Computer Graphics

Lecture 13

Transformation Composition

Homogeneous Coordinates

Affine Transformations

Announcements

TODO Today

- Composing linear transformations
- Homogeneous coordinates
- Affine transformations: change-of-frame view
- Rigid transformations
- Affine composition

TODO Tomorrow

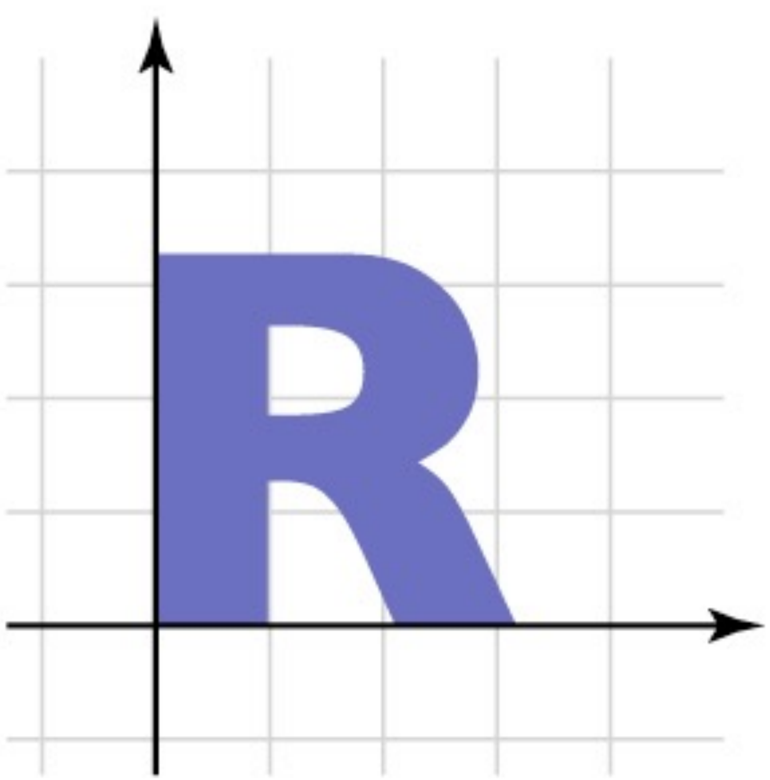
- Similarity transformations
- Inverses
- 3D
- 3D Rotations
- Transforming points vs vectors
- Transforming normals

