2D Geometric Transformations

CS 4620 Lecture 4

A little quick math background

- Notation for sets, functions, mappings
- Linear transformations
- Matrices
  - Matrix-vector multiplication
  - Matrix-matrix multiplication
- Geometry of curves in 2D
  - Implicit representation
  - Explicit representation

Implicit representations

- Equation to tell whether we are on the curve

  $$\{\mathbf{v} \,|\, f(\mathbf{v}) = 0\}$$

- Example: line (orthogonal to **u**, distance *k* from **0**)

  $$\{\mathbf{v} \,|\, \mathbf{v} \cdot \mathbf{u} + k = 0\}$$ (**u** is a unit vector)

- Example: circle (center **p**, radius *r*)

  $$\{\mathbf{v} \,|\, (\mathbf{v} - \mathbf{p}) \cdot (\mathbf{v} - \mathbf{p}) - r^2 = 0\}$$

- Always define boundary of region
  - (if *f* is continuous)

Explicit representations

- Also called parametric
- Equation to map domain into plane

$$\{f(t) \mid t \in D\}$$

- Example: line (containing **p**, parallel to **u**)

$$\{\mathbf{p} + t\mathbf{u} \mid t \in \mathbb{R}\}$$

- Example: circle (center **b**, radius *r*)

$$\{\mathbf{p} + r[\cos t \;\; \sin t]^T \mid t \in [0, 2\pi)\}$$

- Like tracing out the path of a particle over time
- Variable *t* is the “parameter”

Transforming geometry

- Move a subset of the plane using a mapping from the plane to itself

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

- Parametric representation:

$$\{f(t) \mid t \in D\} \rightarrow \{T(f(t)) \mid t \in D\}$$

- Implicit representation:

$$\{\mathbf{v} \mid f(\mathbf{v}) = 0\} \rightarrow \{T(\mathbf{v}) \mid f(\mathbf{v}) = 0\}$$
$$= \{\mathbf{v} \mid f(T^{-1}(\mathbf{v})) = 0\}$$

Translation

- Simplest transformation: $T(\mathbf{v}) = \mathbf{v} + \mathbf{u}$
- Inverse: $T^{-1}(\mathbf{v}) = \mathbf{v} - \mathbf{u}$
- Example of transforming circle

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)

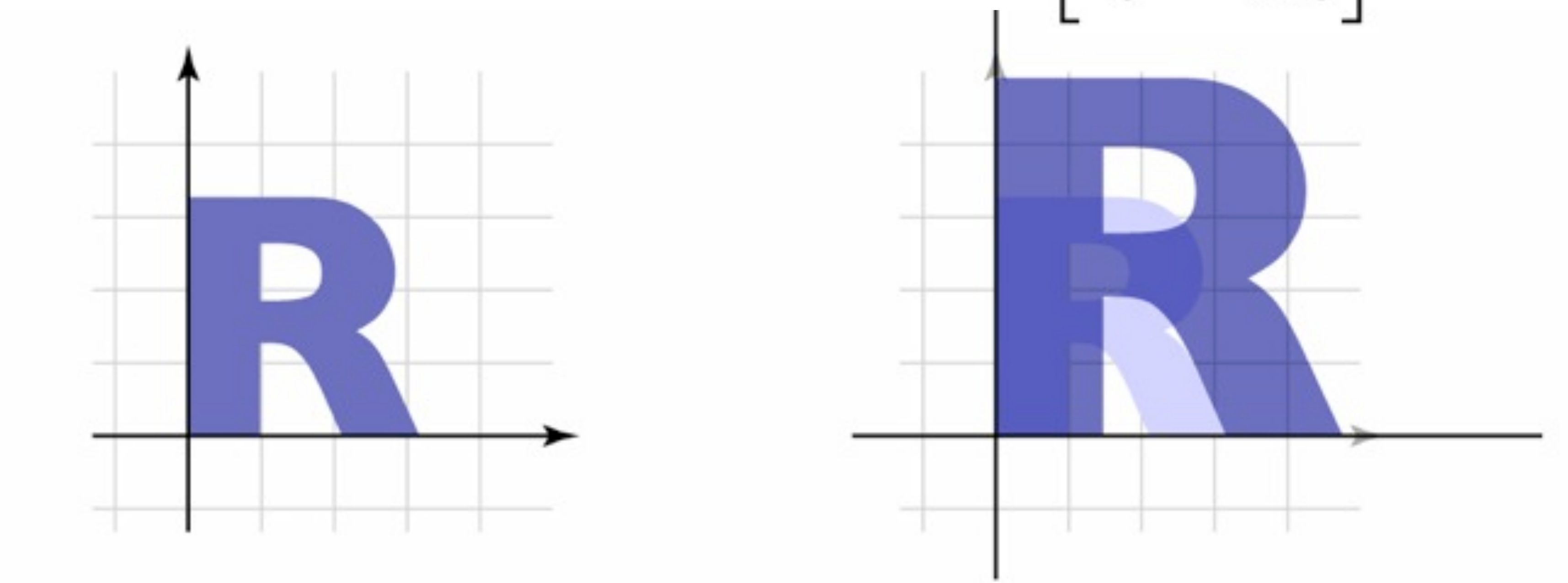
Geometry of 2D linear trans.

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

Linear transformation gallery

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

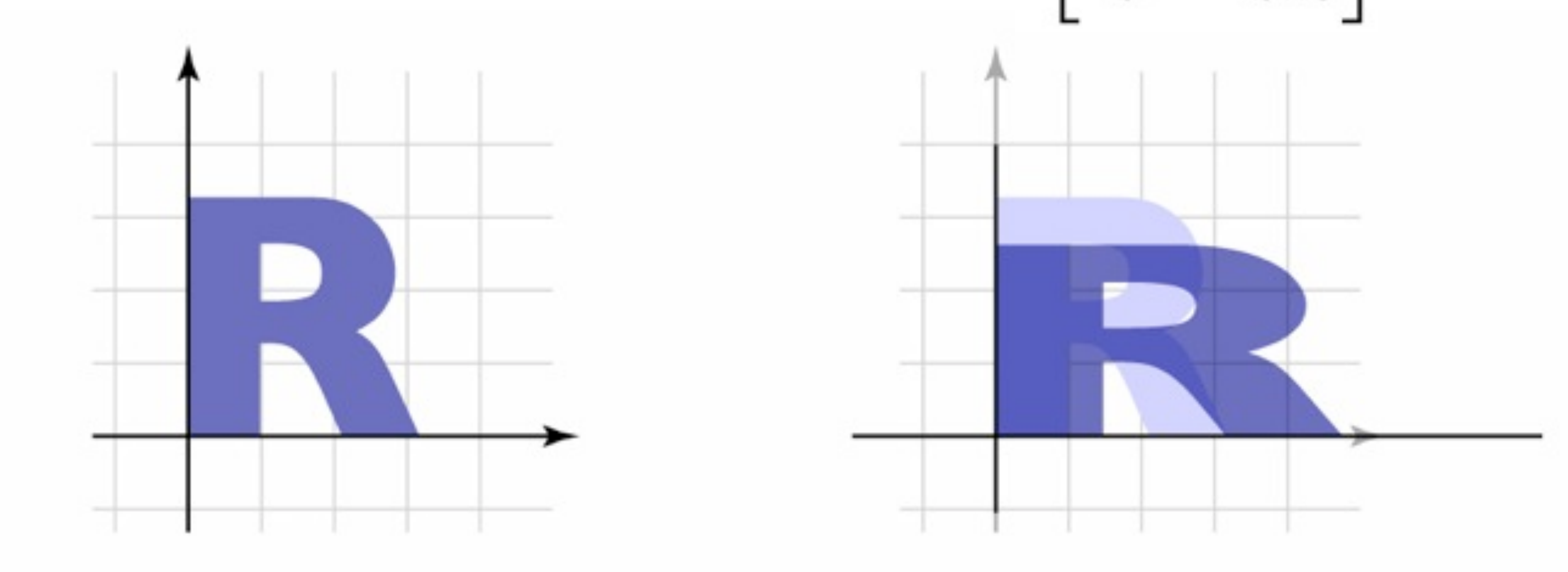
$$\begin{bmatrix} 1.5 & 0 \\ 0 & 1.5 \end{bmatrix}$$



Linear transformation gallery

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

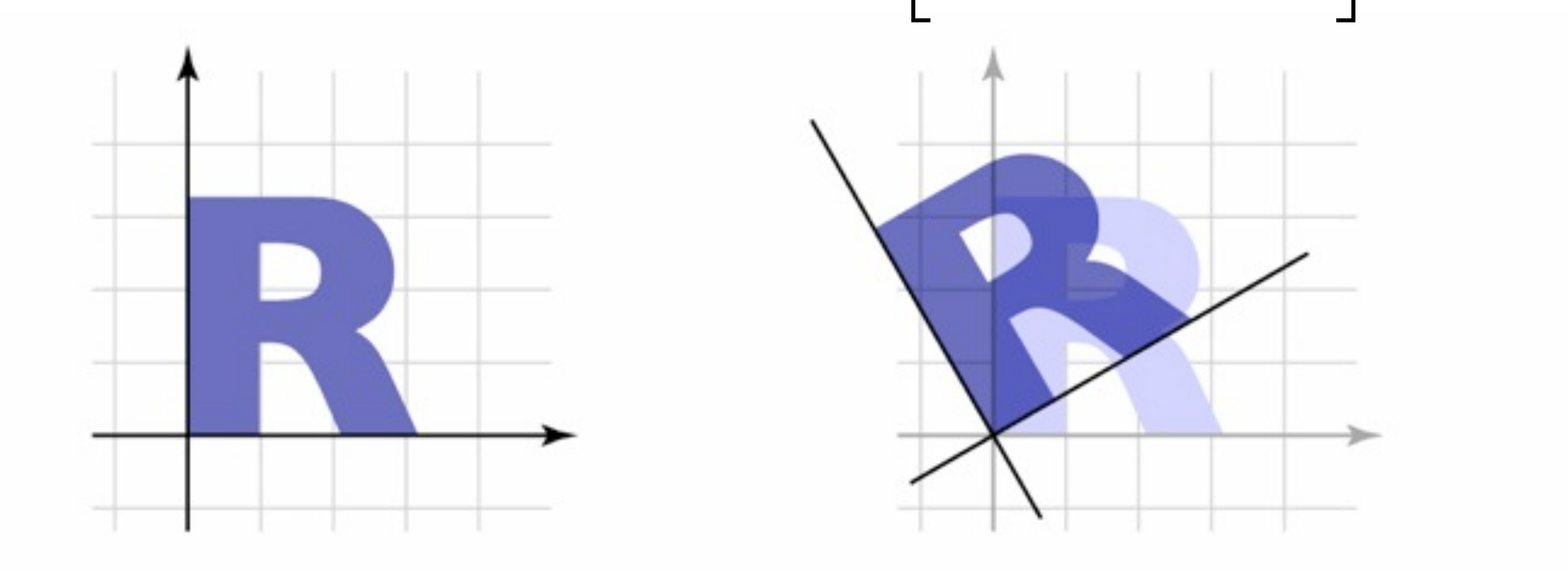
$$\begin{bmatrix} 1.5 & 0 \\ 0 & 0.8 \end{bmatrix}$$



