

Geometric Transformations

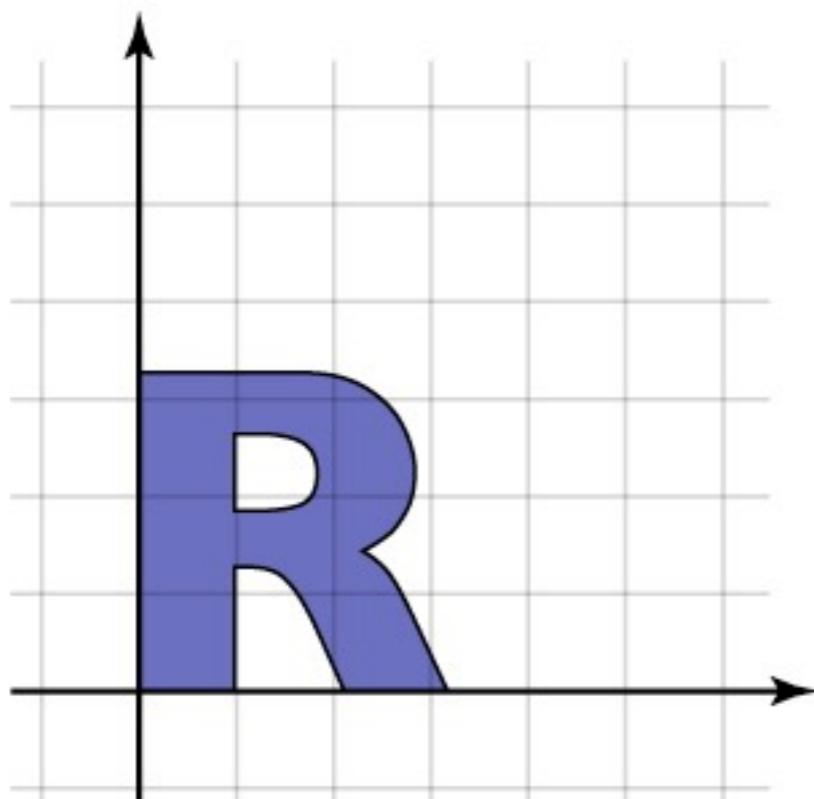
CS 4620 Lecture 10

A little quick math background

- **Notation for sets, functions, mappings**
- **Linear and affine transformations**
- **Matrices**
 - Matrix-vector multiplication
 - Matrix-matrix multiplication
- **Implicit vs. explicit geometry**

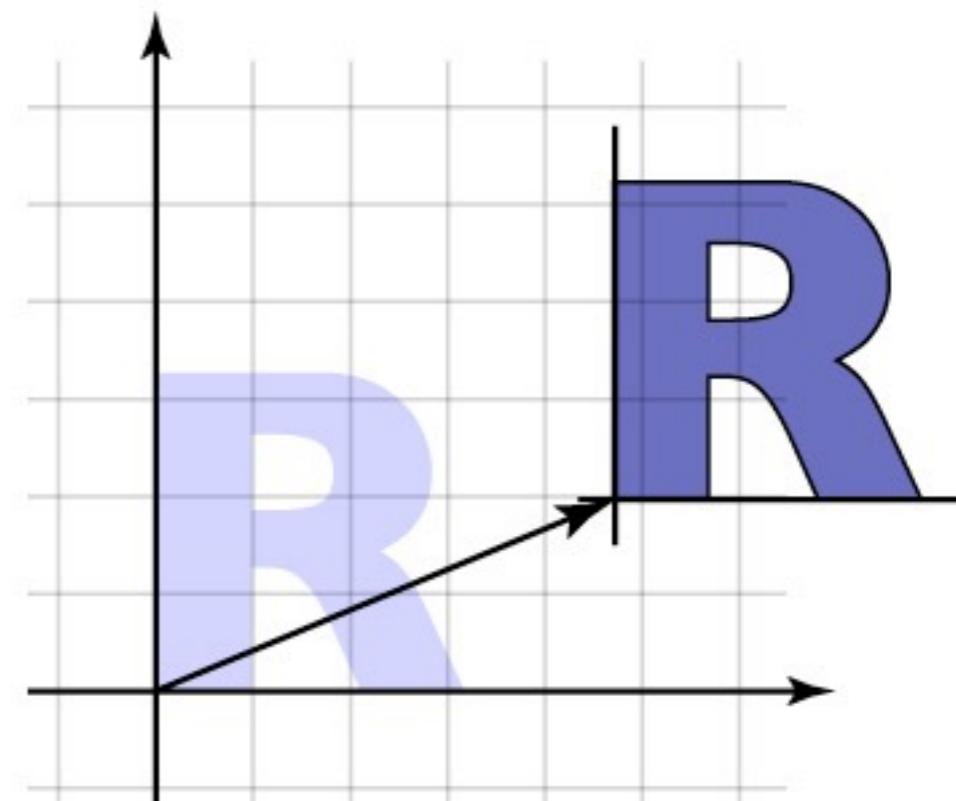
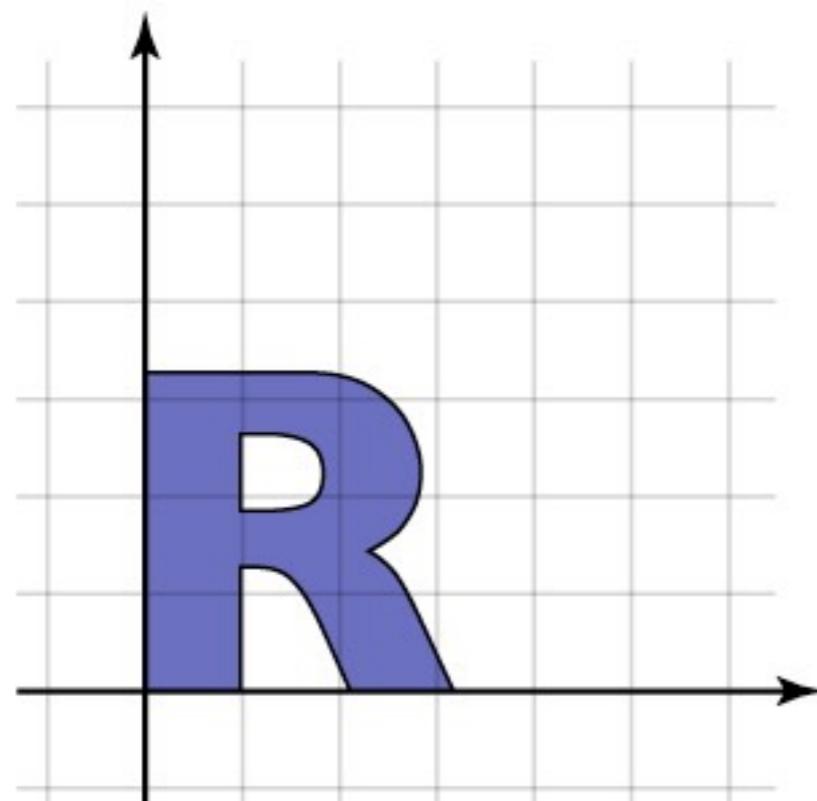
Translation

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



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)

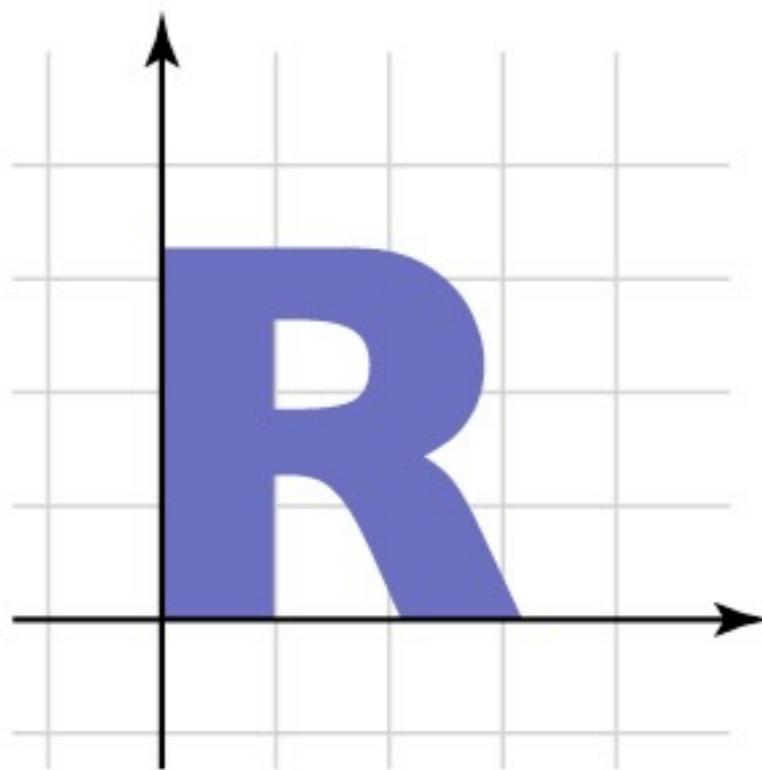
Geometry of 2D linear trans.

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

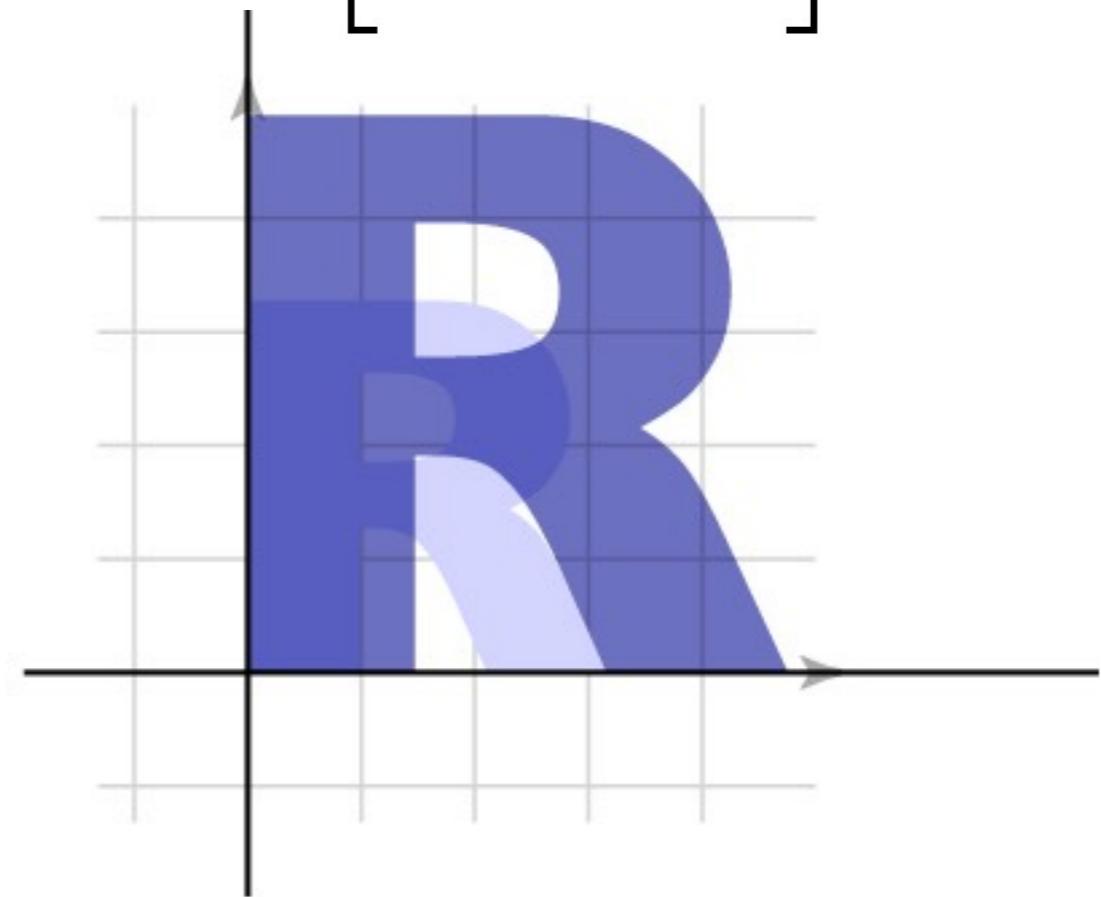
Linear transformation gallery

- **Uniform scale**

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



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

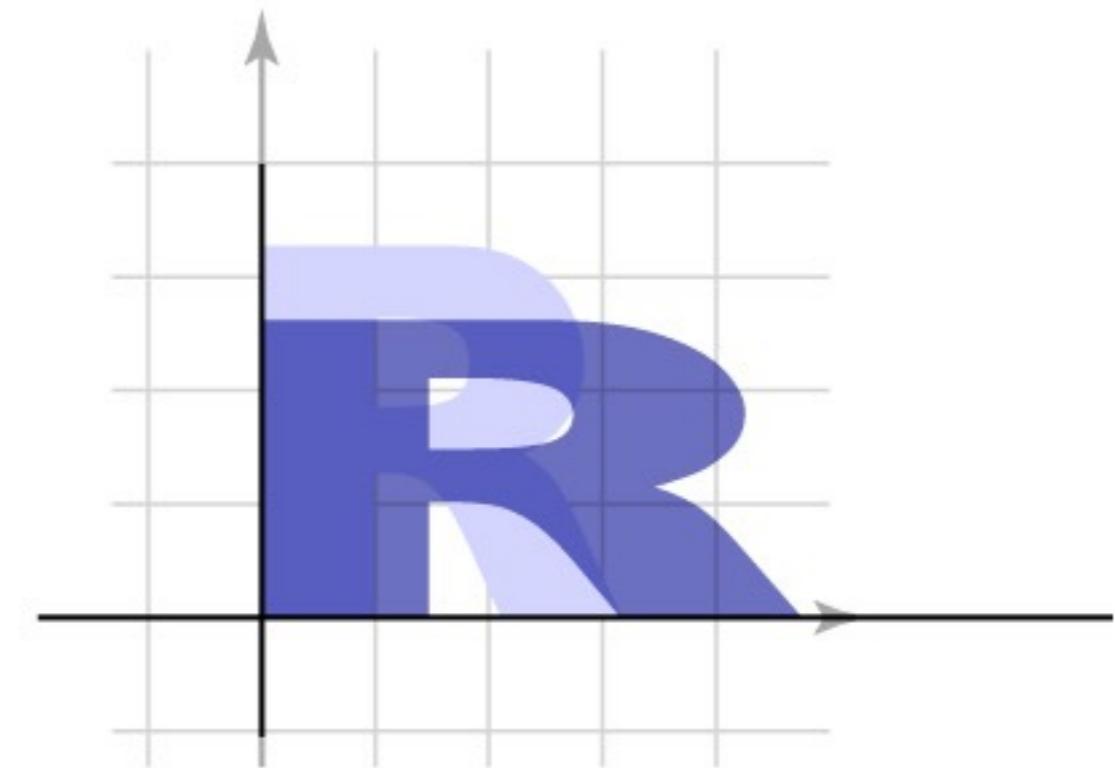
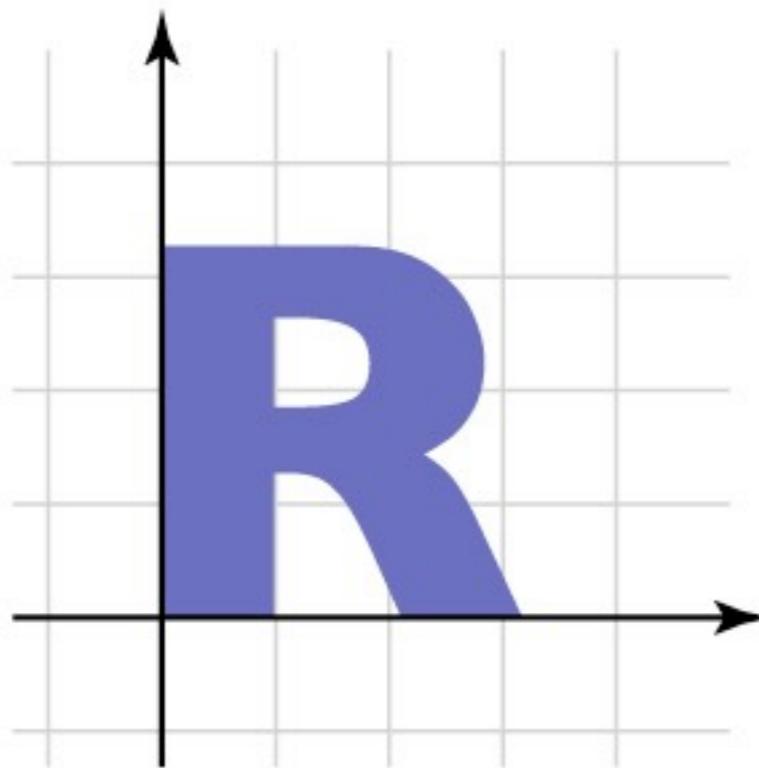


Linear transformation gallery

- **Nonuniform scale**

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

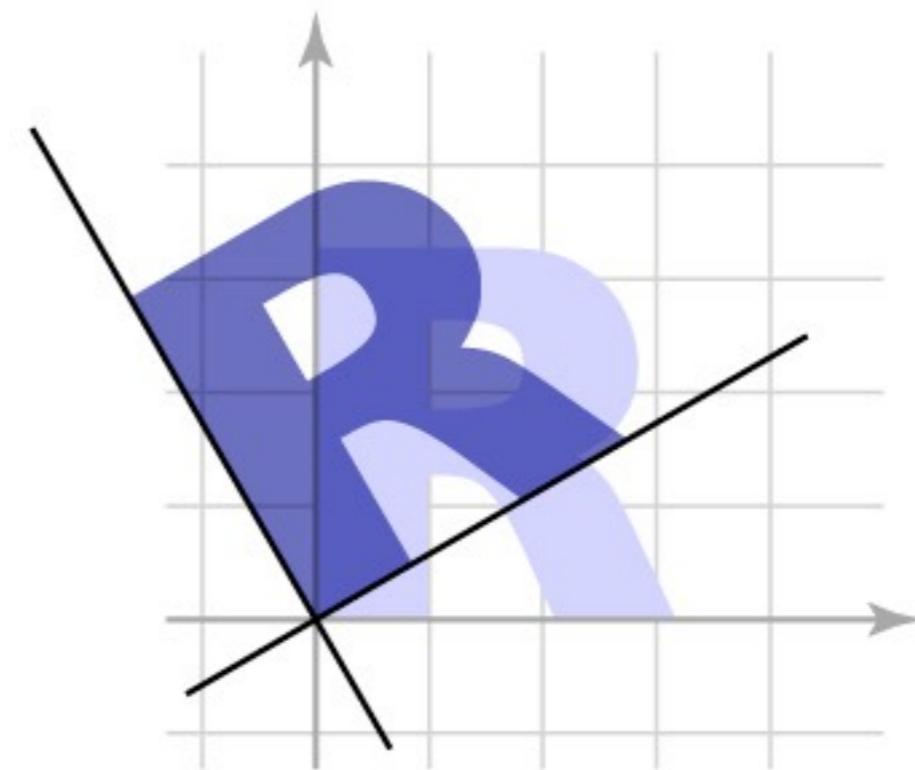
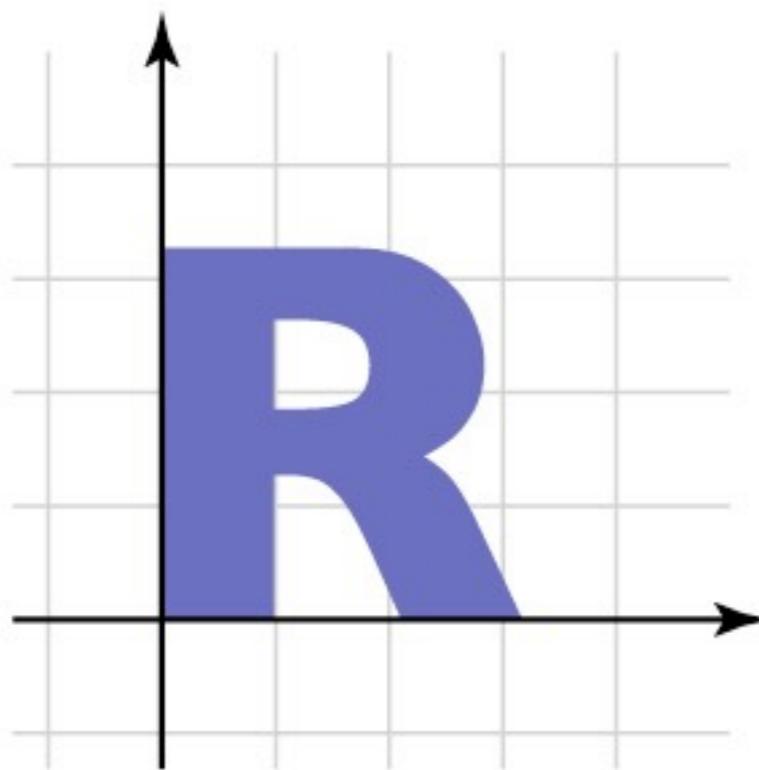
$$\begin{bmatrix} 1.5 & 0 \\ 0 & 0.8 \end{bmatrix}$$



Linear transformation gallery

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

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

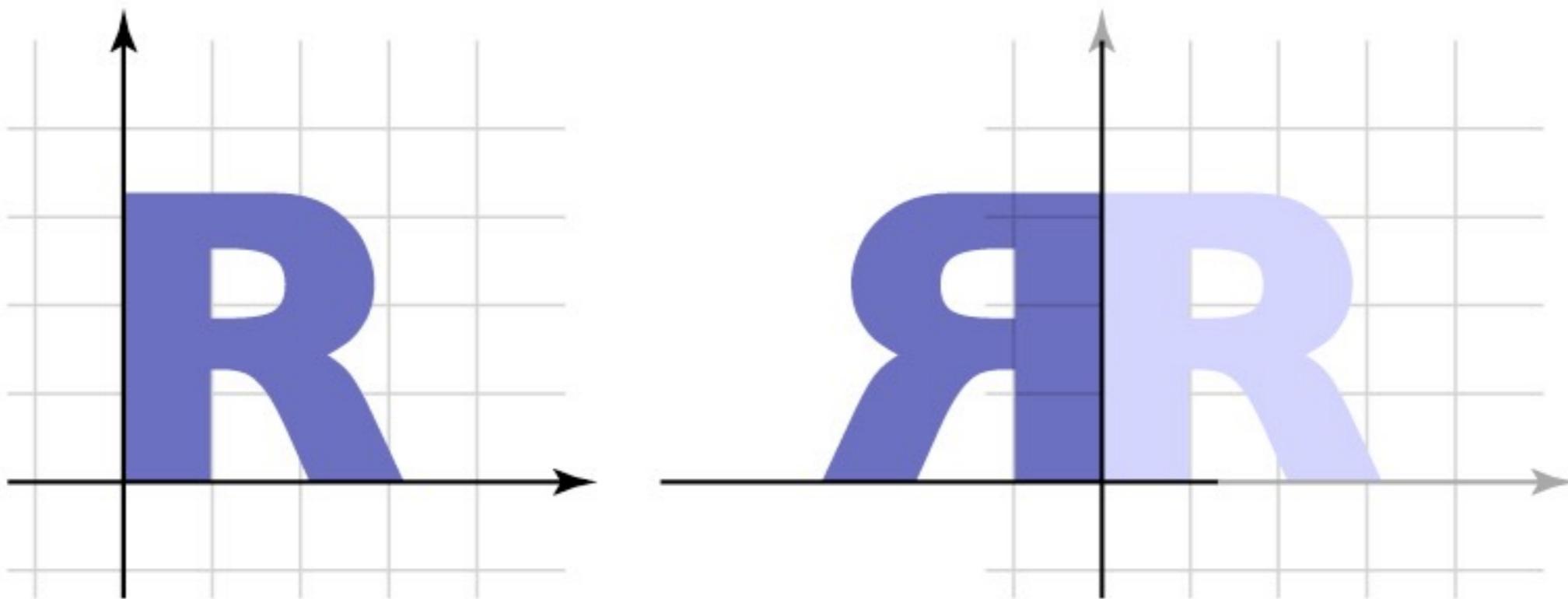


Linear transformation gallery

- **Reflection**

- can consider it a special case of nonuniform scale

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



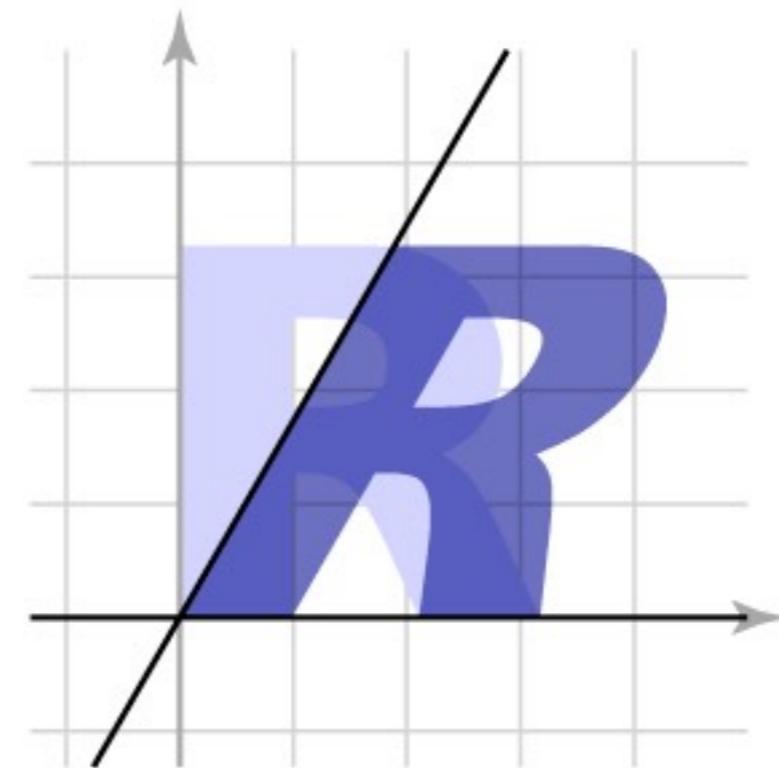
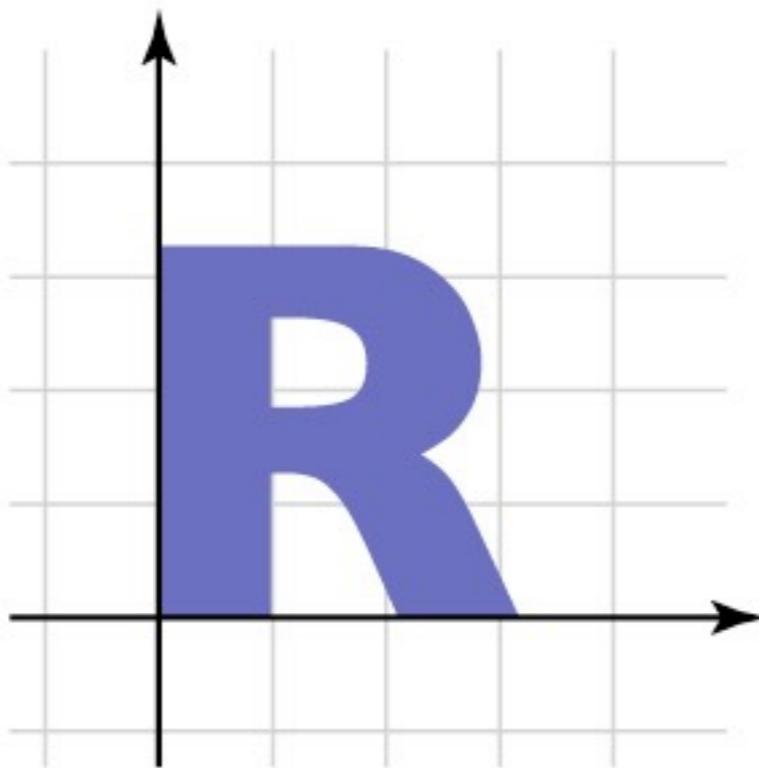
Linear transformation gallery