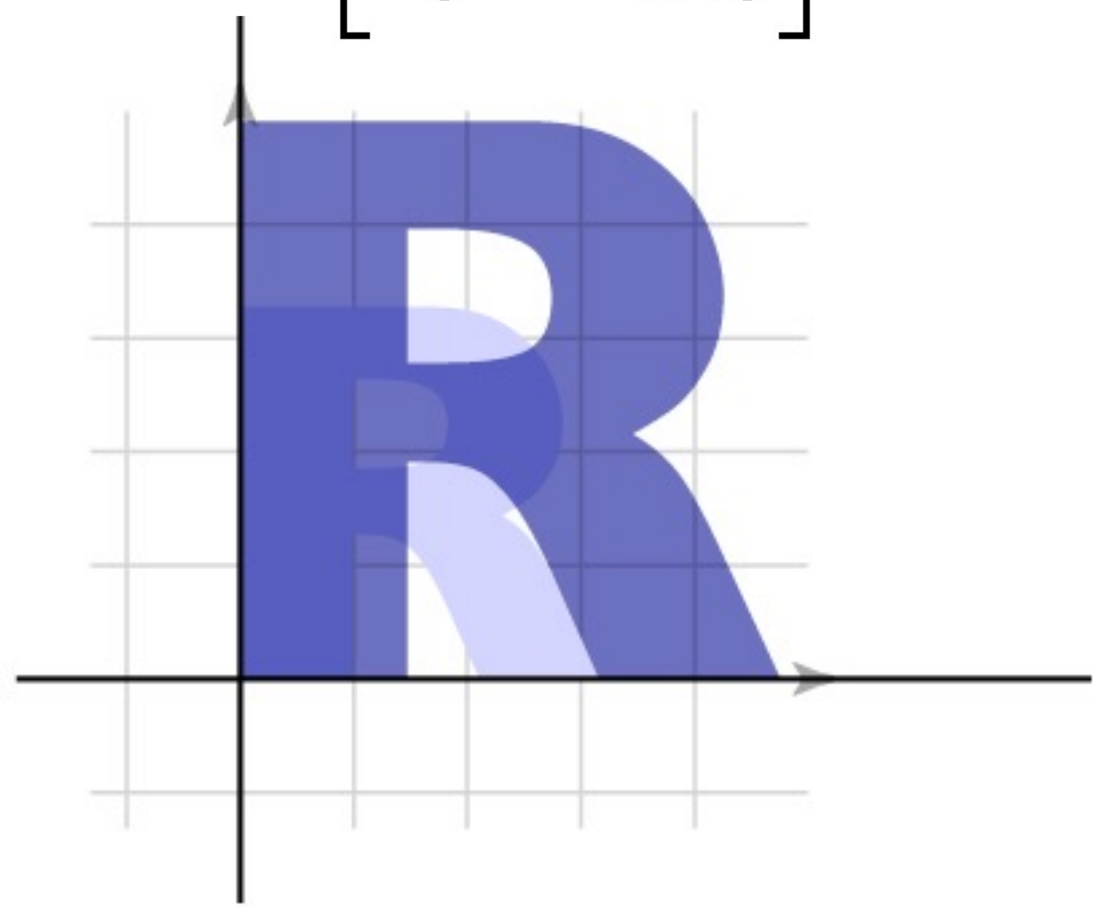
Last time: 2D Matrix Transformations

Linear transformation gallery

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

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

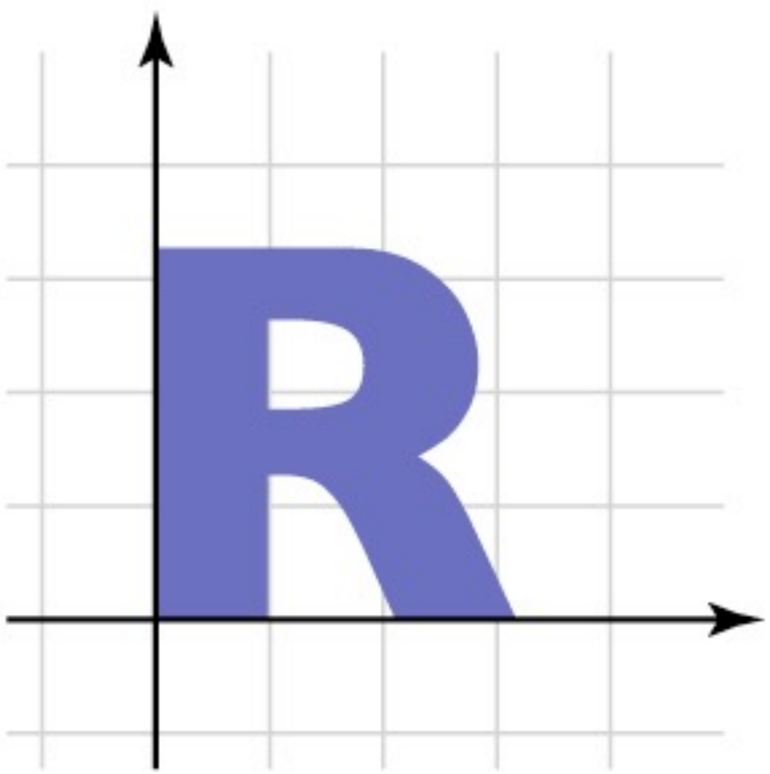


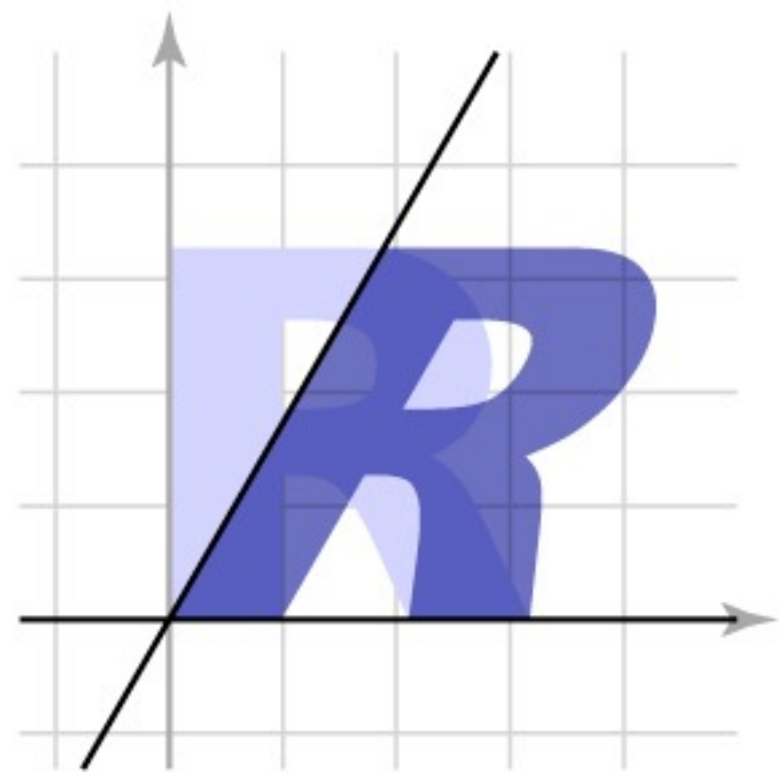