Linear transformation gallery

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

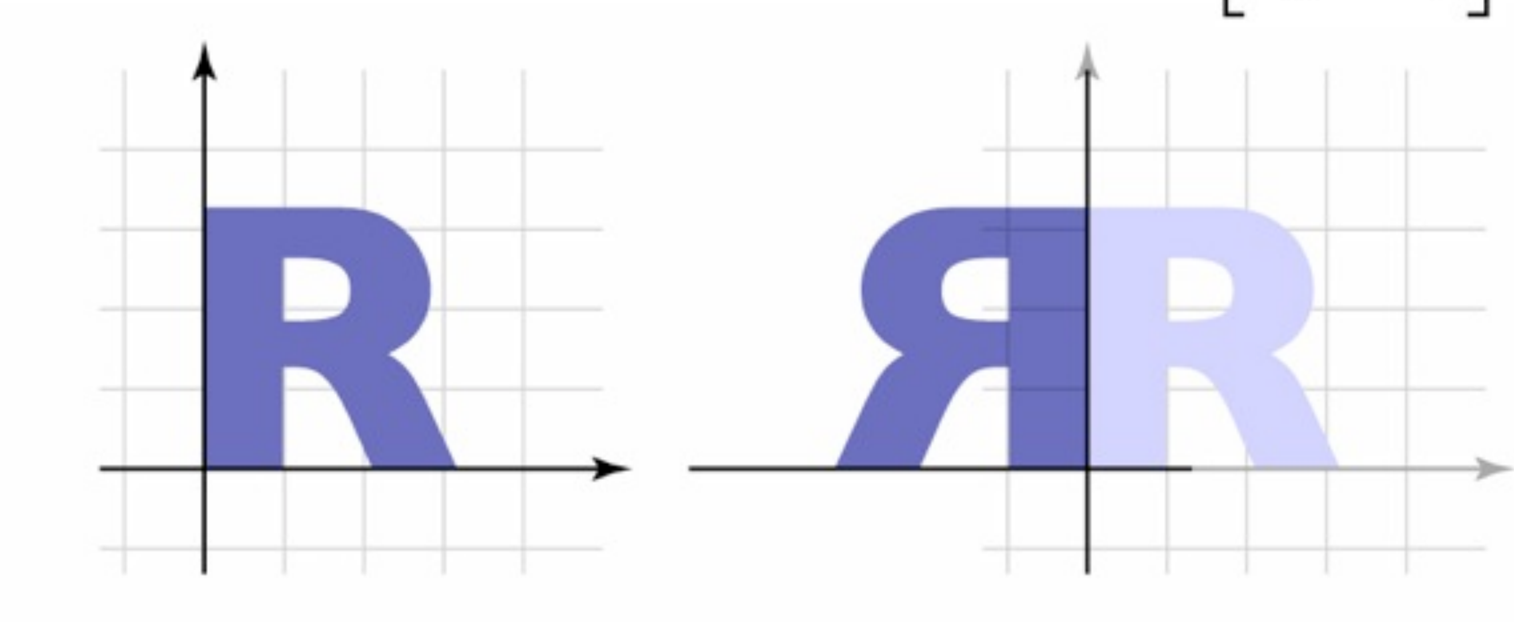
$$\begin{bmatrix}0.866 & -0.5 \\ 0.5 & 0.866\end{bmatrix}$$



Linear transformation gallery

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

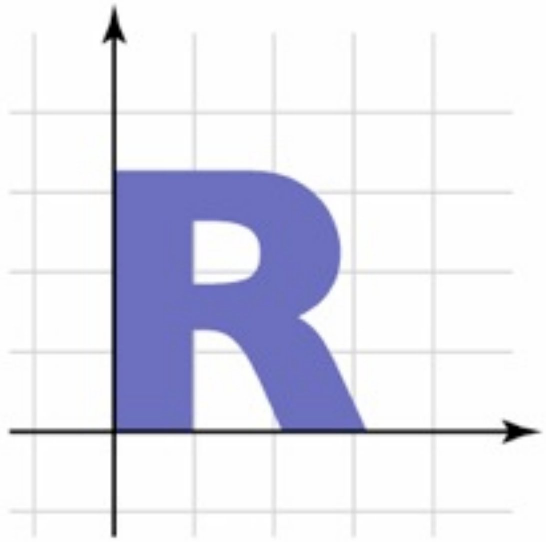
$$\begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix}$$

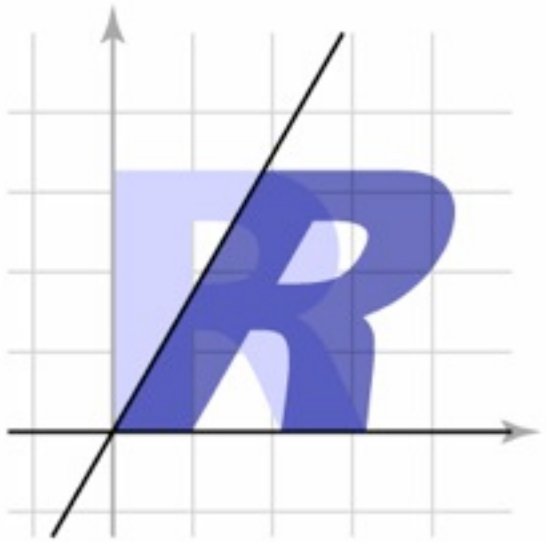


Linear transformation gallery

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

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





Composing transformations

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

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

- Translation by $\mathbf{u}_T$ then by $\mathbf{u}_S$ is translation by $\mathbf{u}_T + \mathbf{u}_S$
  - commutative!

Composing transformations

- Linear transformations also 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 $M_S M_T$
  - only sometimes commutative
    - e.g. rotations & uniform scales
    - e.g. non-uniform scales w/o rotation
  - Note $M_S M_T$, or $S \circ T$, is T first, then S

Combining linear with translation

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

    $= (M_S M_T)\mathbf{p} + (M_S\mathbf{u}_T + \mathbf{u}_S)$
  - e. g. $S(T(0)) = S(\mathbf{u}_T)$
- 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 $\mathbf{u}_S + M_S\mathbf{u}_T$
  - This will work but is a little awkward

Homogeneous coordinates

- A trick for representing the foregoing more elegantly
- Extra component *w* for vectors, extra row/column for matrices
  - for affine, can always keep $w = 1$
- Represent linear transformations with dummy extra row and column

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

Homogeneous coordinates

- Represent translation using the extra column

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

Homogeneous coordinates

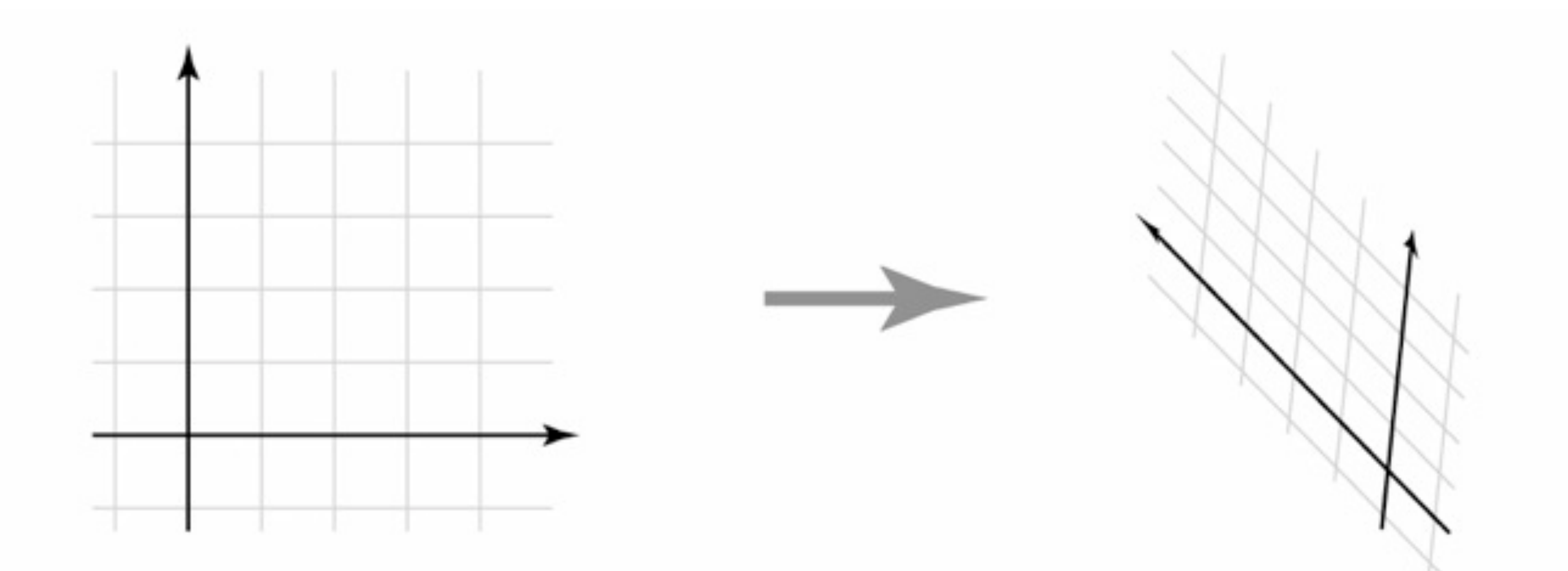
- Composition just works, by 3x3 matrix multiplication

$$\begin{bmatrix} M_S & \mathbf{u}_S \\ 0 & 1 \end{bmatrix} \begin{bmatrix} M_T & \mathbf{u}_T \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \mathbf{p} \\ 1 \end{bmatrix}$$
$$= \begin{bmatrix} (M_S M_T)\mathbf{p} + (M_S \mathbf{u}_T + \mathbf{u}_S) \\ 1 \end{bmatrix}$$