- **Shear**

- can also build these from rotations and nonuniform scales

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

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



Composing transformations

- **Want to move an object, then move it some more**
 - $\mathbf{p} \rightarrow T(\mathbf{p}) \rightarrow S(T(\mathbf{p})) = (S \circ T)(\mathbf{p})$
- **We need to represent $S \circ T$ (“ 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

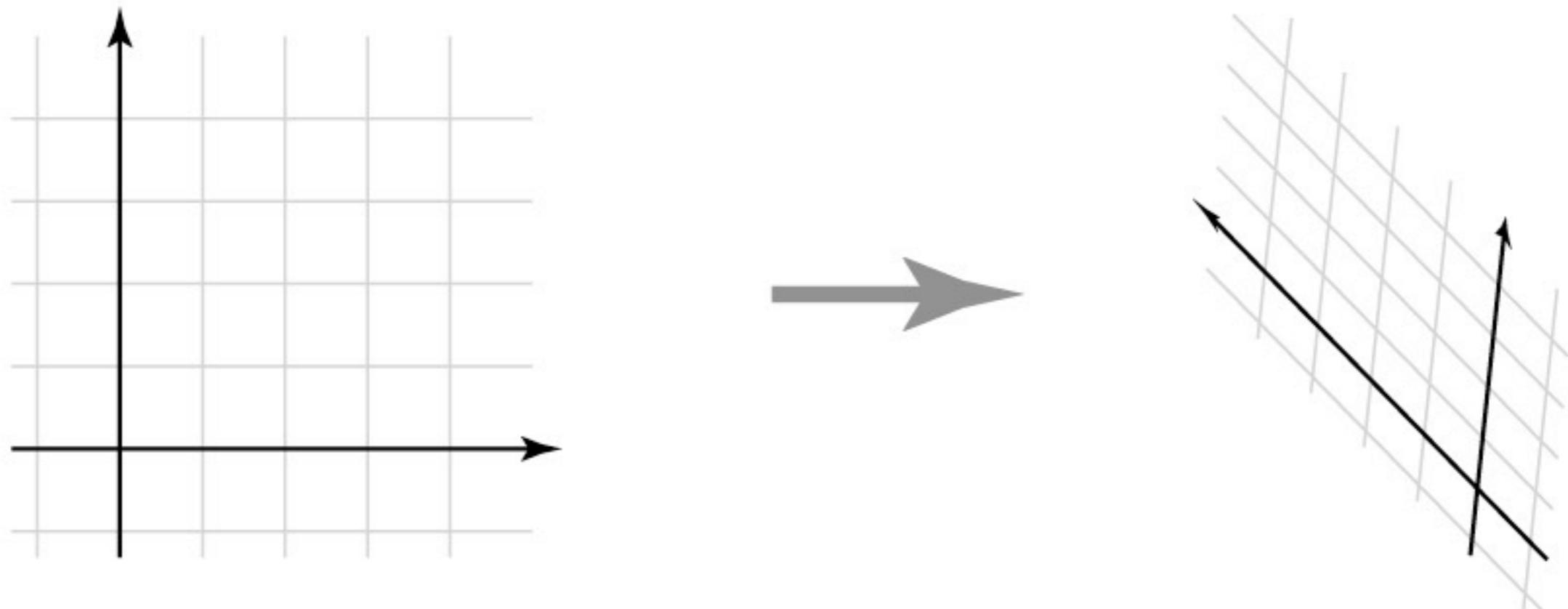
- **Composition just works, by 3x3 matrix multiplication**

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

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

Affine transformations

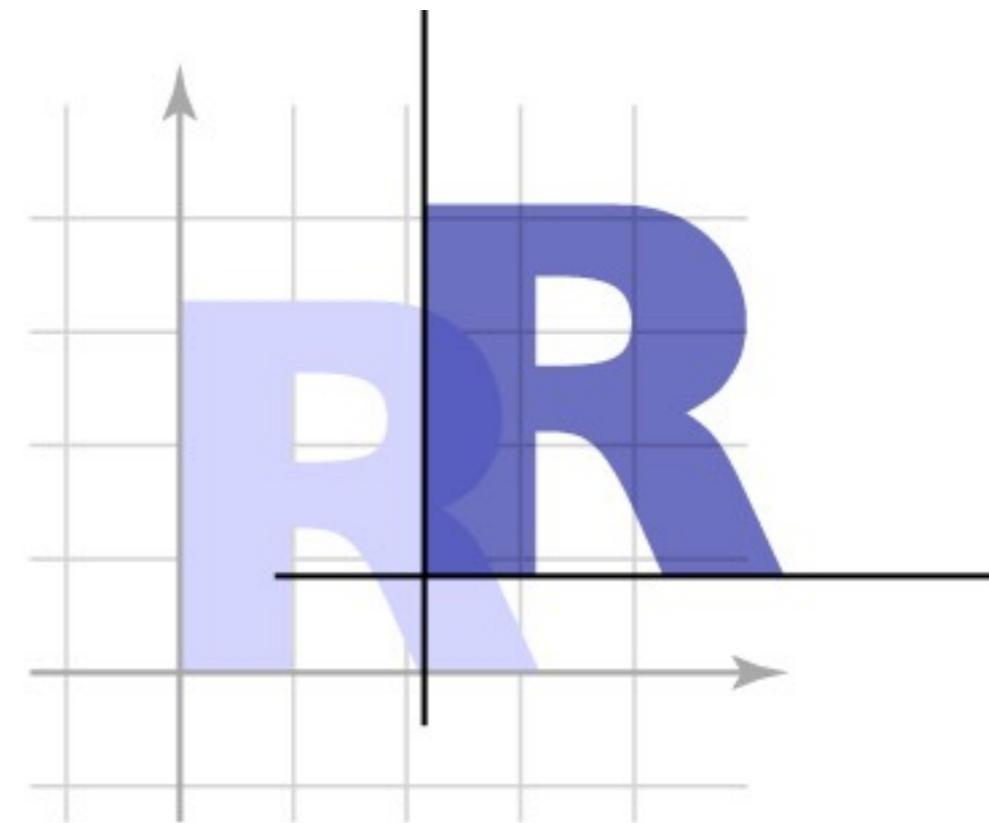
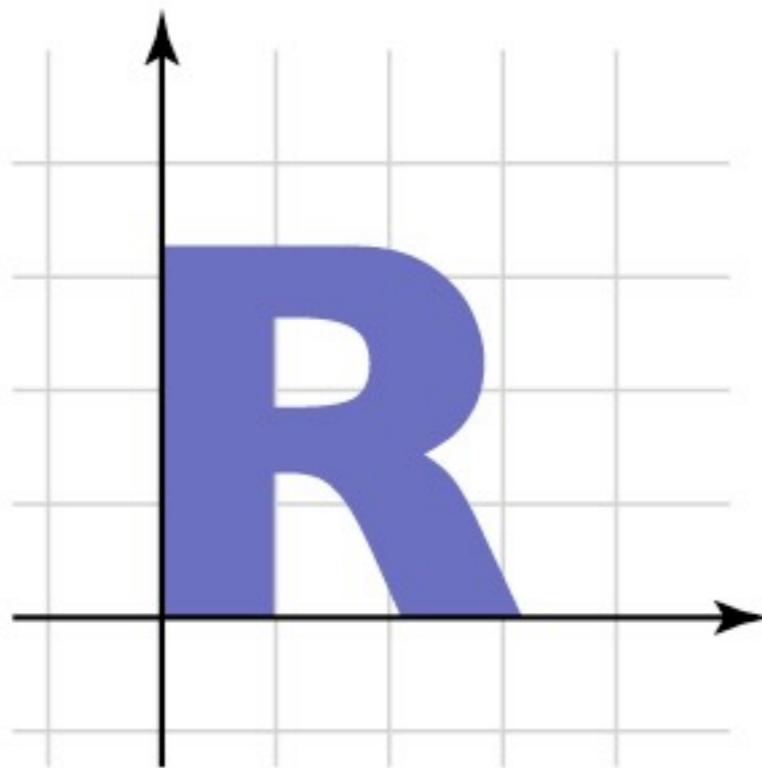
- **The set of transformations we have been looking at is known as the “affine” transformations**
 - straight lines preserved; parallel lines preserved
 - ratios of lengths along lines preserved (midpoints preserved)



Affine transformation gallery

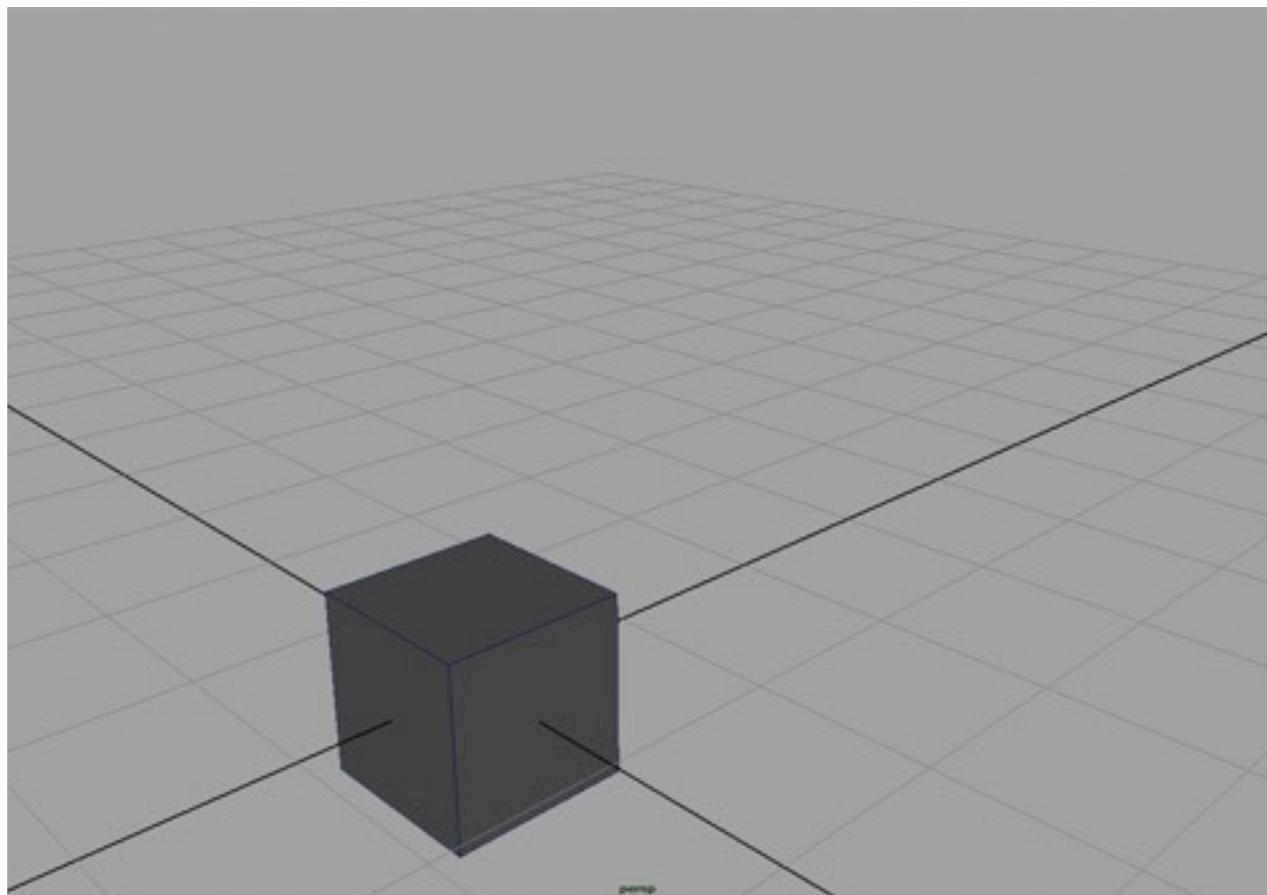
- **Translation**

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



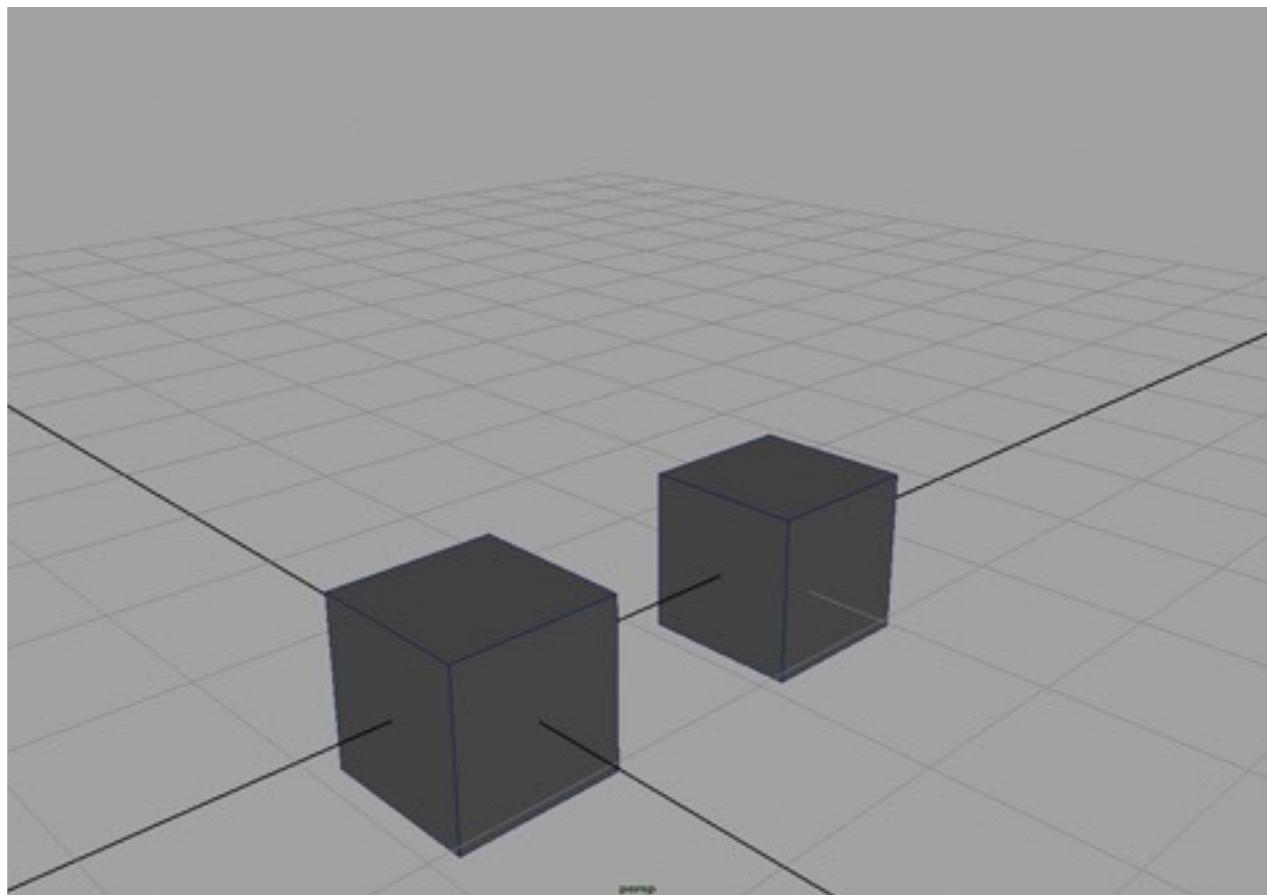
Translation

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$



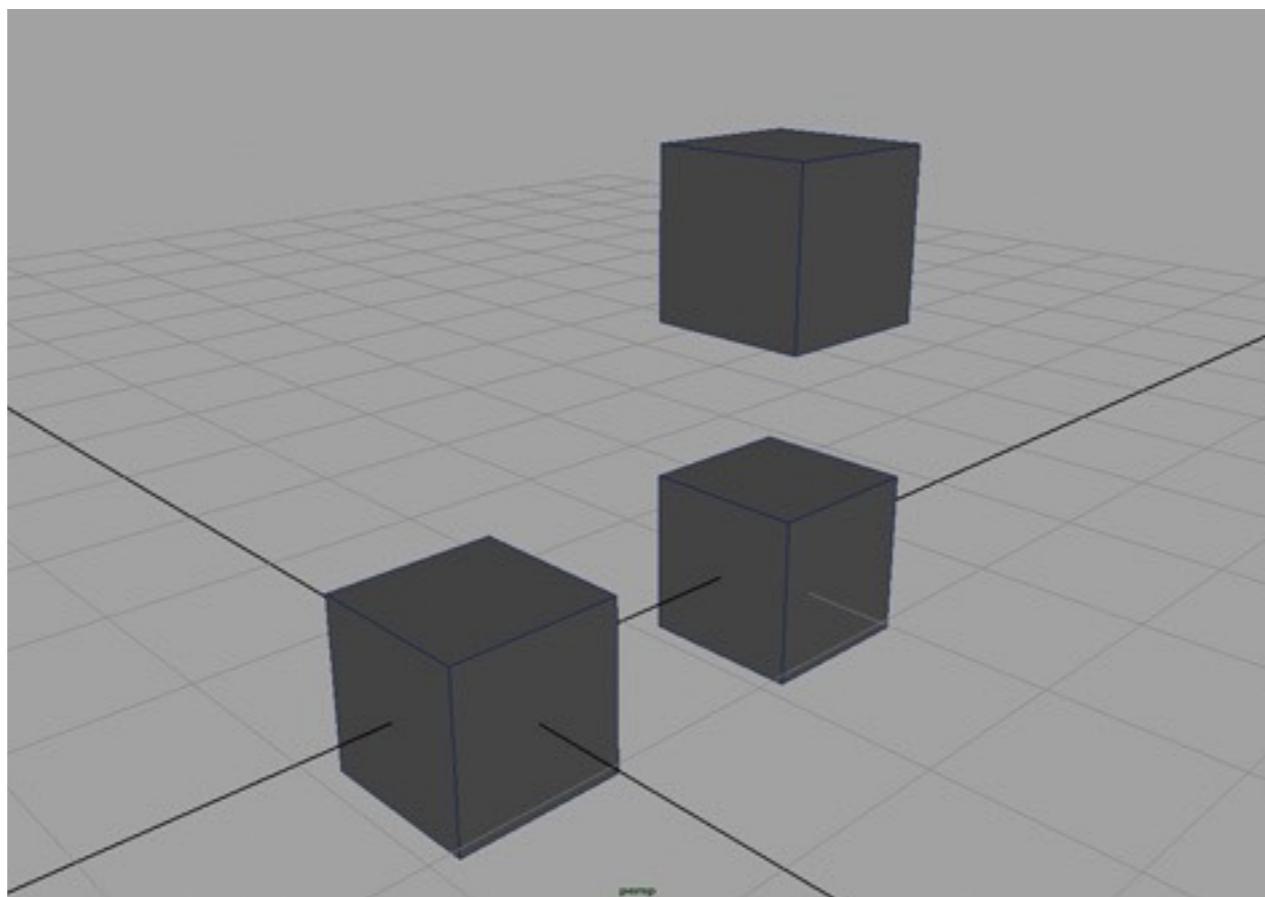
Translation

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$



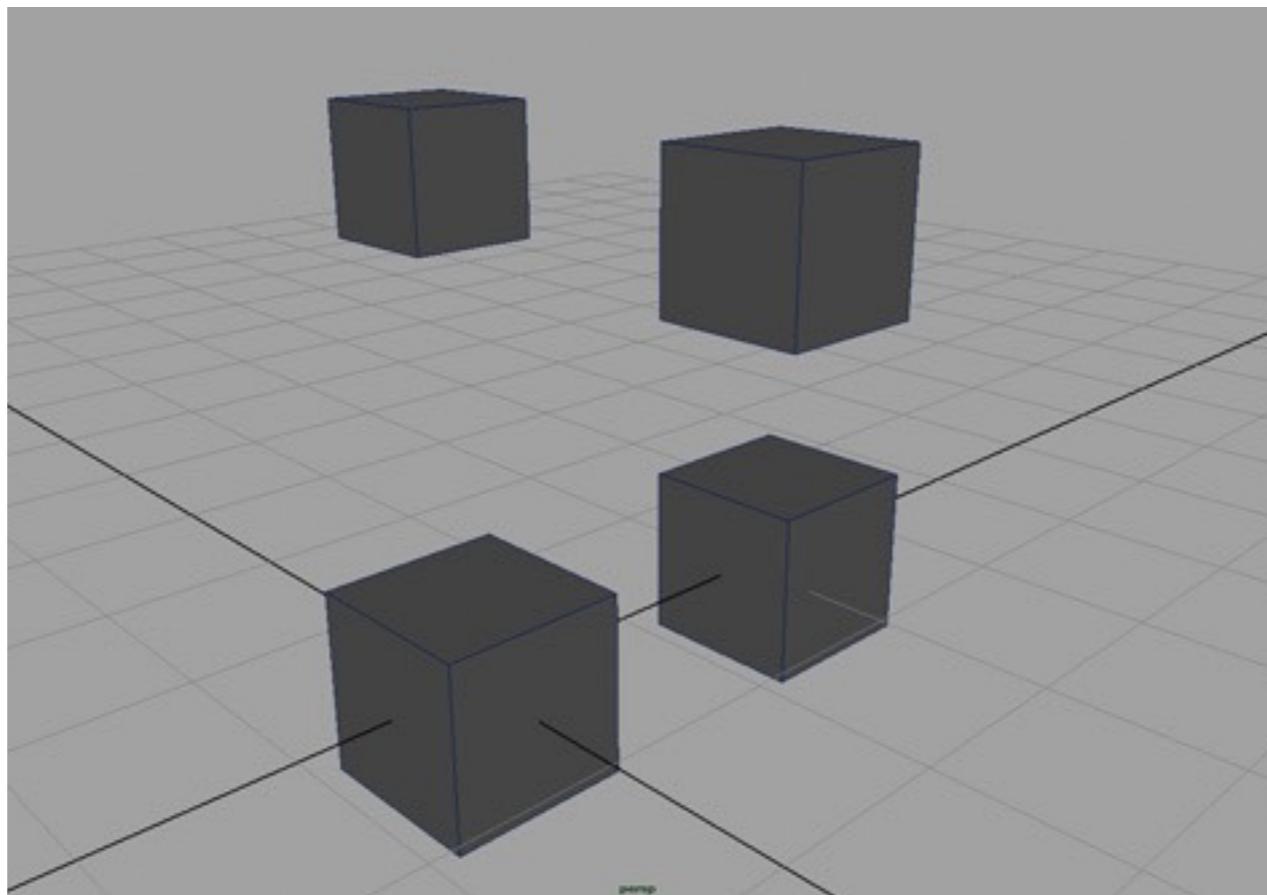
Translation

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$



Translation

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

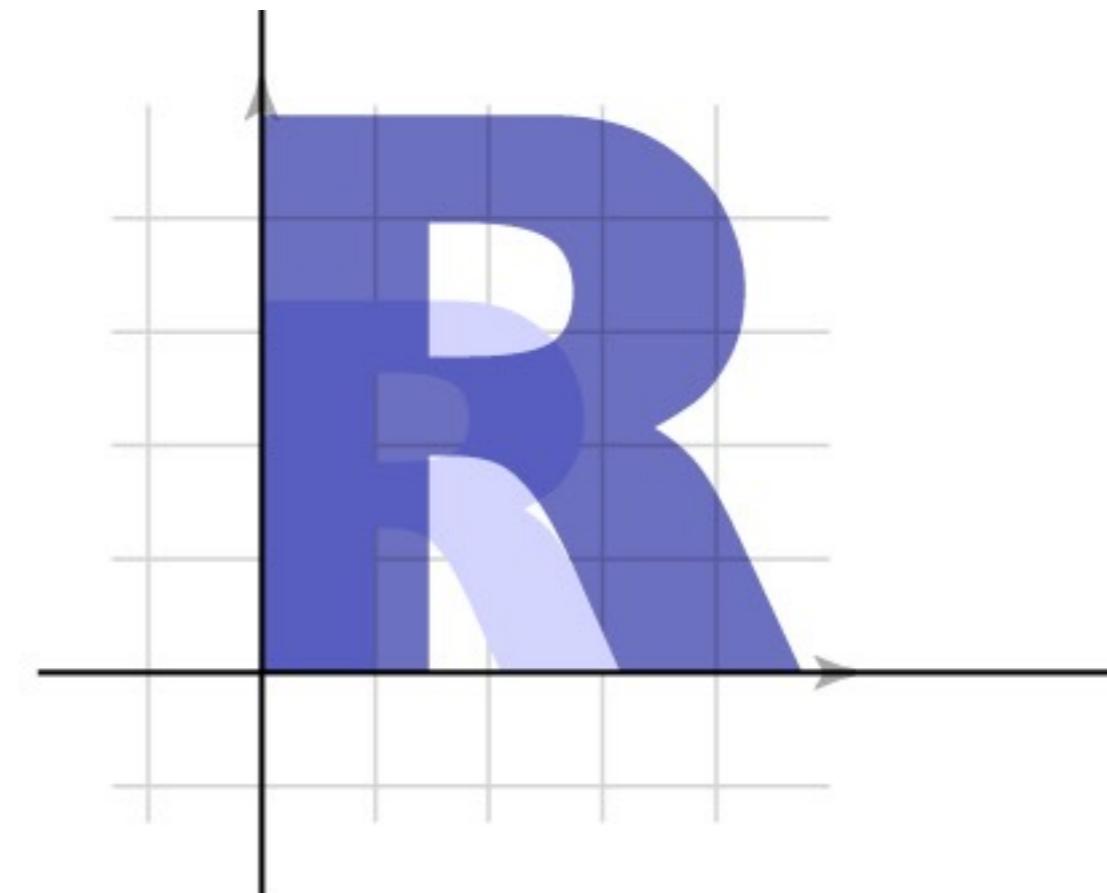
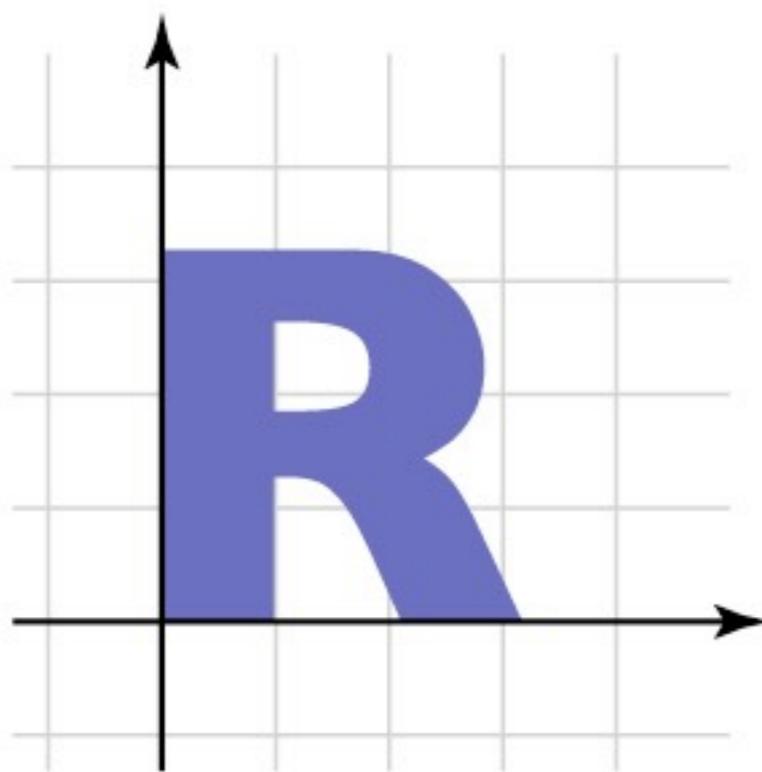