Linear transformation gallery

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

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



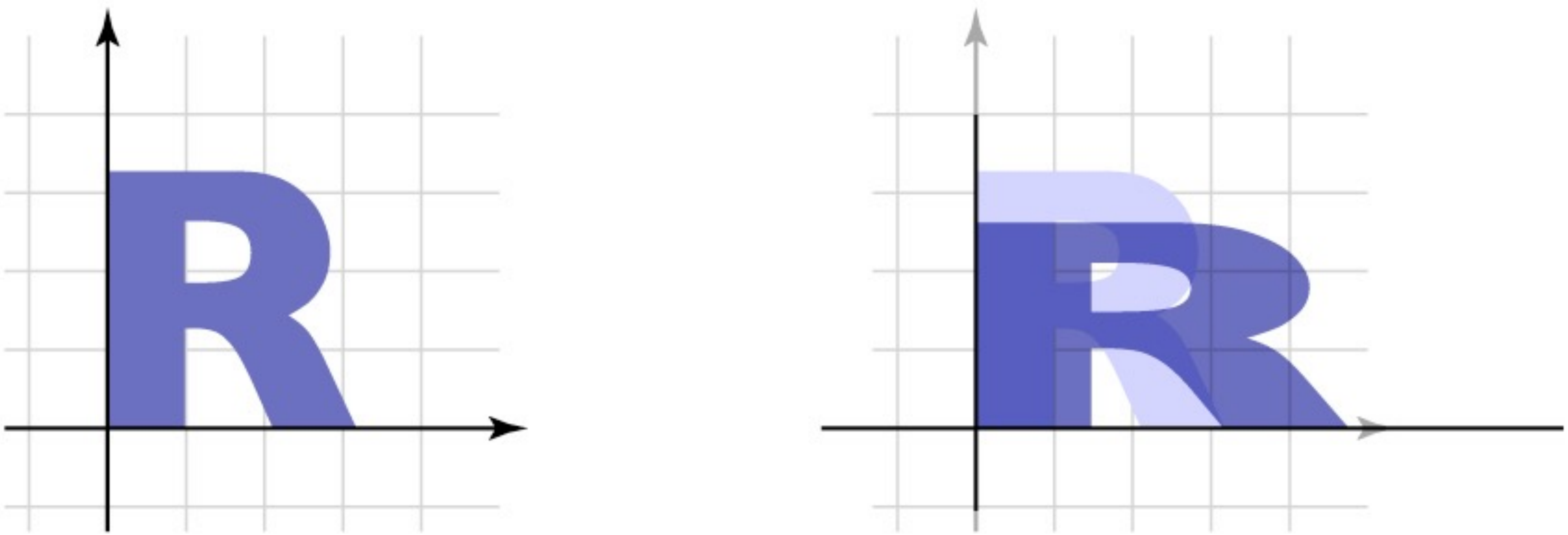


Linear transformation gallery

- Nonuniform scale

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