- This is exactly the same as carrying around *M* and **u**
  – but cleaner
  – and generalizes in useful ways as we'll see later

Affine transformations

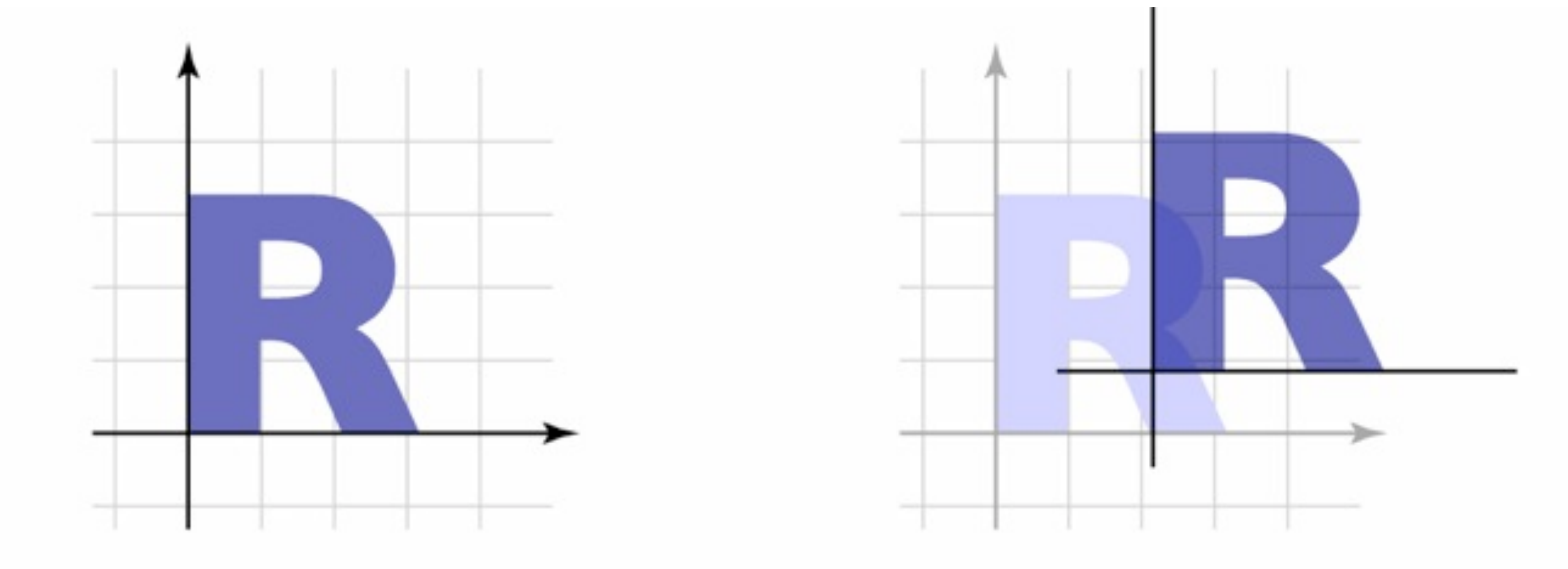
- The set of transformations we have been looking at is known as the "affine" transformations
  - straight lines preserved; parallel lines preserved
  - ratios of lengths along lines preserved (midpoints preserved)



Affine transformation gallery

- Translation

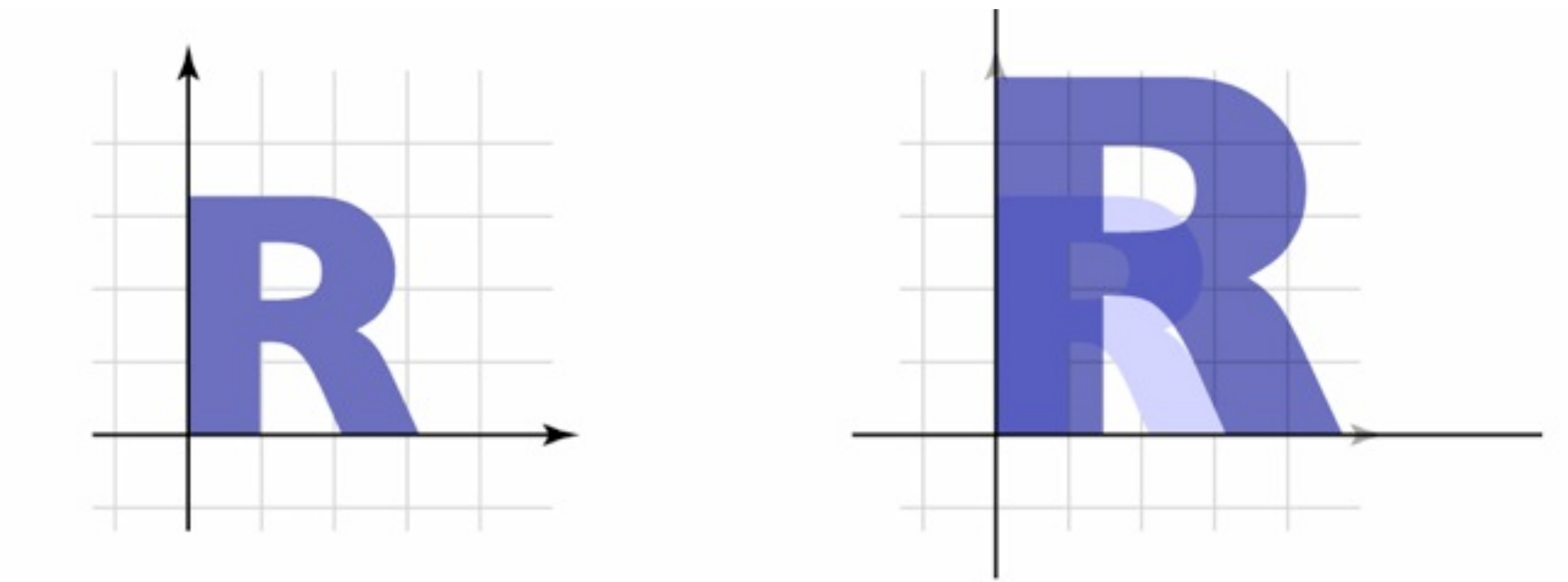
$$\begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \quad \begin{bmatrix} 1 & 0 & 2.15 \\ 0 & 1 & 0.85 \\ 0 & 0 & 1 \end{bmatrix}$$



Affine transformation gallery

- Uniform scale

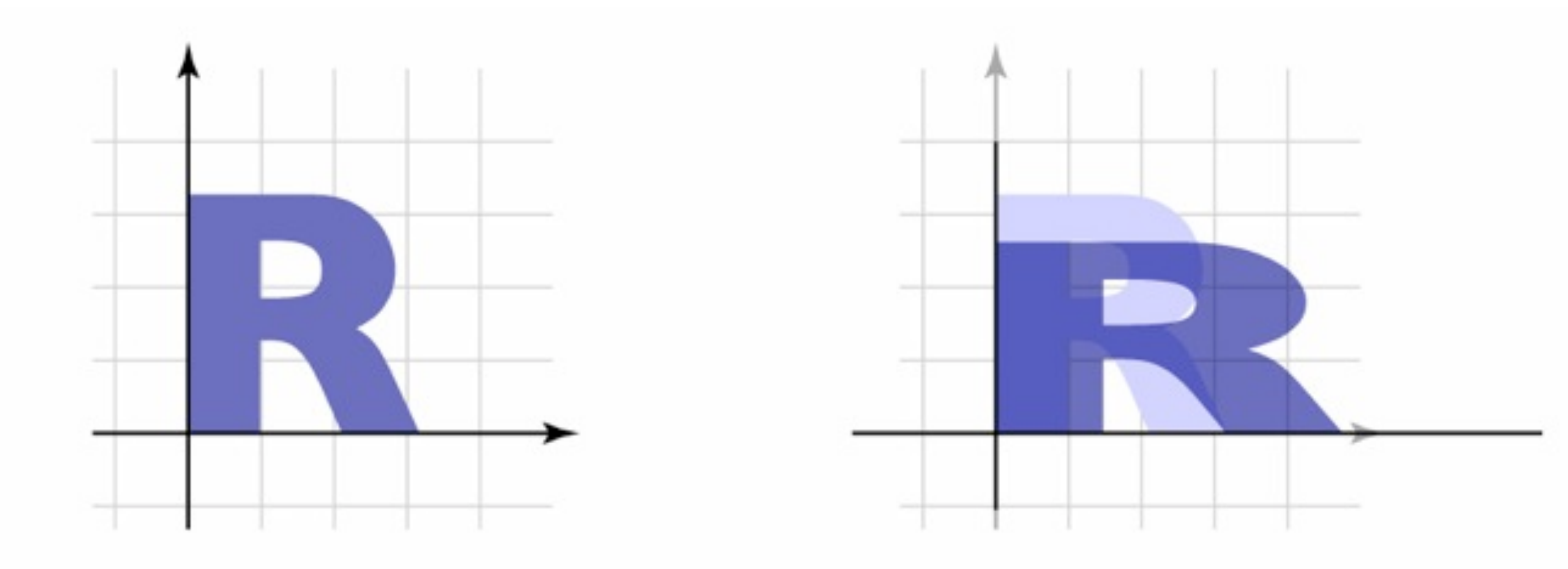
$$\begin{bmatrix} s & 0 & 0 \\ 0 & s & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad \begin{bmatrix} 1.5 & 0 & 0 \\ 0 & 1.5 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$