Affine transformation gallery

- **Uniform scale**

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

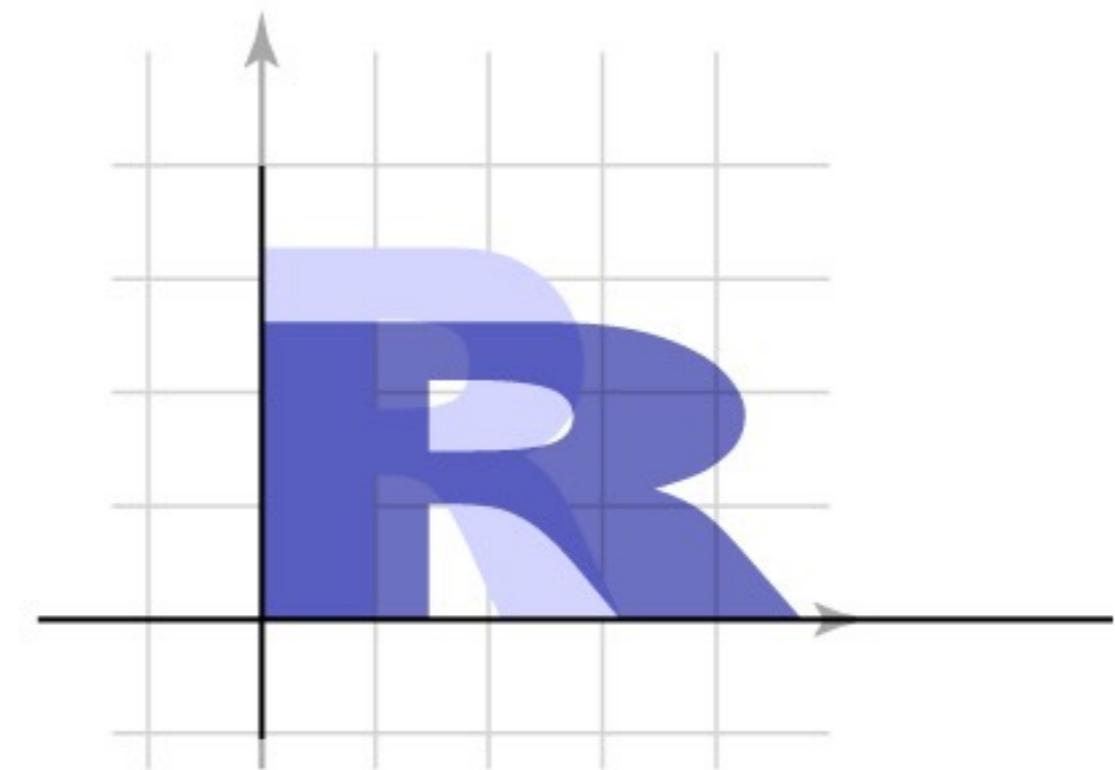
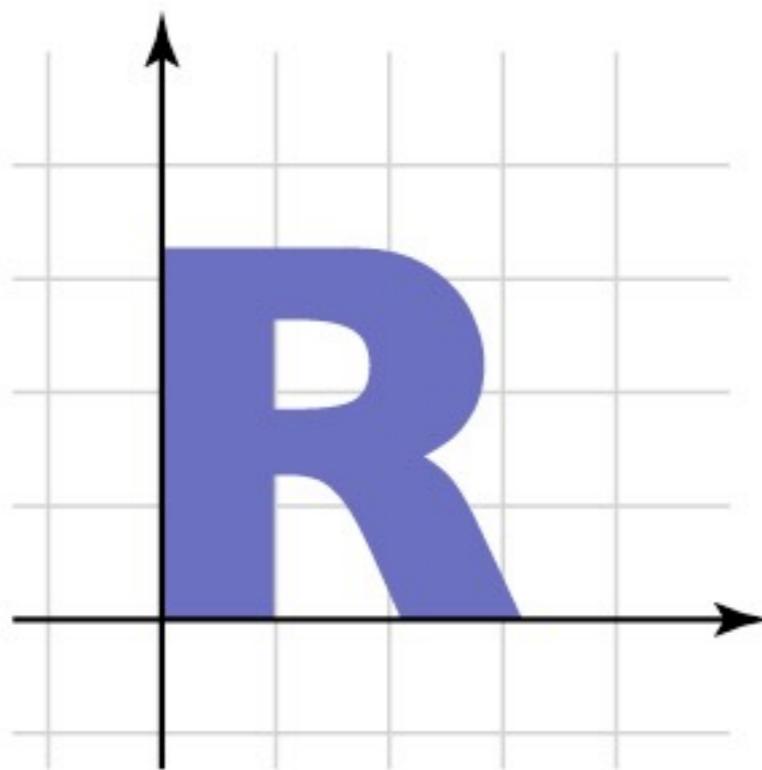
$$\begin{bmatrix} 1.5 & 0 & 0 \\ 0 & 1.5 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$



Affine transformation gallery

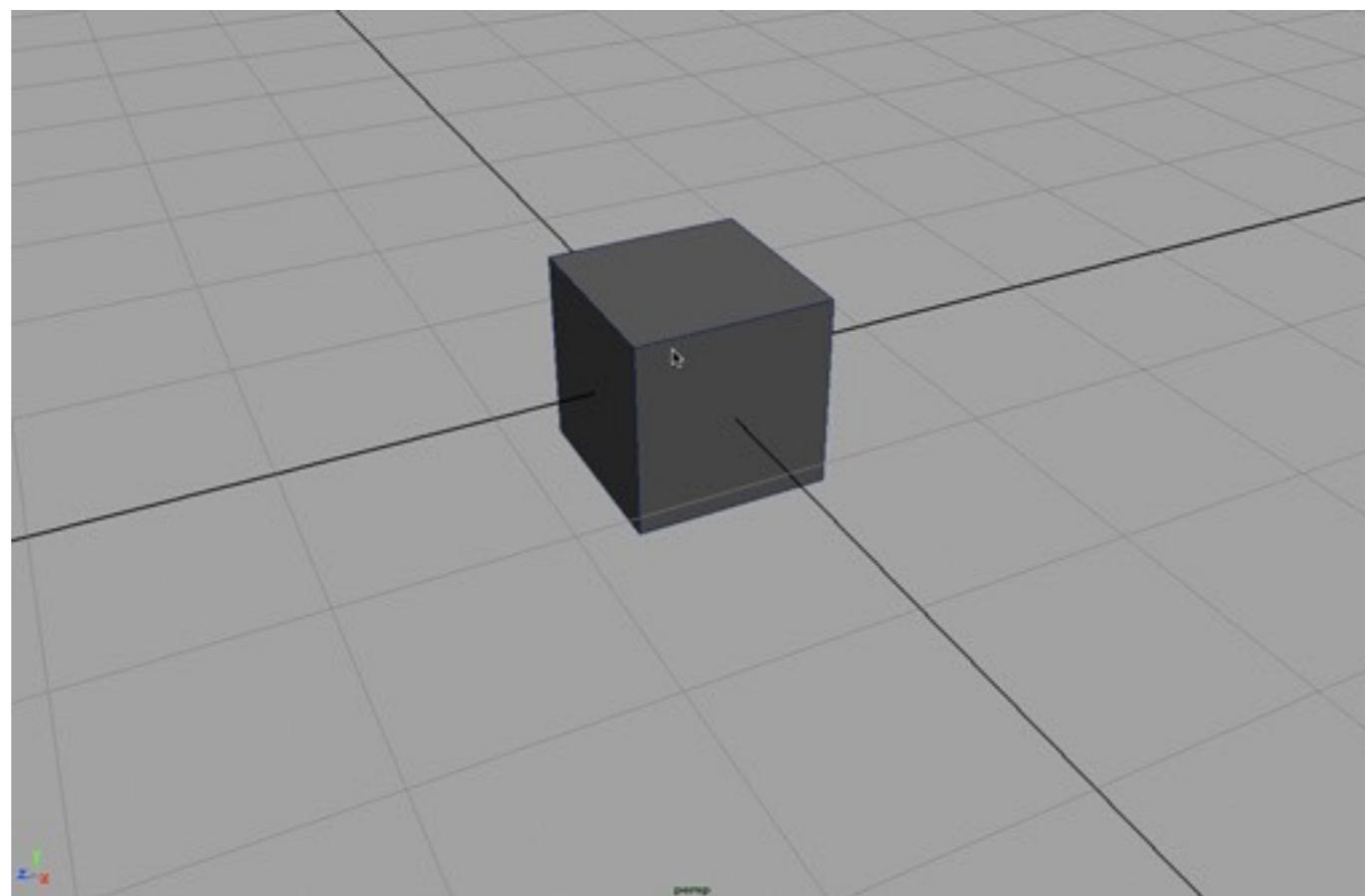
- **Nonuniform scale**

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



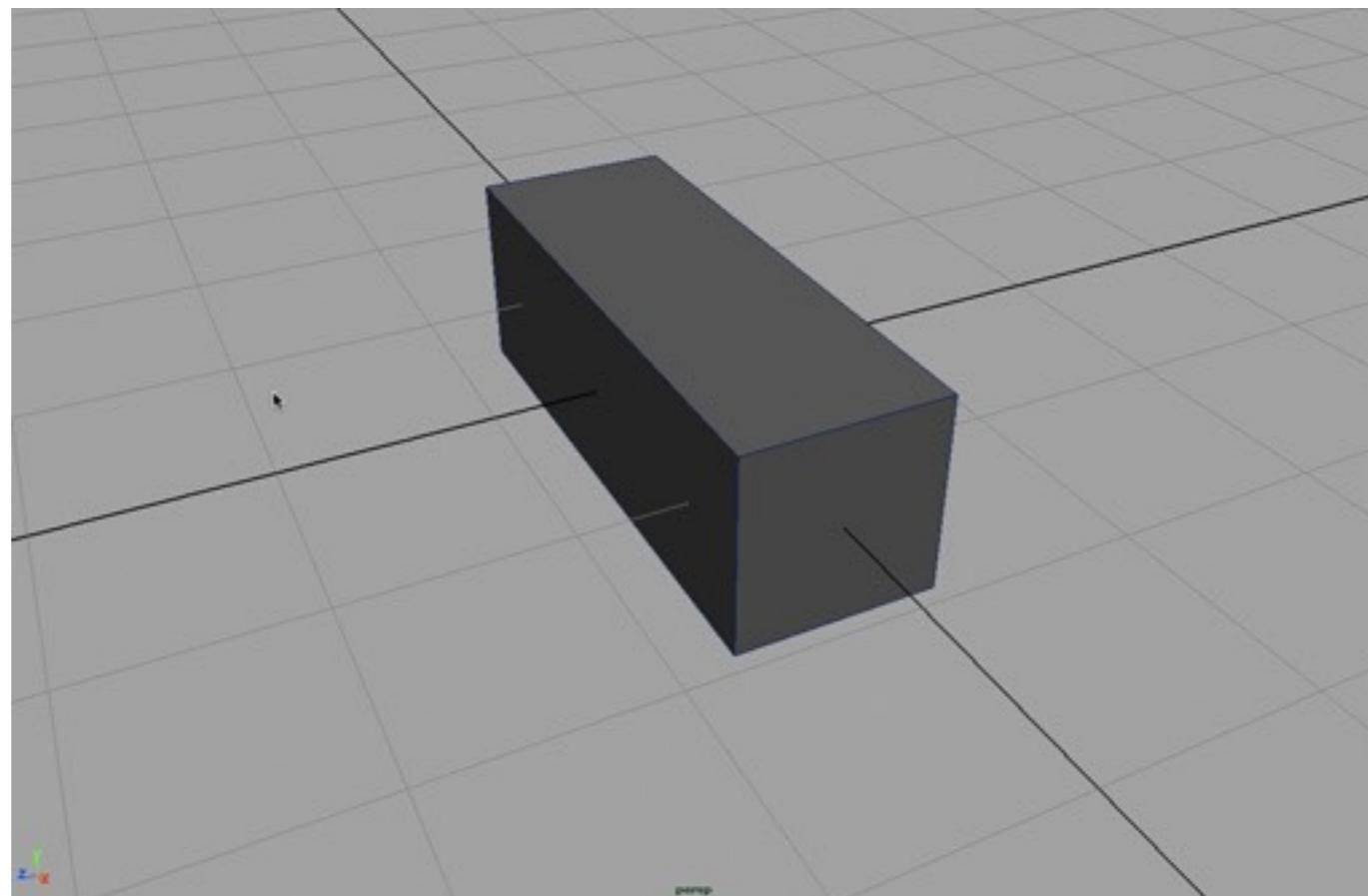
Scaling

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$



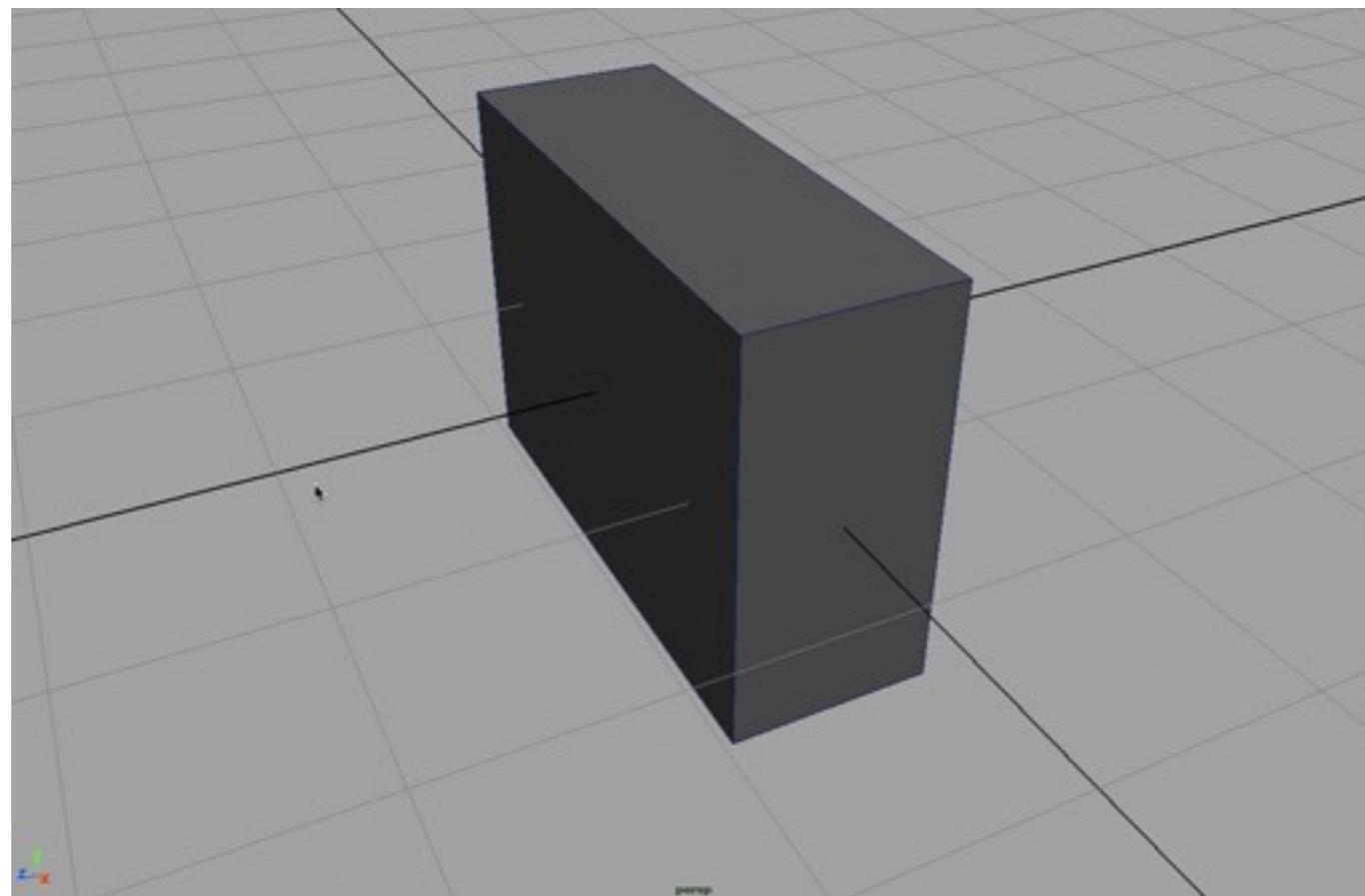
Scaling

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$



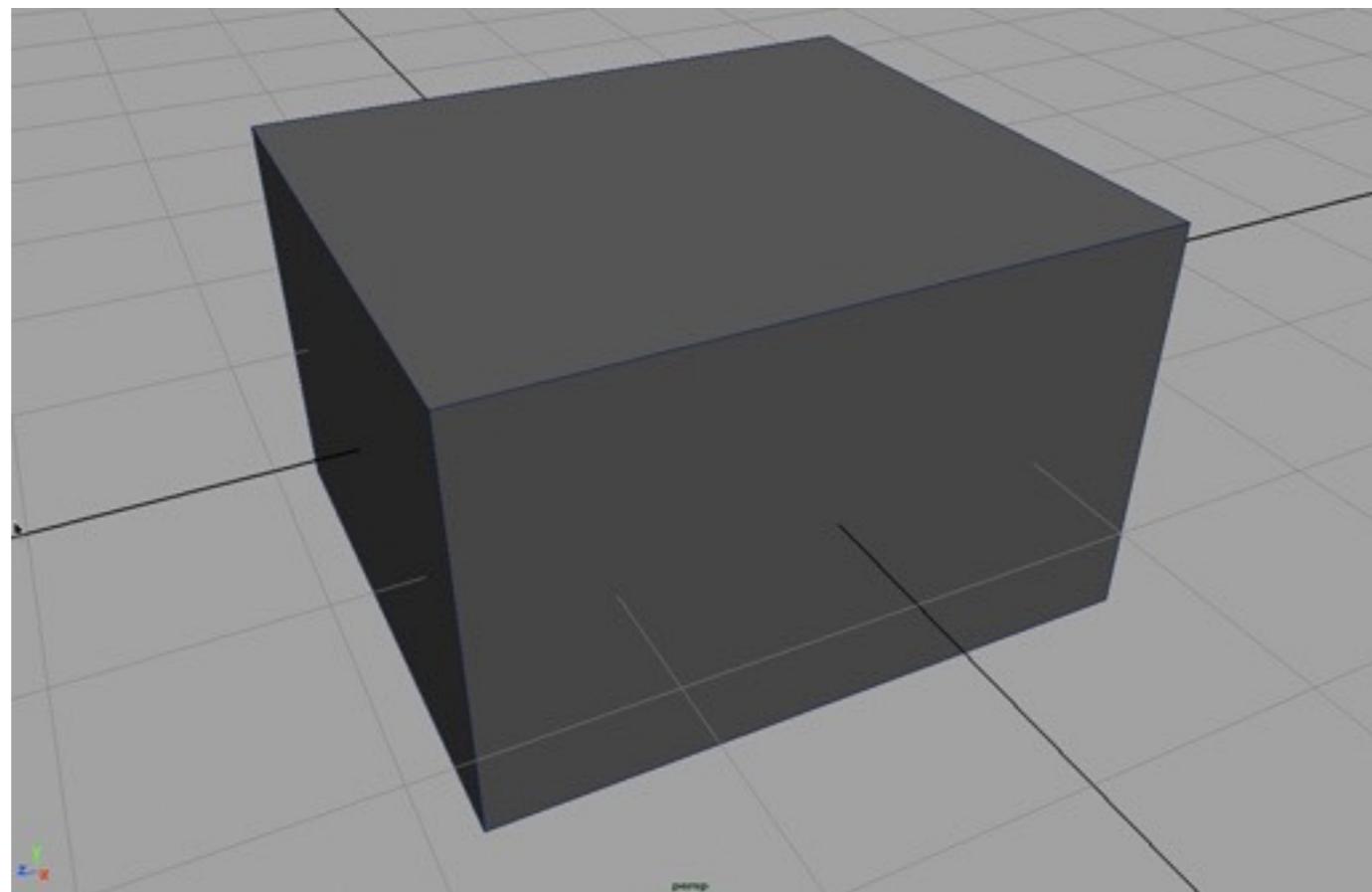
Scaling

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$



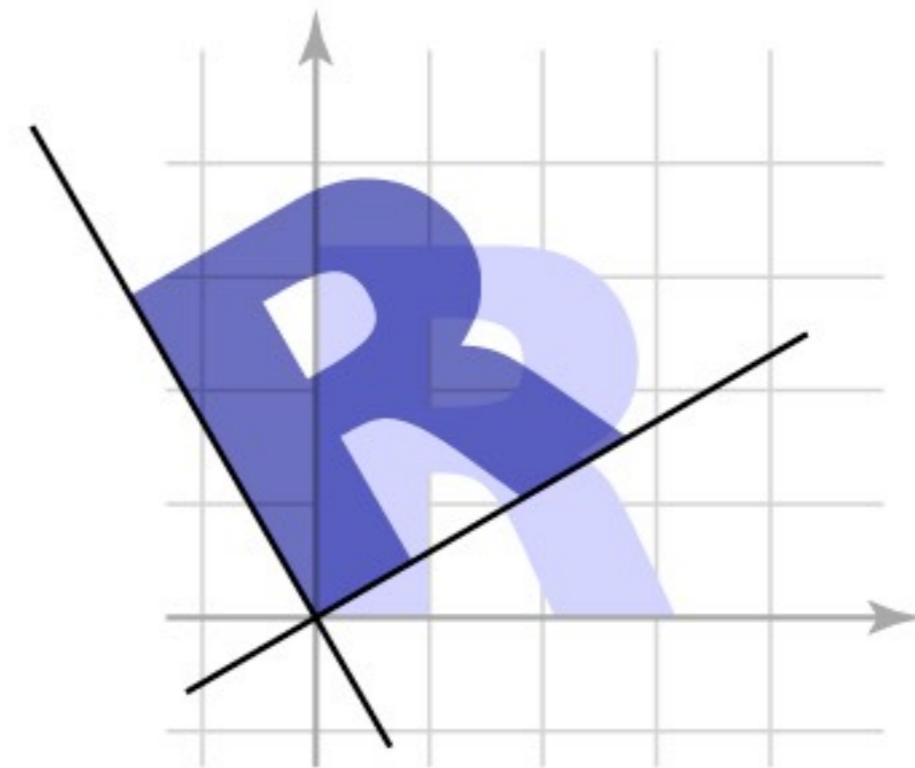
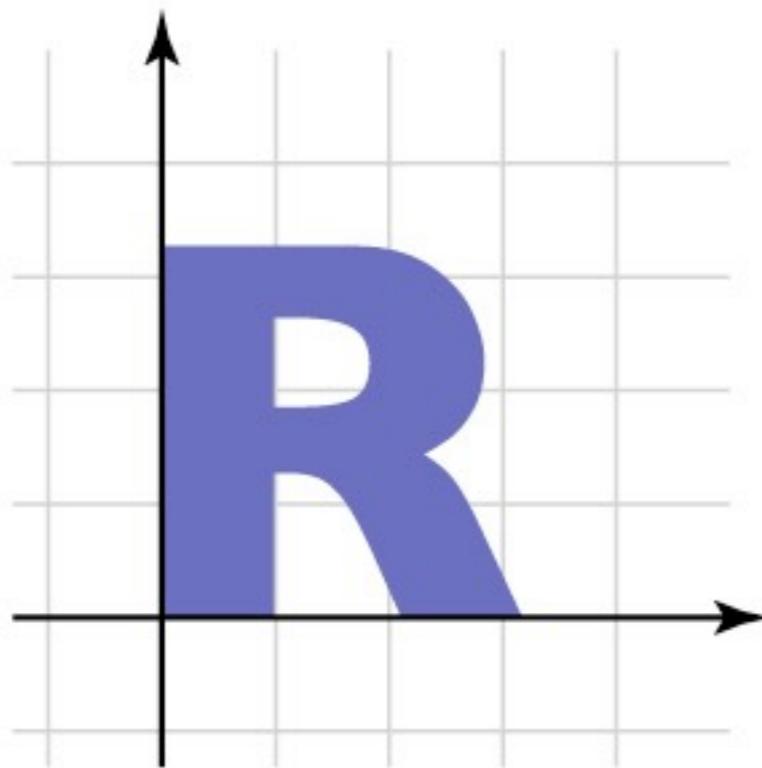
Scaling

