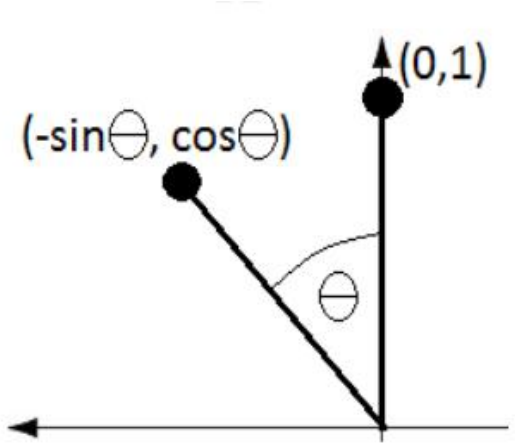
$$\begin{bmatrix} 1.5 & 0 \\ 0 & 0.8 \end{bmatrix}$$





Rotation



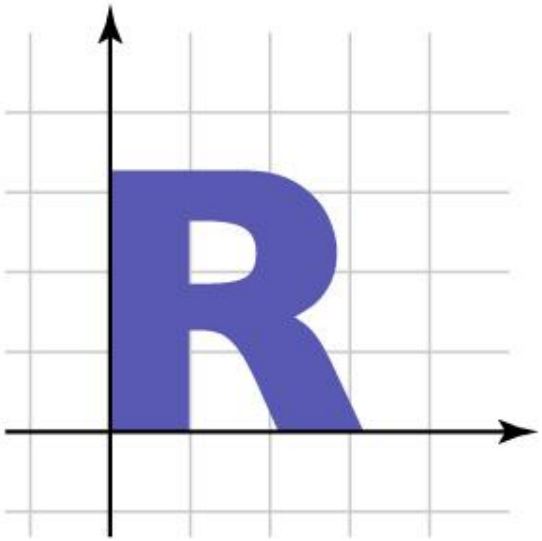


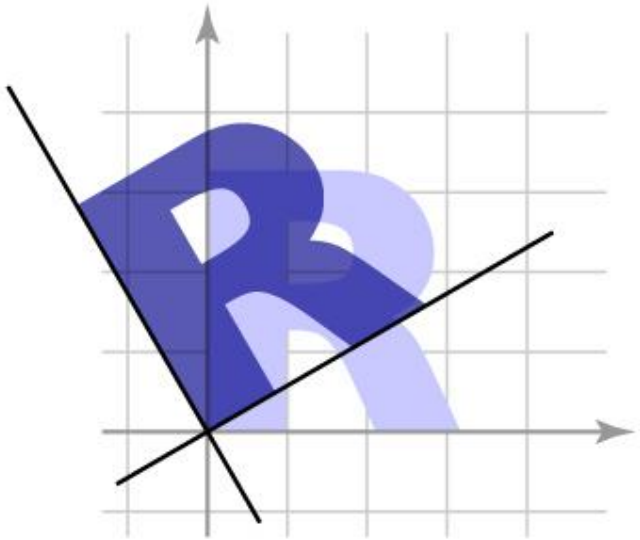
$$\Rightarrow R_\theta = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$$

Linear transformation gallery

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

$$\begin{bmatrix} 0.866 & -.05 \\ 0.5 & 0.866 \end{bmatrix}$$

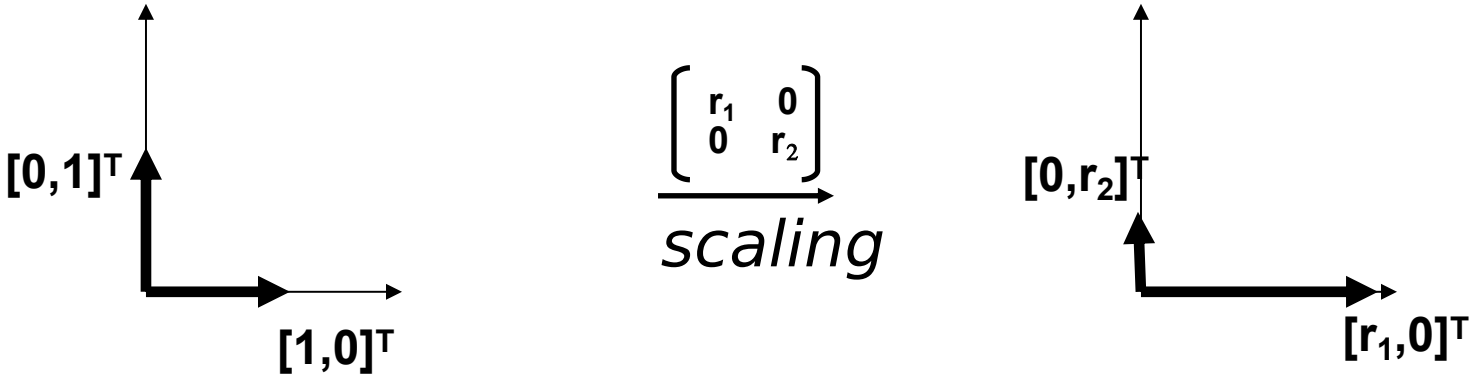


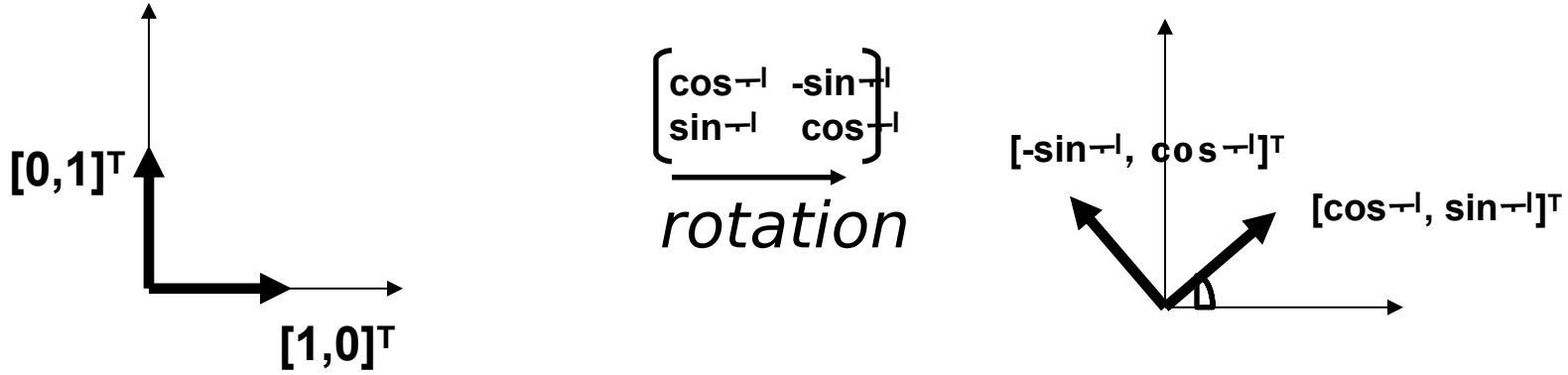


(Rotate 30 deg ccw)