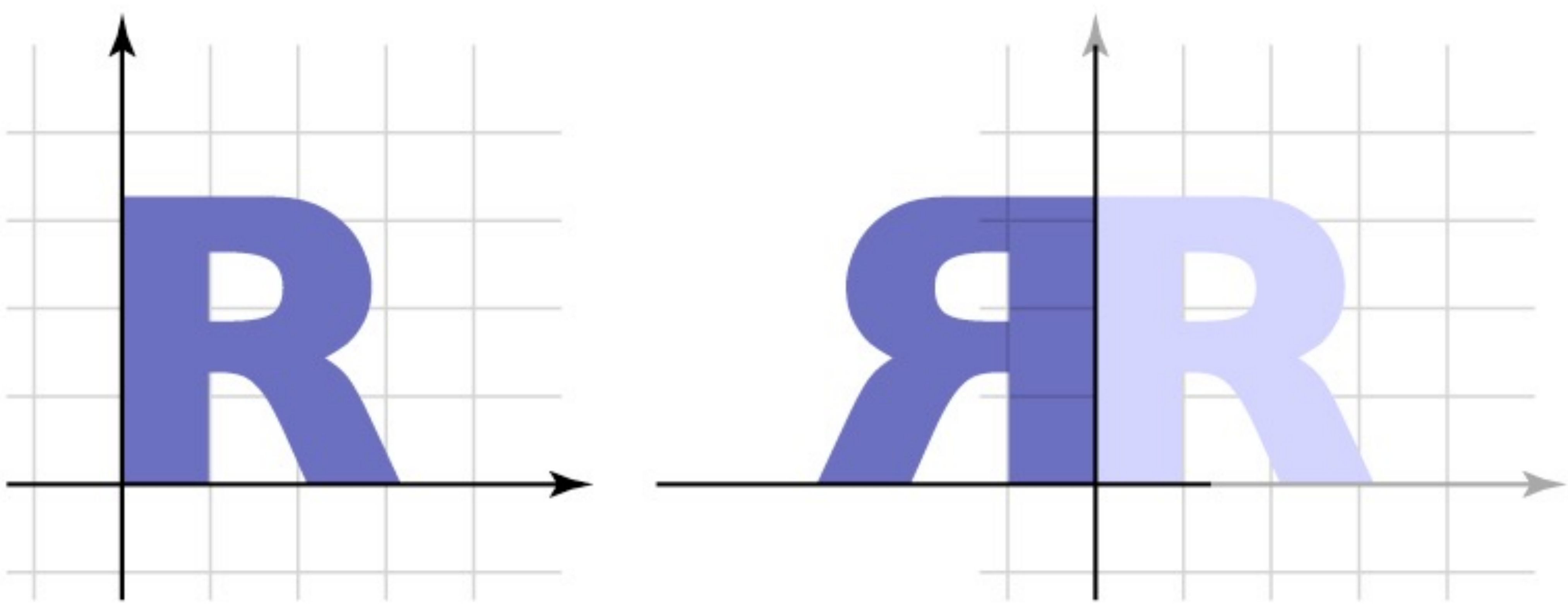
$$\begin{bmatrix} 1.5 & 0 \\ 0 & 0.8 \end{bmatrix}$$



Linear transformation gallery

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

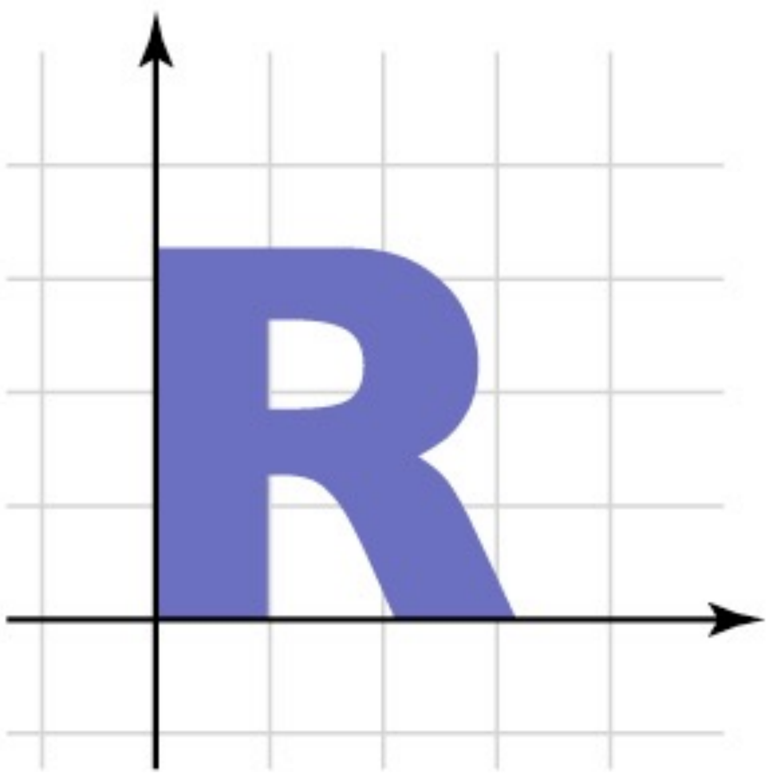
$$\begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix}$$