Affine transformation gallery

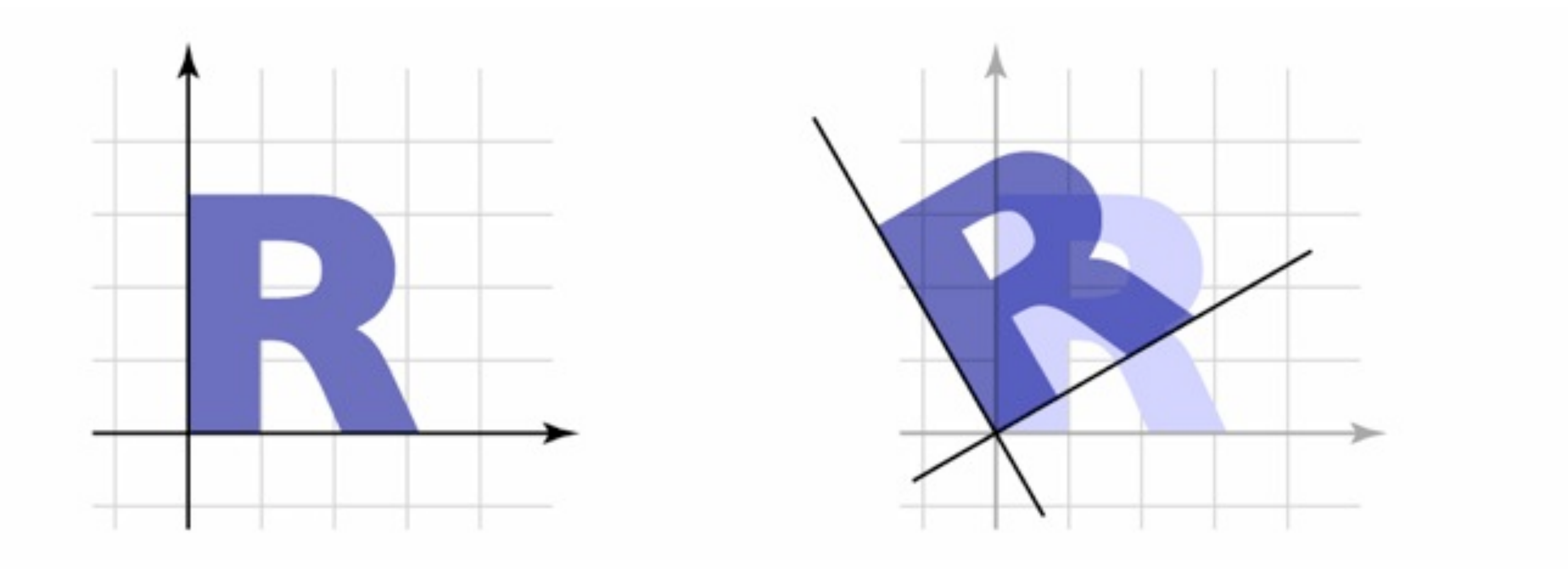
- Nonuniform scale

$$\begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad \begin{bmatrix} 1.5 & 0 & 0 \\ 0 & 0.8 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$



Affine transformation gallery

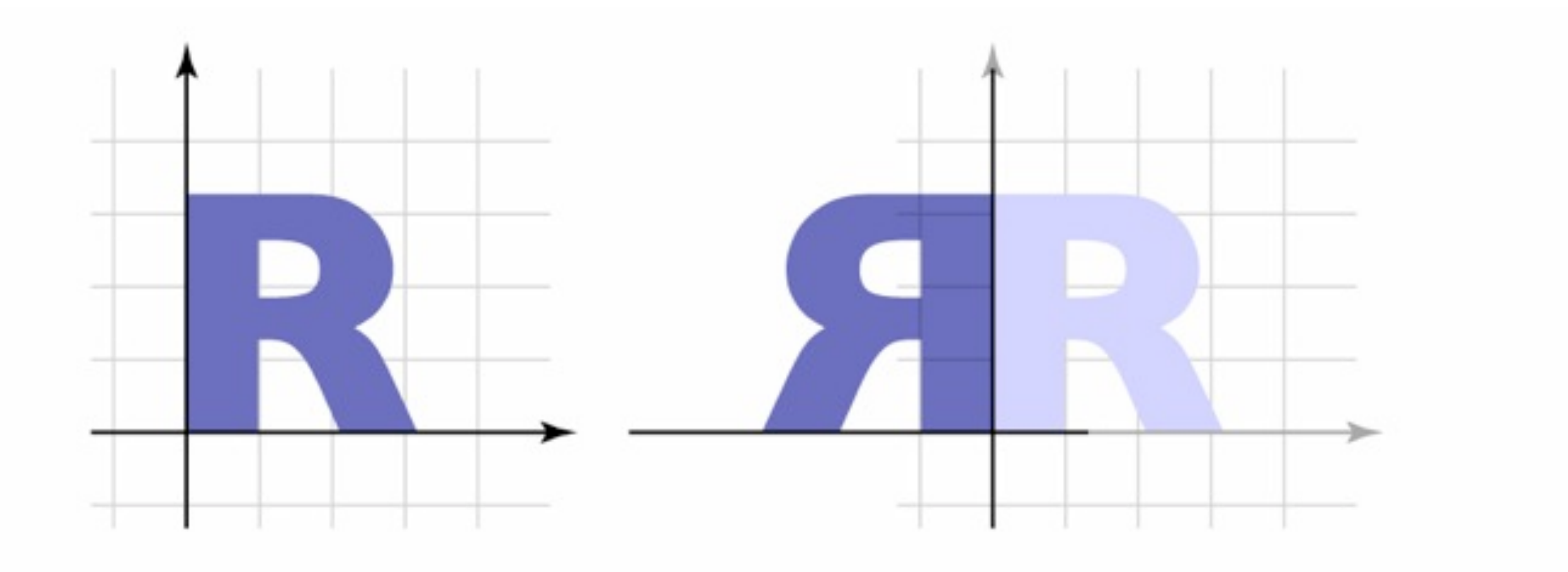
- Rotation $\begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$ $\begin{bmatrix} 0.866 & -0.5 & 0 \\ 0.5 & 0.866 & 0 \\ 0 & 0 & 1 \end{bmatrix}$



Affine transformation gallery

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

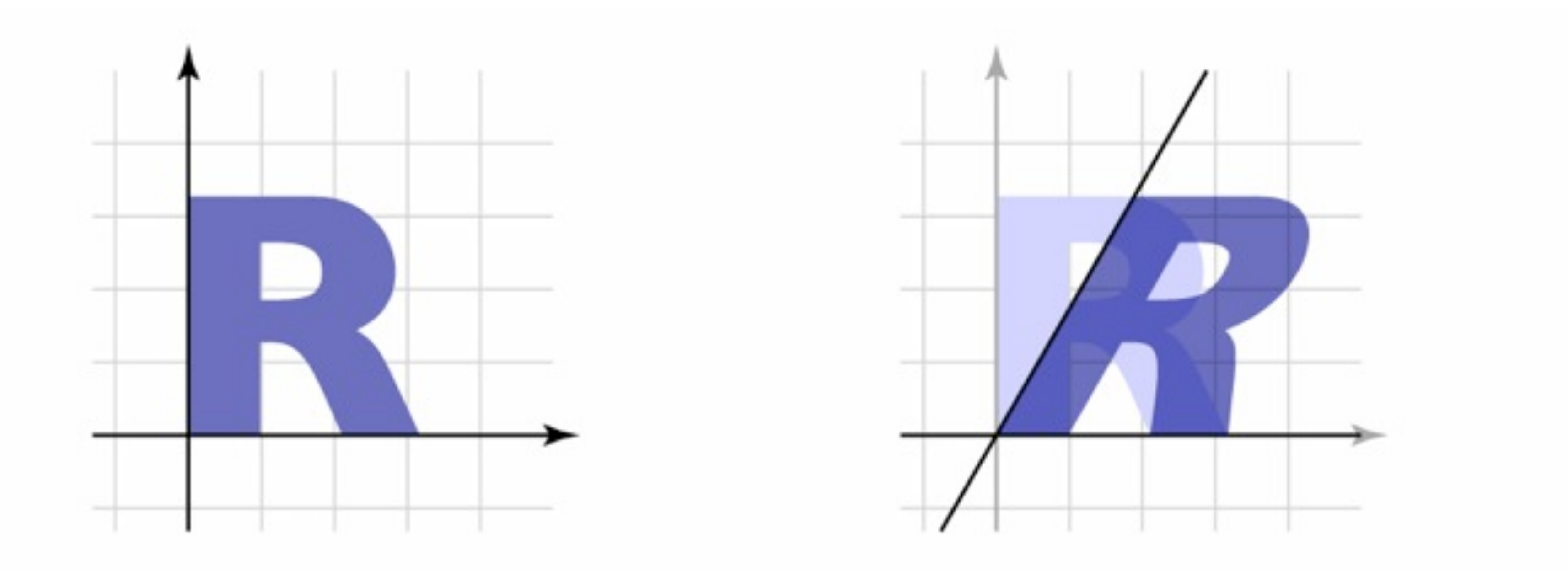
$$\begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$