Matrices: Scaling, Rotation, Identity

- Scaling without rotation =| "**diagonal** matrix"
- Rotation without stretching =| "**orthonormal** matrix" **O**
- **Identity** ("do nothing") matrix = unit scaling, no rotation

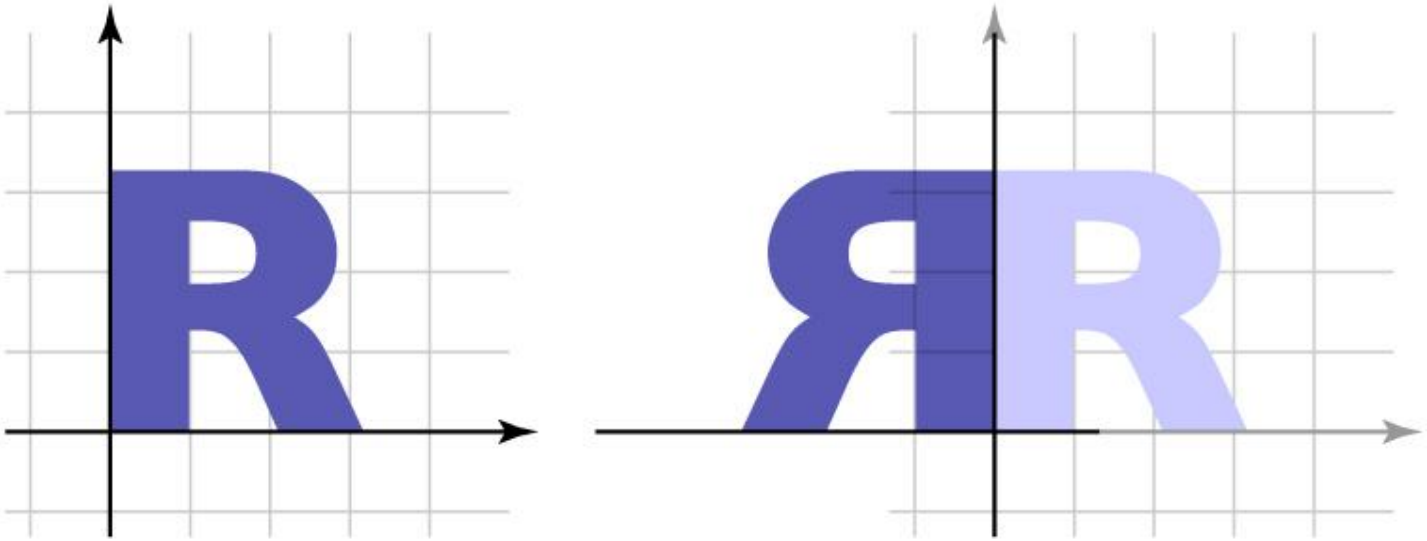




Linear transformation gallery

- Reflection
  - can consider it a special case of nonuniform scale

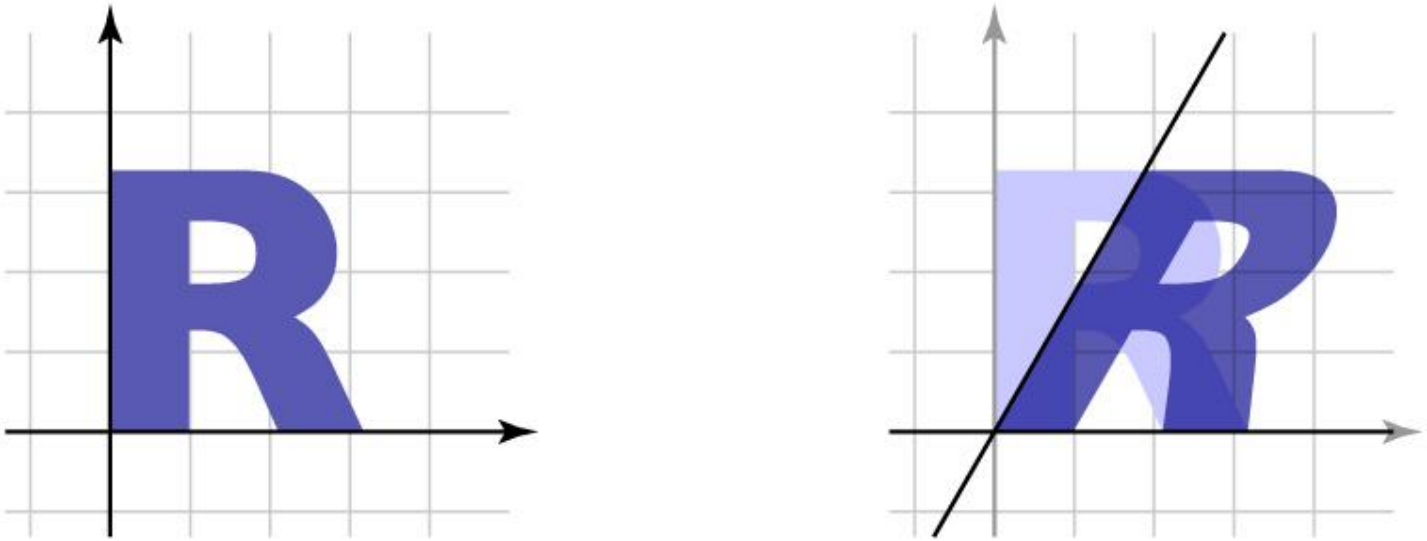
$$\begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix}$$