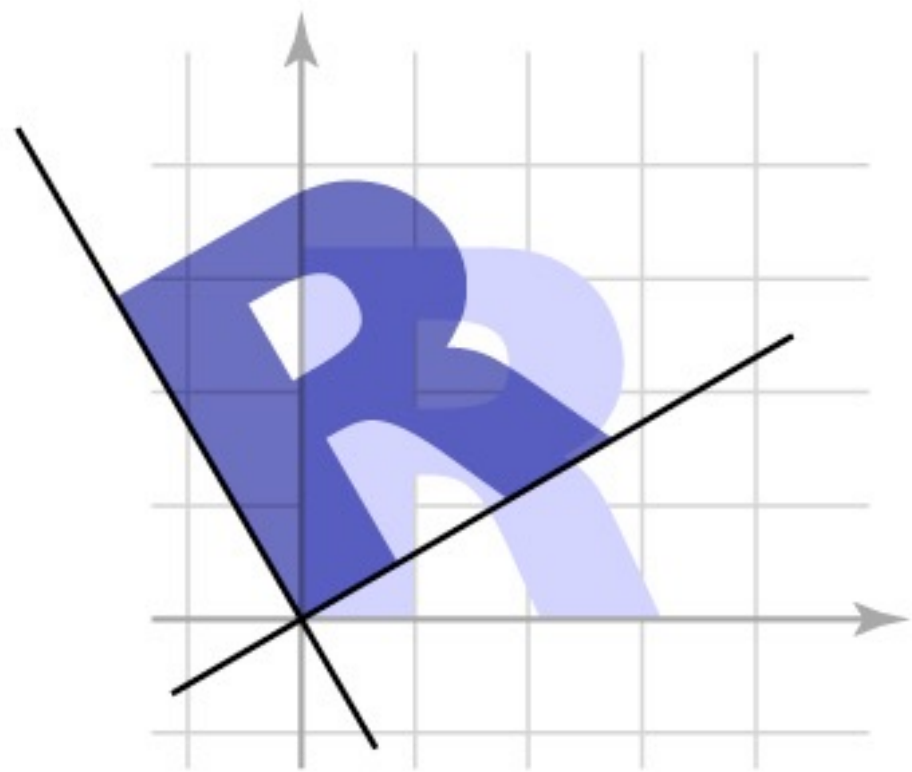


Linear transformation gallery

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

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





Composing Linear Transformations

$$A\colon \mathbb{R}^2 \to \mathbb{R}^2$$

`Vec2 A(in)::Vec2`

$$B\colon \mathbb{R}^2 \to \mathbb{R}^2$$

`Vec2 B(in::Vec2)`

$$A * B\colon \mathbb{R}^2 \to \mathbb{R}^2$$

`A(B(x::Vec2) => Vec2`

Composing Linear Transformations

$$A\colon \mathbb{R}^2 \to \mathbb{R}^2$$

```
Vec2 A(in)::Vec2
```

$$B\colon \mathbb{R}^2 \to \mathbb{R}^2$$

```
Vec2 B(in::Vec2)
```

$$A * B\colon \mathbb{R}^2 \to \mathbb{R}^2$$

```
A(B(x::Vec2) => Vec2
```

Example:

$$\begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix} \qquad \begin{bmatrix} 0.866 & -0.5 \\ 0.5 & 0.866 \end{bmatrix}$$