Affine transformation gallery

- Shear

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

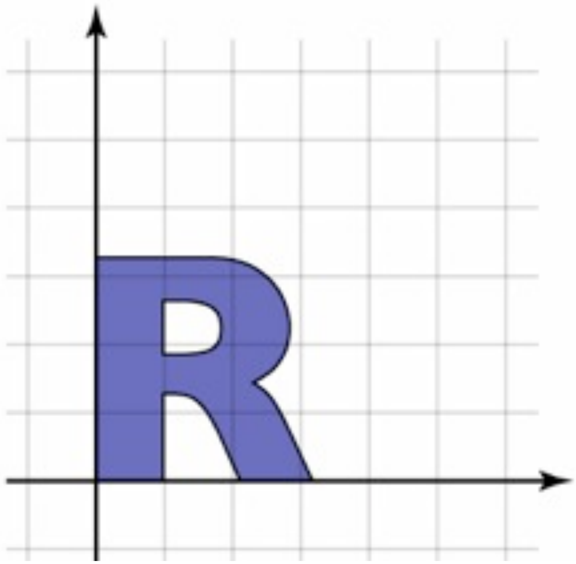


General affine transformations

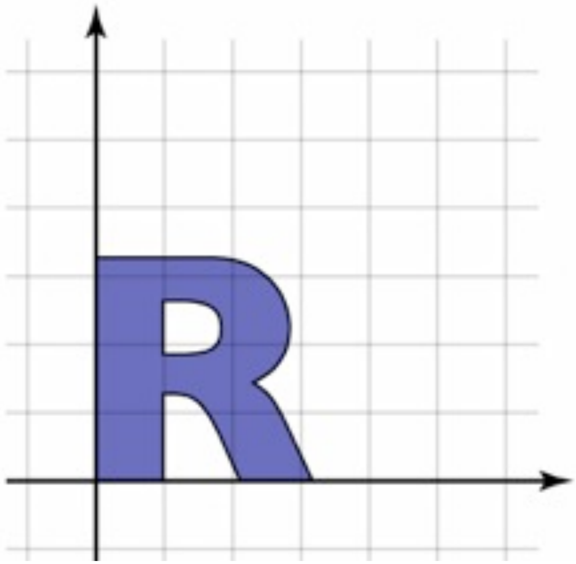
- The previous slides showed "canonical" examples of the types of affine transformations
- Generally, transformations contain elements of multiple types
  - often define them as products of canonical transforms
  - sometimes work with their properties more directly