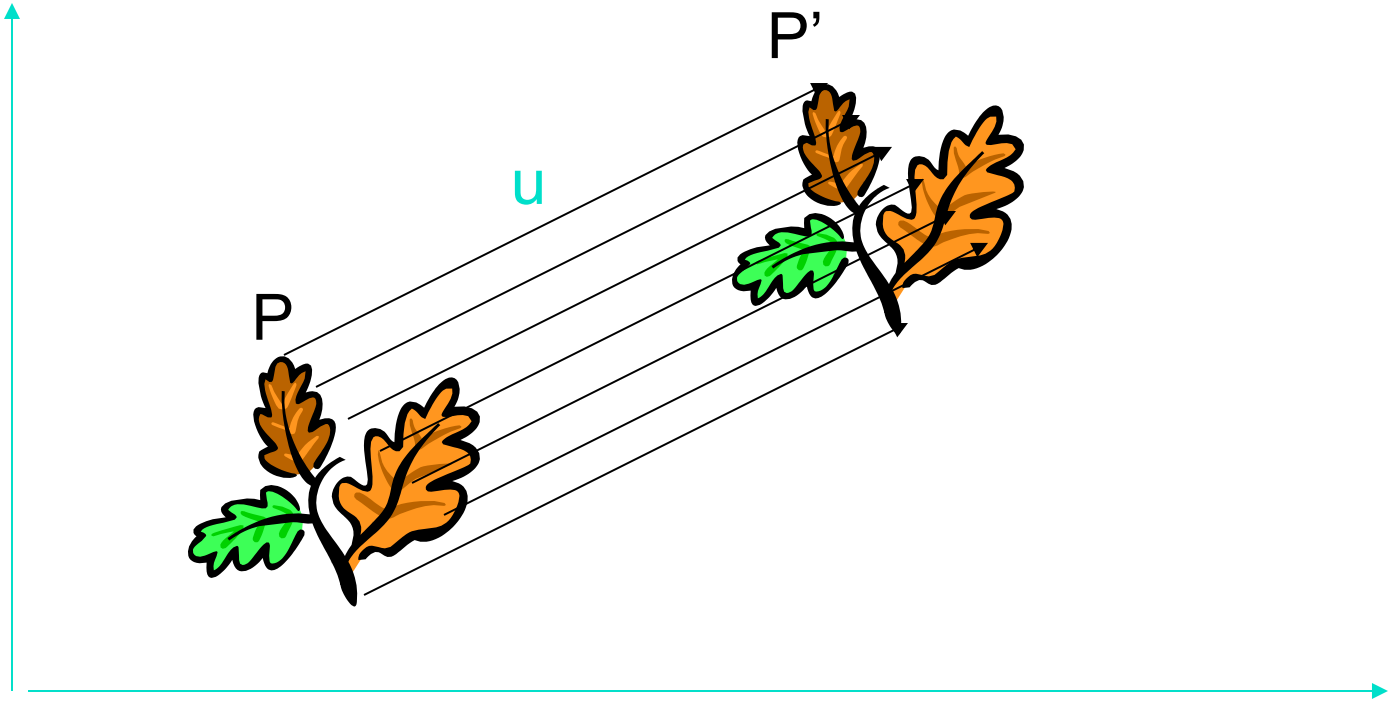
Linear transformation gallery

- Shear $\begin{bmatrix} 1 & a \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} x + ay \\ y \end{bmatrix}$

$$\begin{bmatrix} 1 & 0.5 \\ 0 & 1 \end{bmatrix}$$

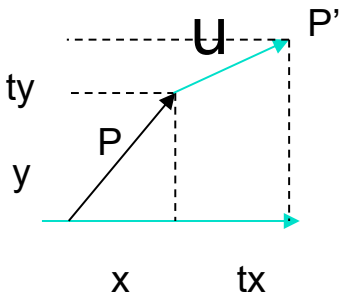


Translation



$$T(\mathbf{v}) = \mathbf{v} + \mathbf{u}$$

$$T^{-1}(\mathbf{v}) = \mathbf{v} - \mathbf{u}$$



Is translation linear transformation?

- No. because it cannot be represented using a simple matrix multiplication.

- We can express using vector addition:

$$T(\mathbf{v}) = \mathbf{v} + \mathbf{u}$$

- Combining with linear transformation:

$$T(\mathbf{v}) = M\mathbf{v} + \mathbf{u}$$

➡ **Affine transformation**

Review

- Linear transformation
  - Scale, rotation, reflection, shear
  - Represented as matrix multiplication

$$T(\mathbf{v}) = M\mathbf{v}$$

- Translation
  - Not a linear transformation
  - Can be expressed using vector addition

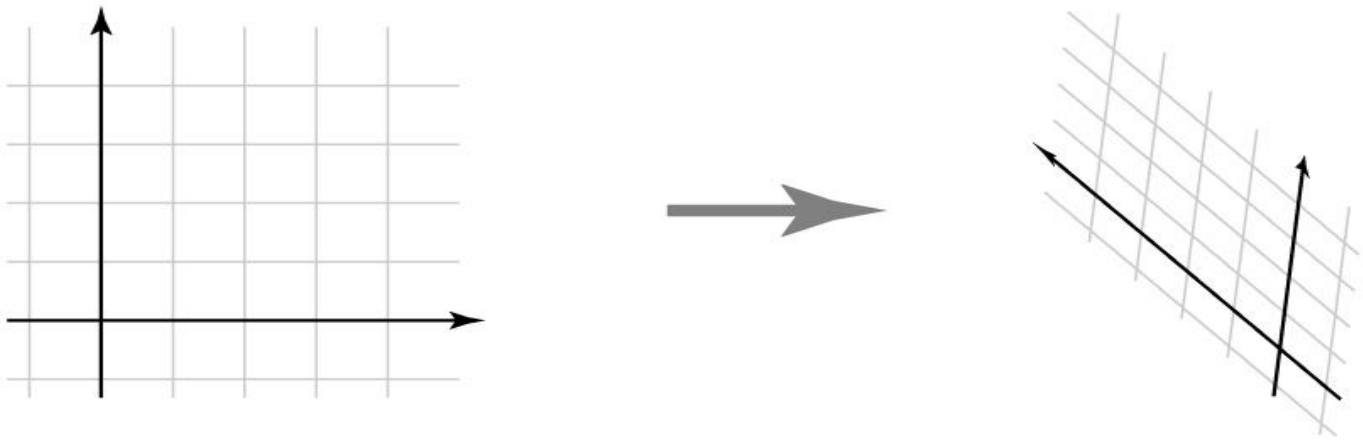
$$T(\mathbf{v}) = \mathbf{v} + \mathbf{u}$$

Affine Transformation

- Linear transformation + Translation

$$T(\mathbf{v}) = M\mathbf{v} + \mathbf{u}$$

- Preserves lines
- Preserves parallel lines
- Preserves ratios of distance along a line
- These properties are inherited from linear transformations.