Composing Linear Transformations

$$A\colon \mathbb{R}^2 \to \mathbb{R}^2$$

```
Vec2 A(in)::Vec2
```

$$B\colon \mathbb{R}^2 \to \mathbb{R}^2$$

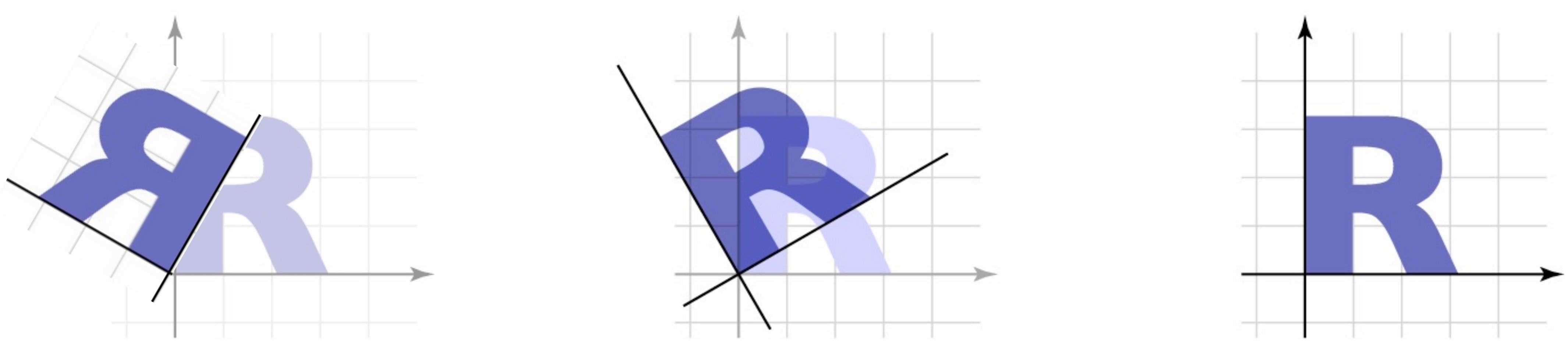
```
Vec2 B(in::Vec2)
```

$$A * B\colon \mathbb{R}^2 \to \mathbb{R}^2$$

```
A(B(x::Vec2) => Vec2
```

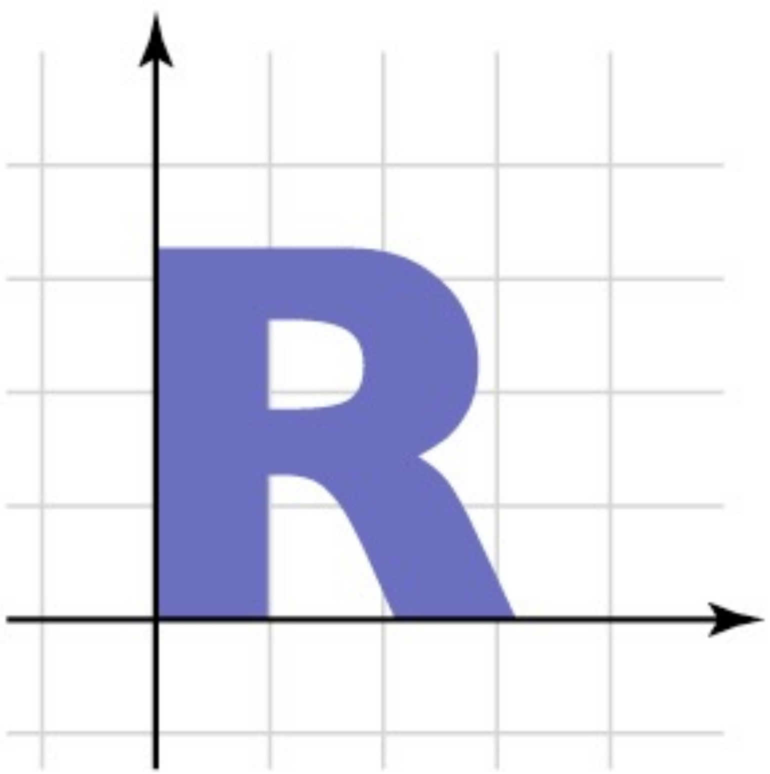
Example:

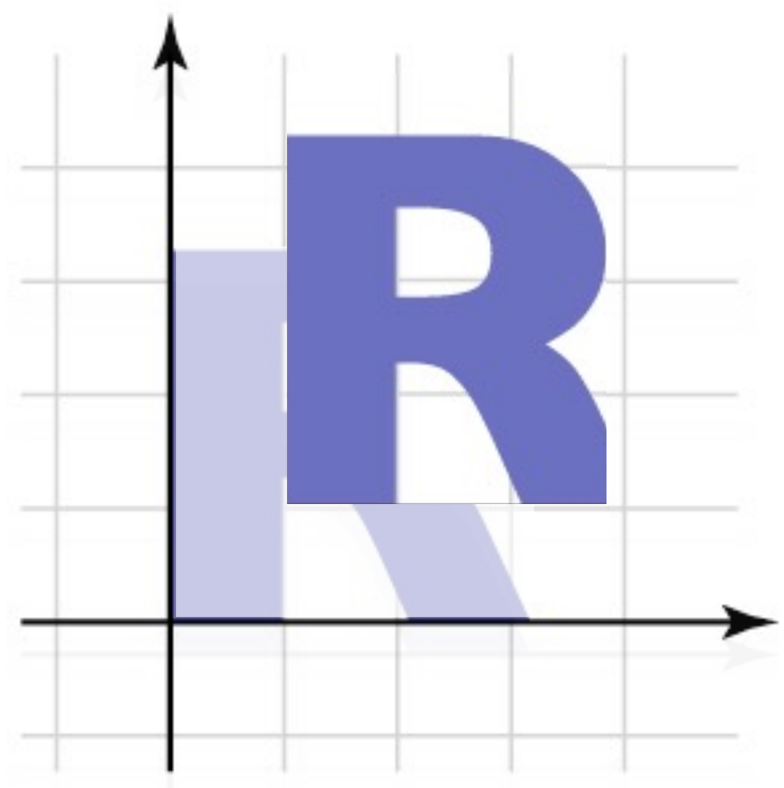
$$\begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix} \qquad \begin{bmatrix} 0.866 & -0.5 \\ 0.5 & 0.866 \end{bmatrix}$$



Linear transformation gallery

- Translation





About that translation thing…

- To the notes!