Composite affine transformations

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



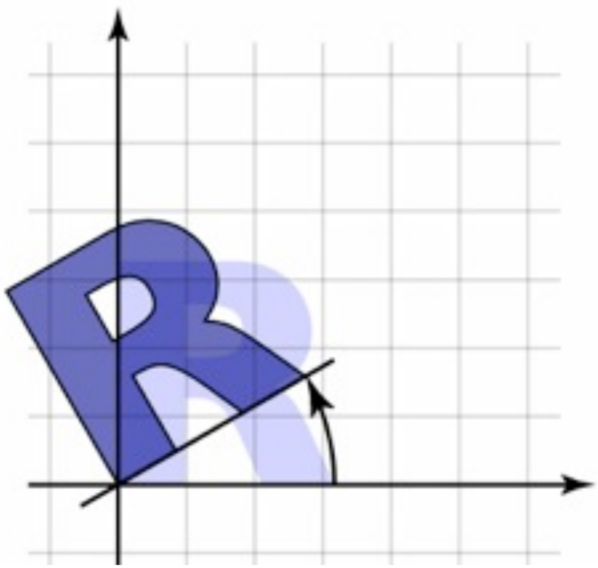
rotate, then translate



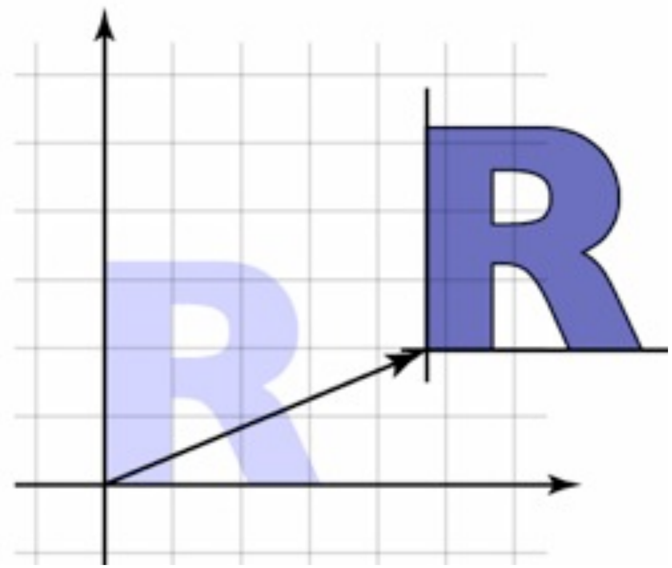
translate, then rotate

Composite affine transformations

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



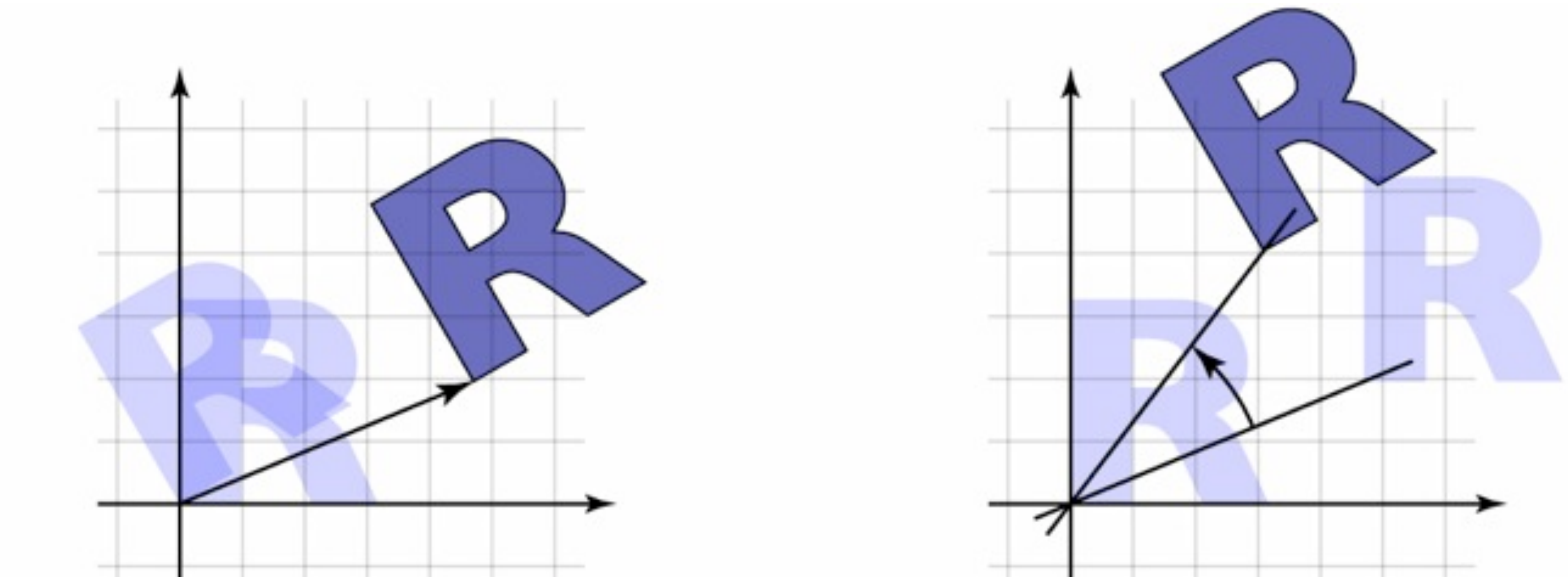
rotate, then translate



translate, then rotate

Composite affine transformations

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

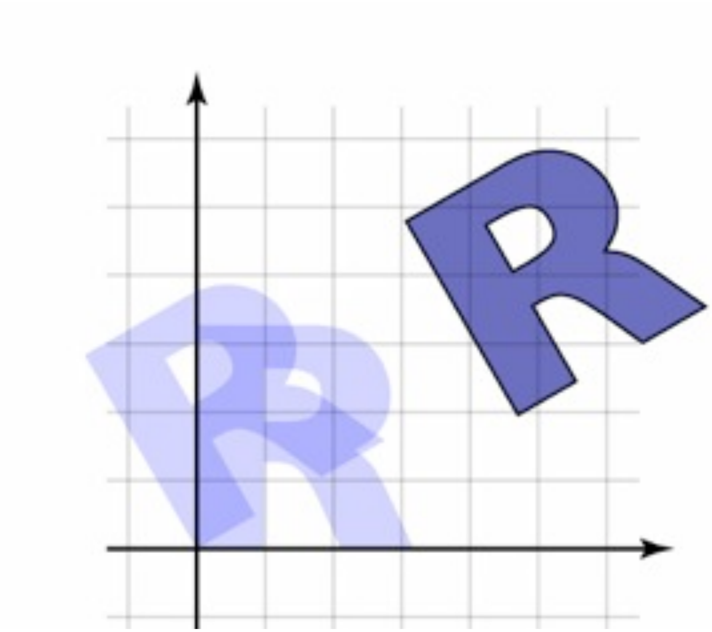


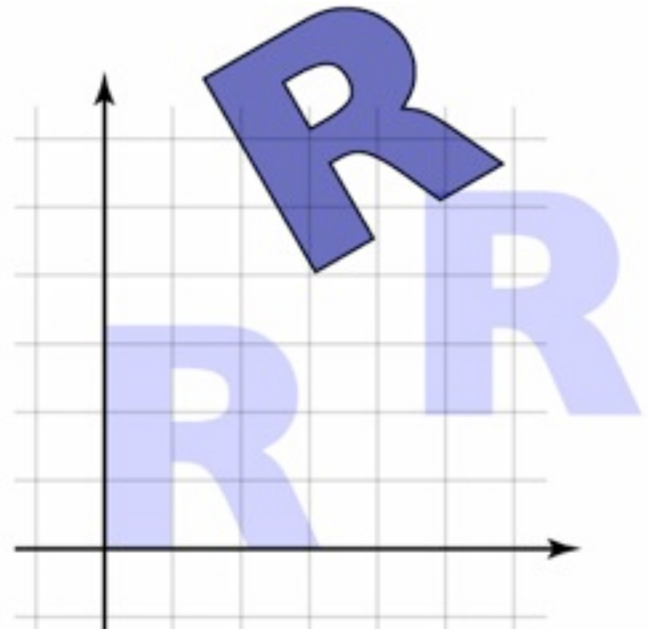
rotate, then translate

translate, then rotate

Composite affine transformations

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

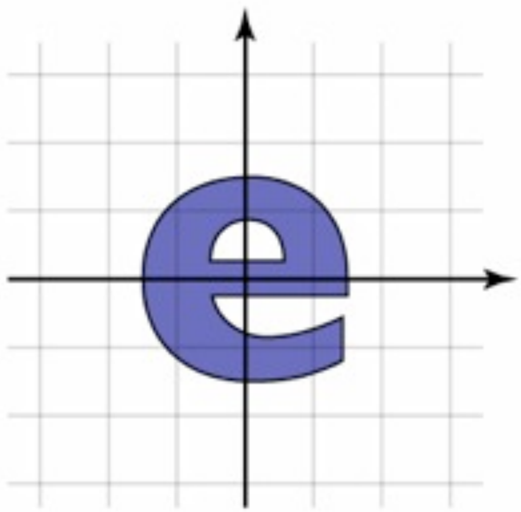


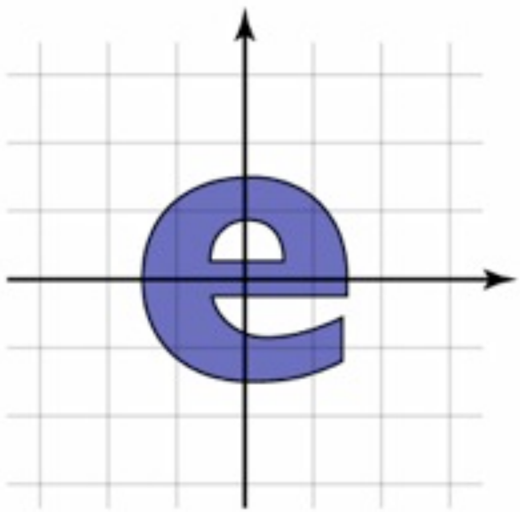


rotate, then translate translate, then rotate

Composite affine transformations

- Another example



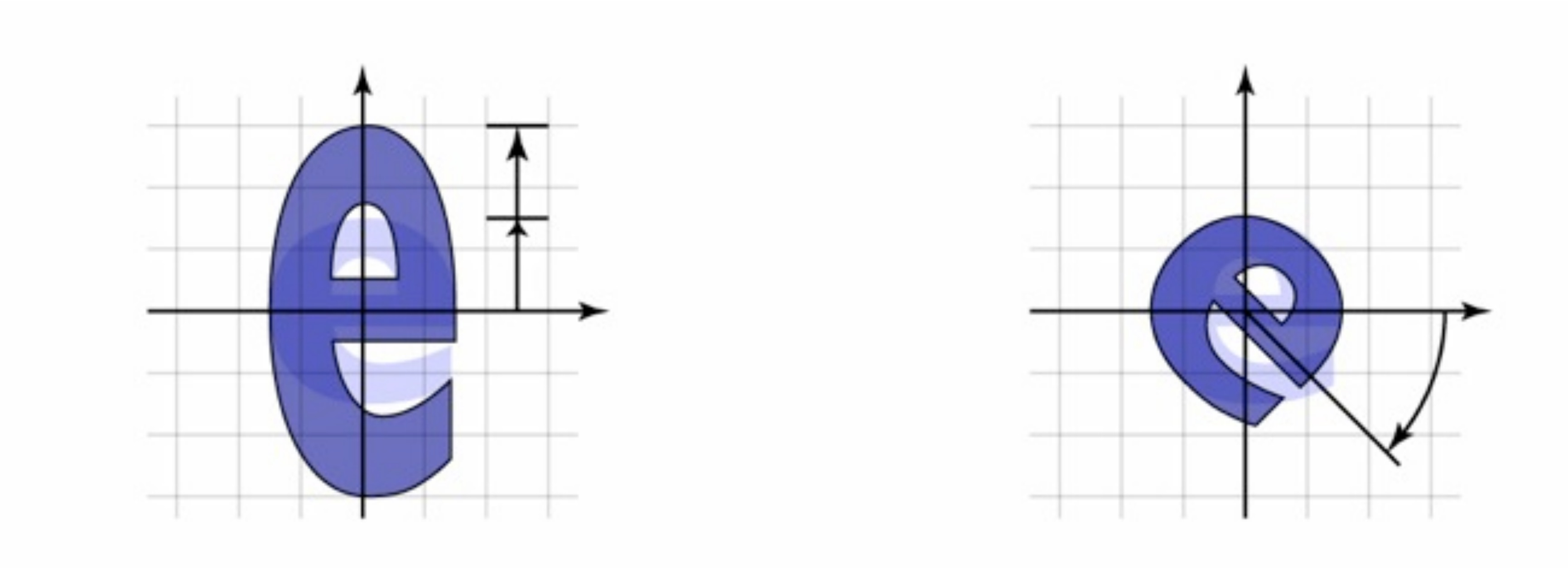


scale, then rotate

rotate, then scale

Composite affine transformations

- Another example

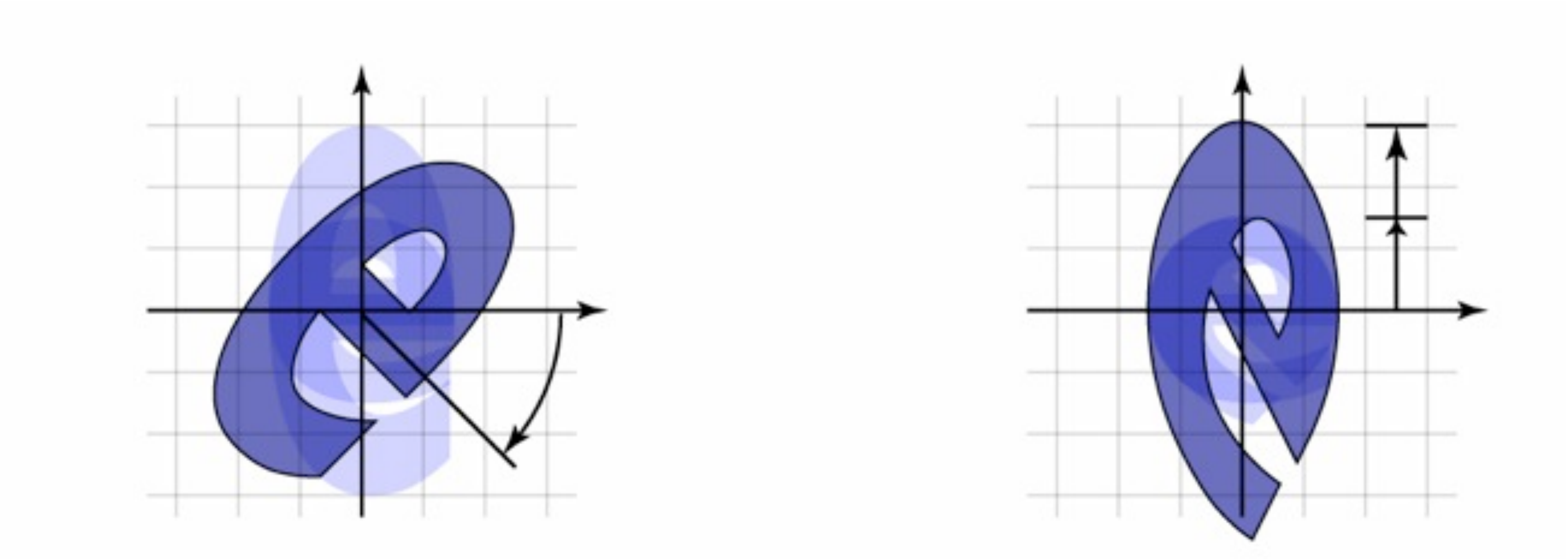


scale, then rotate

rotate, then scale

Composite affine transformations

- Another example

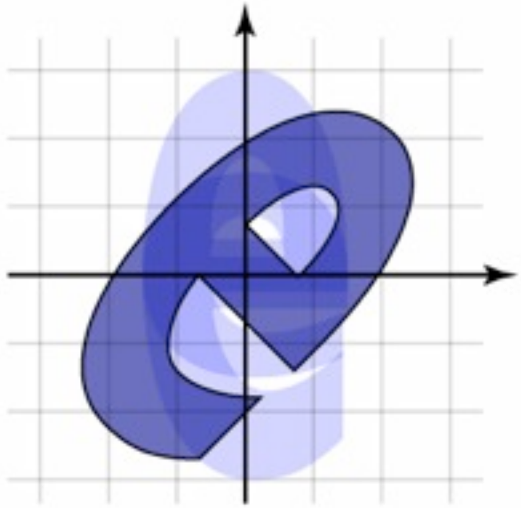


scale, then rotate

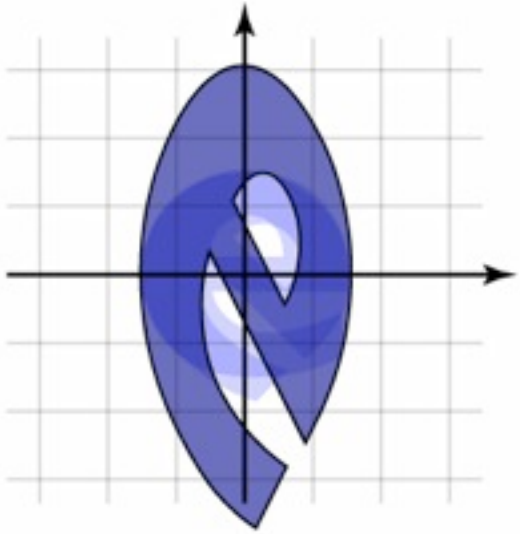
rotate, then scale

Composite affine transformations

- Another example



scale, then rotate



rotate, then scale

Rigid motions

- A transform made up of only translation and rotation is a *rigid motion* or a *rigid body transformation*
- The linear part is an orthonormal matrix

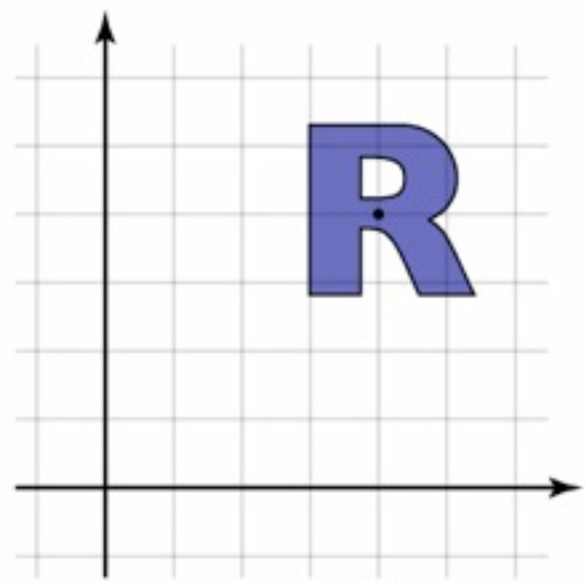
$$R = \begin{bmatrix} Q & \mathbf{u} \\ 0 & 1 \end{bmatrix}$$

- Inverse of orthonormal matrix is transpose
  – so inverse of rigid motion is easy:

$$R^{-1}R = \begin{bmatrix} Q^T & -Q^T\mathbf{u} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} Q & \mathbf{u} \\ 0 & 1 \end{bmatrix}$$

Composing to change axes

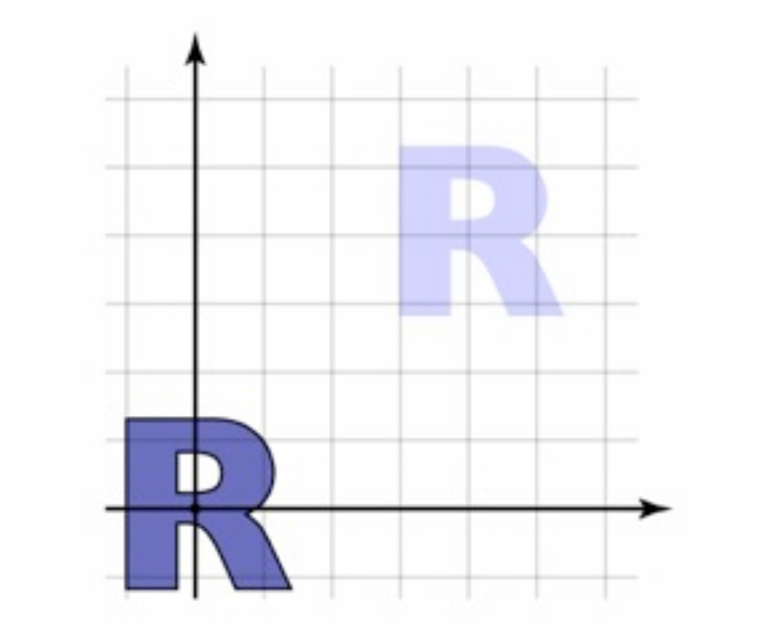
- Want to rotate about a particular point
  - could work out formulas directly…
- Know how to rotate about the origin
  - so translate that point to the origin



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

Composing to change axes

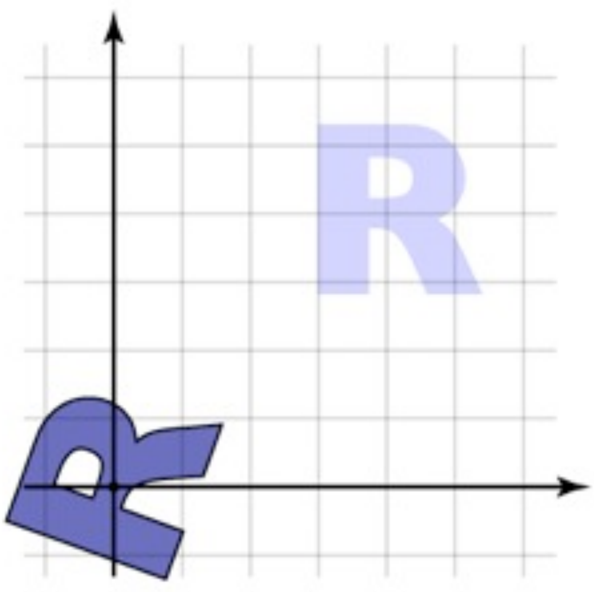
- Want to rotate about a particular point
  - could work out formulas directly…
- Know how to rotate about the origin
  - so translate that point to the origin



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

Composing to change axes

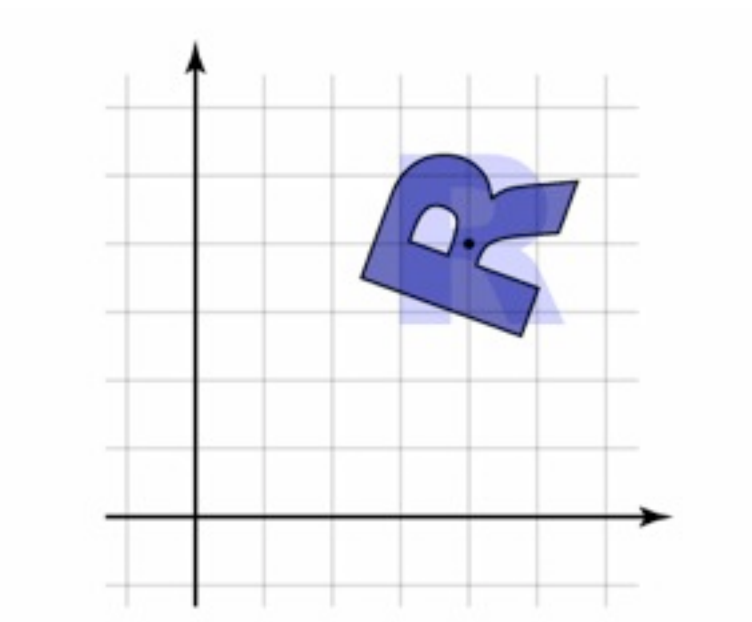
- Want to rotate about a particular point
  - could work out formulas directly…
- Know how to rotate about the origin
  - so translate that point to the origin



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

Composing to change axes

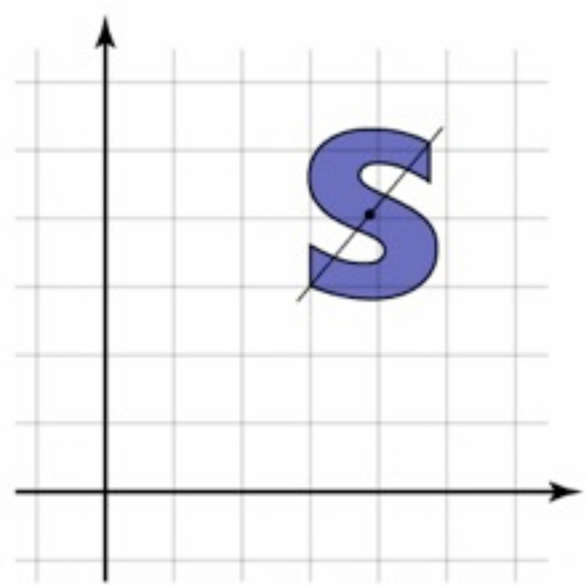
- Want to rotate about a particular point
  - could work out formulas directly…
- Know how to rotate about the origin
  - so translate that point to the origin



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

Composing to change axes

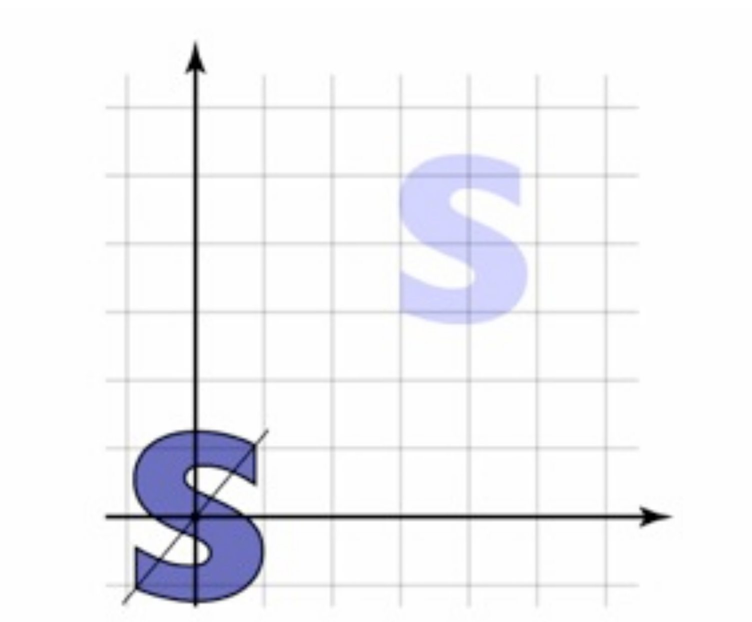
- Want to scale along a particular axis and point
- Know how to scale along the y axis at the origin
  - so translate to the origin and rotate to align axes



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

Composing to change axes

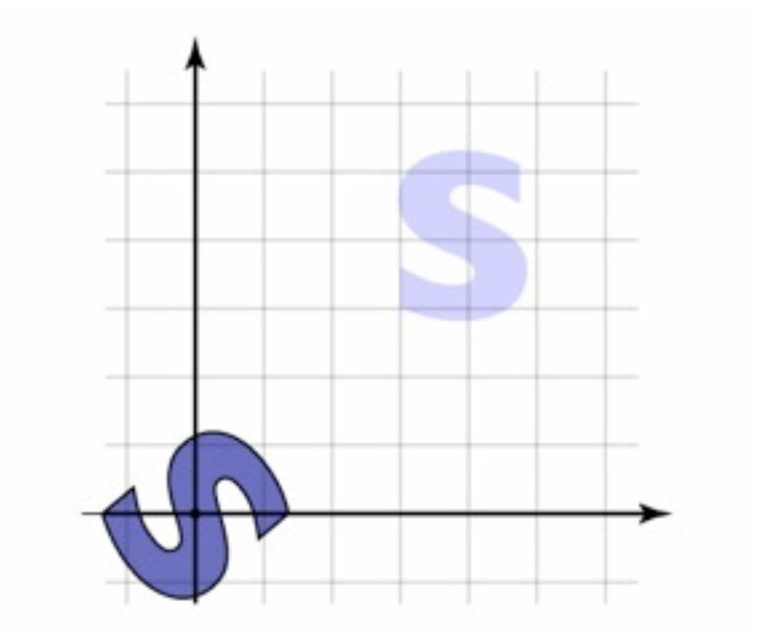
- Want to scale along a particular axis and point
- Know how to scale along the y axis at the origin
  - so translate to the origin and rotate to align axes