Composing Transformations & Homogeneous Coordinates

Composing transformations

- Want to move an object, then move it some more
  - $\mathbf{p} \rightarrow T(\mathbf{p}) \rightarrow S(T(\mathbf{p})) = (S \circ T)(\mathbf{p})$
- We need to represent *S* o *T* ("S compose T")
  - and would like to use the same representation as for *S* and *T*

Composing transformations

- Composing linear transformations is straightforward
  - $$T(\mathbf{p}) = M_T\mathbf{p}; S(\mathbf{p}) = M_S\mathbf{p}$$
    $$(S \circ T)(\mathbf{p}) = M_S M_T \mathbf{p}$$
- Transforming first by M_T then by M_S is the same as transforming by $M_S M_T$
  - Note $M_S M_T$, or S o T, is T first, then S

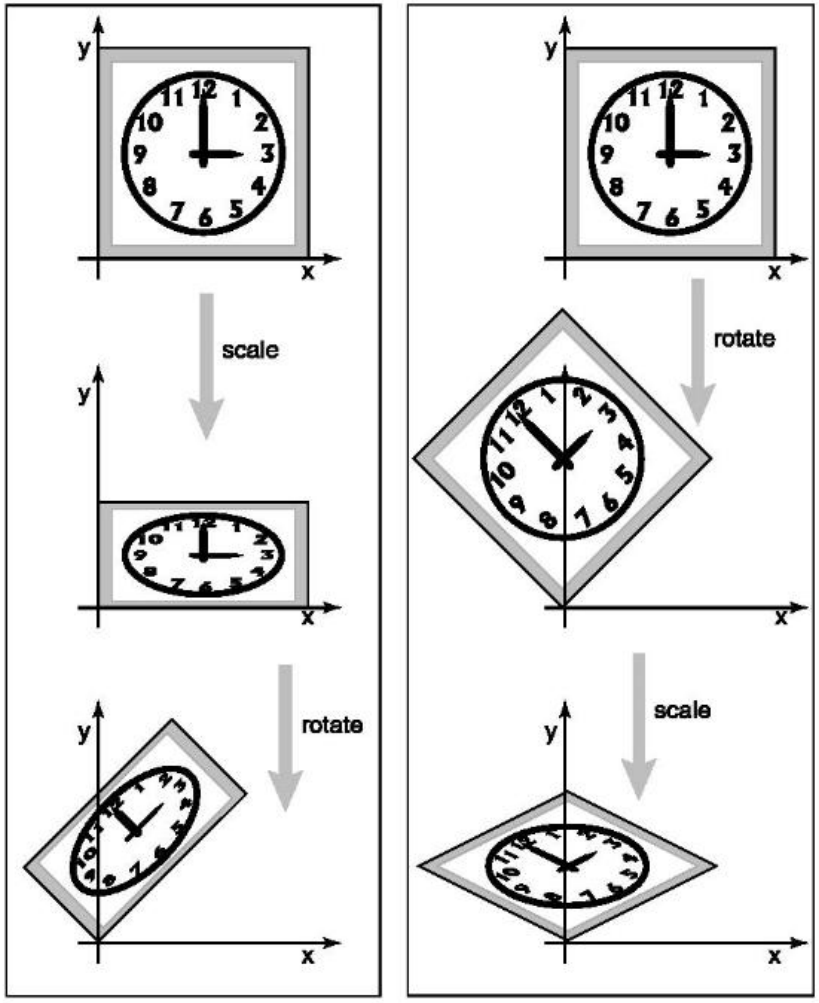
Order matters!

- Note that matrix multiplication is associative, but **not commutative**

$$(AB)C = A(BC)$$

$$AB \neq BA$$

- only sometimes commutative
  - e.g. rotations & uniform scales
  - e.g. non-uniform scales w/o rotation