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

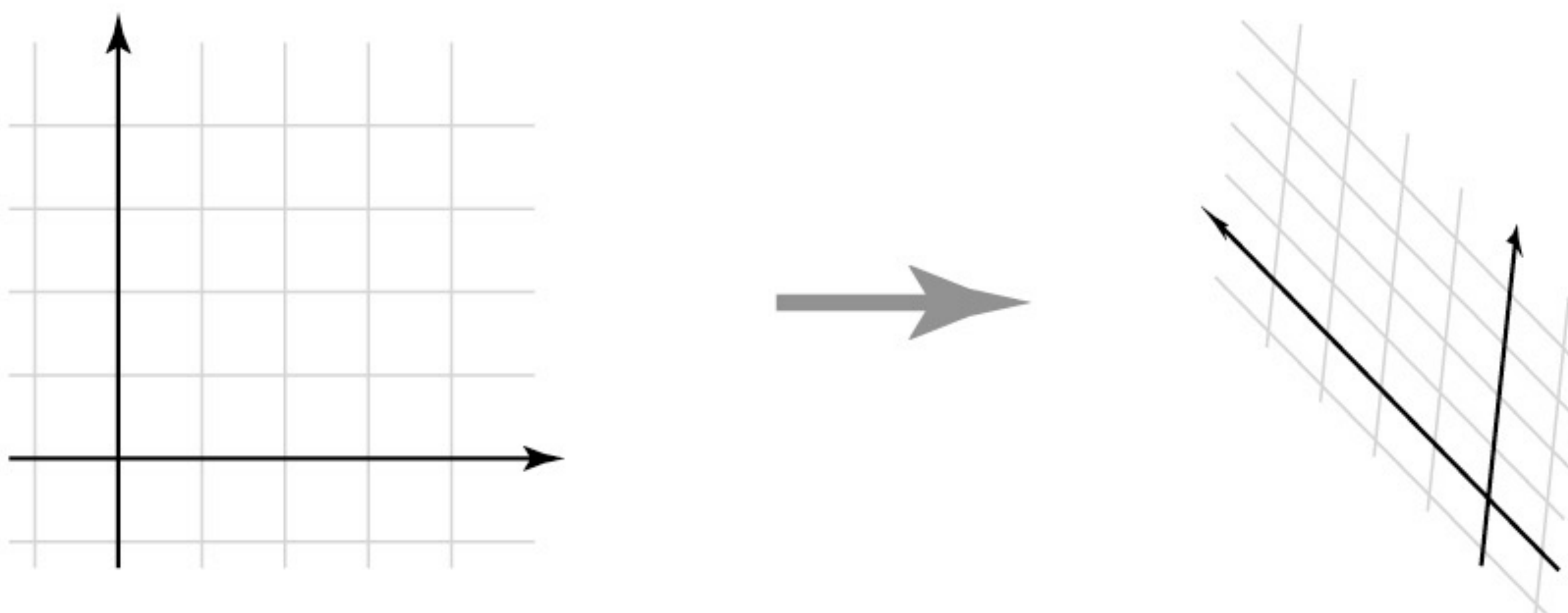
- Composition just works, by 3x3 matrix multiplication

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

- 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

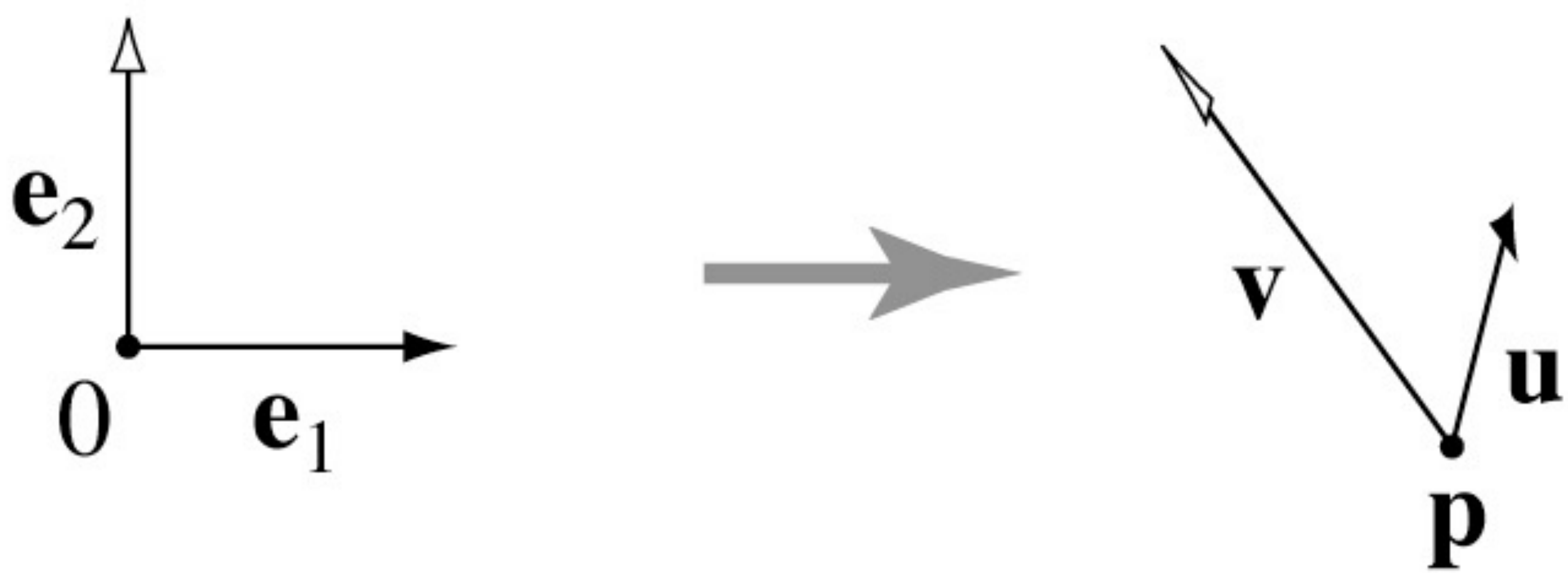
- The set of transformations we have been looking at is known as the "affine" transformations
  - straight lines preserved; parallel lines preserved
  - ratios of lengths along lines preserved (midpoints preserved)



Affine change of coordinates

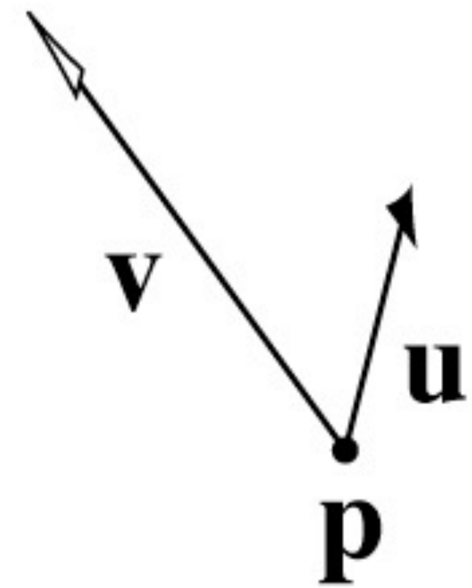
- Six degrees of freedom

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



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



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

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:

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

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

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

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

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

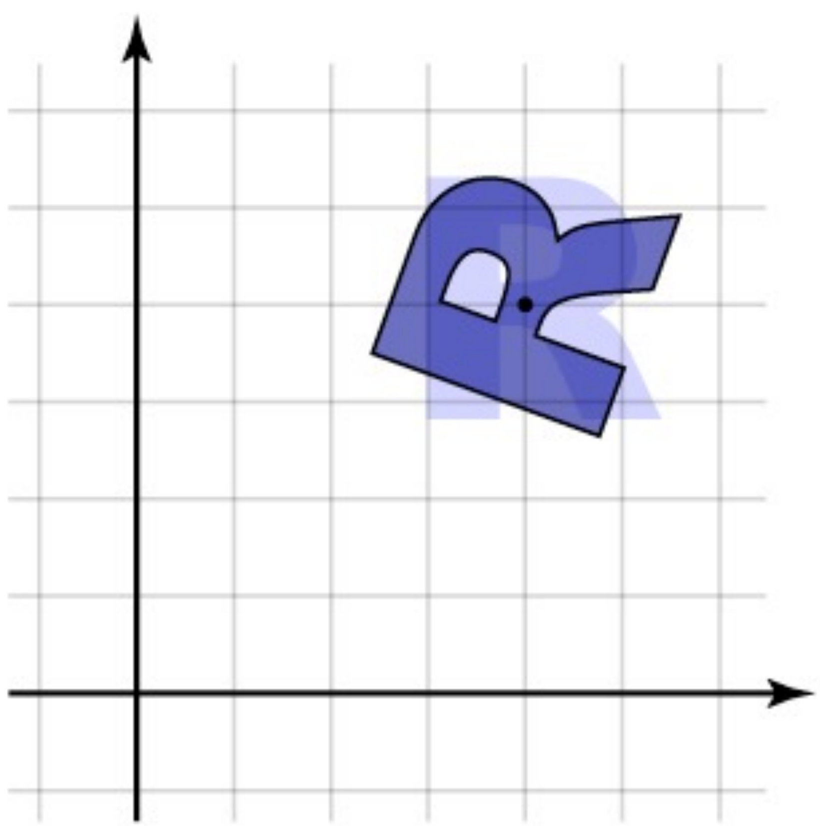
Affine Composition

- Composition just works, by 3x3 matrix multiplication

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

Affine Composition Example: Rotation about not-the-origin

- 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



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

Similarity Transformations

- When we move an object to the canonical frame to apply a transformation, we are 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*