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

Composing to change axes

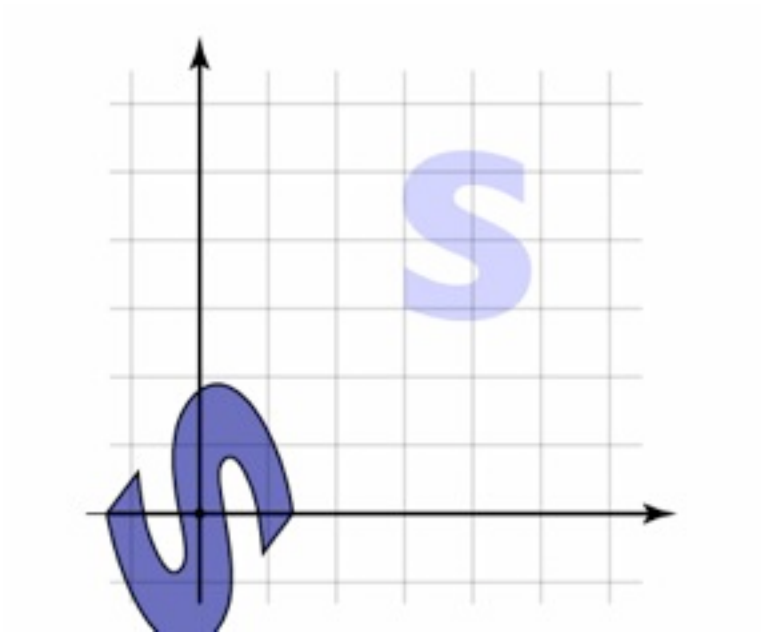
- Want to scale along a particular axis and point
- Know how to scale along the y axis at the origin
  - so translate to the origin and rotate to align axes



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

Composing to change axes

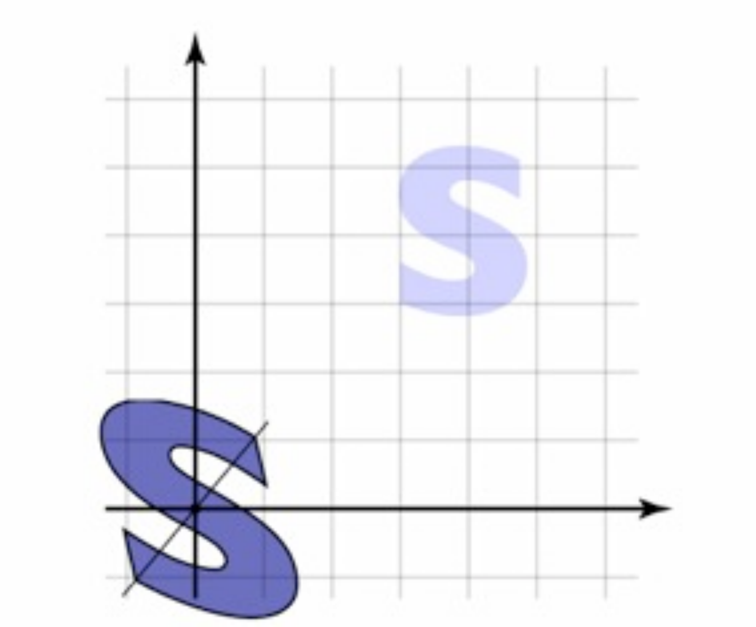
- Want to scale along a particular axis and point
- Know how to scale along the y axis at the origin
  - so translate to the origin and rotate to align axes



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

Composing to change axes

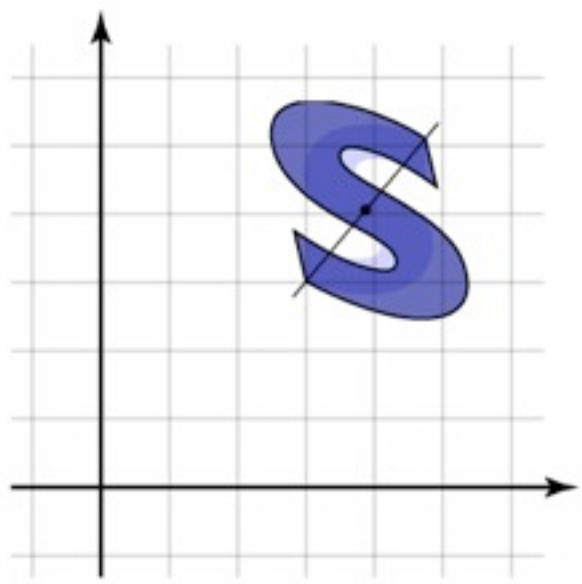
- Want to scale along a particular axis and point
- Know how to scale along the y axis at the origin
  - so translate to the origin and rotate to align axes



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

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



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

Transforming points and vectors

- Recall distinction points vs. vectors
  - vectors are just offsets (differences between points)
  - points have a location
    - represented by vector offset from a fixed origin
- Points and vectors transform differently
  - points respond to translation; vectors do not

$$\mathbf{v} = \mathbf{p} - \mathbf{q}$$

$$T(\mathbf{x}) = M\mathbf{x} + \mathbf{t}$$

$$\begin{aligned} T(\mathbf{p} - \mathbf{q}) &= M\mathbf{p} + \mathbf{t} - (M\mathbf{q} + \mathbf{t}) \\ &= M(\mathbf{p} - \mathbf{q}) + (\mathbf{t} - \mathbf{t}) = M\mathbf{v} \end{aligned}$$

Transforming points and vectors

- Homogeneous coords. let us exclude translation
  - just put 0 rather than 1 in the last place

$$\begin{bmatrix} M & \mathbf{t} \\ \mathbf{0}^T & 1 \end{bmatrix} \begin{bmatrix} \mathbf{p} \\ 1 \end{bmatrix} = \begin{bmatrix} M\mathbf{p} + \mathbf{t} \\ 1 \end{bmatrix} \qquad \begin{bmatrix} M & \mathbf{t} \\ \mathbf{0}^T & 1 \end{bmatrix} \begin{bmatrix} \mathbf{v} \\ 0 \end{bmatrix} = \begin{bmatrix} M\mathbf{v} \\ 0 \end{bmatrix}$$

  - and note that subtracting two points cancels the extra coordinate, resulting in a vector!
- Preview: projective transformations
  - what's really going on with this last coordinate?
  - think of R^2 embedded in R^3: all affine xfs. preserve $z=1$ plane
  - could have other transforms; project back to $z=1$

More math background

- Coordinate systems
  - Expressing vectors with respect to bases
  - Linear transformations as changes of basis

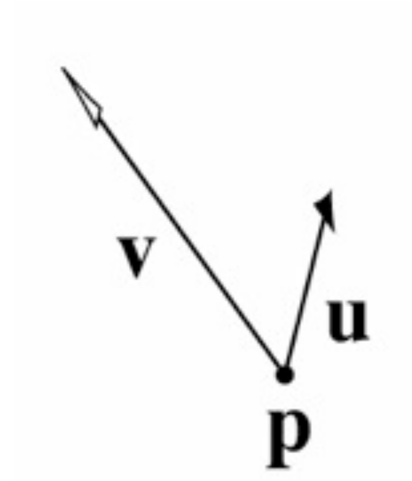
Affine change of coordinates

- Six degrees of freedom

$$\begin{bmatrix} a_1 & a_2 & a_3 \\ a_4 & a_5 & a_6 \\ 0 & 0 & 1 \end{bmatrix} \quad \text{or} \quad \begin{bmatrix} \mathbf{u} & \mathbf{v} & \mathbf{p} \\ 0 & 0 & 1 \end{bmatrix}$$

Affine change of coordinates

- Coordinate frame: point plus basis
- Interpretation: transformation changes representation of point from one basis to another
- "Frame to canonical" matrix has frame in columns
  - takes points represented in frame
  - represents them in canonical basis
  - e.g. [0 0], [1 0], [0 1]
- Seems backward but bears thinking about

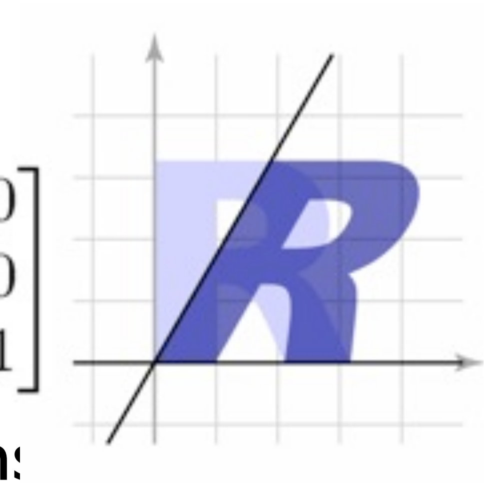


$$\begin{bmatrix} \mathbf{u} & \mathbf{v} & \mathbf{p} \\ 0 & 0 & 1 \end{bmatrix}$$

Affine change of coordinates

- A new way to "read off" the matrix
  - e.g. shear from earlier
  - can look at picture, see effect on basis vectors, write down matrix

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



- Also an easy way to construct transforms
  - e. g. scale by 2 across direction (1,2)

Affine change of coordinates

- When we move an object to the origin to apply a transformation, we are really changing coordinates
  - the transformation is easy to express in object's frame
  - so define it there and transform it

$$T_e = F T_F F^{-1}$$

  - T_e is the transformation expressed wrt. $\{e_1, e_2\}$
  - T_F is the transformation expressed in natural frame
  - F is the frame-to-canonical matrix $[u\ v\ p]$
- This is a *similarity transformation*

Coordinate frame summary

- Frame = point plus basis
- Frame matrix (frame-to-canonical) is

$$F = \begin{bmatrix} \mathbf{u} & \mathbf{v} & \mathbf{p} \\ 0 & 0 & 1 \end{bmatrix}$$

- Move points to and from frame by multiplying with *F*

$$p_e = F p_F \quad p_F = F^{-1} p_e$$

- Move transformations using similarity transforms

$$T_e = F T_F F^{-1} \quad T_F = F^{-1} T_e F$$