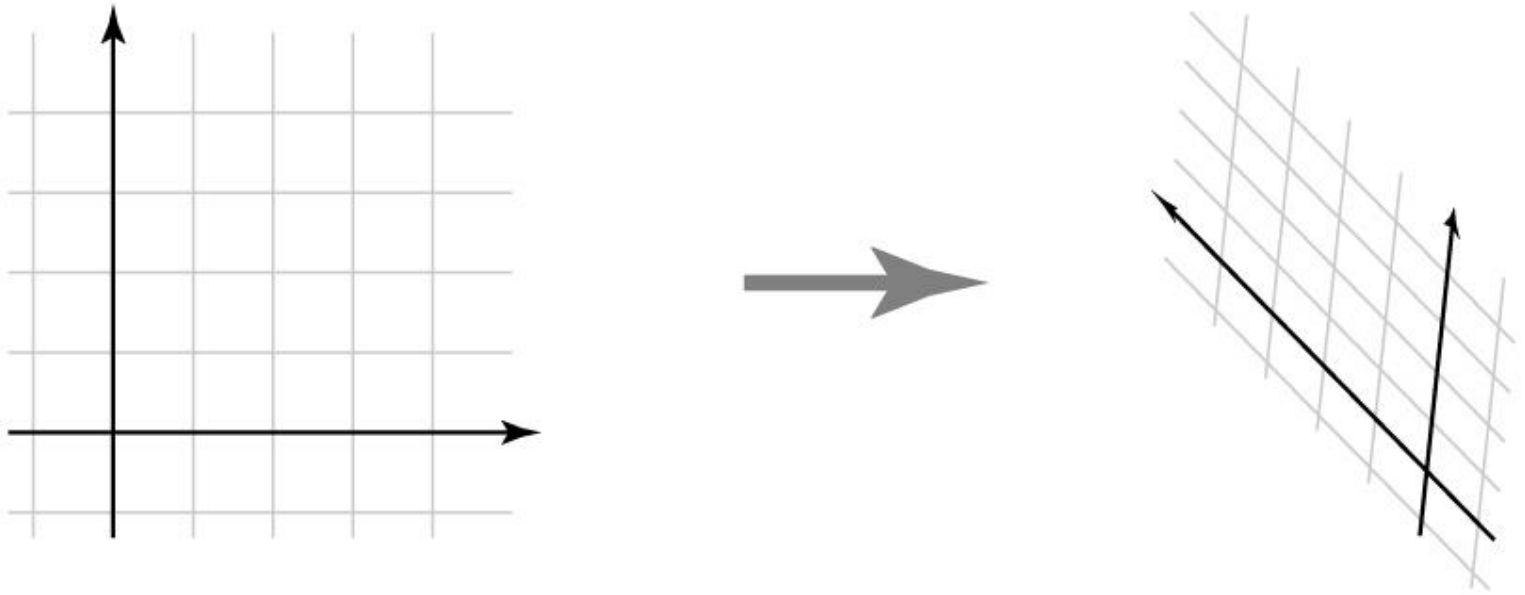
Composing translations

- Composing translations is easy
  - $\mathbf{p} \rightarrow T(\mathbf{p}) \rightarrow S(T(\mathbf{p})) = (S \circ T)(\mathbf{p})$
- Translation by $\mathbf{u}_T$ then by $\mathbf{u}_S$ is translation by $\mathbf{u}_T + \mathbf{u}_S$
- commutative!

$$T(\mathbf{p}) = \mathbf{p} + \mathbf{u}_T; S(\mathbf{p}) = \mathbf{p} + \mathbf{u}_S$$
$$(S \circ T)(\mathbf{p}) = \mathbf{p} + (\mathbf{u}_T + \mathbf{u}_S)$$

Affine transformations

- straight lines preserved; parallel lines preserved
- ratios of lengths along lines preserved (midpoints preserved)
- Origin



Combining linear with translation

- Need to use both in single framework
- Can represent arbitrary seq. as
  - $T(\mathbf{p}) = M_T\mathbf{p} + \mathbf{u}_T$
  - $S(\mathbf{p}) = M_S\mathbf{p} + \mathbf{u}_S$
  - $(S \circ T)(\mathbf{p}) = M_S(M_T\mathbf{p} + \mathbf{u}_T) + \mathbf{u}_S$

    $= (M_S M_T)\mathbf{p} + (M_S\mathbf{u}_T + \mathbf{u}_S)$
- Transforming by M_T and $\mathbf{u}_T$, then by M_S and $\mathbf{u}_S$, is the same as transforming by $M_S M_T$ and $M_s\mathbf{u}_T + \mathbf{u}_S$
- This will work but is a little awkward

Homogeneous Coordinates

- Key idea: Represent 2D points using 3 numbers
- Extra component *w* for vectors, extra row/column for matrices
  - For points, can always keep $w = 1$
  - 2D point x, y -| 3D vector [x, y, 1]^T.
- Linear transformations are represented as:

$$\begin{bmatrix} a & b & 0 \\ c & d & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} ax + by \\ cx + dy \\ 1 \end{bmatrix}$$

Homogeneous Coordinates

- Translations are represented as:

$$\begin{bmatrix} 1 & 0 & t \\ 0 & 1 & s \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} x+t \\ y+s \\ 1 \end{bmatrix}$$

- Affine transformations are represented as:

linear part

$$\begin{bmatrix} m_{11} & m_{12} & u_x \\ m_{21} & m_{22} & u_y \\ 0 & 0 & 1 \end{bmatrix}$$

translational part

Homogeneous Coordinates

- **Composing affine transformations** just works by 3x3 **matrix multiplication**

$$T(\mathbf{p}) = M_T\mathbf{p} + \mathbf{u}_T$$
$$S(\mathbf{p}) = M_S\mathbf{p} + \mathbf{u}_S$$

$$T(\mathbf{p}) = \begin{bmatrix} \overset{2x2}{M_S} & \overset{2x1}{\mathbf{u}_S} \\ 0 & 1 \end{bmatrix} \qquad S(\mathbf{p}) = \begin{bmatrix} \overset{2x2}{M_T} & \overset{2x1}{\mathbf{u}_T} \\ 0 & 1 \end{bmatrix}$$

Homogeneous Coordinates

- **Composing affine transformations** just works by 3x3 **matrix multiplication**

$$(S \circ T)(\mathbf{p}) = \begin{bmatrix} \overset{2x2}{M_S} & \overset{2x1}{\mathbf{u}_S} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \overset{2x2}{M_T} & \overset{2x1}{\mathbf{u}_T} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \overset{2x1}{\mathbf{p}} \\ 1 \end{bmatrix}$$

$$= \begin{bmatrix} (M_S M_T)\mathbf{p} + (M_S \mathbf{u}_T + \mathbf{u}_S) \\ 1 \end{bmatrix}$$

- Much cleaner

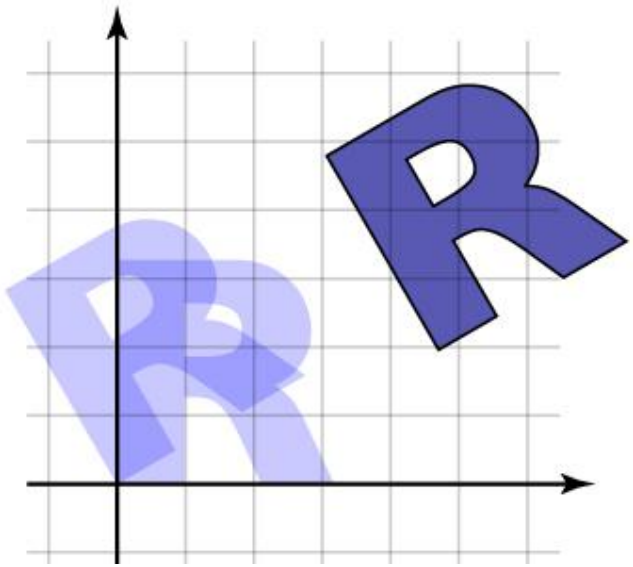
Summary: Homogeneous Coordinates in 2D

- Use $(\mathbf{x},\mathbf{y},1)^T$ instead of $(x,y)^T$ for **2D points**
- Use **3x3 matrices** instead of 2x2 matrices **for 2D linear transformations**
- Use **3x3 matrices** instead of vector additions **for 2D translations**