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$



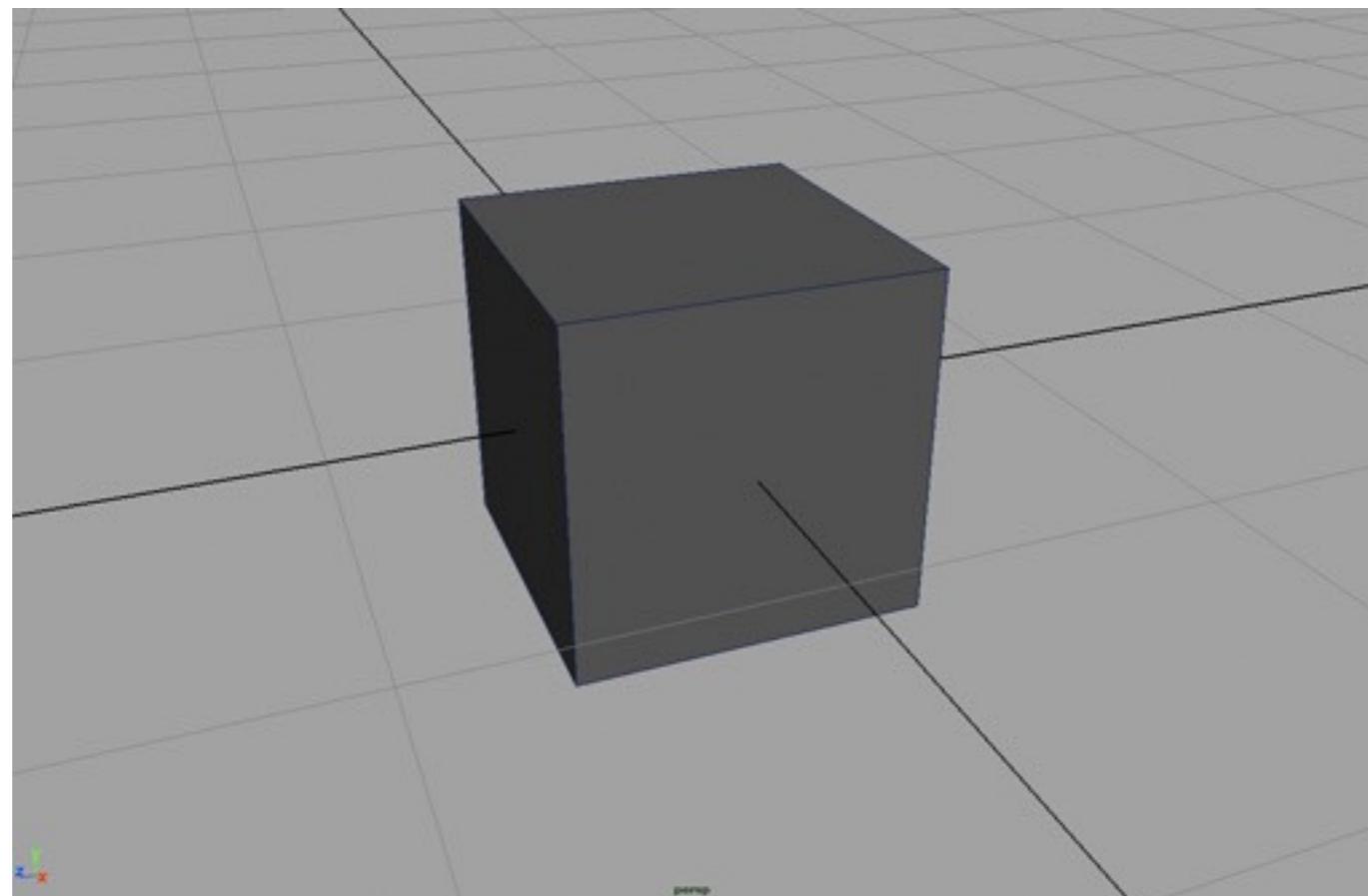
Affine transformation gallery

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



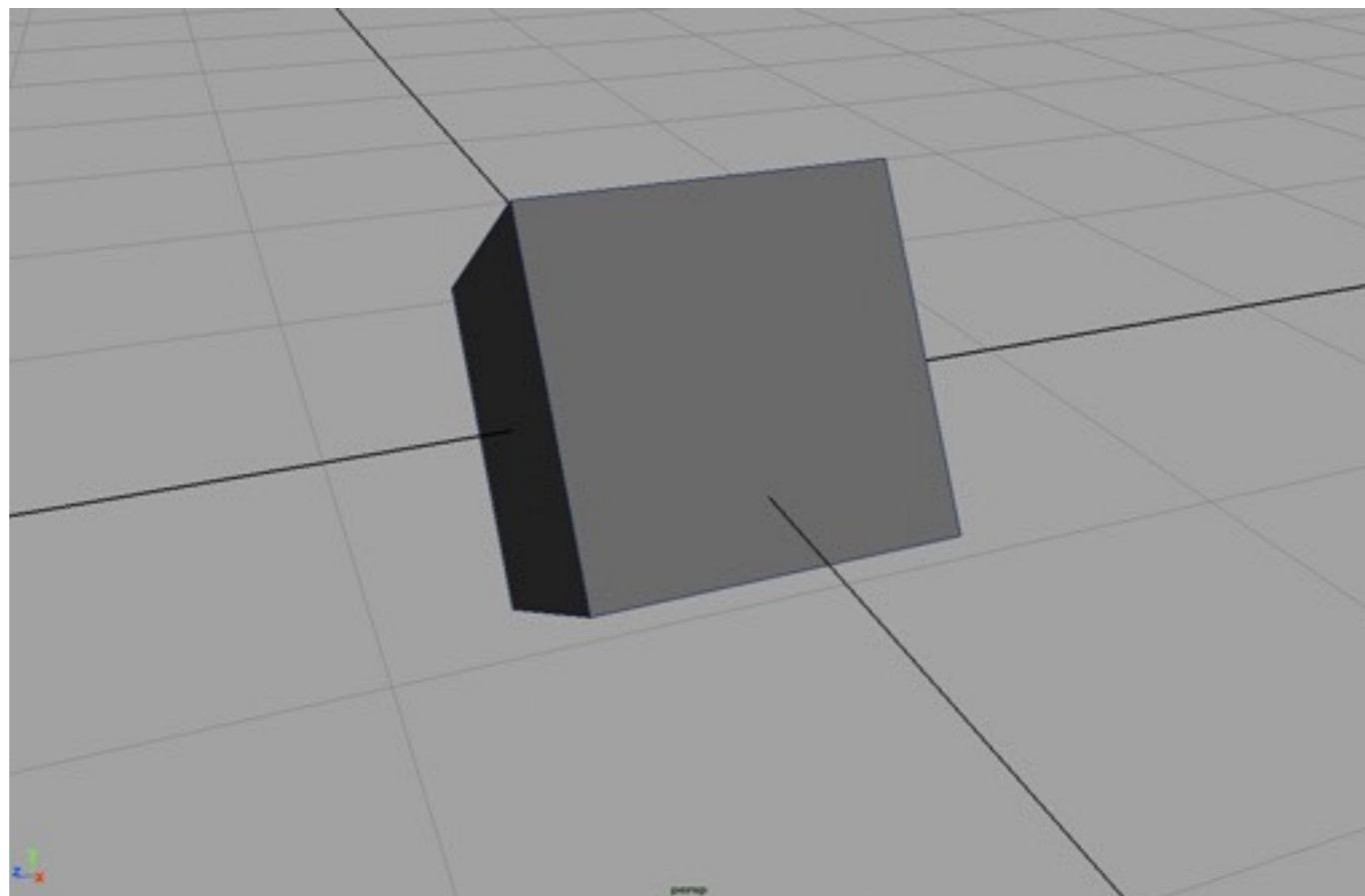
Rotation about **z** axis

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



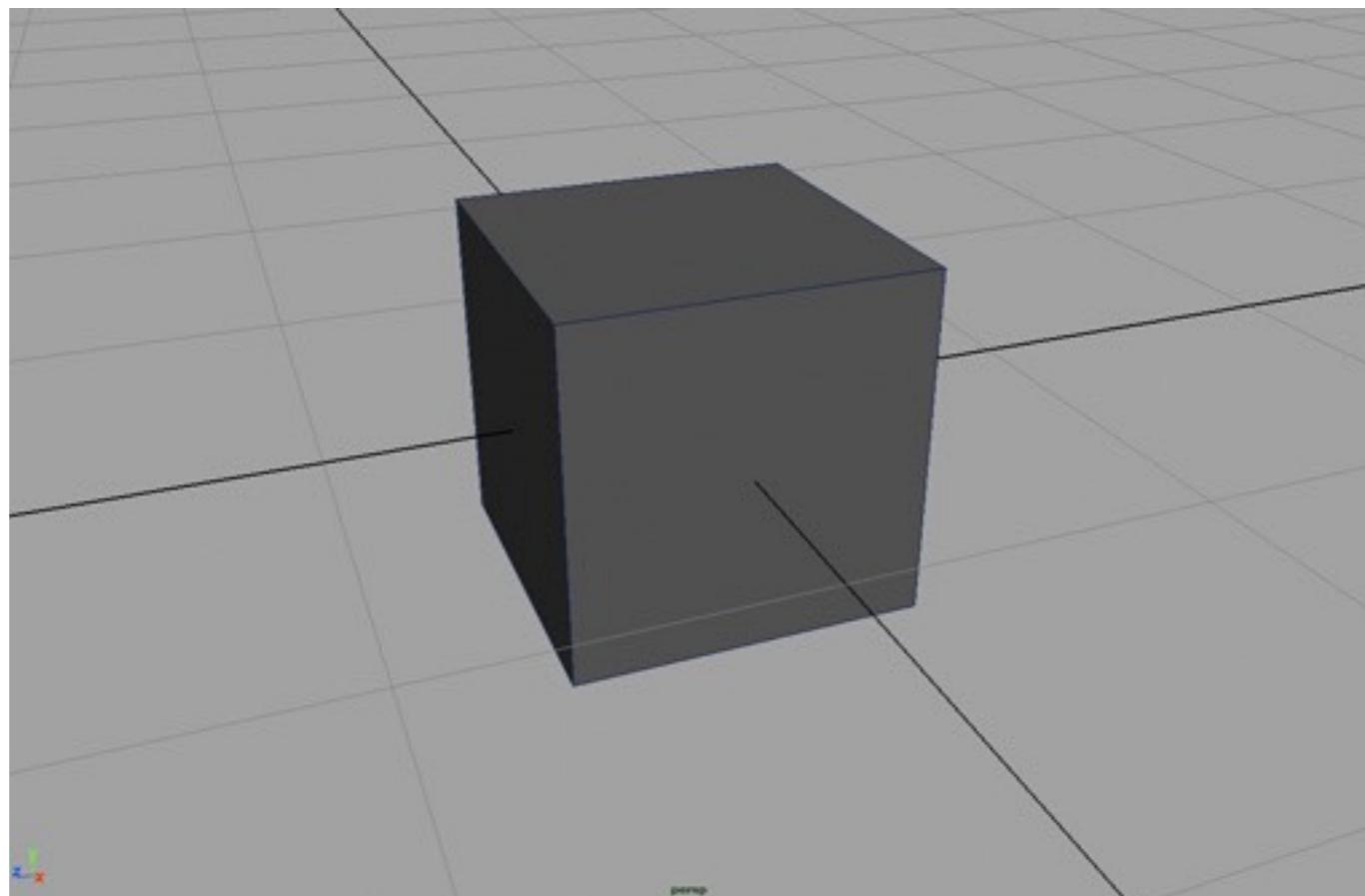
Rotation about **z** axis

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



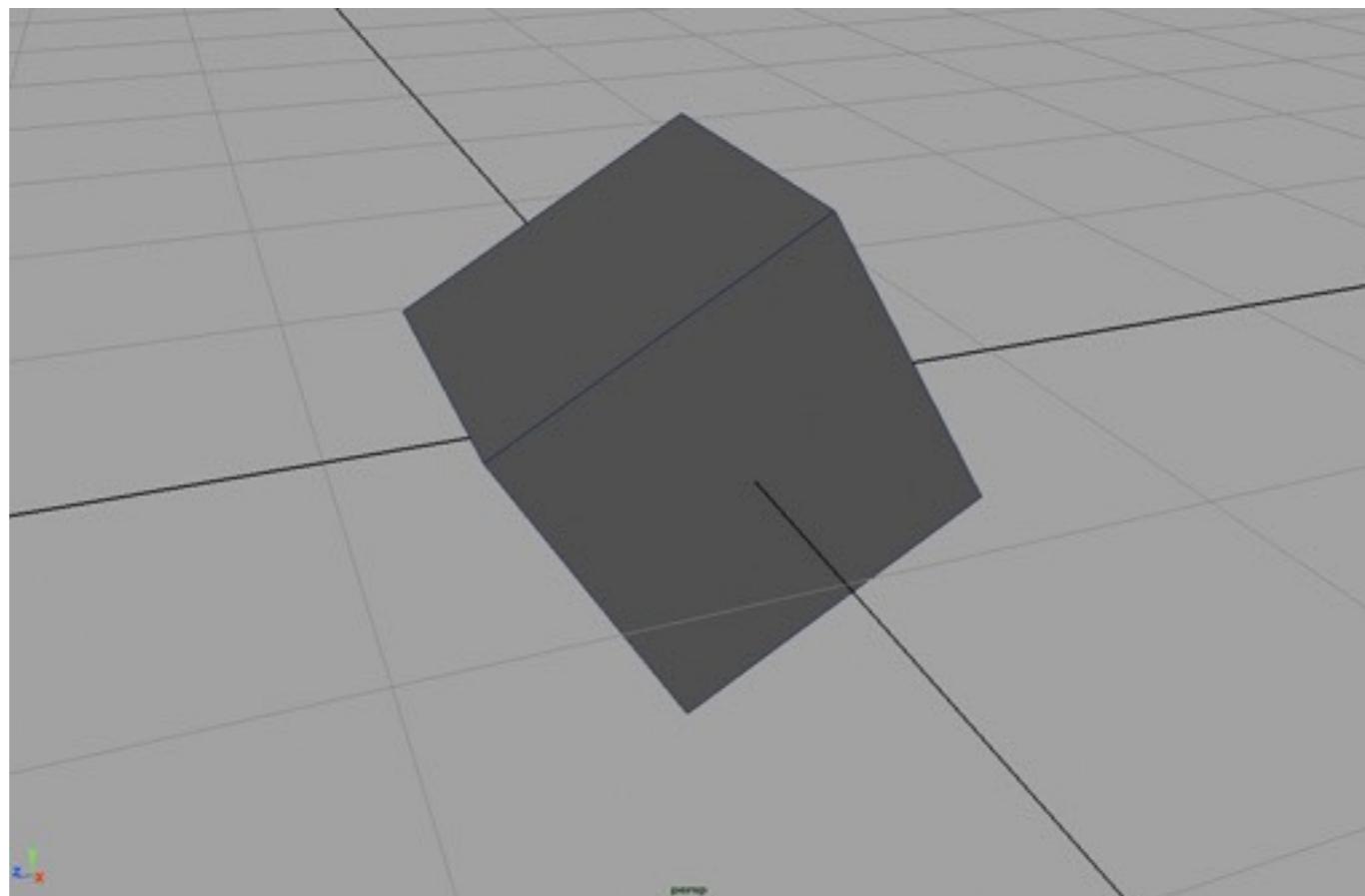
Rotation about **x** axis

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



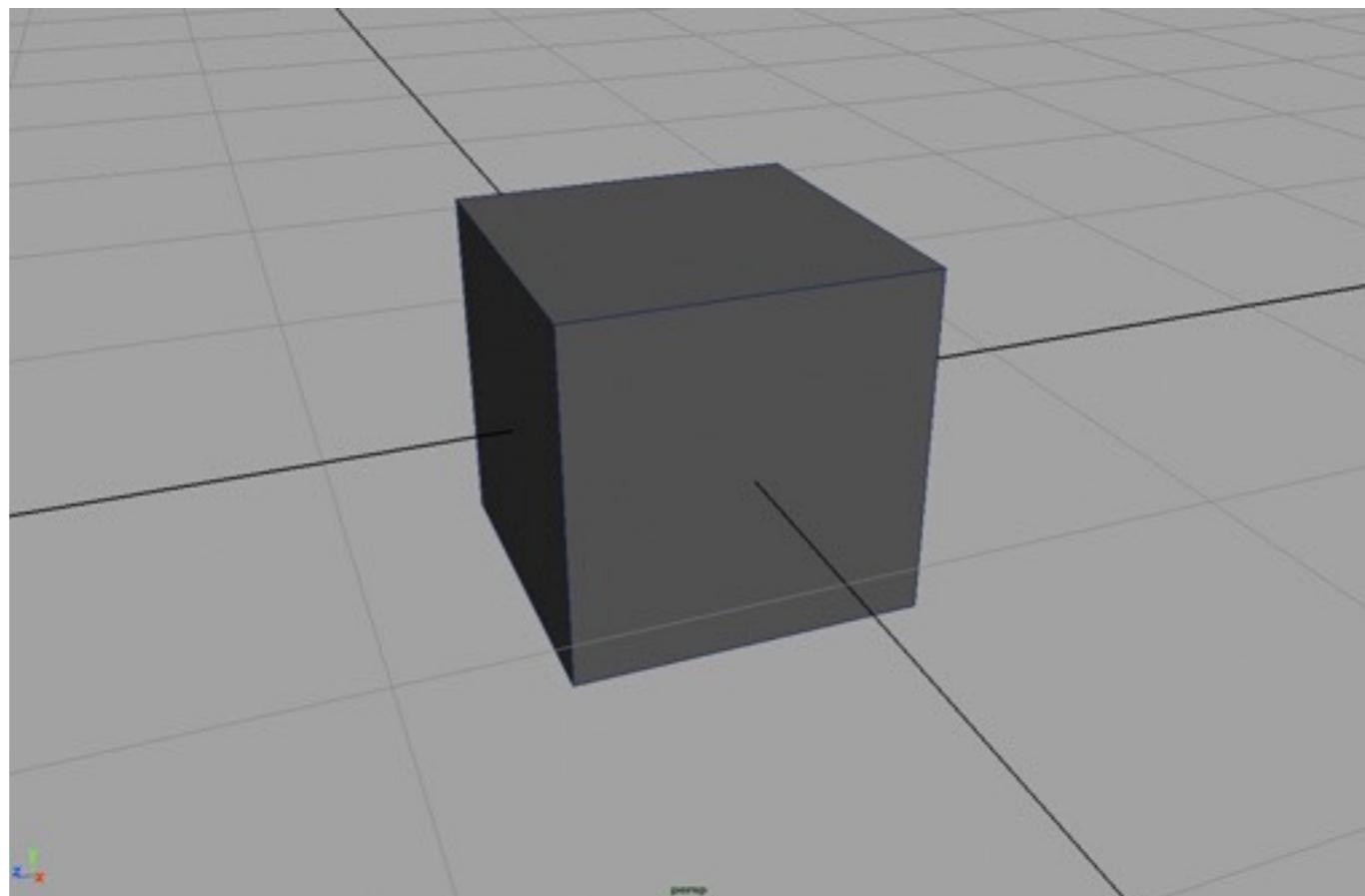
Rotation about **x** axis

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



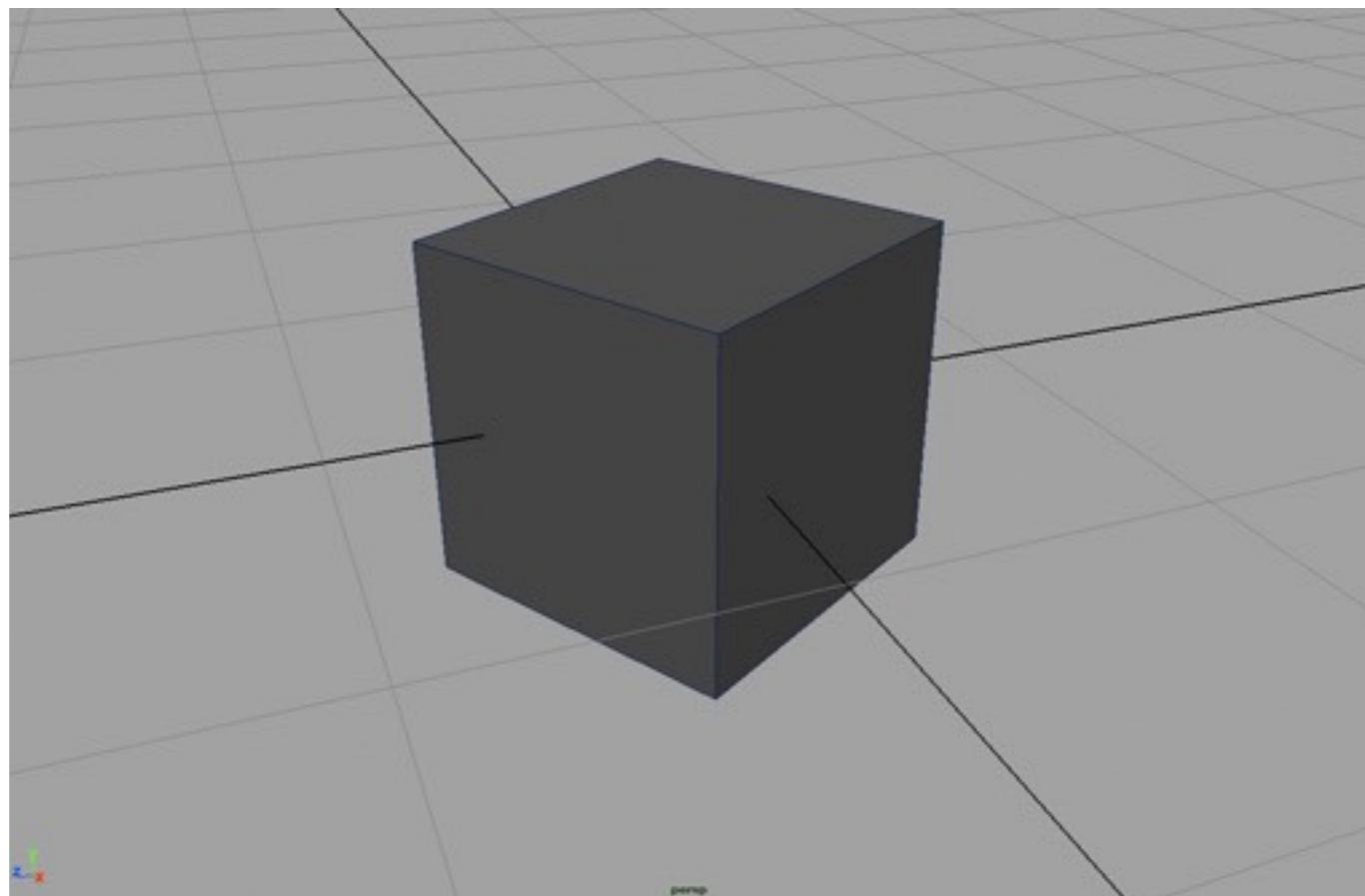
Rotation about **y** axis

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



Rotation about **y** axis

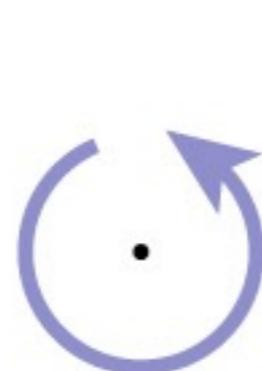
$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos \theta & 0 & \sin \theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin \theta & 0 & \cos \theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$



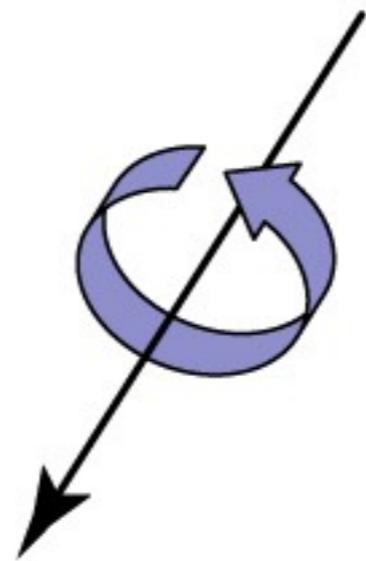
General Rotation Matrices

- **A rotation in 2D is around a point**
- **A rotation in 3D is around an axis**
 - so 3D rotation is w.r.t a line, not just a point
 - there are many more 3D rotations than 2D
 - a 3D space around a given point, not just 1D

convention: positive
rotation is CCW



2D

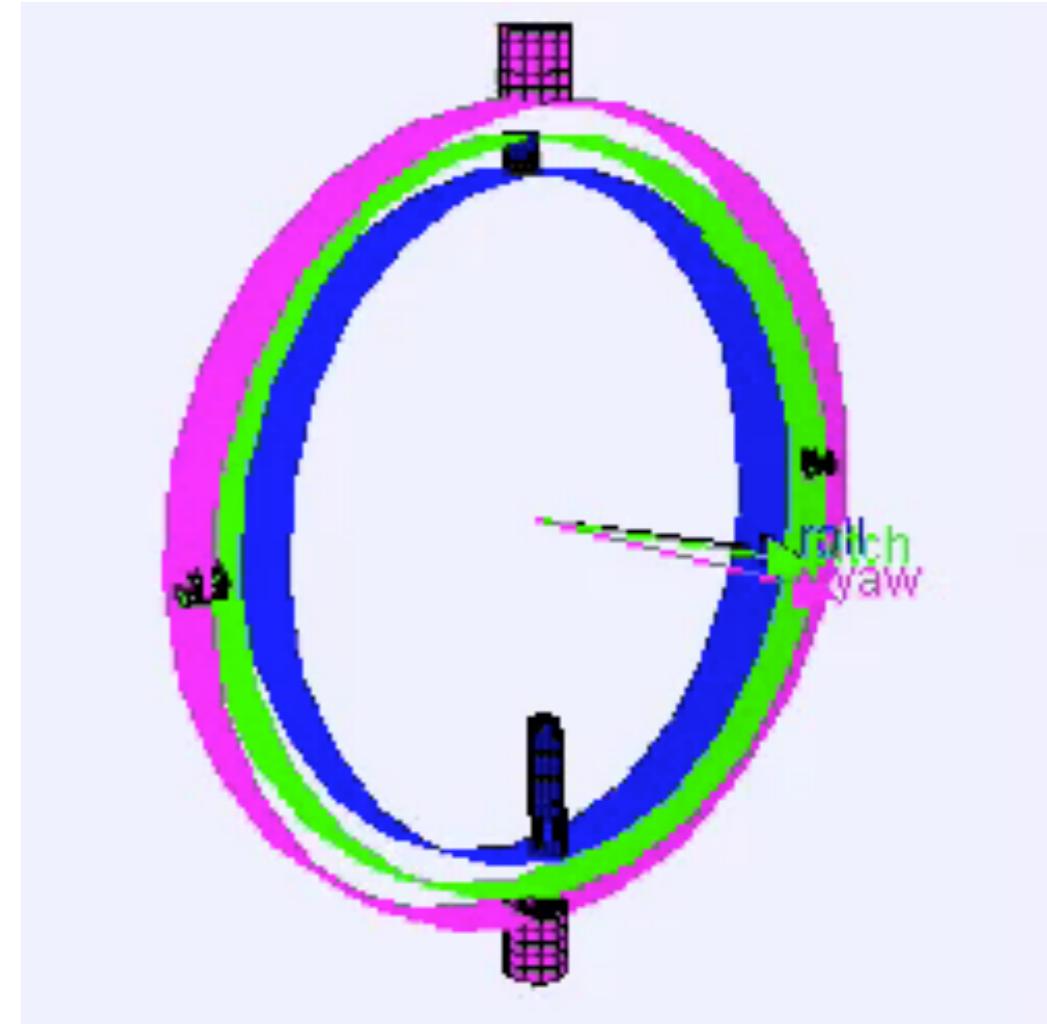


3D

convention: positive
rotation is CCW
when axis vector is
pointing at you

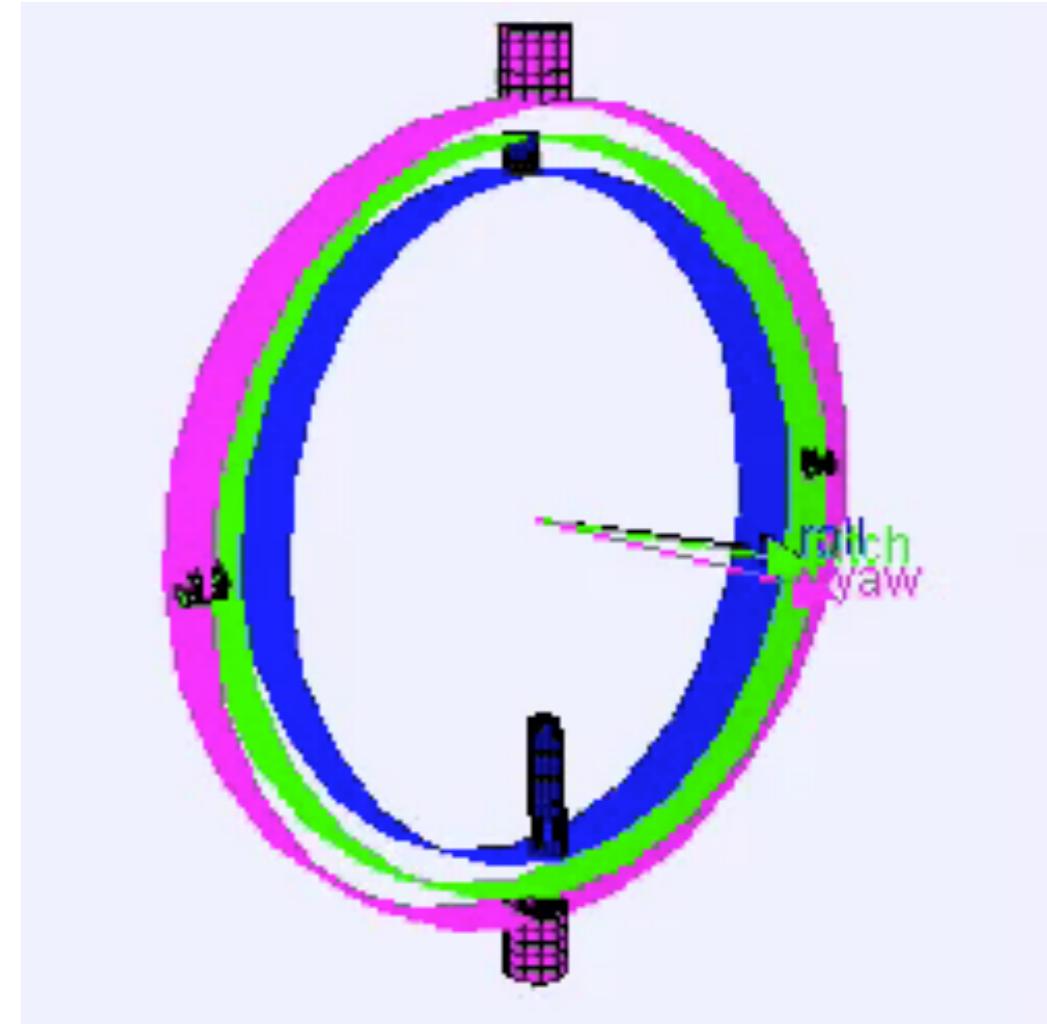
Euler angles

- An object can be oriented arbitrarily
- Euler angles: simply compose three coord. axis rotations
 - e.g. x, then y, then z: $R(\theta_x, \theta_y, \theta_z) = R_z(\theta_z)R_y(\theta_y)R_x(\theta_x)$
 - “heading, attitude, bank”
(common for airplanes)
 - “roll, pitch, yaw”
(common for vehicles)
 - “pan, tilt, roll”
(common for cameras)

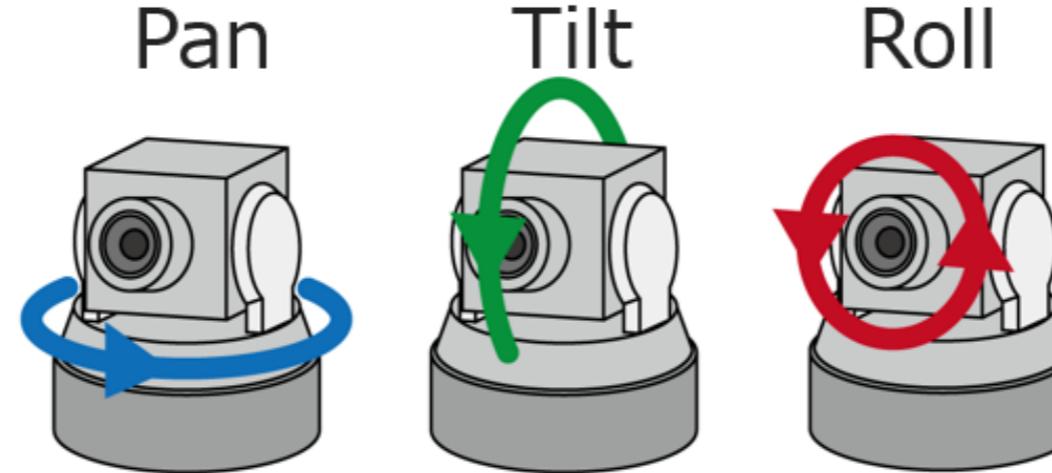


Euler angles

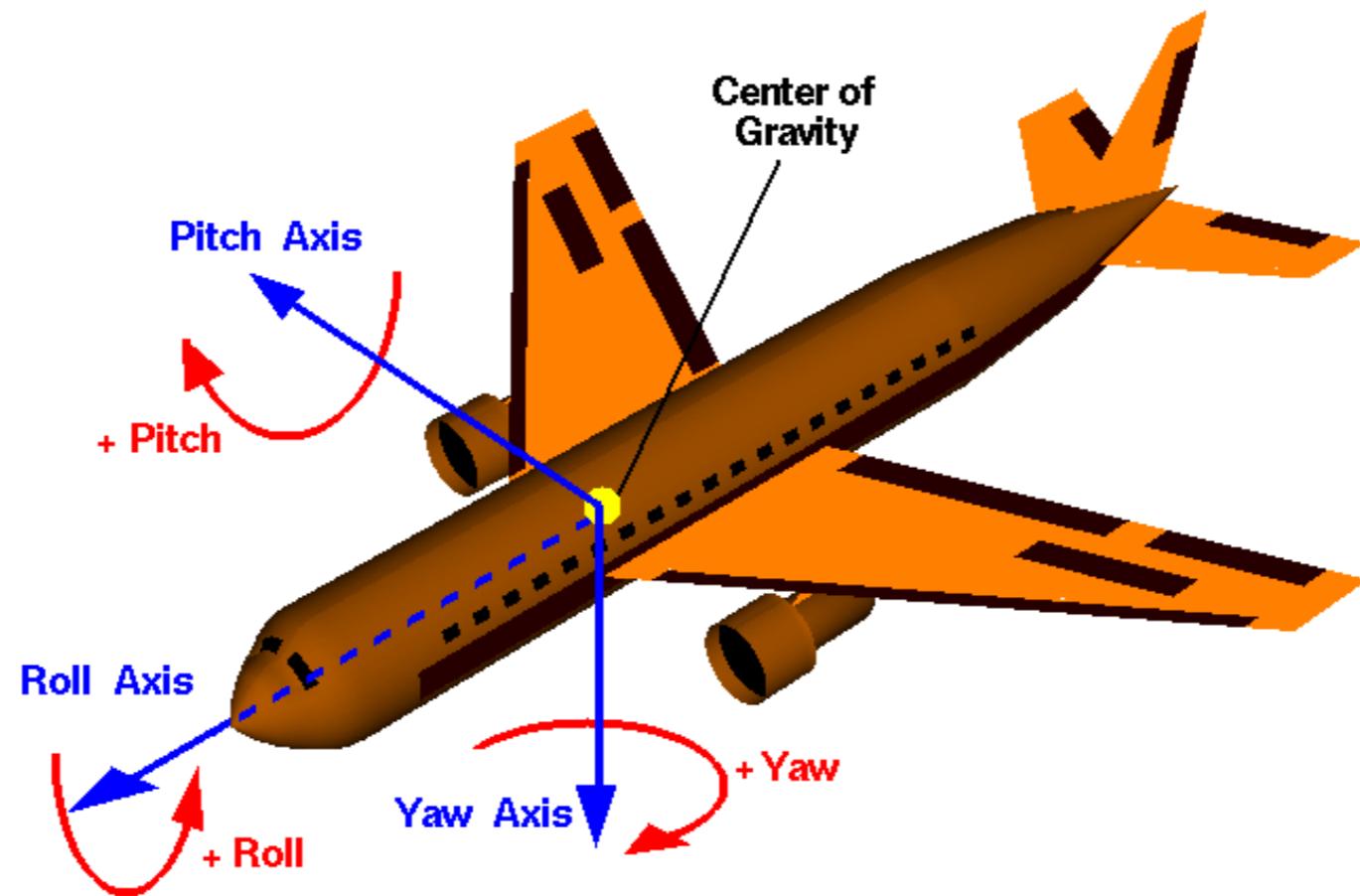
- An object can be oriented arbitrarily
- Euler angles: simply compose three coord. axis rotations
 - e.g. x, then y, then z: $R(\theta_x, \theta_y, \theta_z) = R_z(\theta_z)R_y(\theta_y)R_x(\theta_x)$
 - “heading, attitude, bank”
(common for airplanes)
 - “roll, pitch, yaw”
(common for vehicles)
 - “pan, tilt, roll”
(common for cameras)



Euler angles in applications



Ishikawa Watanabe Laboratory



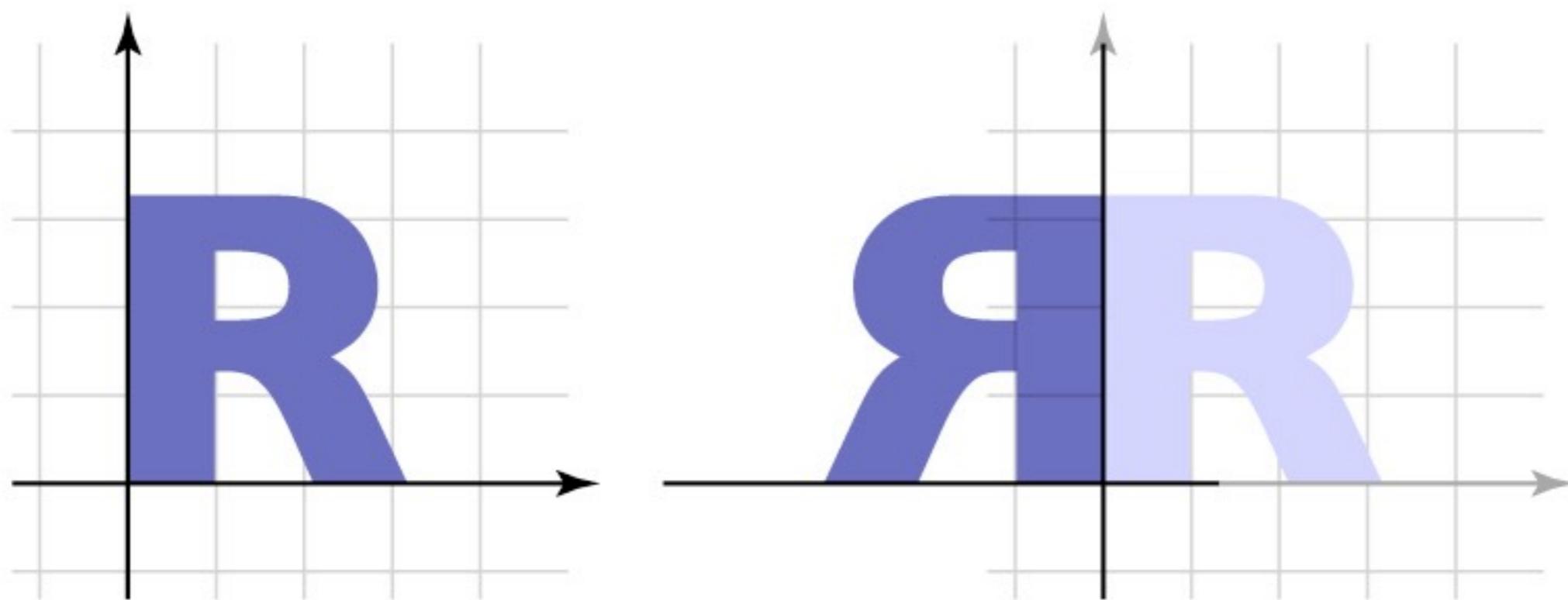
Wikipedia

Affine transformation gallery

- **Reflection**

- can consider it a special case
of nonuniform scale

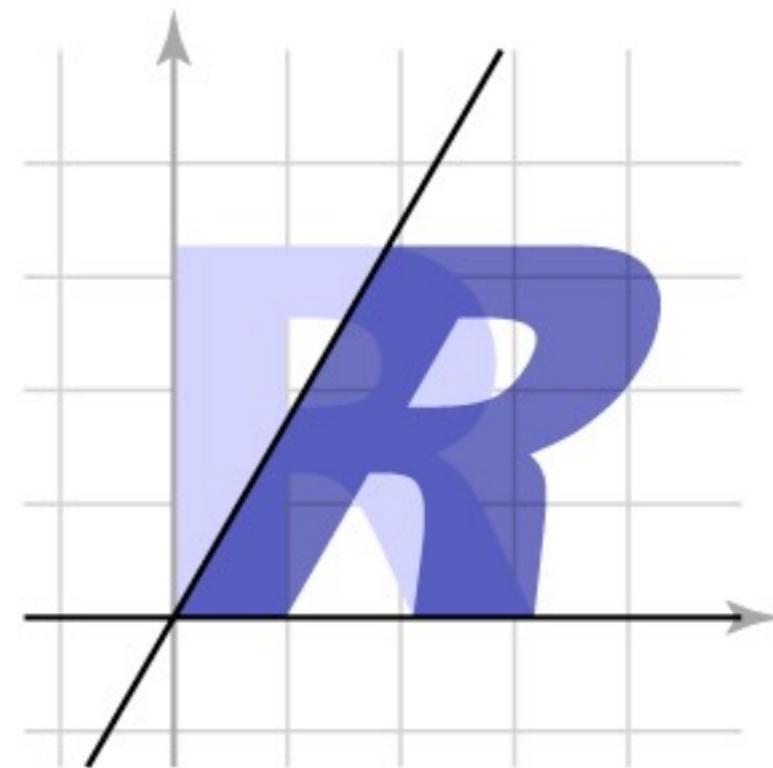
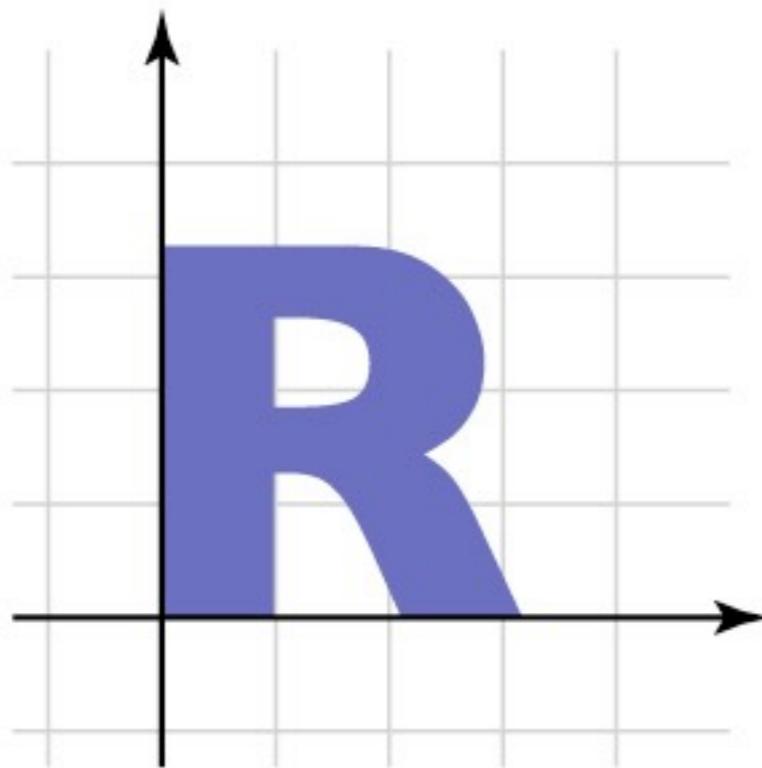
$$\begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$



Affine transformation gallery

- **Shear**
 - can still also build these from rotations and nonuniform scales

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



Transforming geometry

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

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

- How to do this depends on how the geometry is described:
 - parametric (explicit) generally requires computing T
 - implicit generally requires computing the inverse of T

Implicit representations

- **Equation to tell whether we are on the curve**

$$\{\mathbf{v} \mid f(\mathbf{v}) = 0\}$$

- **Example: line (orthogonal to \mathbf{u} , distance k from $\mathbf{0}$)**

$$\{\mathbf{v} \mid \mathbf{v} \cdot \mathbf{u} + k = 0\} \quad (\mathbf{u} \text{ is a unit vector})$$

- **Example: circle (center \mathbf{p} , radius r)**

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

- **Always define boundary of region**

- (if f is continuous)

Examples of implicit surfaces

- **Blobby models**

- start with a smooth function of distance from a point \mathbf{p} :

$$f(\mathbf{x}) = g(\|\mathbf{x} - \mathbf{p}\|)$$

$$g(0) = 1$$

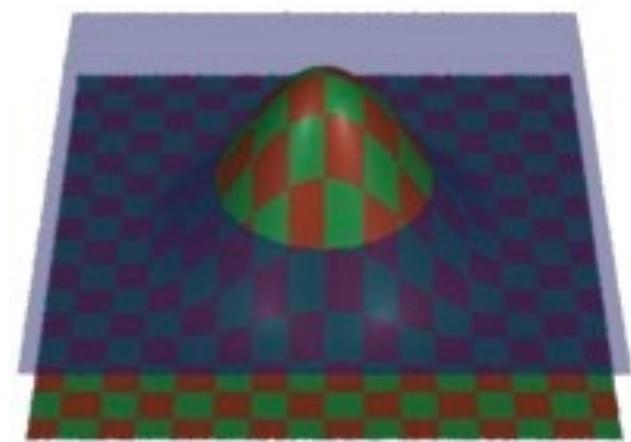
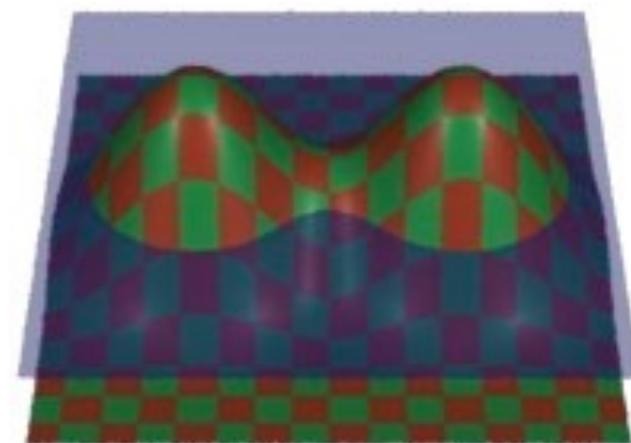
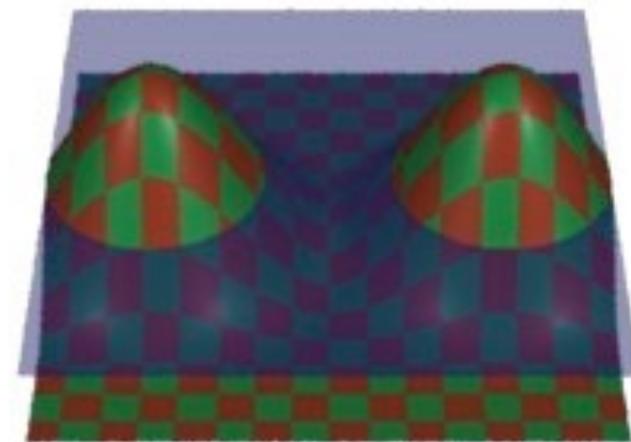
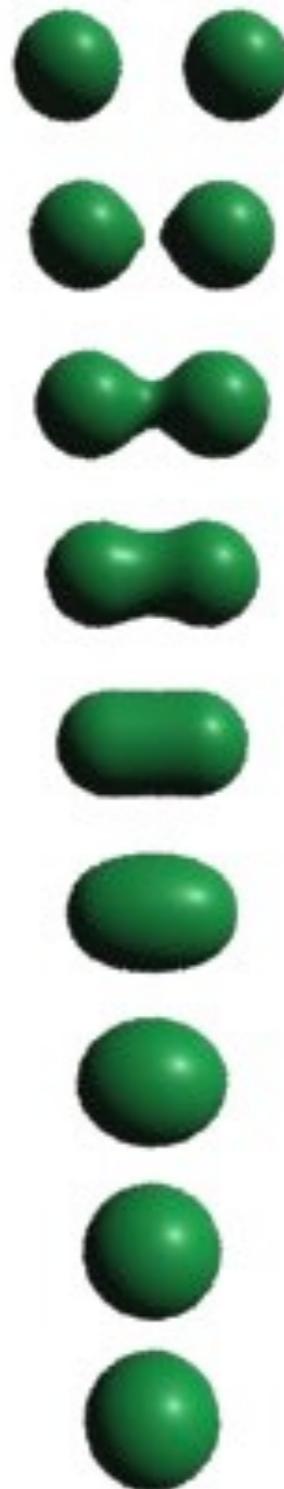
$$r > 1 \implies g(r) = 0$$

- an implicit surface

$$\{\mathbf{x} \mid f(\mathbf{x}) = 0\}$$

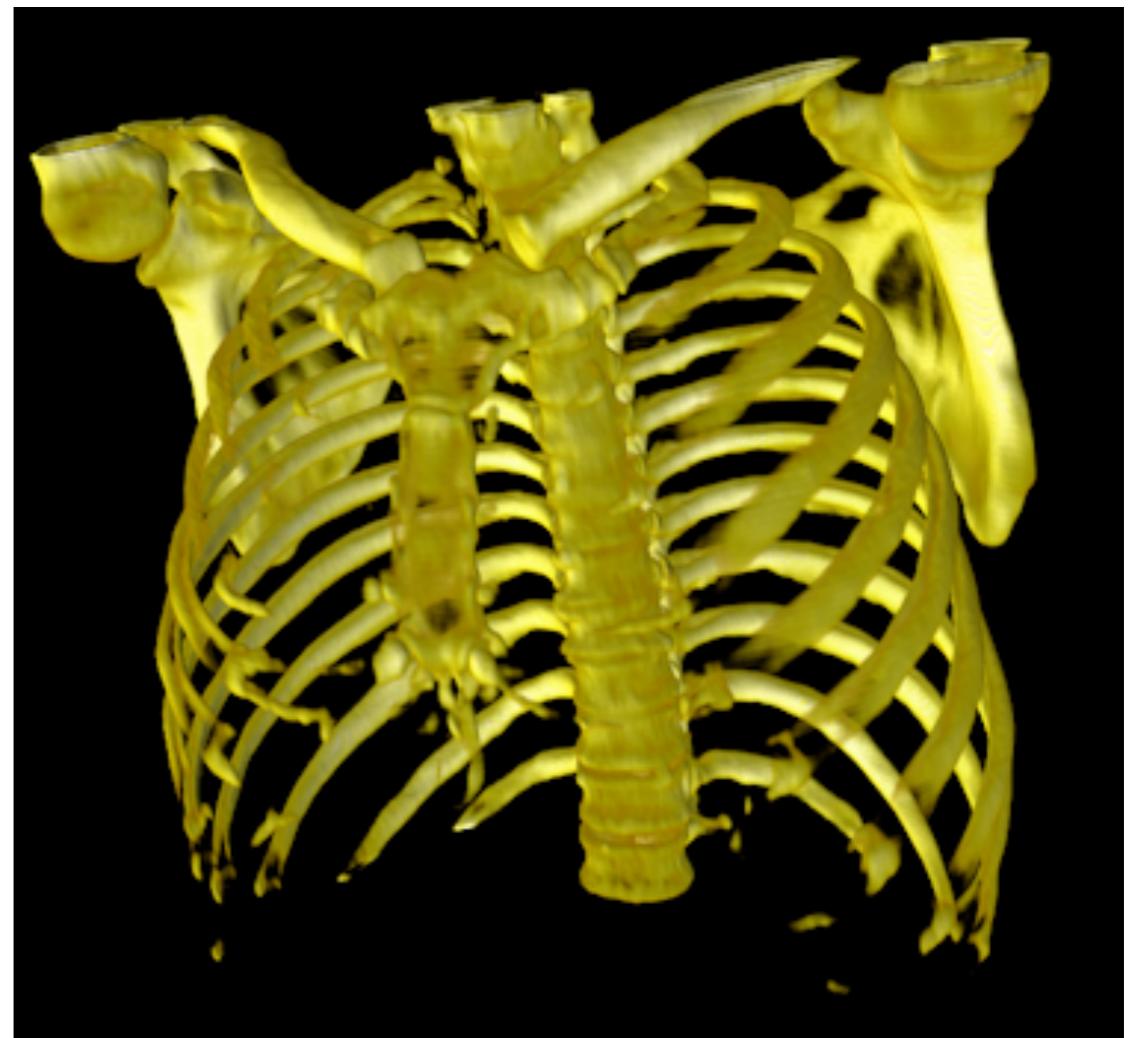
is a sphere, but if we add together several such primitives we can get interesting shapes

$$F(\mathbf{x}) = \sum_i f(\mathbf{x}_i)$$



Examples of implicit surfaces

- **Isosurfaces of volumetric data**
 - volumetric data gives you a function of 3 spatial variables, such as mass density.
 - isosurfaces of this density field generally correspond to surfaces of parts of the scanned object.
 - the function is provided via a regular 3D grid of samples



[santesoft.com]

Explicit representations

- **Also called parametric**
- **Equation to map domain into plane**

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

- **Example: line (containing \mathbf{p} , parallel to \mathbf{u})**

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

- **Example: circle (center \mathbf{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”**

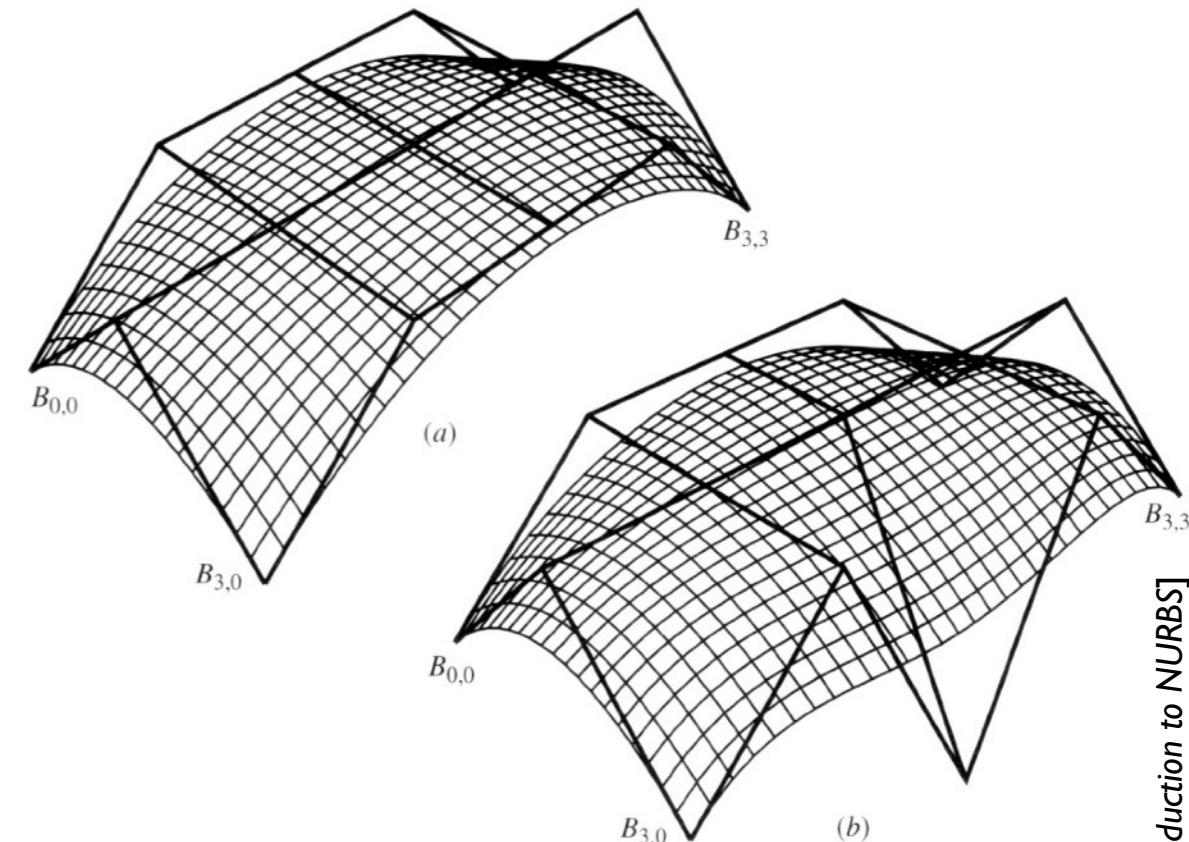
Examples of explicit representations

- **Splines**
 - univariate polynomials mapping from 1D to 2D or 3D

$$\mathbf{x}(t) = (p(t), q(t))$$

- **Spline surfaces**
 - bivariate polynomials mapping from 2D to 3D

$$\mathbf{x}(s, t) = (p(s, t), q(s, t), r(s, t))$$



- **Triangle meshes**
 - each triangle is a linear mapping from 2D to 3D

Transforming geometry

- **Move a subset of space using a mapping from the space 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:**

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

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

Properties of Matrices

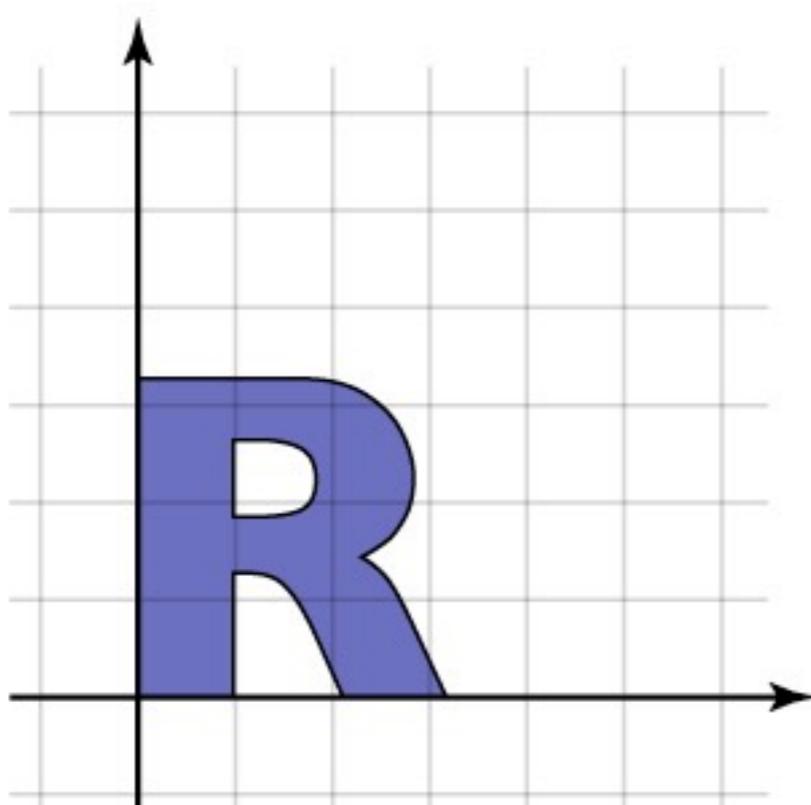
- **Translations: linear part is the identity**
- **Scales: linear part is diagonal**
- **Rotations: linear part is orthogonal**
 - columns of R are mutually orthonormal: $R^T R = I$
 - rows of R are also mutually orthonormal: $RR^T = I$
 - also, determinant of R is 1.0: $\det R = 1$
(distinguishes rotations from reflections)

General affine transformations

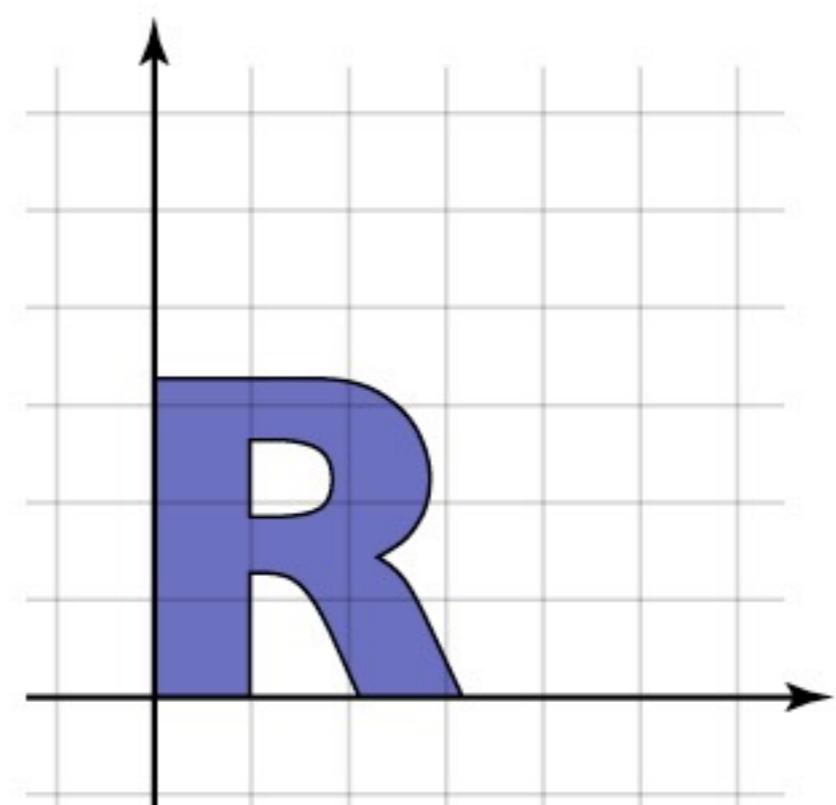
- **The previous slides showed “canonical” examples of the types of affine transformations**
- **Generally, transformations contain elements of multiple types**
 - often define them as products of canonical transforms
 - sometimes work with their properties more directly

Composite affine transformations

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



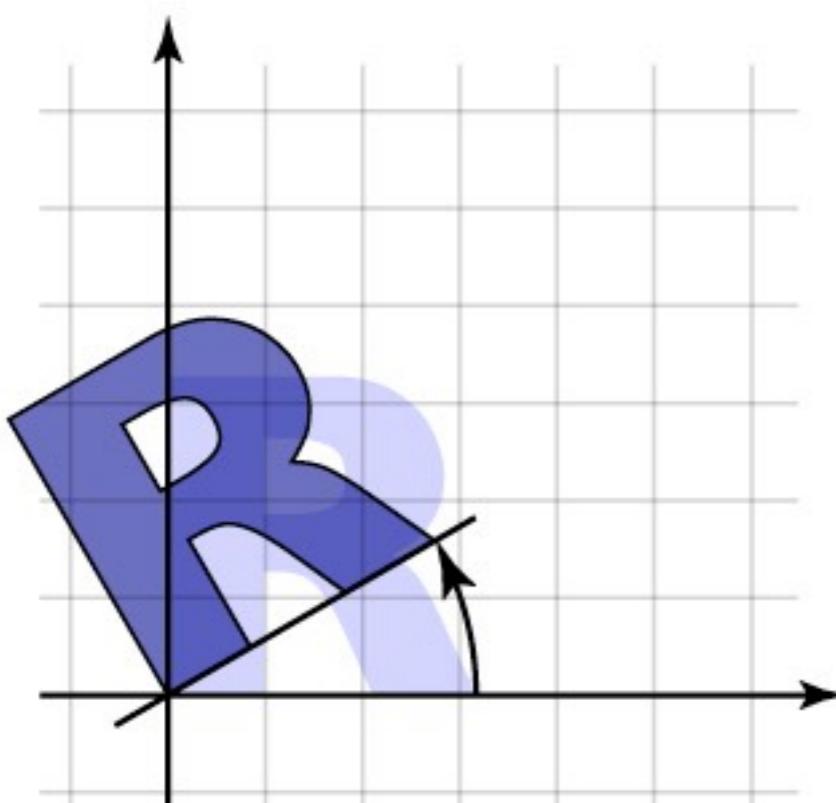
rotate, then translate



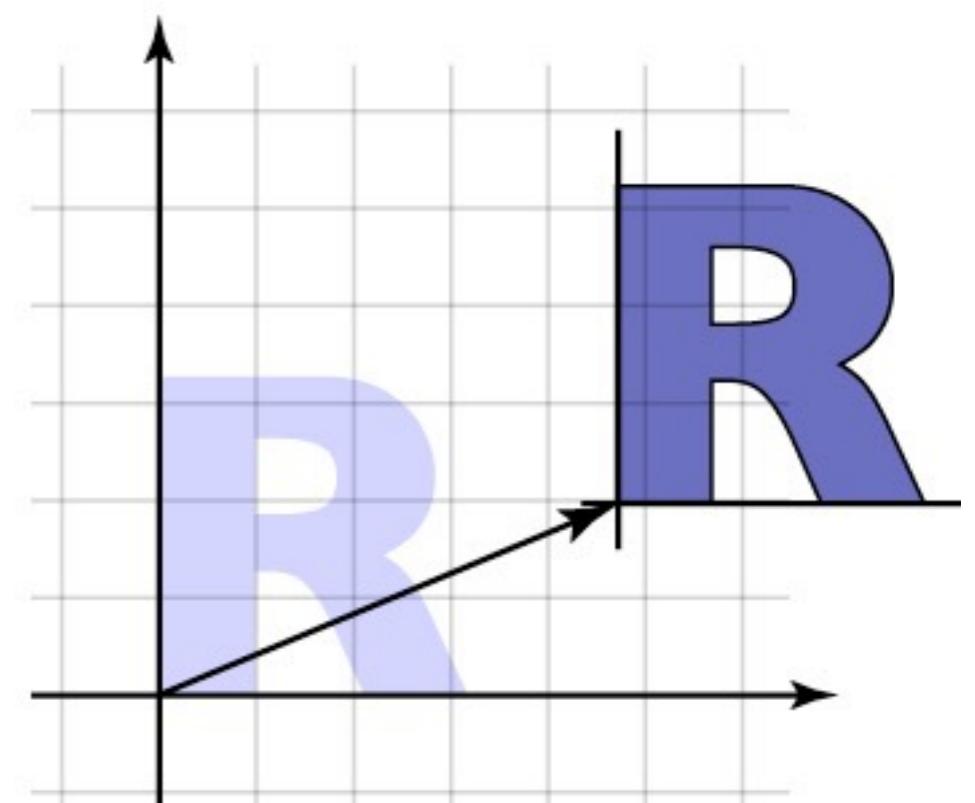
translate, then rotate

Composite affine transformations

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



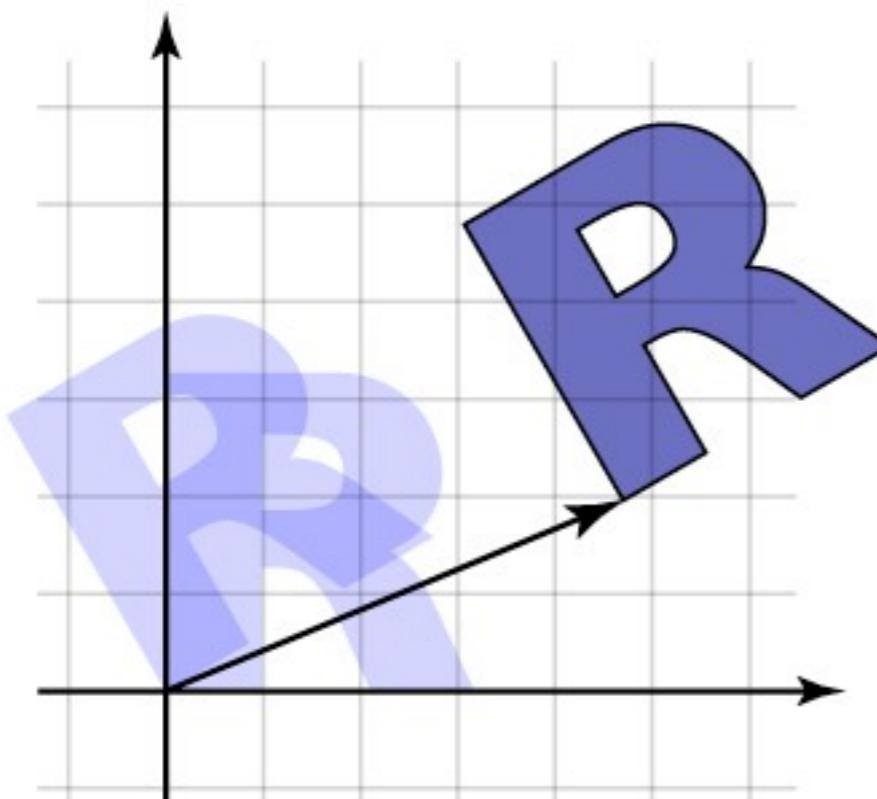
rotate, then translate



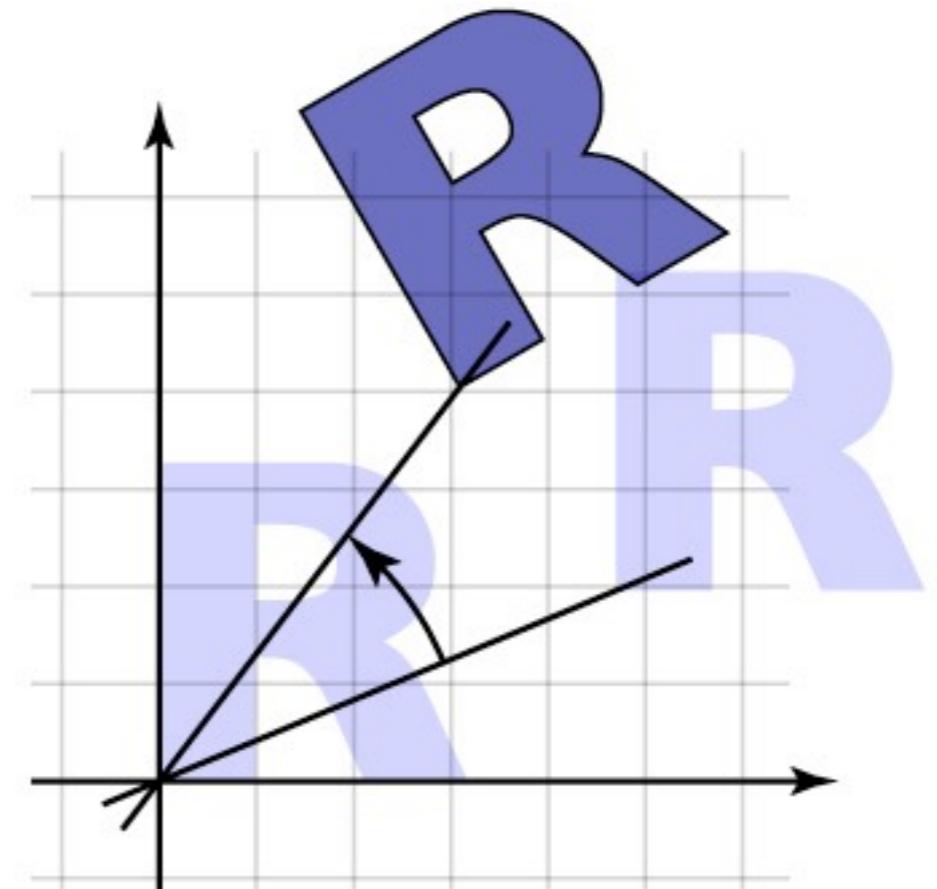
translate, then rotate

Composite affine transformations

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



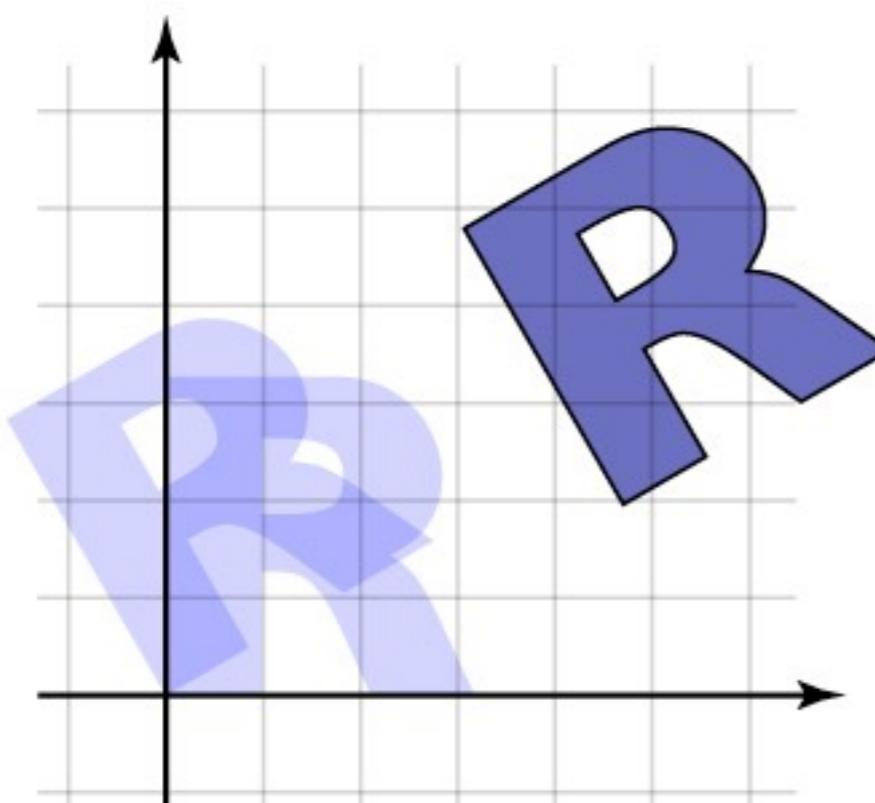
rotate, then translate



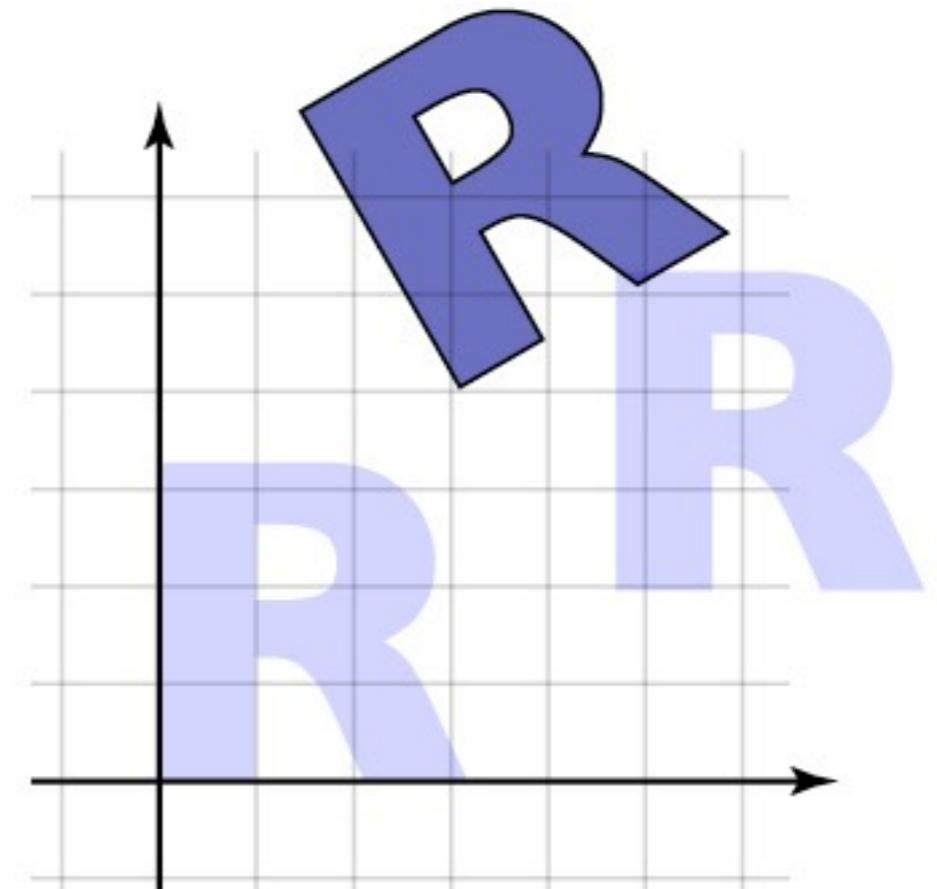
translate, then rotate

Composite affine transformations

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



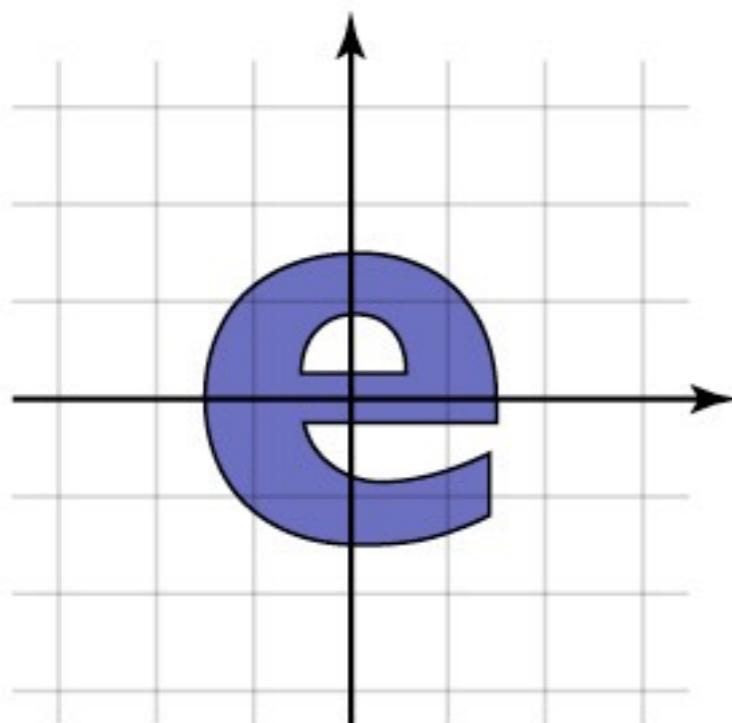
rotate, then translate



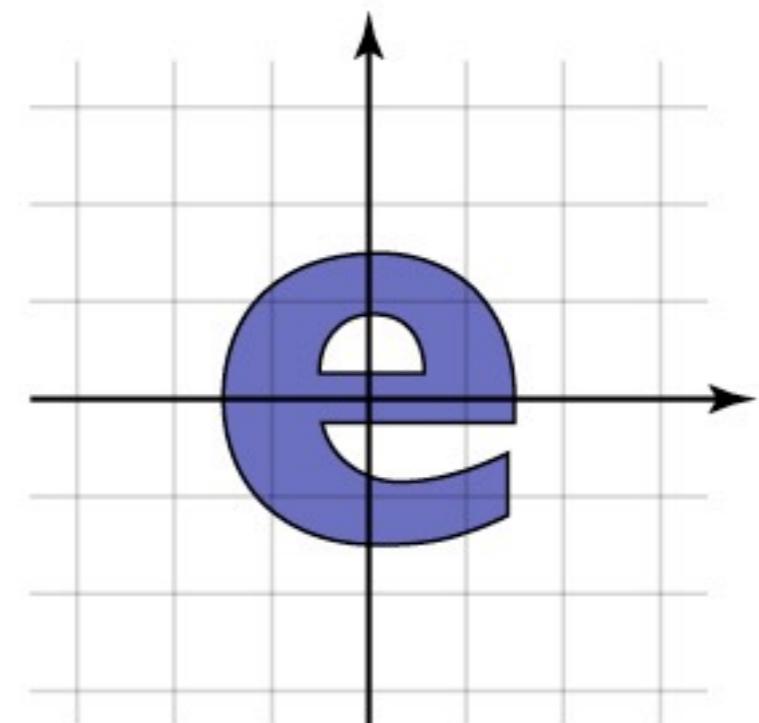
translate, then rotate

Composite affine transformations

- **Another example**



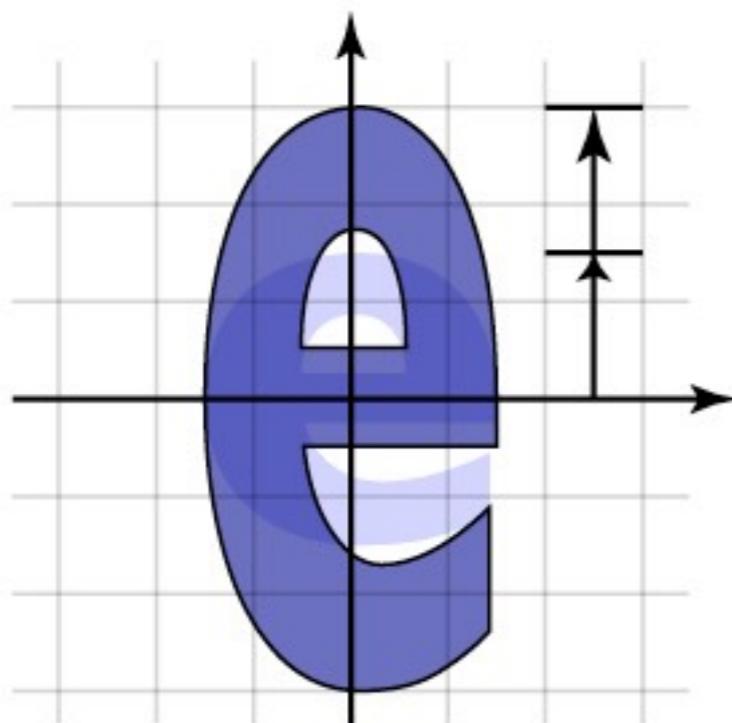
scale, then rotate



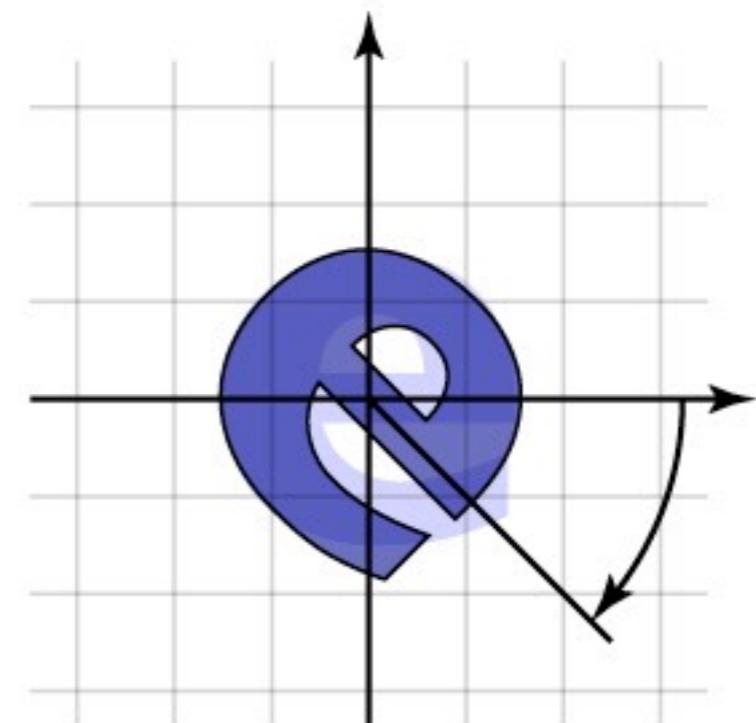
rotate, then scale

Composite affine transformations

- **Another example**



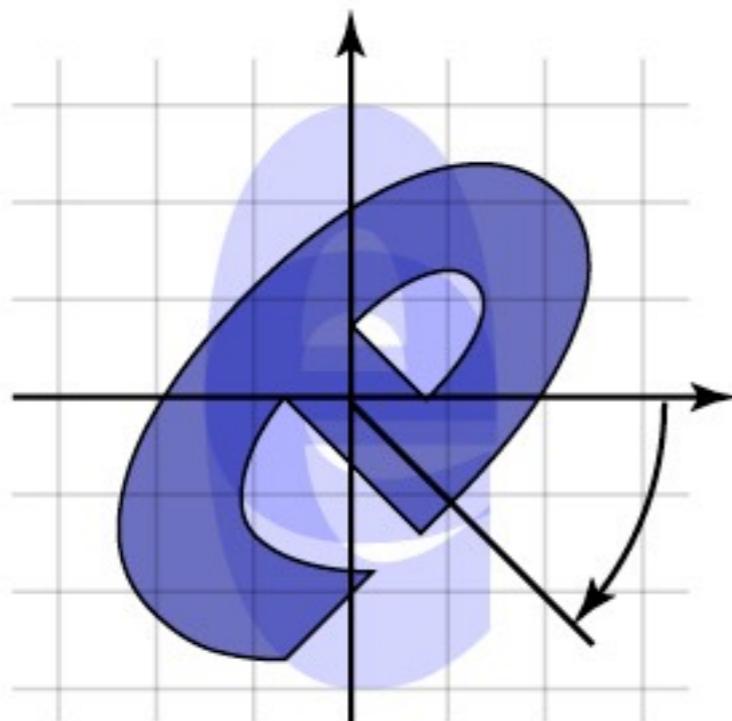
scale, then rotate



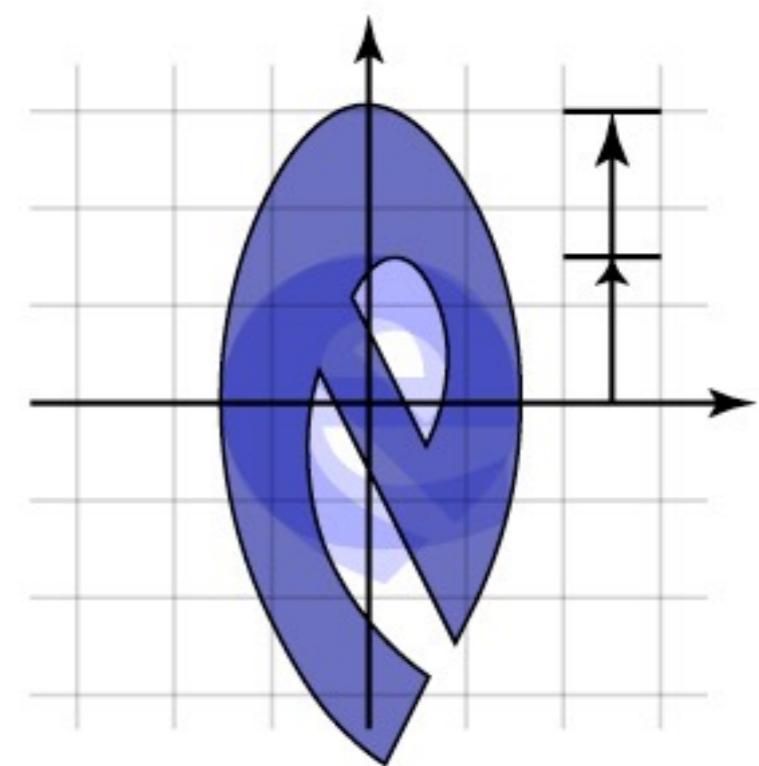
rotate, then scale

Composite affine transformations

- **Another example**



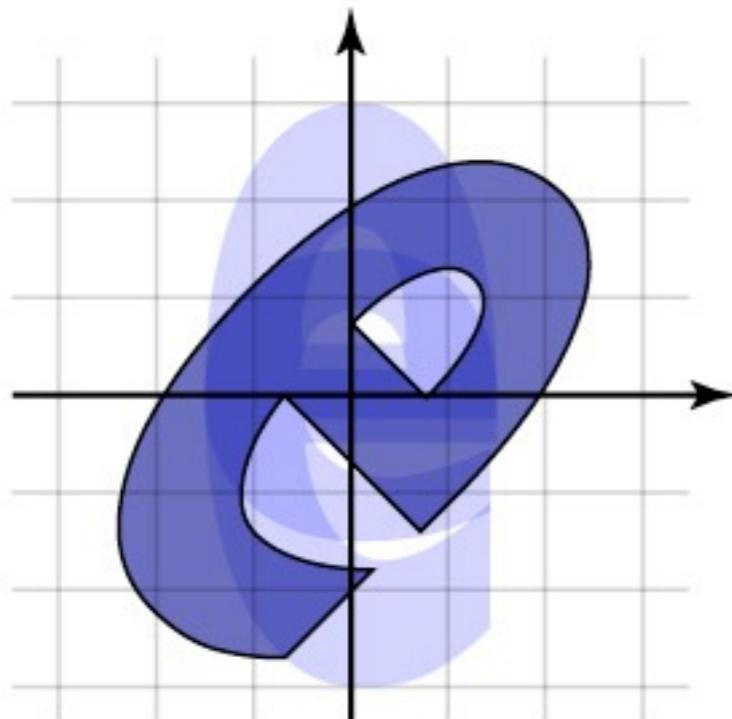
scale, then rotate



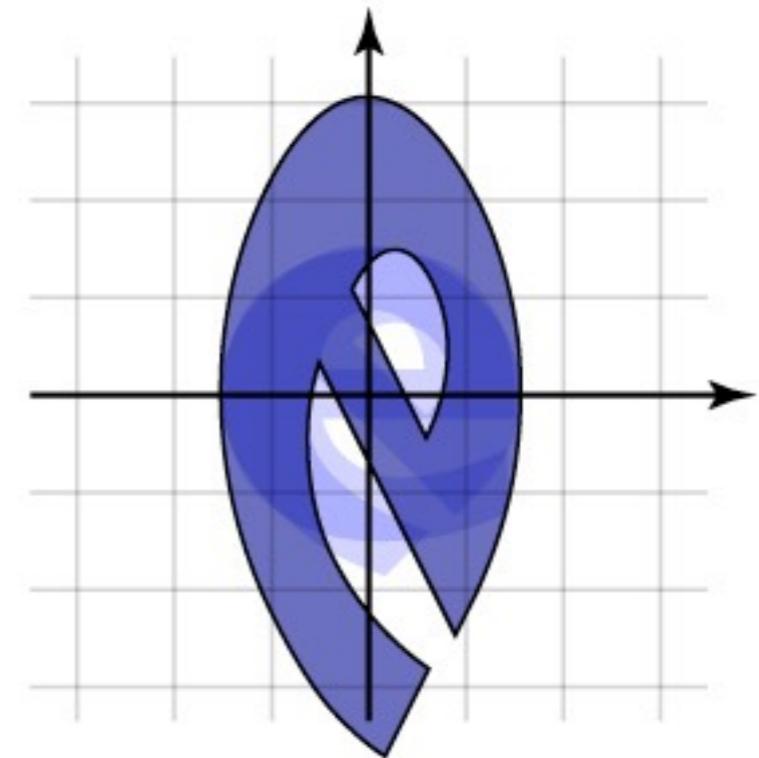
rotate, then scale

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

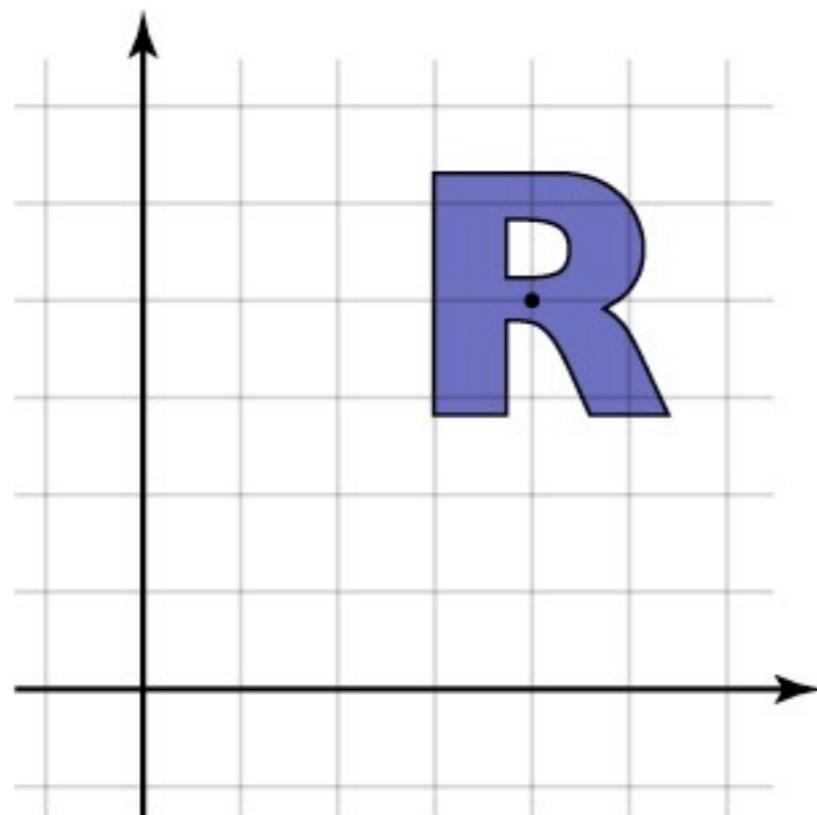
$$R = \begin{bmatrix} Q & \mathbf{u} \\ 0 & 1 \end{bmatrix}$$

- 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

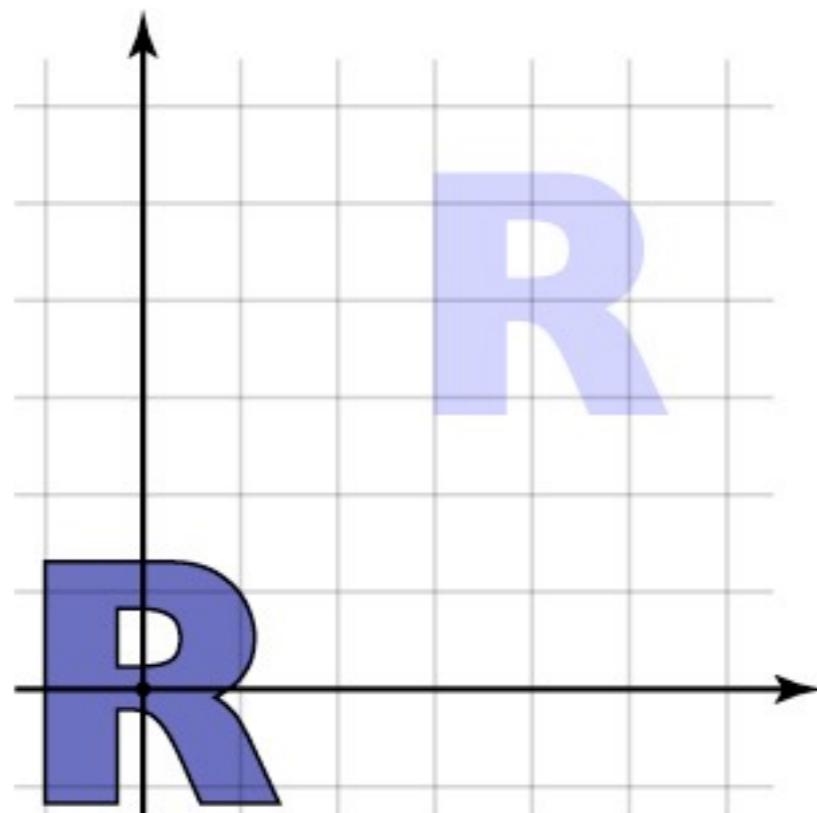
- **Want to rotate about a particular point**
 - could work out formulas directly...
- **Know how to rotate about the origin**
 - so translate that point to the origin



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

Composing to change axes

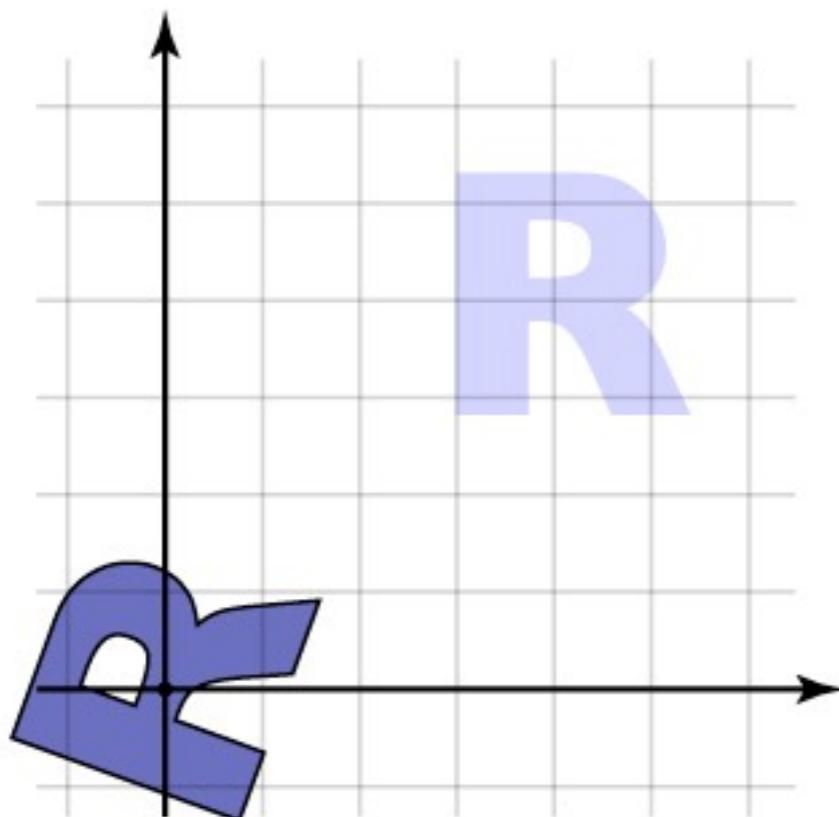
- **Want to rotate about a particular point**
 - could work out formulas directly...
- **Know how to rotate about the origin**
 - so translate that point to the origin



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

Composing to change axes

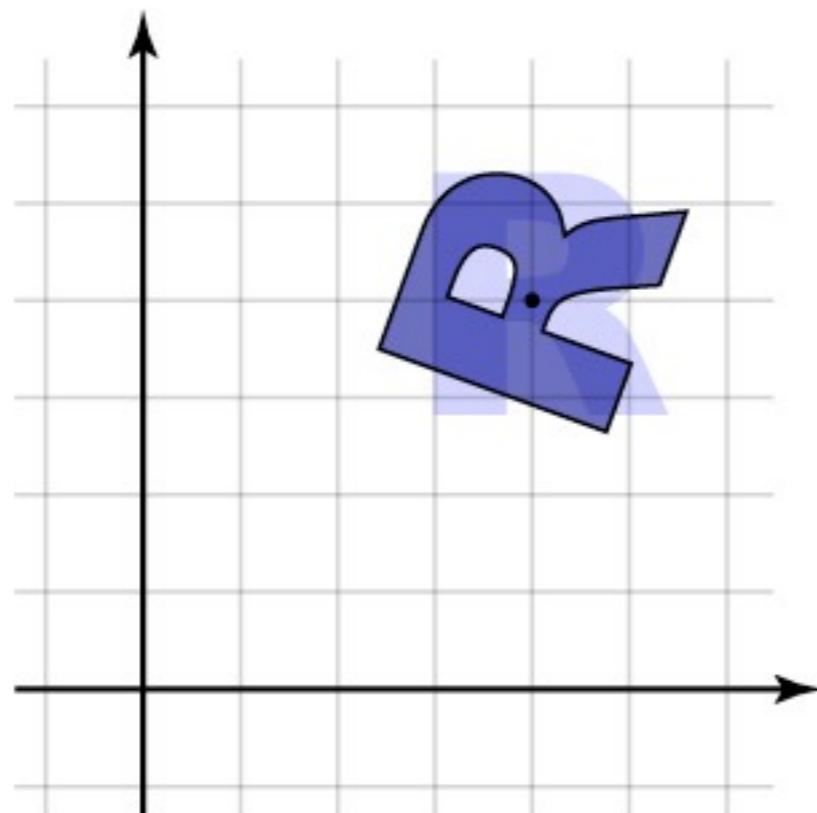
- **Want to rotate about a particular point**
 - could work out formulas directly...
- **Know how to rotate about the origin**
 - so translate that point to the origin



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

Composing to change axes

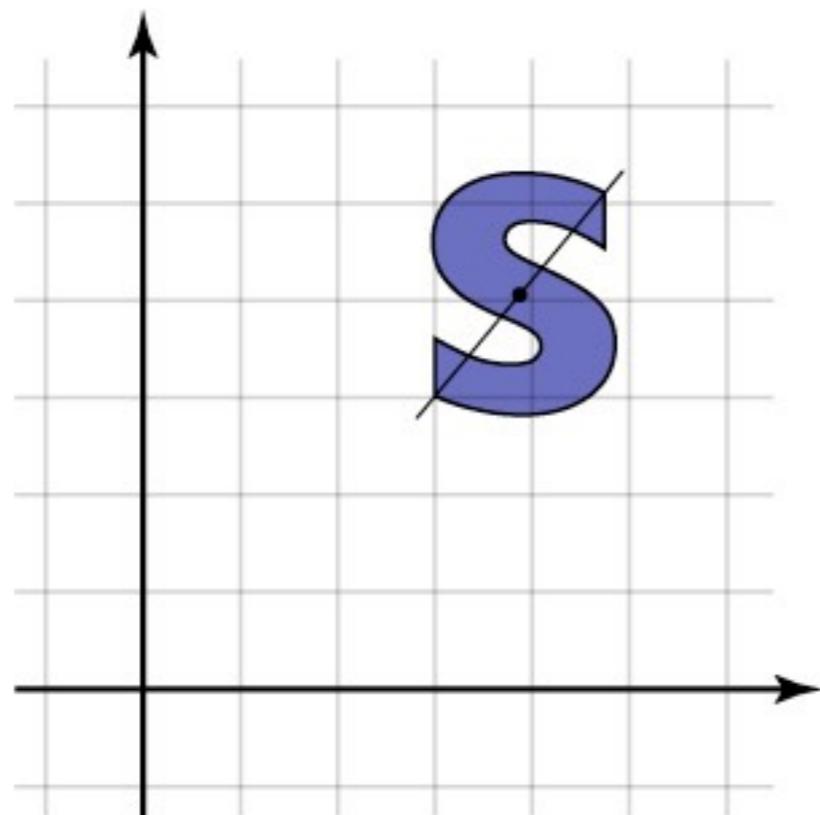
- **Want to rotate about a particular point**
 - could work out formulas directly...
- **Know how to rotate about the origin**
 - so translate that point to the origin



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

Composing to change axes

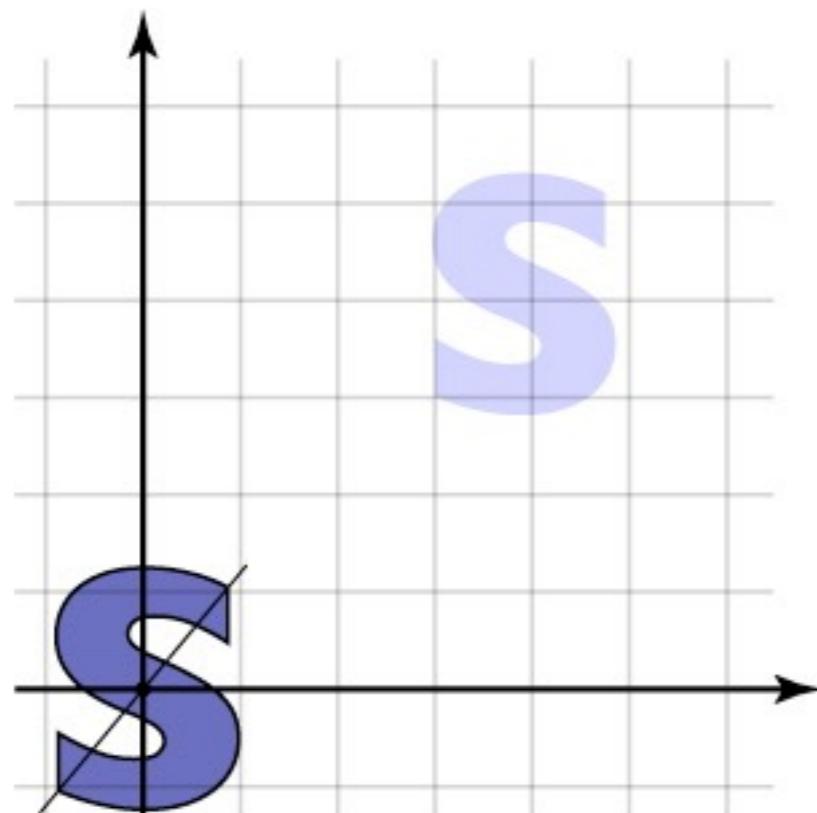
- **Want to scale along a particular axis and point**
- **Know how to scale along the y axis at the origin**
 - so translate to the origin and rotate to align axes



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

Composing to change axes

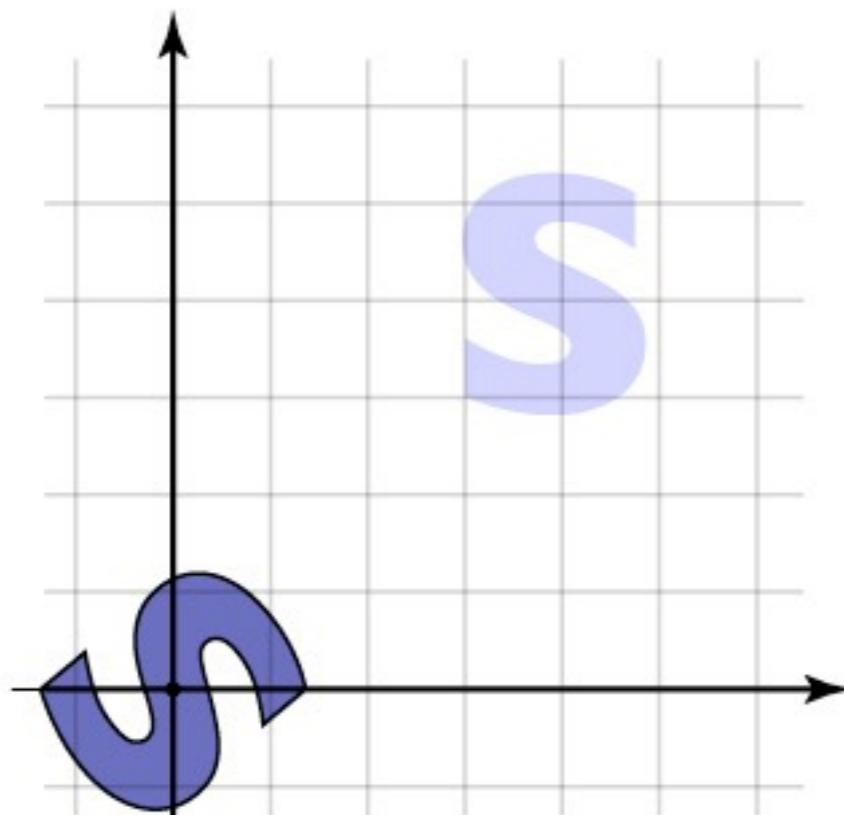
- **Want to scale along a particular axis and point**
- **Know how to scale along the y axis at the origin**
 - so translate to the origin and rotate to align axes



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

Composing to change axes

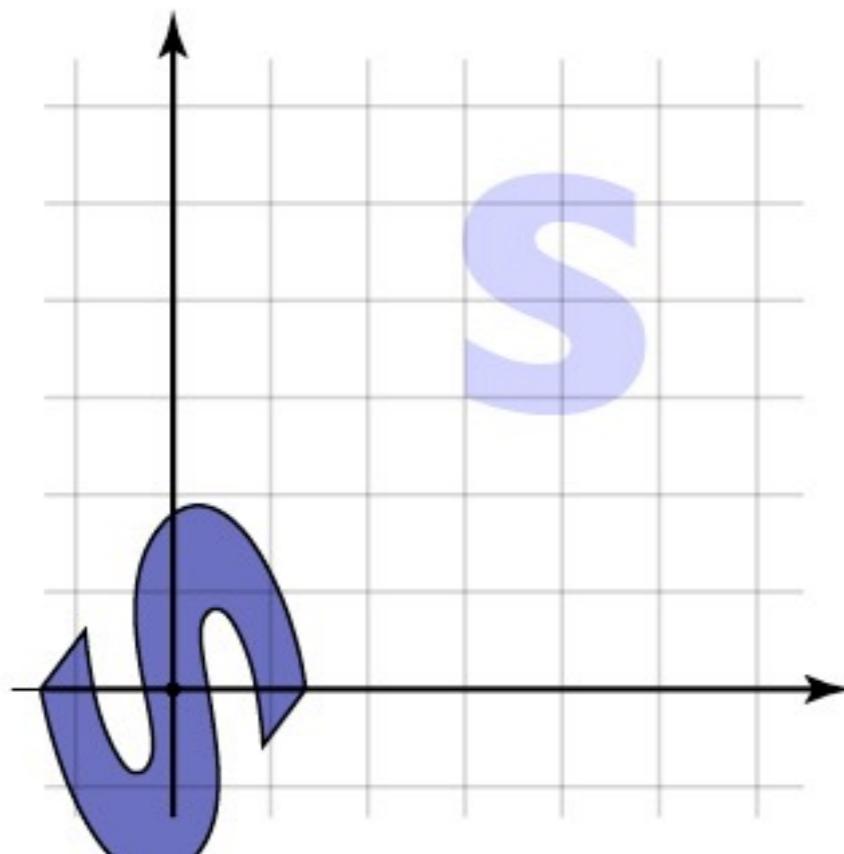
- **Want to scale along a particular axis and point**
- **Know how to scale along the y axis at the origin**
 - so translate to the origin and rotate to align axes



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

Composing to change axes

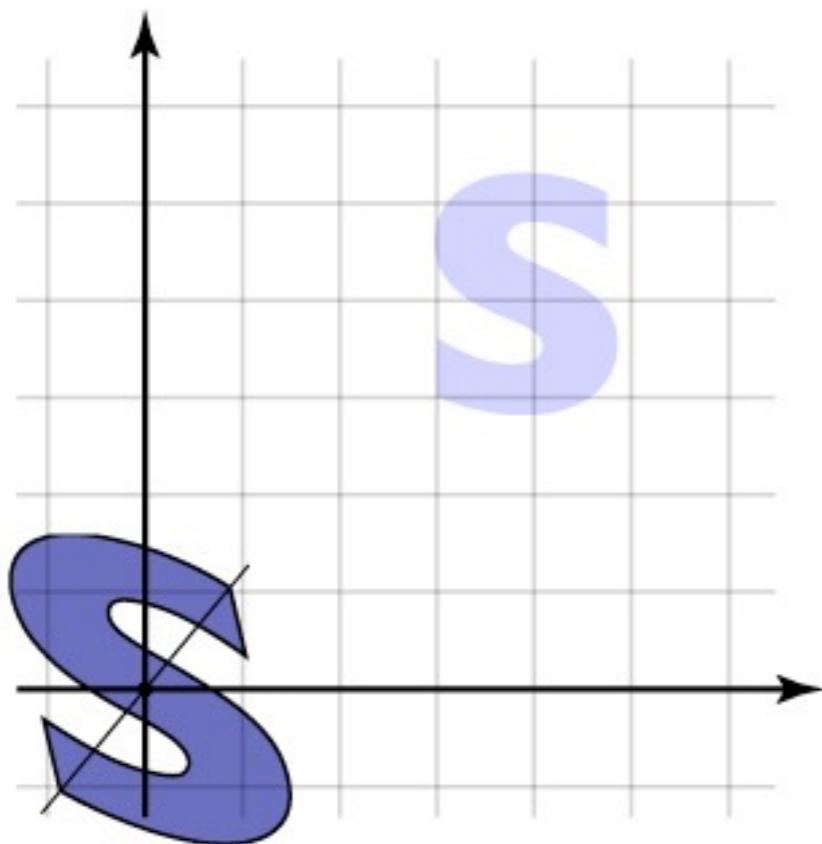
- **Want to scale along a particular axis and point**
- **Know how to scale along the y axis at the origin**
 - so translate to the origin and rotate to align axes



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

Composing to change axes

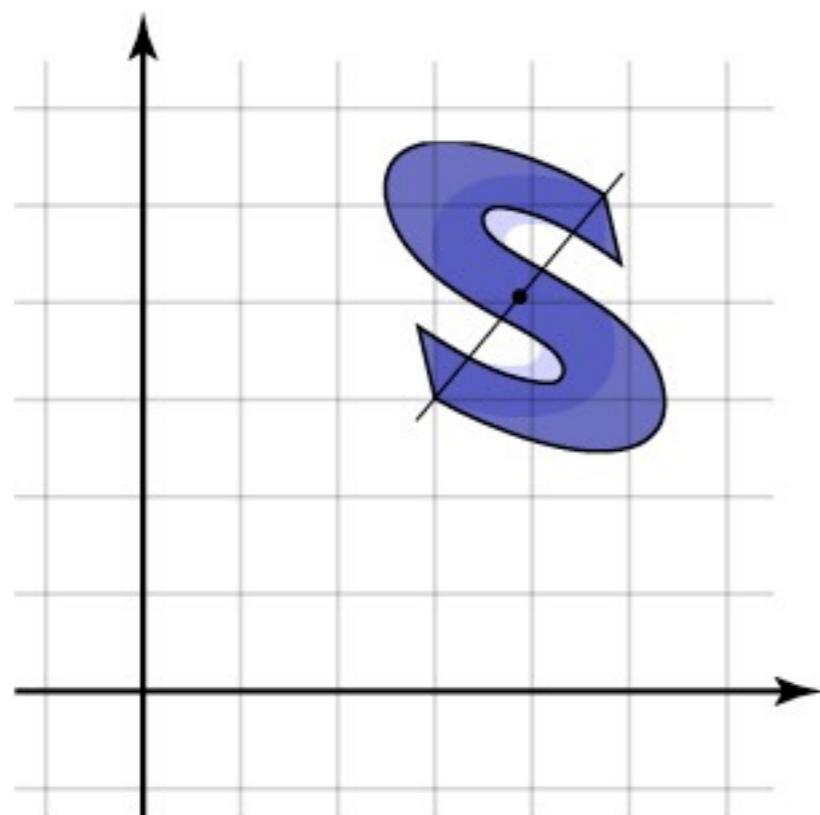
- **Want to scale along a particular axis and point**
- **Know how to scale along the y axis at the origin**
 - so translate to the origin and rotate to align axes



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

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

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

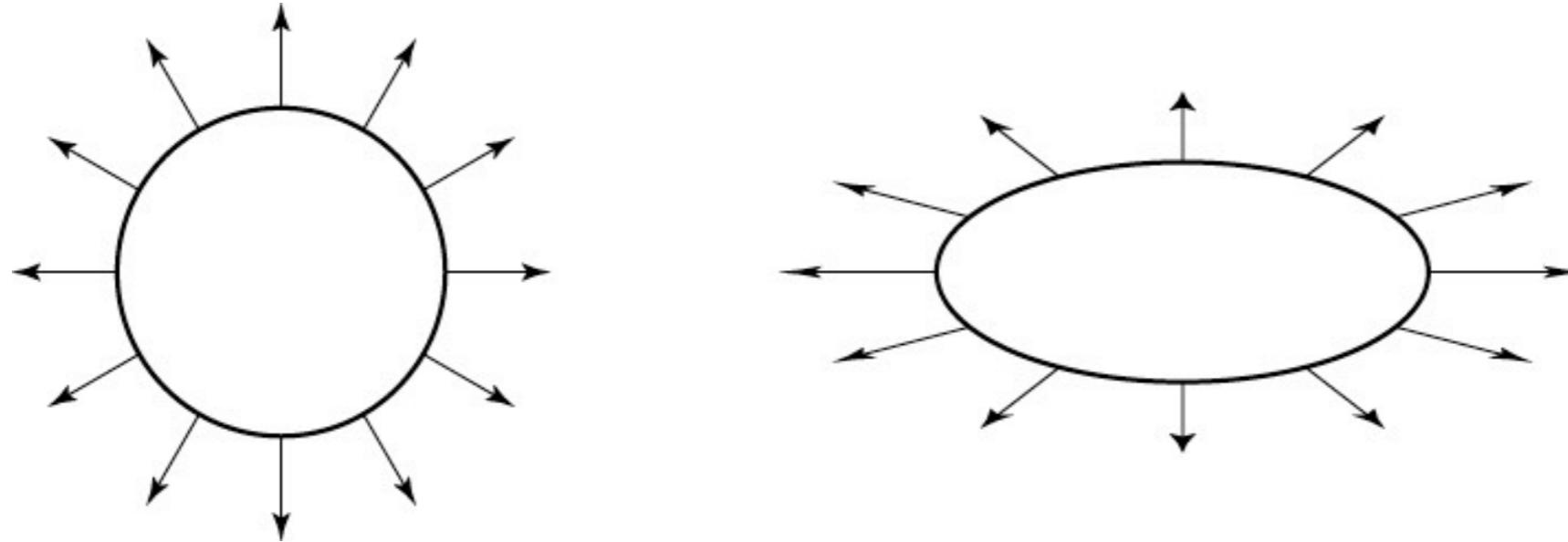
- and note that subtracting two points cancels the extra coordinate, resulting in a vector!

- **Preview: projective transformations**

- what's really going on with this last coordinate?
 - think of \mathbb{R}^2 embedded in \mathbb{R}^3 : all affine xfs. preserve $\mathbf{z}=1$ plane
 - could have other transforms; project back to $\mathbf{z}=1$

Transforming normal vectors

- **Transforming surface normals**
 - differences of points (and therefore tangents) transform OK
 - normals do not; therefore use inverse transpose matrix



$$\text{have: } \mathbf{t} \cdot \mathbf{n} = \mathbf{t}^T \mathbf{n} = 0$$

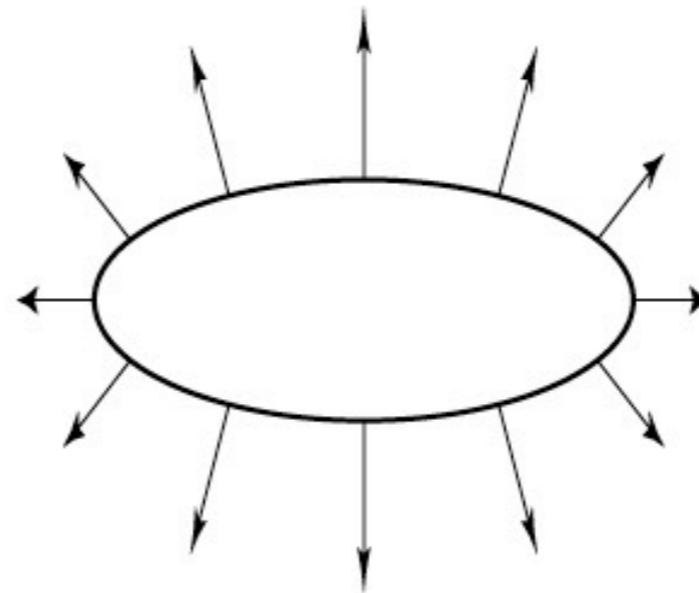
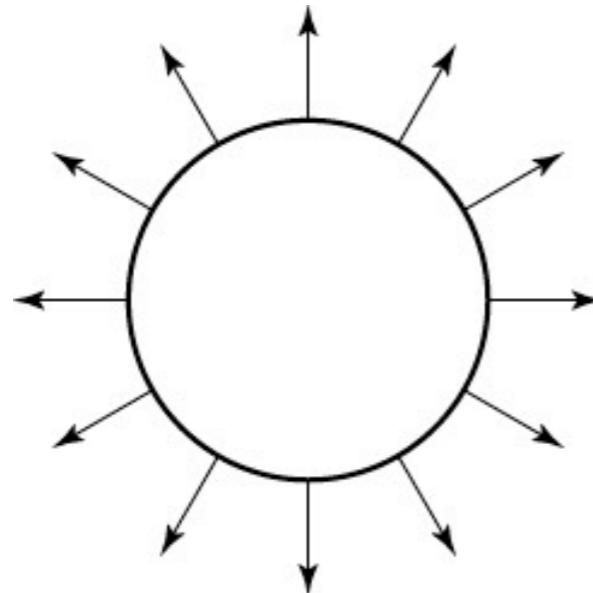
$$\text{want: } M\mathbf{t} \cdot X\mathbf{n} = \mathbf{t}^T M^T X \mathbf{n} = 0$$

$$\text{so set } X = (M^T)^{-1}$$

$$\text{then: } M\mathbf{t} \cdot X\mathbf{n} = \mathbf{t}^T M^T (M^T)^{-1} \mathbf{n} = \mathbf{t}^T \mathbf{n} = 0$$

Transforming normal vectors

- **Transforming surface normals**
 - differences of points (and therefore tangents) transform OK
 - normals do not; therefore use inverse transpose matrix



$$\text{have: } \mathbf{t} \cdot \mathbf{n} = \mathbf{t}^T \mathbf{n} = 0$$

$$\text{want: } M\mathbf{t} \cdot X\mathbf{n} = \mathbf{t}^T M^T X \mathbf{n} = 0$$

$$\text{so set } X = (M^T)^{-1}$$

$$\text{then: } M\mathbf{t} \cdot X\mathbf{n} = \mathbf{t}^T M^T (M^T)^{-1} \mathbf{n} = \mathbf{t}^T \mathbf{n} = 0$$

More math background

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

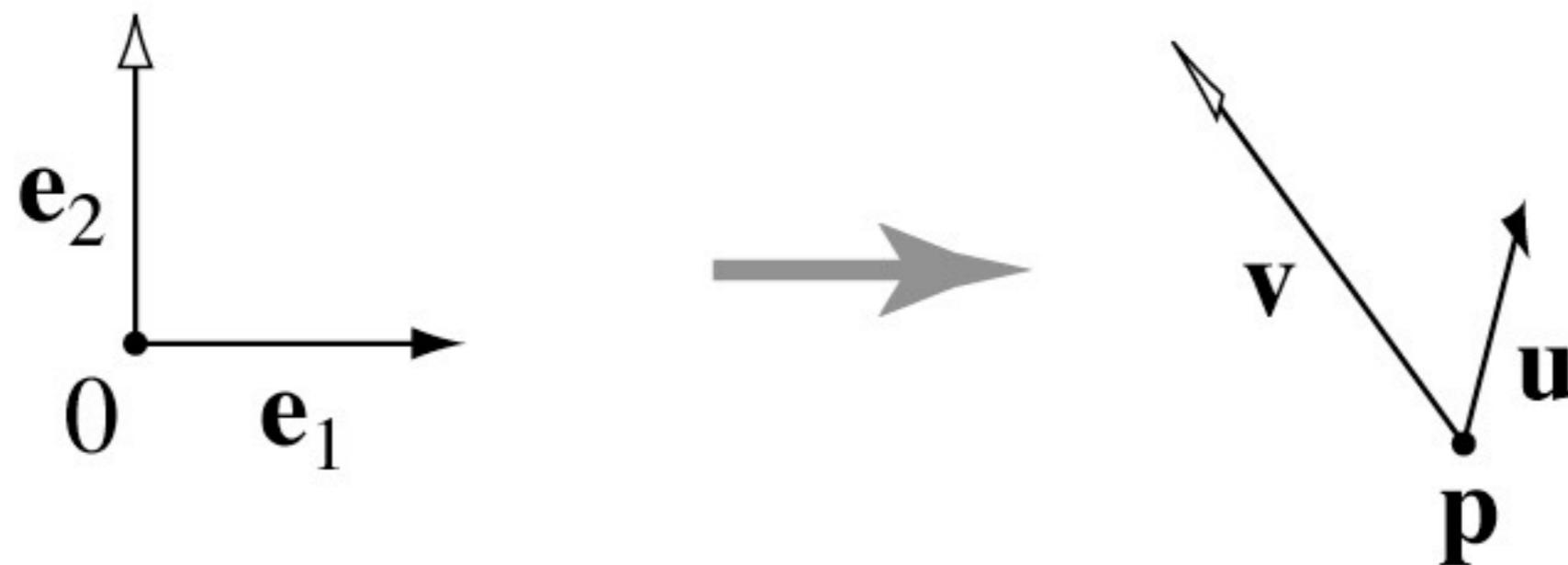
Affine change of coordinates

- **Six degrees of freedom**

$$\begin{bmatrix} a_1 & a_2 & a_3 \\ a_4 & a_5 & a_6 \\ 0 & 0 & 1 \end{bmatrix}$$

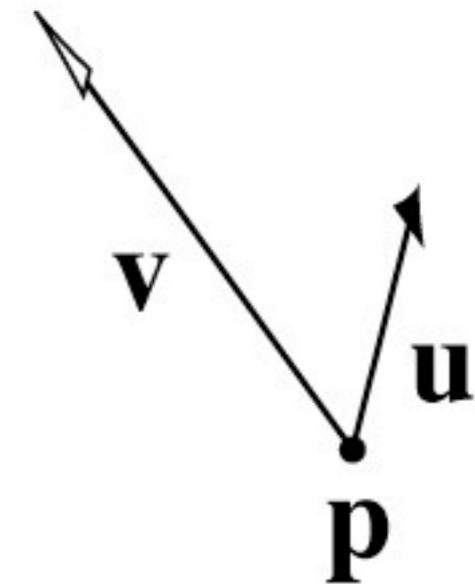
or

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



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**

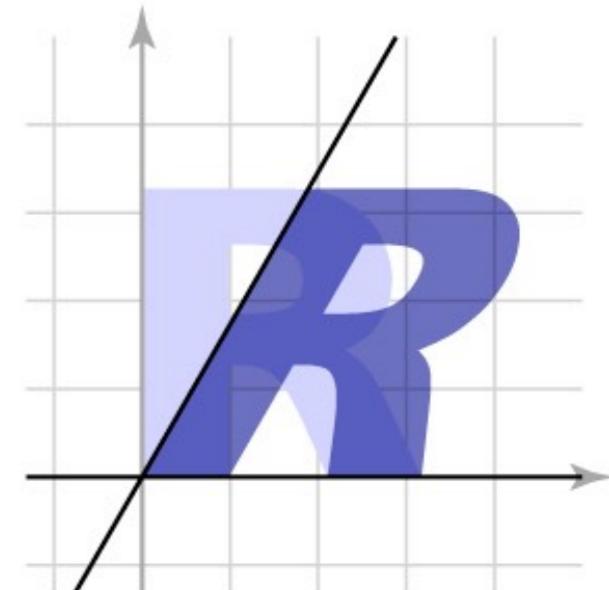


$$\begin{bmatrix} \mathbf{u} & \mathbf{v} & \mathbf{p} \\ 0 & 0 & 1 \end{bmatrix}$$

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

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



Affine change of coordinates

- **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. $\{\mathbf{e}_1, \mathbf{e}_2\}$
- T_F is the transformation expressed in natural frame
- F is the frame-to-canonical matrix $[u \ v \ p]$
- **This is a *similarity transformation***

Building general rotations

- **Using elementary transforms you need three**
 - translate axis to pass through origin
 - rotate about y to get into x - y plane
 - rotate about z to align with x axis
- **Alternative: construct frame and change coordinates**
 - choose p, u, v, w to be orthonormal frame with p and u matching the rotation axis
 - apply similarity transform $T = F R_x(\theta) F^{-1}$

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**

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

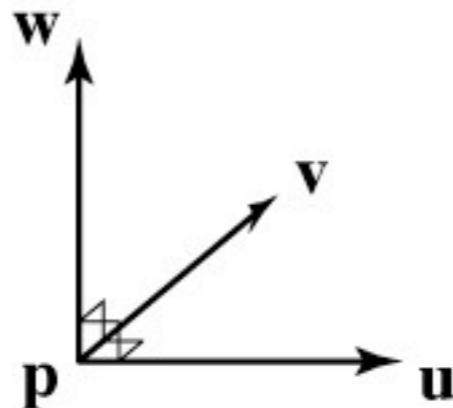
- **Move transformations using similarity transforms**

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

Orthonormal frames in 3D

- **Useful tools for constructing transformations**
- **Recall rigid motions**
 - affine transforms with pure rotation
 - columns (and rows) form right handed ONB
 - that is, an **orthonormal basis**

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



Building 3D frames

- **Given a vector \mathbf{a} and a secondary vector \mathbf{b}**
 - The \mathbf{u} axis should be parallel to \mathbf{a} ; the \mathbf{u} – \mathbf{v} plane should contain \mathbf{b}
 - $\mathbf{u} = \mathbf{a} / \|\mathbf{a}\|$
 - $\mathbf{w} = \mathbf{u} \times \mathbf{b}; \mathbf{w} = \mathbf{w} / \|\mathbf{w}\|$
 - $\mathbf{v} = \mathbf{w} \times \mathbf{u}$
- **Given just a vector \mathbf{a}**
 - The \mathbf{u} axis should be parallel to \mathbf{a} ; don't care about orientation about that axis
 - Same process but choose arbitrary \mathbf{b} first
 - Good choice is not near \mathbf{a} : e.g. set smallest entry to 1

Building transforms from points

- **2D affine transformation has 6 degrees of freedom (DOFs)**
 - this is the number of “knobs” we have to set to define one
- **So, 6 constraints suffice to define the transformation**
 - handy kind of constraint: point \mathbf{p} maps to point \mathbf{q} (2 constraints at once)
 - three point constraints add up to constrain all 6 DOFs
(i.e. can map any triangle to any other triangle)
- **3D affine transformation has 12 degrees of freedom**
 - count them from the matrix entries we’re allowed to change
- **So, 12 constraints suffice to define the transformation**
 - in 3D, this is 4 point constraints
(i.e. can map any tetrahedron to any other tetrahedron)