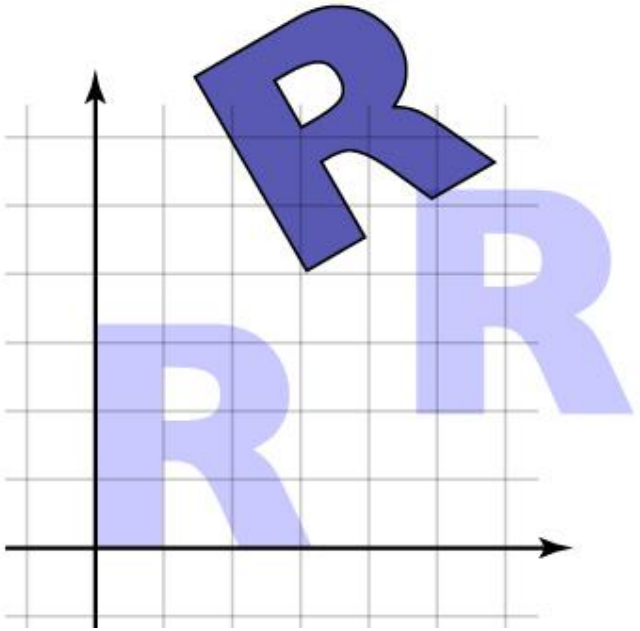
- We can treat linear transformations and translations **in a consistent manner!**

Composite affine transformations

- In general **not** commutative: order matters!



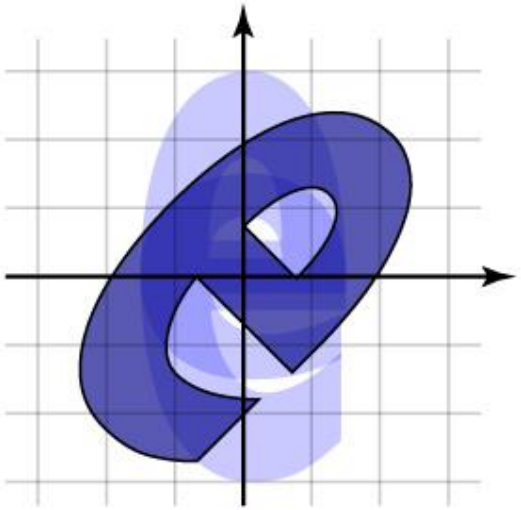
rotate, then translate



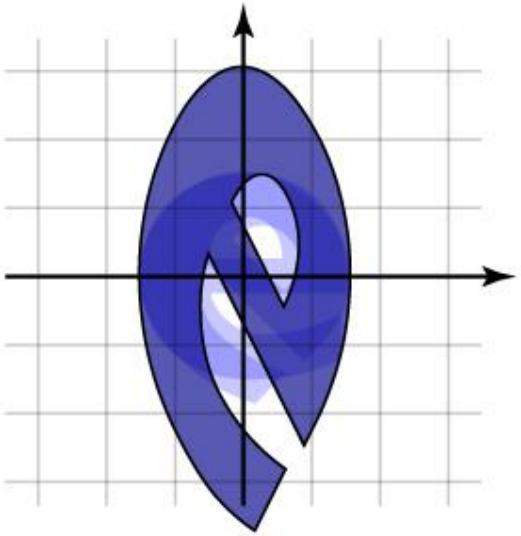
translate, then rotate

Composite affine transformations

- Another example



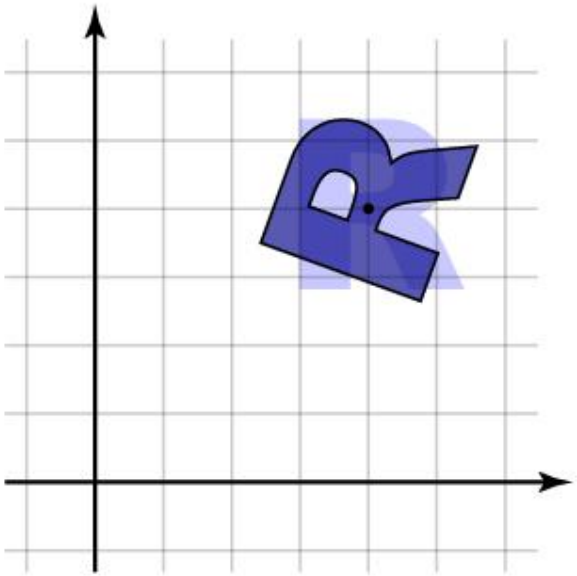
scale, then rotate



rotate, then scale

Composing to change axes

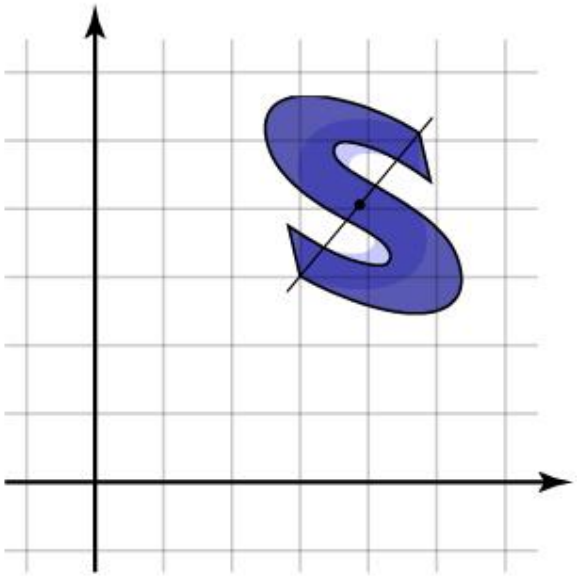
- Want to rotate about a particular point
  - could work out formulas directly…
- Know how to rotate about the origin
  - so translate that point to the origin



$$M = T^{-1}RT$$

Composing to change axes

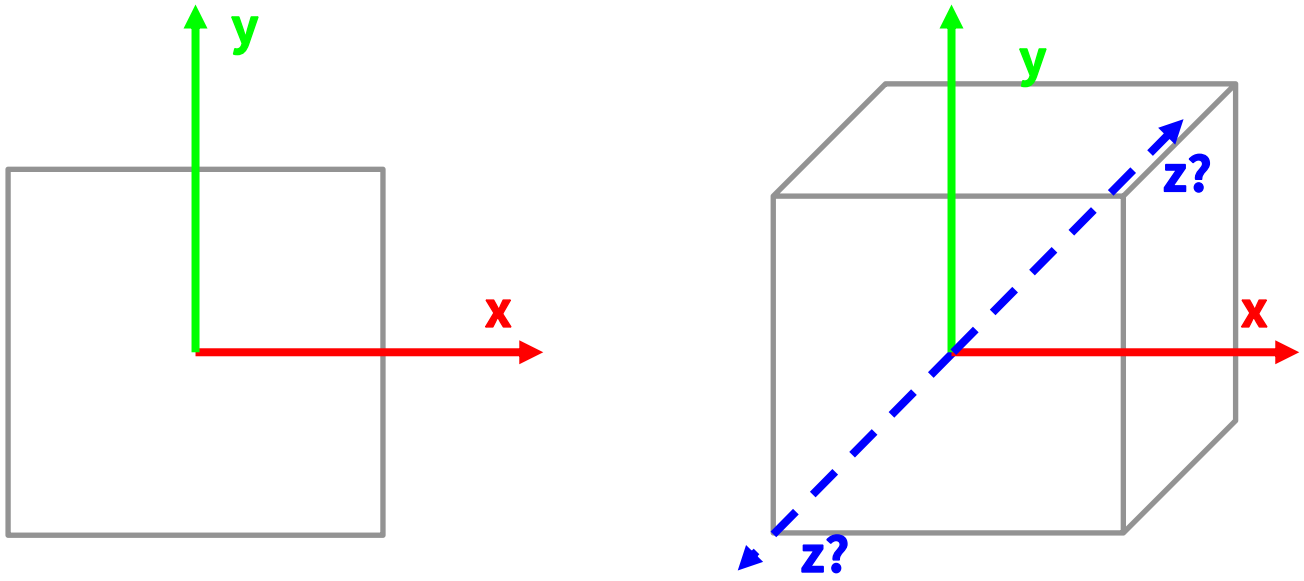
- Want to scale along a particular axis and point
- Know how to scale along the y axis at the origin
  - so translate to the origin and rotate to align axes



$$M = T^{-1}R^{-1}SRT$$

3D Affine Transformations

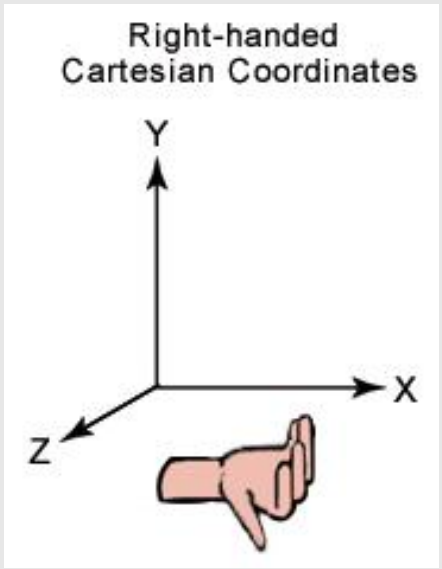
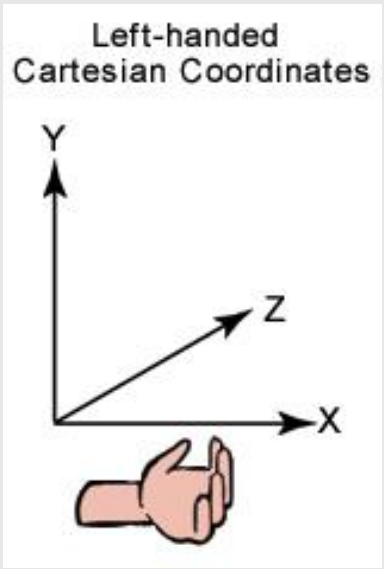
Now, let's move to the 3D world!



- Cartesian coordinate system

Two Types of 3D Cartesian Coordinate Systems

What we're using

		
Positive rotation direction	**counterclockwise** about the axis of rotation 	**clockwise** about the axis of rotation 
Used in…	**OpenGL**, Maya, Houdini, AutoCAD, … Standard for Physics & Math	DirectX, Unity, Unreal, …

Point Representation in Cartesian & Homogeneous Coordinate System

	Cartesian coordinate system	Homogeneous coordinate system
A 2**D point** is represented as…	$\begin{bmatrix} p_x \\ p_y \end{bmatrix}$	$\begin{bmatrix} p_x \\ p_y \\ \boxed{1} \end{bmatrix}$
A 3**D point** is represented as…	$\begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix}$	$\begin{bmatrix} p_x \\ p_y \\ p_z \\ \boxed{1} \end{bmatrix}$

Linear Transformation in 3D

- Linear transformation in 3D can be represented as matrix multiplication of …

3x3 matrix (in Cartesian coordinates)

$$\begin{bmatrix} m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ m_{31} & m_{32} & m_{33} \end{bmatrix} \begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix}$$

or

4x4 matrix (in homogeneous coordinates)

$$\begin{bmatrix} m_{11} & m_{12} & m_{13} & 0 \\ m_{21} & m_{22} & m_{23} & 0 \\ m_{31} & m_{32} & m_{33} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} p_x \\ p_y \\ p_z \\ 1 \end{bmatrix}$$

Linear Transformation in 3D

Scale:

3D

$$\mathbf{S_s} = \begin{bmatrix} \mathbf{S}_x & 0 & 0 \\ 0 & \mathbf{S}_y & 0 \\ 0 & 0 & \mathbf{S}_z \end{bmatrix}$$

3D-H

$$\mathbf{S_s} = \begin{bmatrix} \mathbf{S}_x & 0 & 0 & 0 \\ 0 & \mathbf{S}_y & 0 & 0 \\ 0 & 0 & \mathbf{S}_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Shear (in x, based on y,z position):

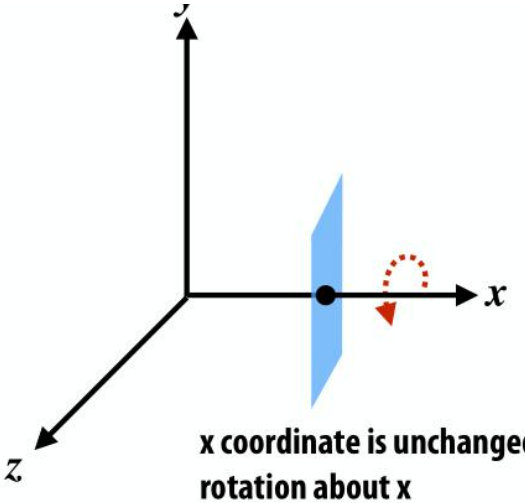
$$\mathbf{H}_{x,\mathbf{d}} = \begin{bmatrix} 1 & \mathbf{d}_y & \mathbf{d}_z \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\mathbf{H}_{x,\mathbf{d}} = \begin{bmatrix} 1 & \mathbf{d}_y & \mathbf{d}_z & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Linear Transformation in 3D

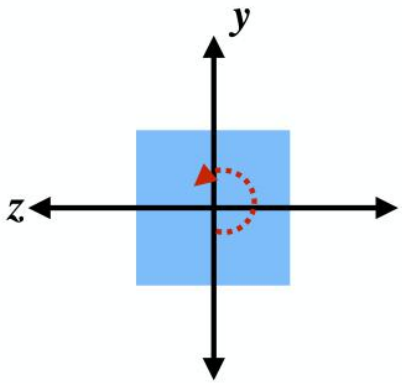
Rotation about x axis:

$$\mathbf{R}_{x,\theta} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{bmatrix}$$



x coordinate is unchanged by rotation about x

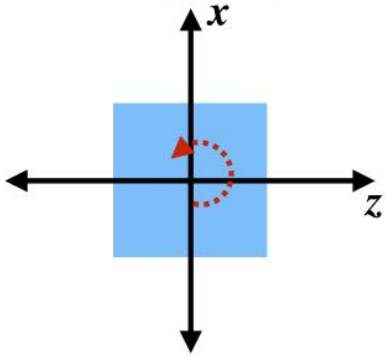
View looking down -x axis:



Rotation about y axis:

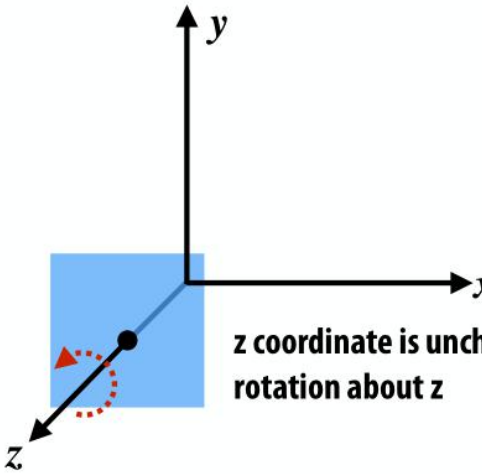
$$\mathbf{R}_{y,\theta} = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix}$$

View looking down -y axis:



Rotation about z axis:

$$\mathbf{R}_{z,\theta} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$



z coordinate is unchanged by rotation about z

Review of Translation in 2D

- Translation in **2D** can be represented as ...

Vector addition (in Cartesian coordinates)

$$T(\mathbf{p}) = \begin{bmatrix} p_x \\ p_y \end{bmatrix} + \begin{bmatrix} u_x \\ u_y \end{bmatrix}$$

Matrix multiplication of **3x3 matrix** (in homogeneous coordinates)

$$\begin{bmatrix} 1 & 0 & u_x \\ 0 & 1 & u_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} p_x \\ p_y \\ 1 \end{bmatrix}$$

Translation in 3D

- Translation in **3D** can be represented as …

Vector addition
(in Cartesian coordinates)

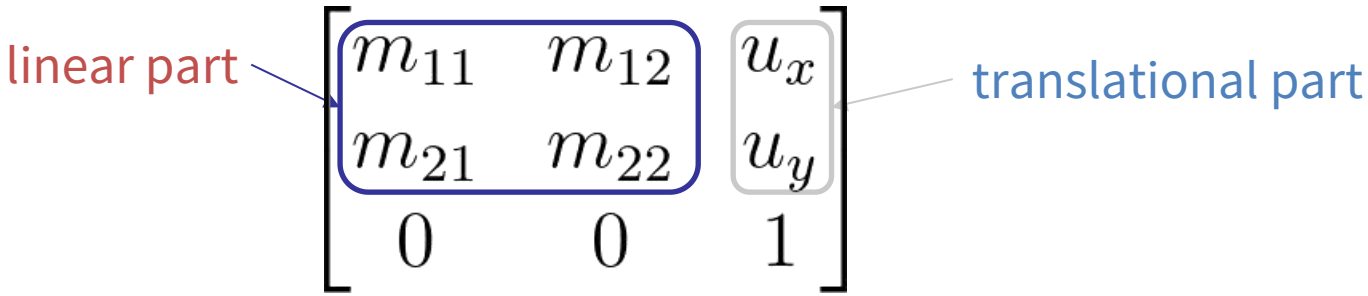
$$T(\mathbf{p}) = \begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix} + \begin{bmatrix} u_x \\ u_y \\ u_z \end{bmatrix}$$

Matrix multiplication of
4x4 matrix
(in homogeneous coordinates)

$$\begin{bmatrix} 1 & 0 & 0 & u_x \\ 0 & 1 & 0 & u_y \\ 0 & 0 & 1 & u_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} p_x \\ p_y \\ p_z \\ 1 \end{bmatrix}$$

Review of Affine Transformation in 2D

- In homogeneous coordinates, 2D affine transformation can be represented as multiplication of 3x3 **matrix**:



Affine Transformation in 3D

- In homogeneous coordinates, 3D affine transformation can be represented as multiplication of **4x4 matrix**: