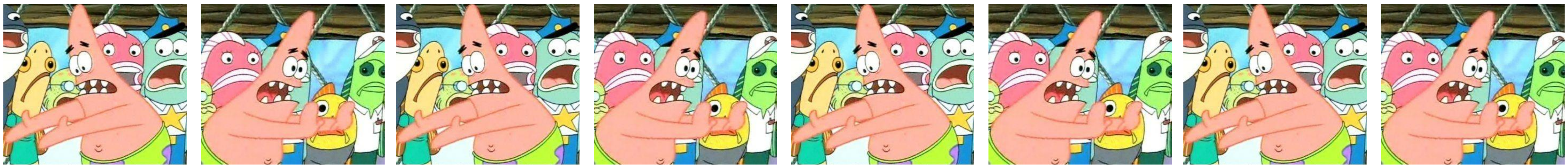


Computer Graphics

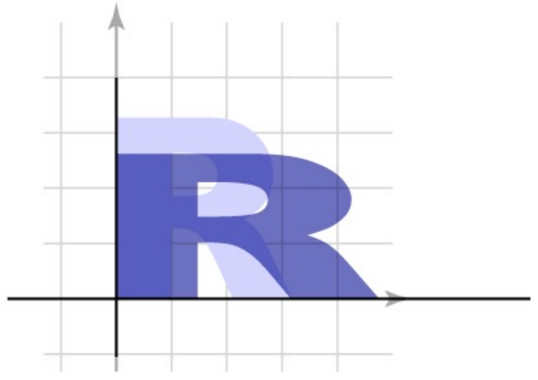
Lecture 14

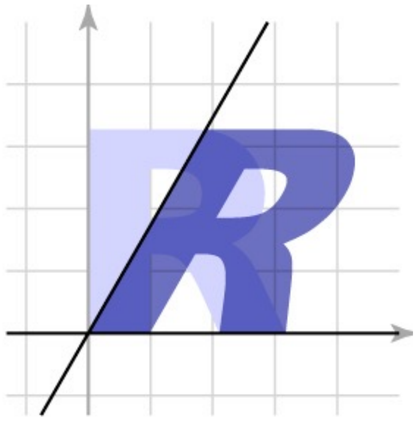
Affine Composition
3D Transformations

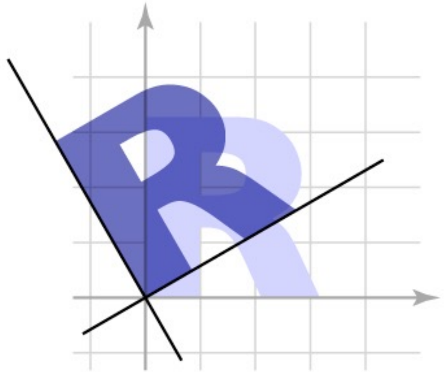
Announcements

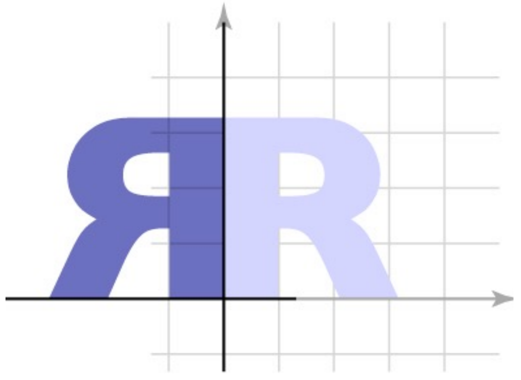
Announcements

Transformations: So Far


scale

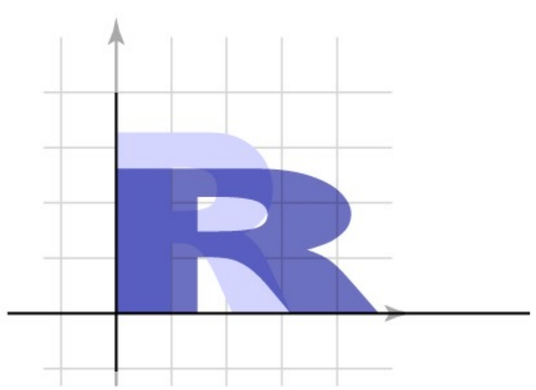

shear


rotate

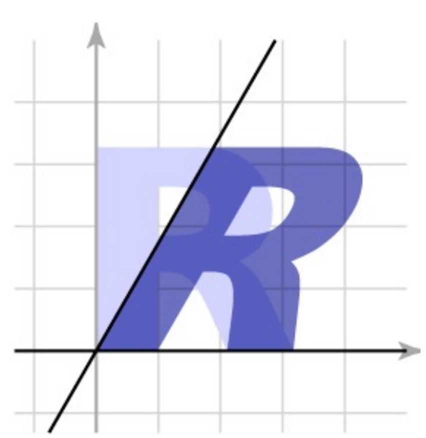

reflect

Transformations: So Far

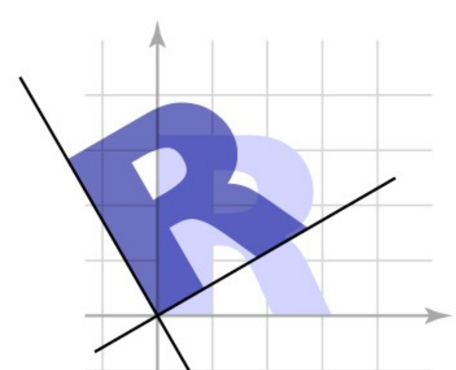
- 2x2 matrices are linear functions that:



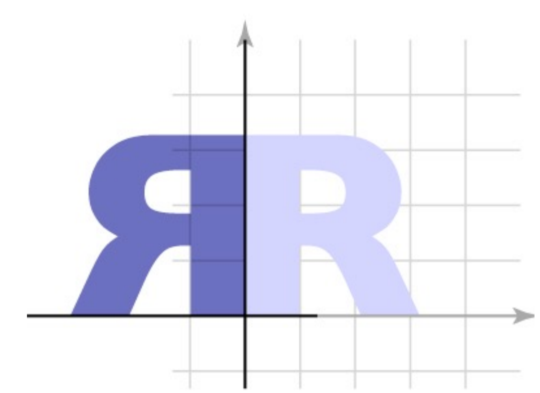
scale



shear



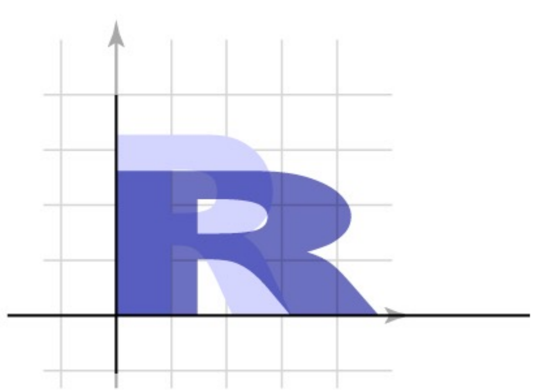
rotate



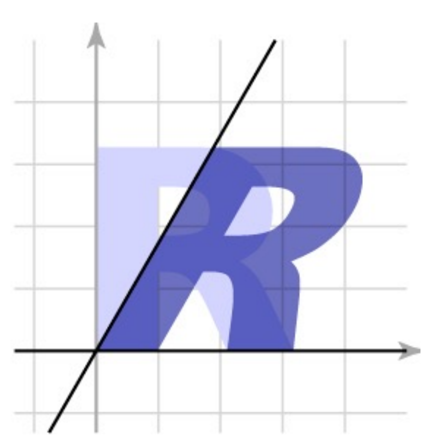
reflect

Transformations: So Far

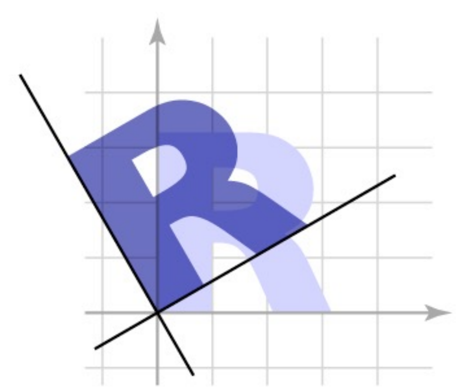
- 2x2 matrices are linear functions that:
  - Move 2D points from one place to another



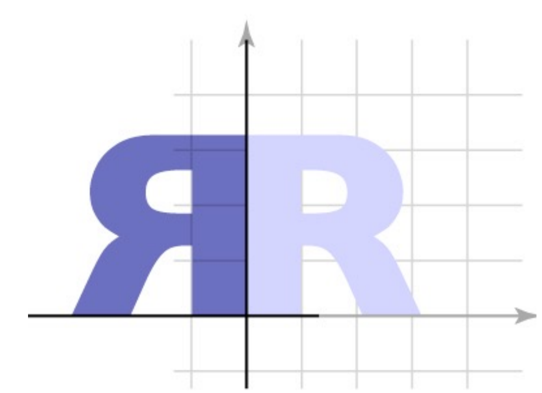
scale



shear



rotate

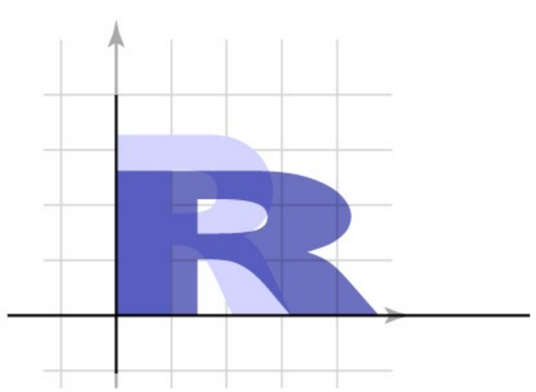


reflect

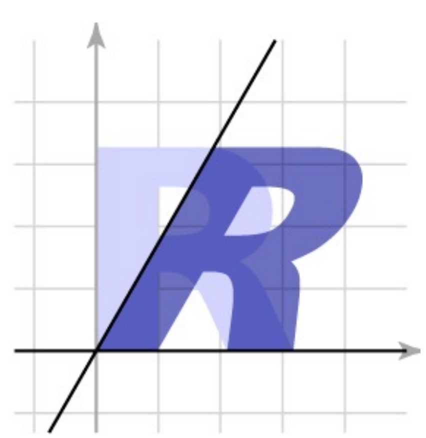
Transformations: So Far

- 2x2 matrices are linear functions that:
    - Move 2D points from one place to another

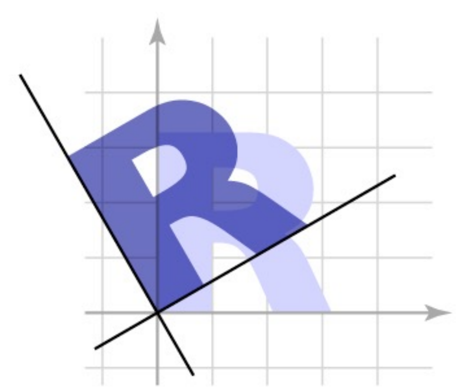
or, **equivalently**:



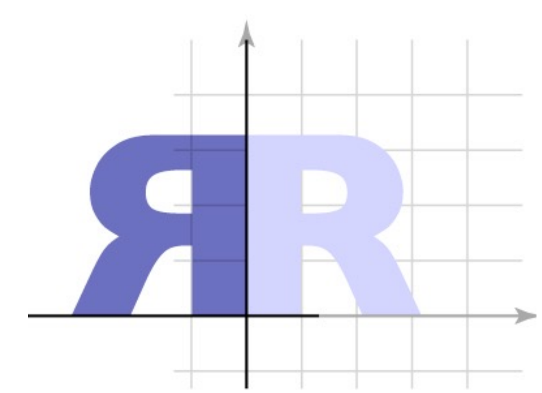
scale



shear



rotate



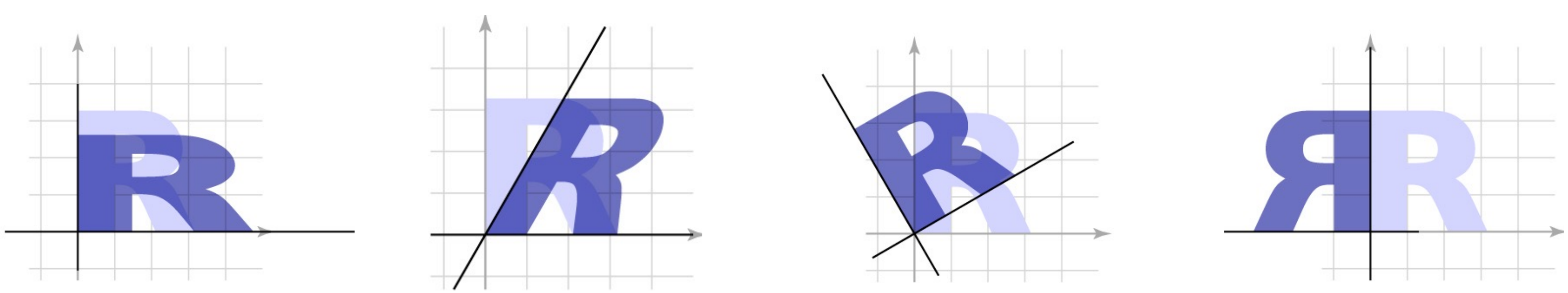
reflect

Transformations: So Far

- 2x2 matrices are linear functions that:
  - Move 2D points from one place to another

or, **equivalently**:

  - Change the basis in which points are represented



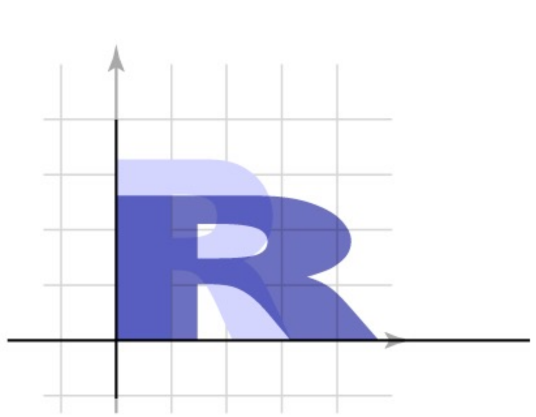
scale shear rotate reflect

Transformations: So Far

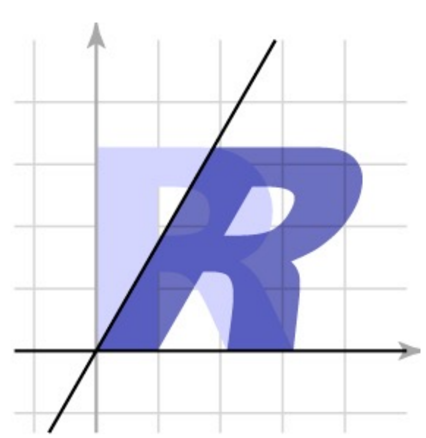
- 2x2 matrices are linear functions that:
  - Move 2D points from one place to another

or, **equivalently**:

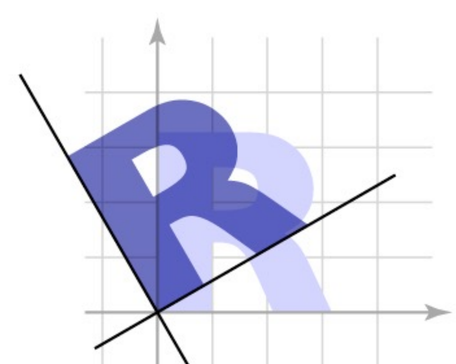
  - Change the basis in which points are represented
- We can:



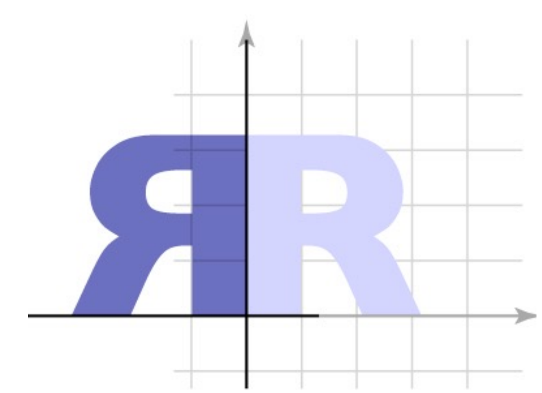
scale



shear



rotate



reflect

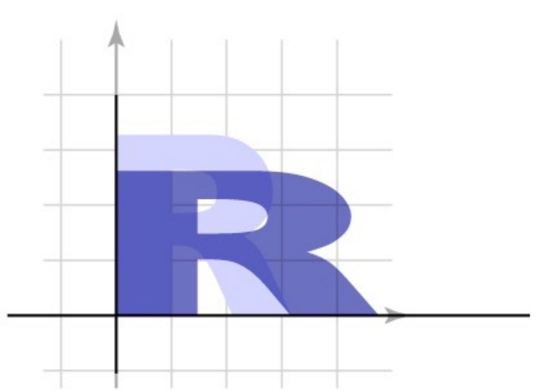
Transformations: So Far

- 2x2 matrices are linear functions that:
  - Move 2D points from one place to another

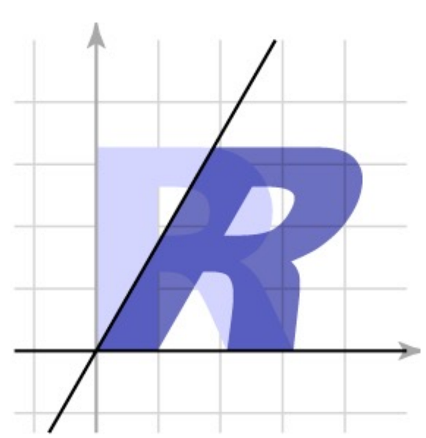
or, **equivalently**:

  - Change the basis in which points are represented
- We can:

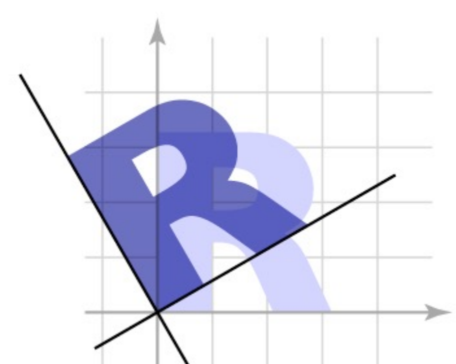
but we can't do translation!



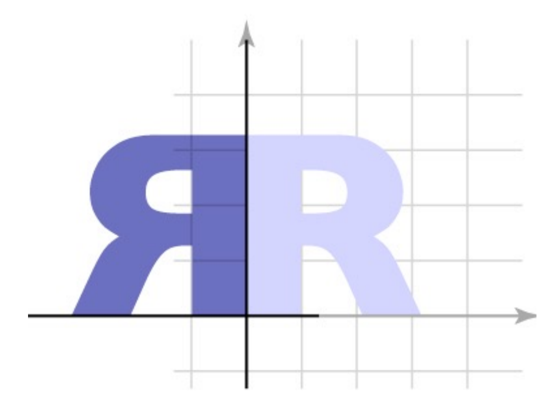
scale



shear



rotate



reflect

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

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

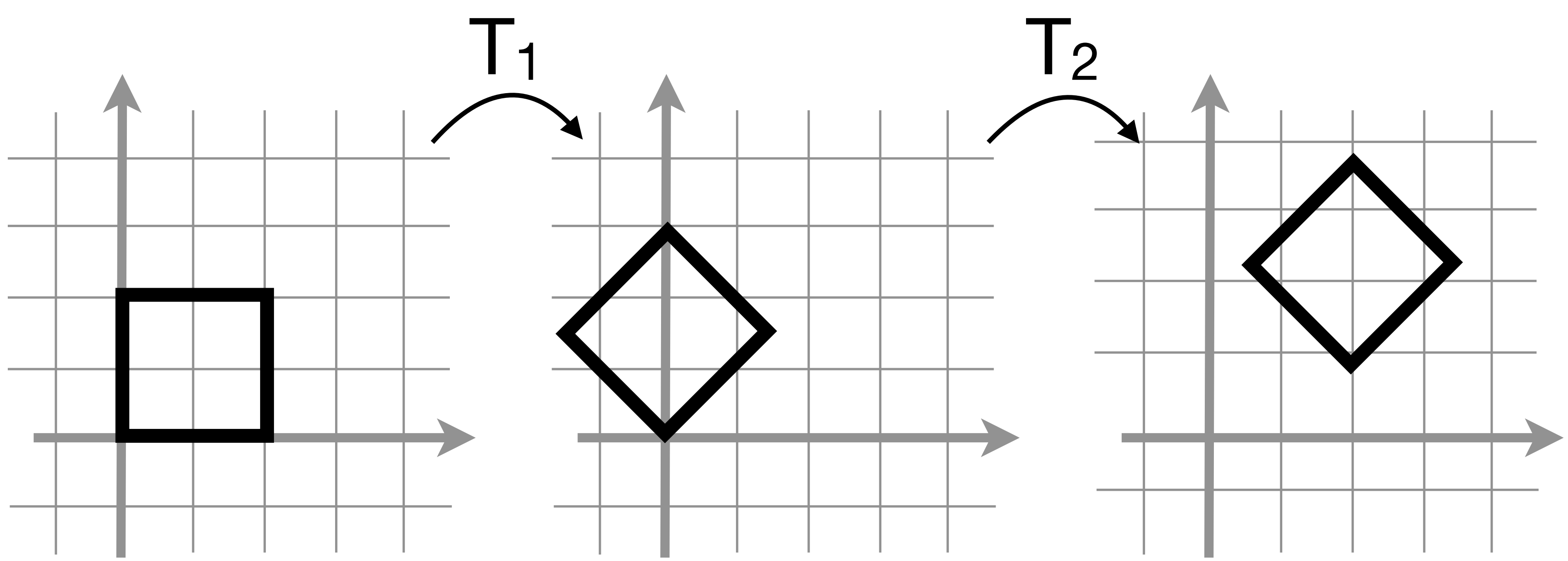
Homogeneous coordinates

- Represent translation using an extra column

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

Transformations: So Far

- Transformations are **composable** via matrix multiplication:

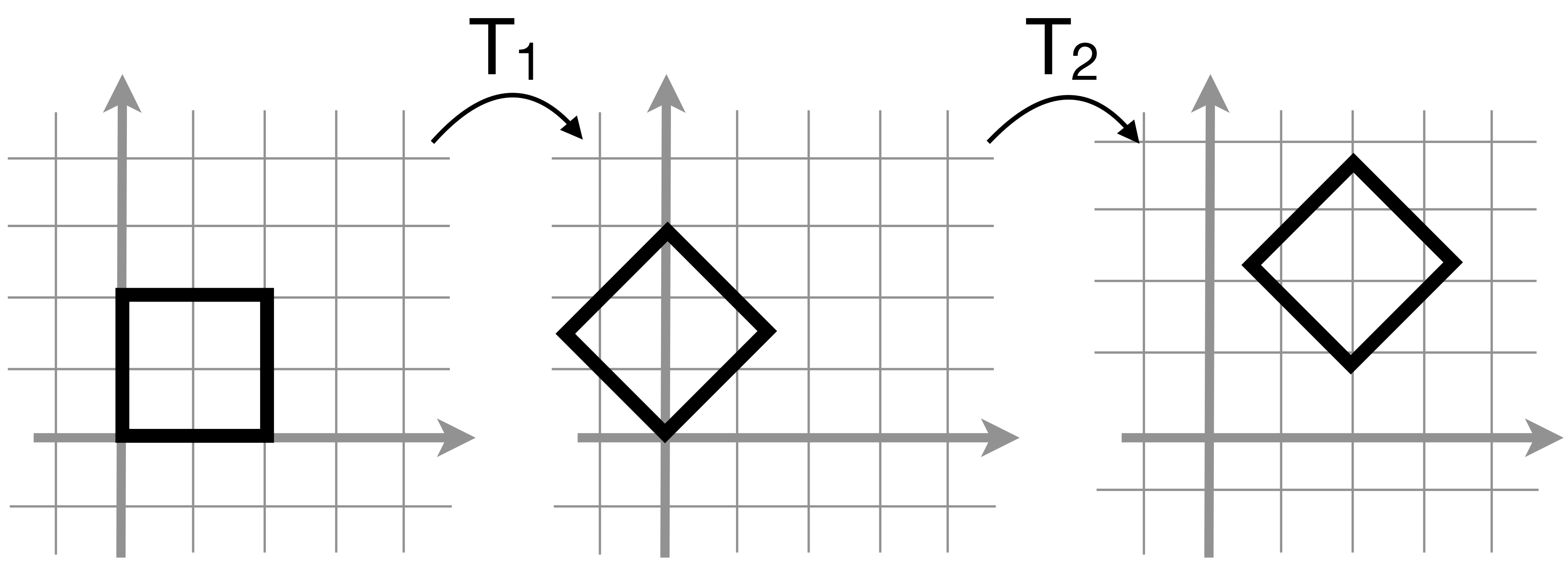


T_1: Rotate 45 CCW

T_2: Translate (1, 0.5)

Transformations: So Far

- Transformations are **composable** via matrix multiplication:



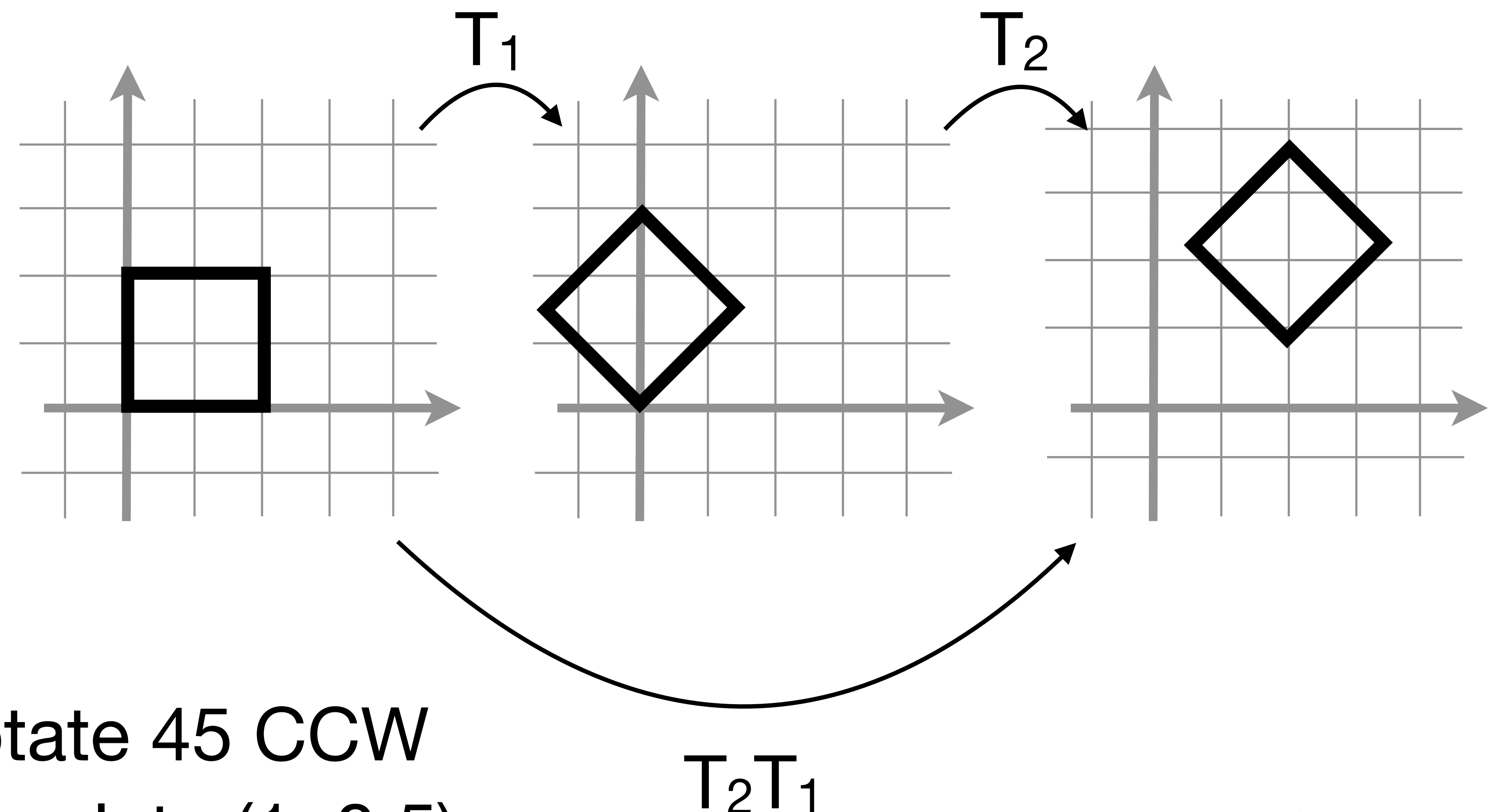
T_1: Rotate 45 CCW

T_2: Translate (1, 0.5)

applied right-to-left!

Transformations: So Far

- Transformations are **composable** via matrix multiplication:



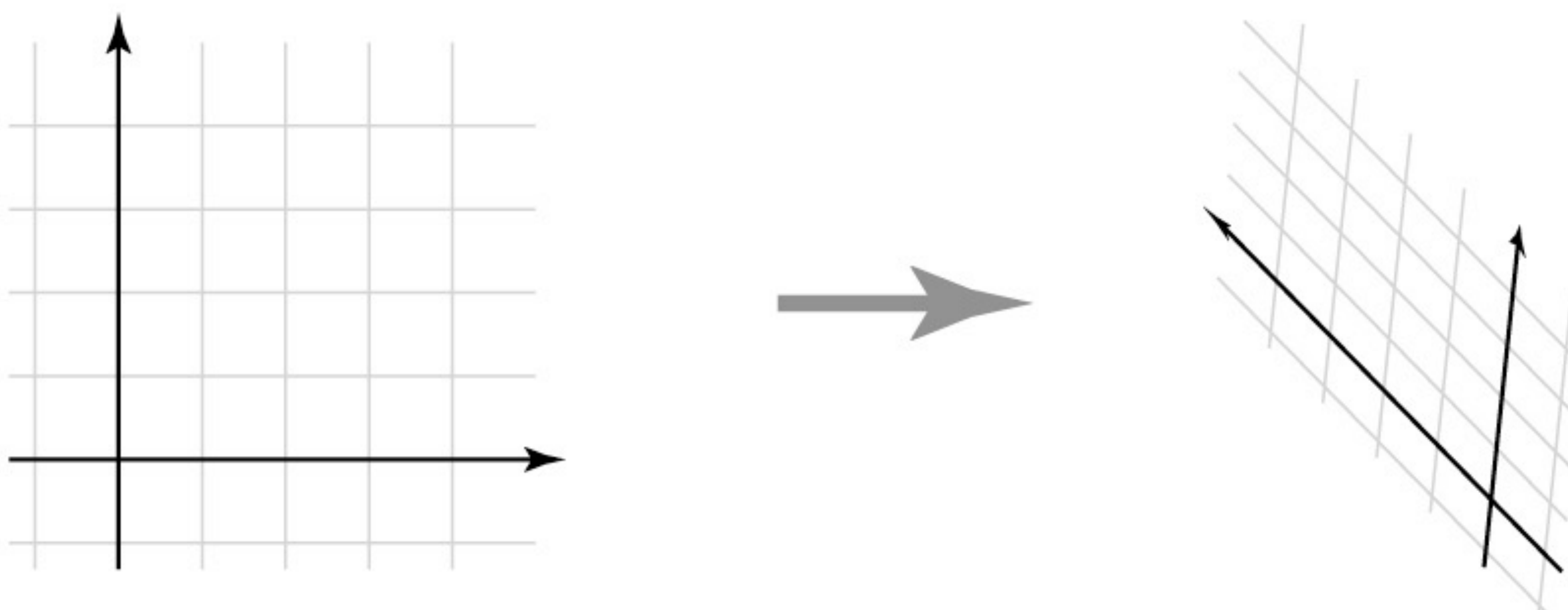
T_1: Rotate 45 CCW

T_2: Translate (1, 0.5)

applied right-to-left!

Affine transformations

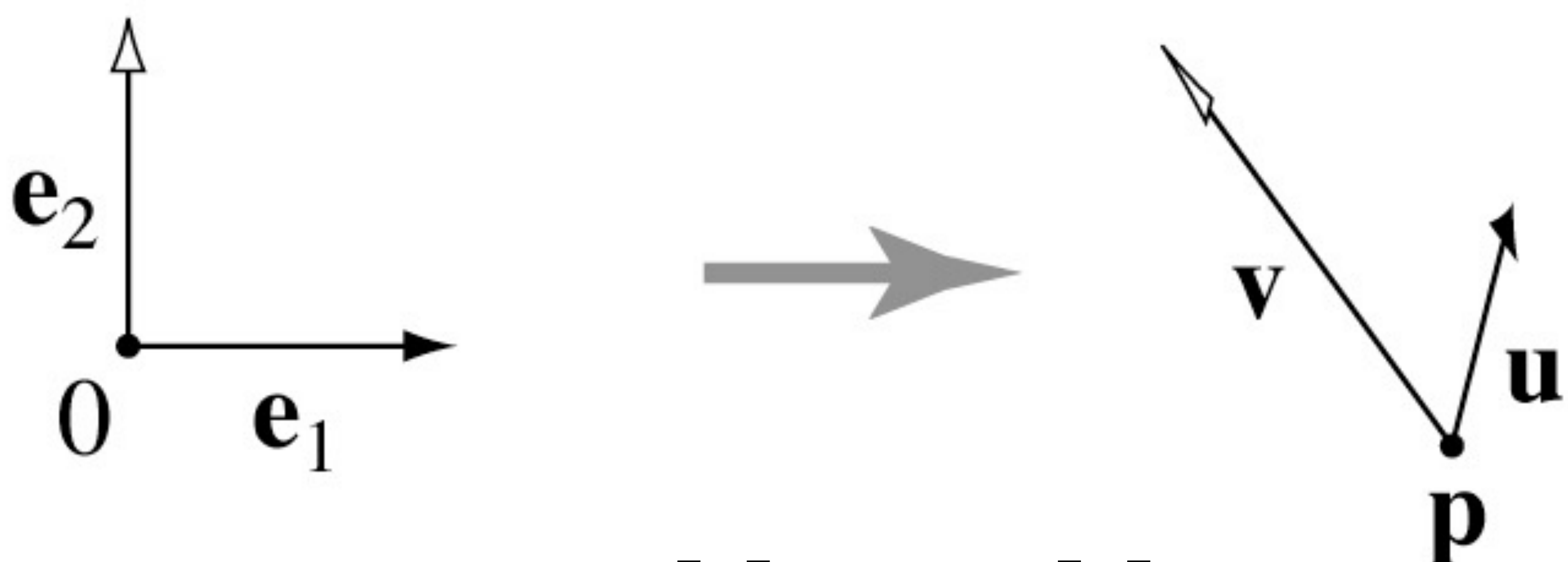
- The set of transformations we have been looking at is known as the "affine" transformations
  - straight lines preserved; parallel lines preserved
  - ratios of lengths along lines preserved (midpoints preserved)



Affine change of coordinates

- Six degrees of freedom

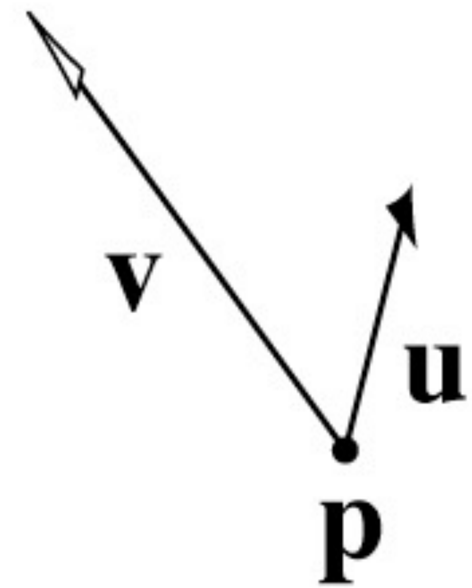
$$\begin{bmatrix} a_1 & a_2 & a_3 \\ a_4 & a_5 & a_6 \\ 0 & 0 & 1 \end{bmatrix} \quad \text{or} \quad \begin{bmatrix} \mathbf{u} & \mathbf{v} & \mathbf{p} \\ 0 & 0 & 1 \end{bmatrix}$$



"canonical" basis: $e_1 = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \quad e_2 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$

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



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

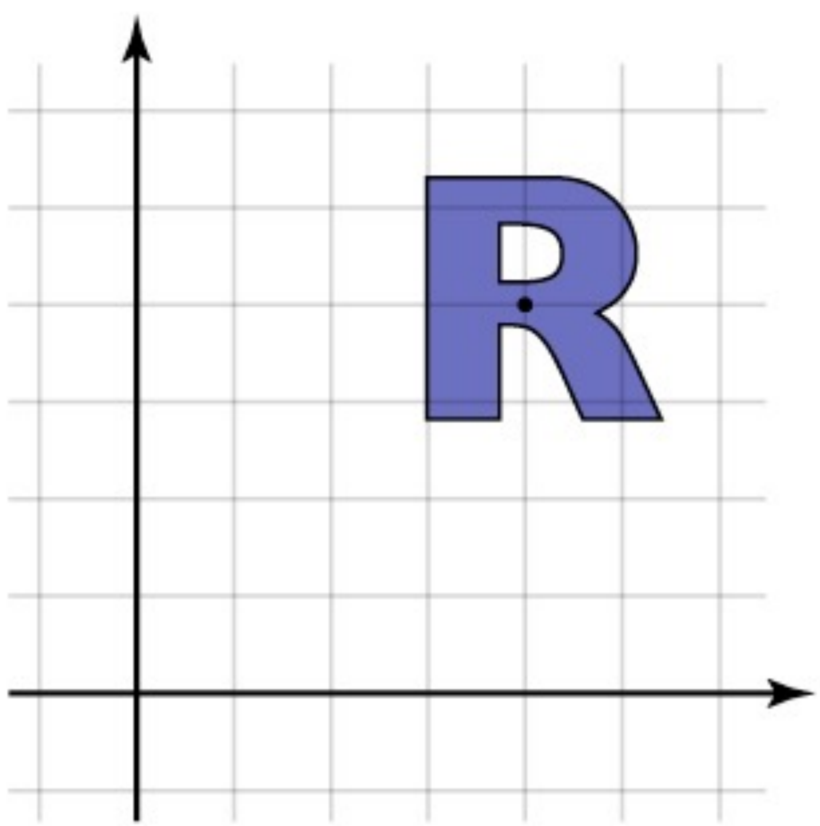
Rigid motions

- 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

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

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

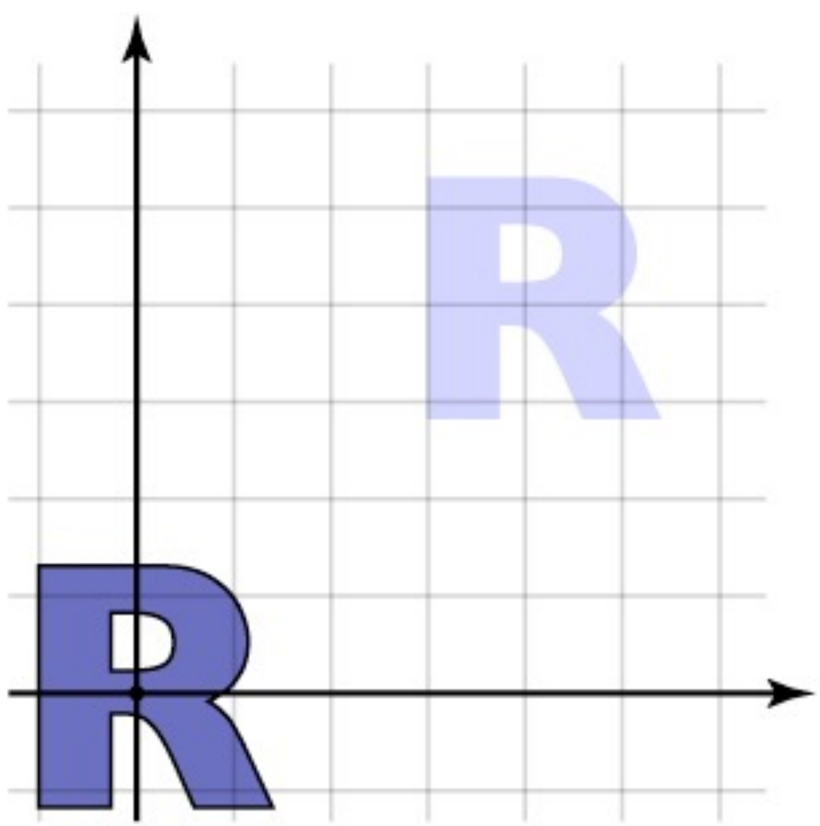


$$M = T^{-1}RT$$

Reminder: R = $\begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$

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

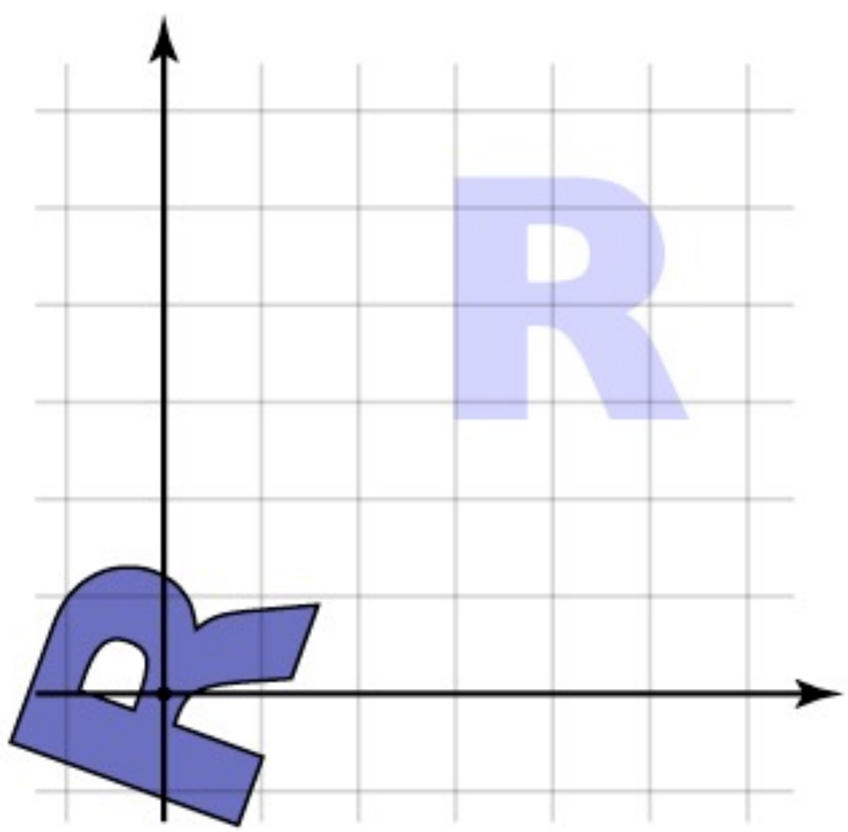


$$M = T^{-1}RT$$

Reminder: R = $\begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$

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

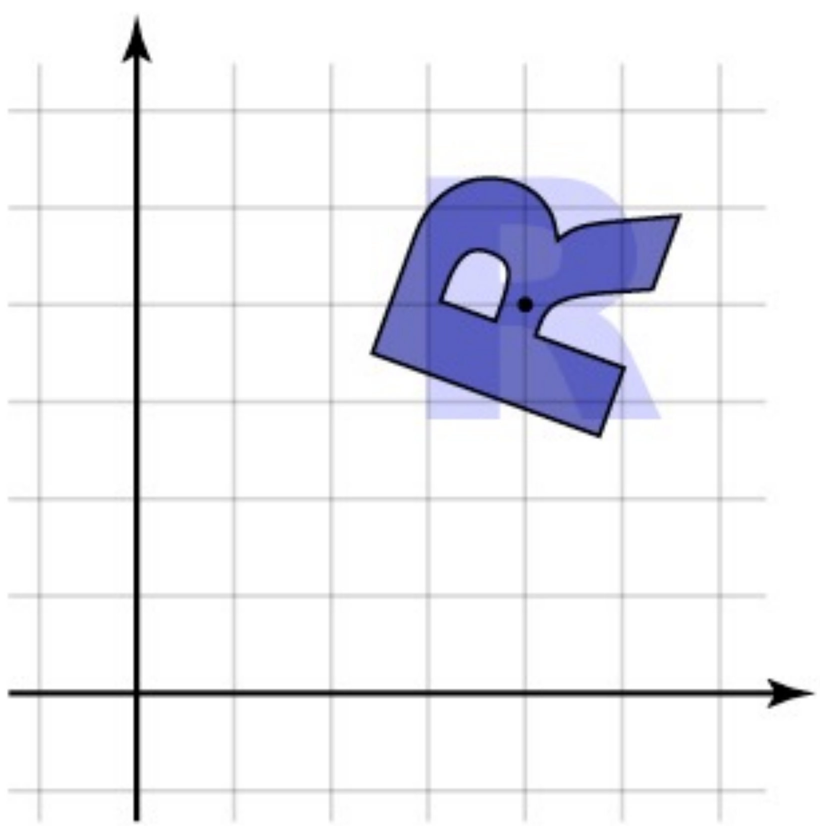


$$M = T^{-1}RT$$

Reminder: R = $\begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$

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

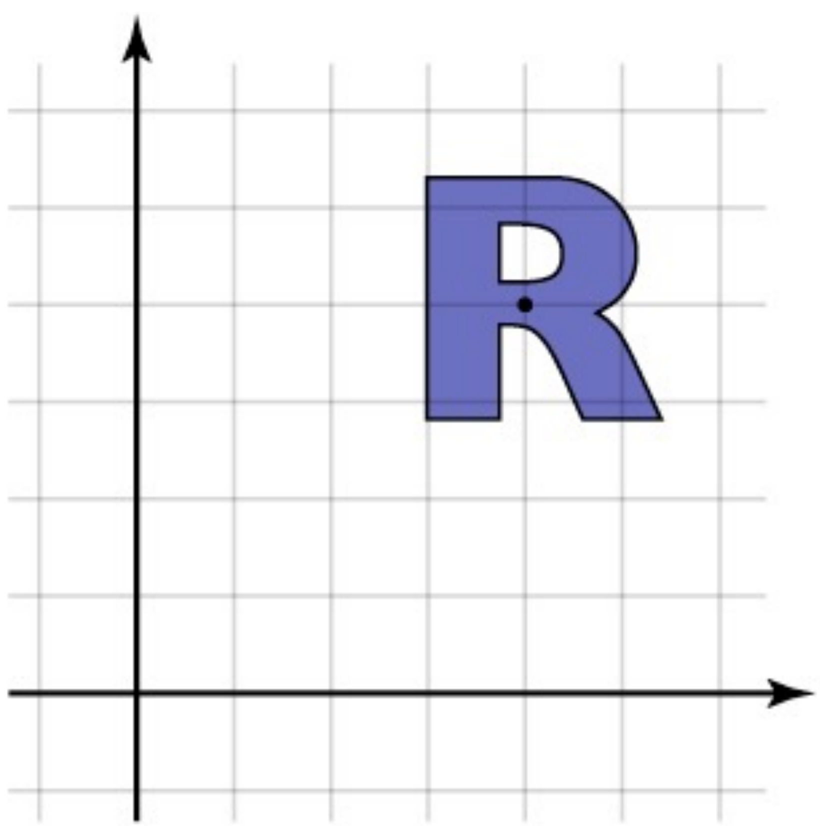


$$M = T^{-1}RT$$

Reminder: R = $\begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$

Exercise: 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

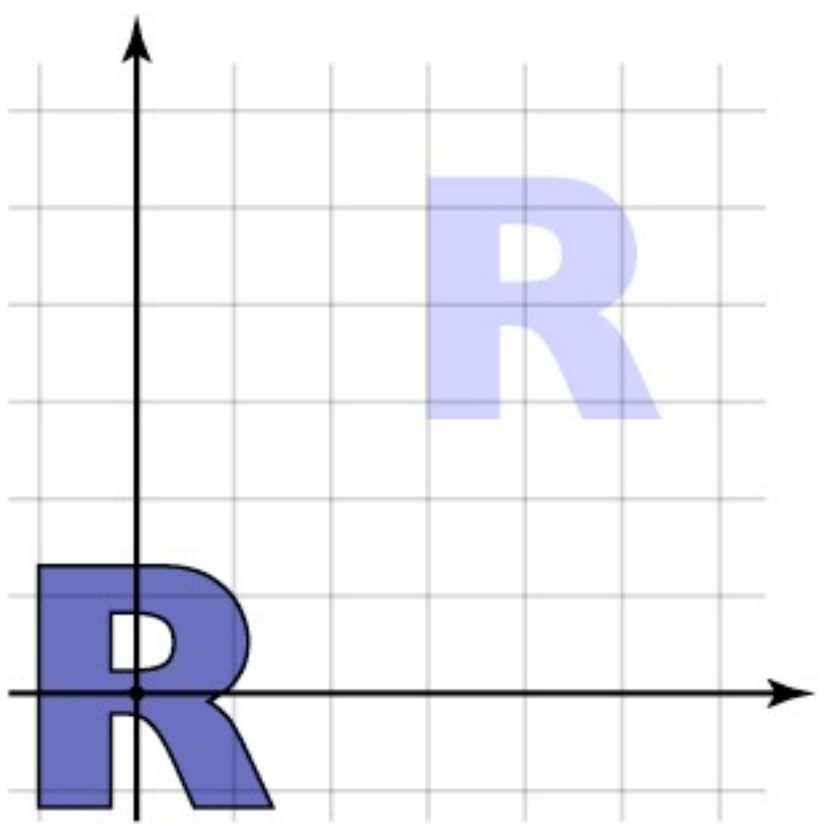


$$M = T^{-1}RT$$

Reminder: R = $\begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$

Exercise:
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

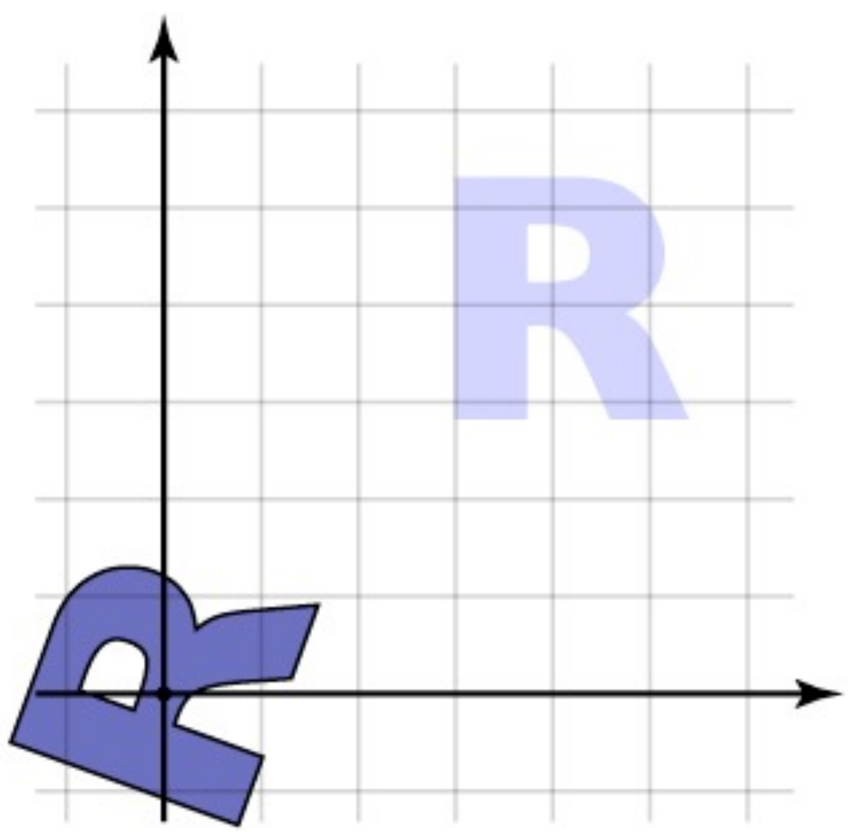


$$M = T^{-1}RT$$

Reminder: R = $\begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$

Exercise:
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

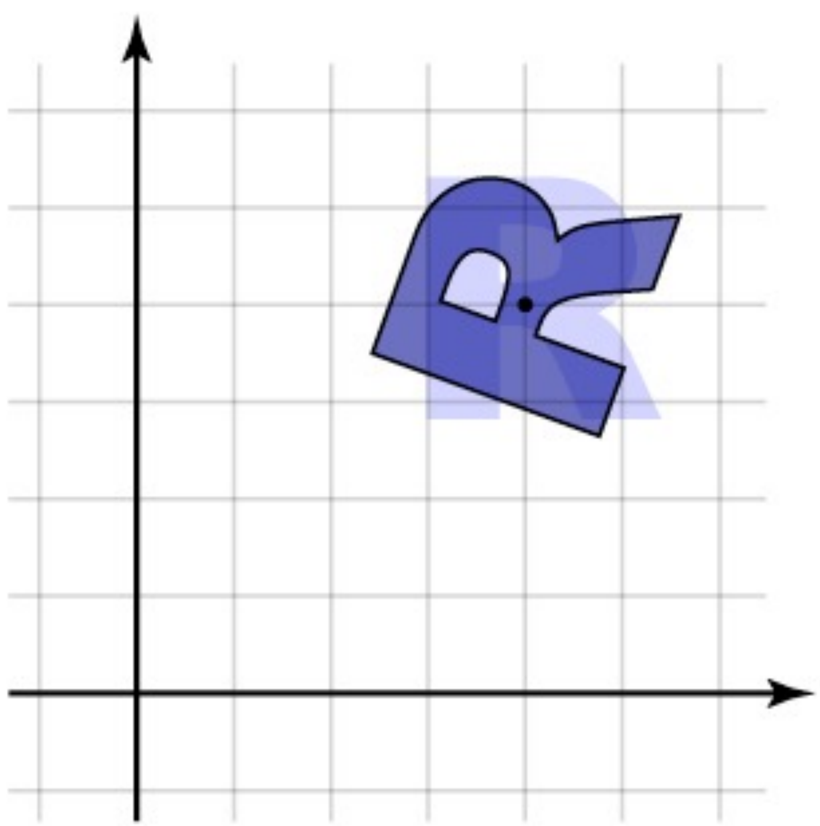


$$M = T^{-1} R T$$

Reminder: R = $\begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$

Exercise:
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



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

Reminder: R = $\begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$

Exercise: Rotate around not-the-origin

- Coordinate Frame interpretation:

  1. Move to origin: change to a new frame w/ origin at **p**

  2. Rotate around origin in that frame

  3. Move back to **p**: change from frame back to canonical

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*

How do we find F^{-1} ?

- Can always invert a matrix algebraically
- Simple cases can be done geometrically:
  - translation: negate tx, ty
  - rotation: rotate by -theta
  - scale: scale by 1/s

How do we find F^{-1} ?

- Rigid transformations: $R = \begin{bmatrix} Q & \mathbf{u} \\ 0 & 1 \end{bmatrix}$
- Linear part (Q) is orthogonal matrix
  - $Q^{-1} = Q^T$
- Inverse can be derived:

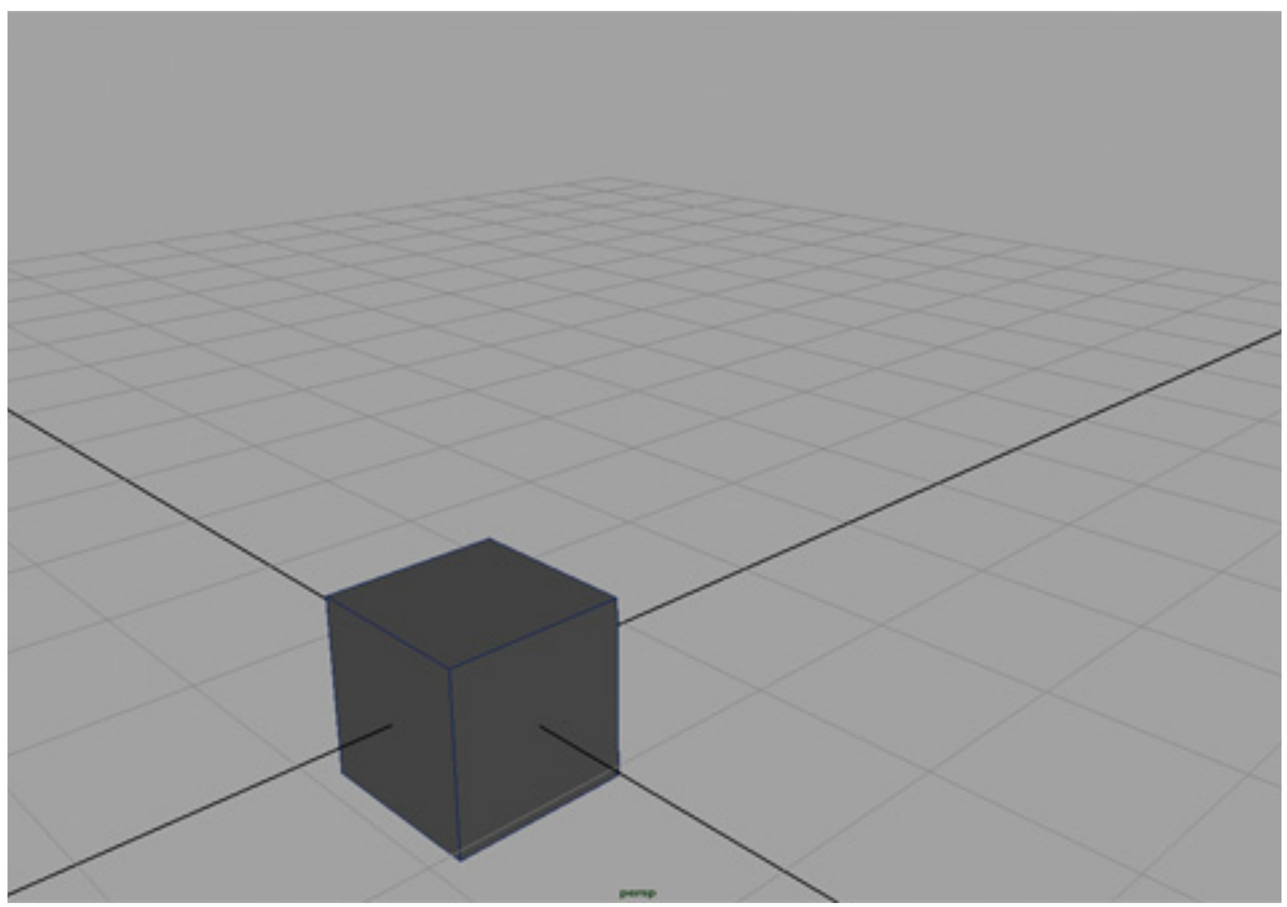
$$R^{-1}R = \begin{bmatrix} Q^T & -Q^T\mathbf{u} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} Q & \mathbf{u} \\ 0 & 1 \end{bmatrix}$$

Transformations in 3D

- Pretty much the same stuff
    - but with one additional D

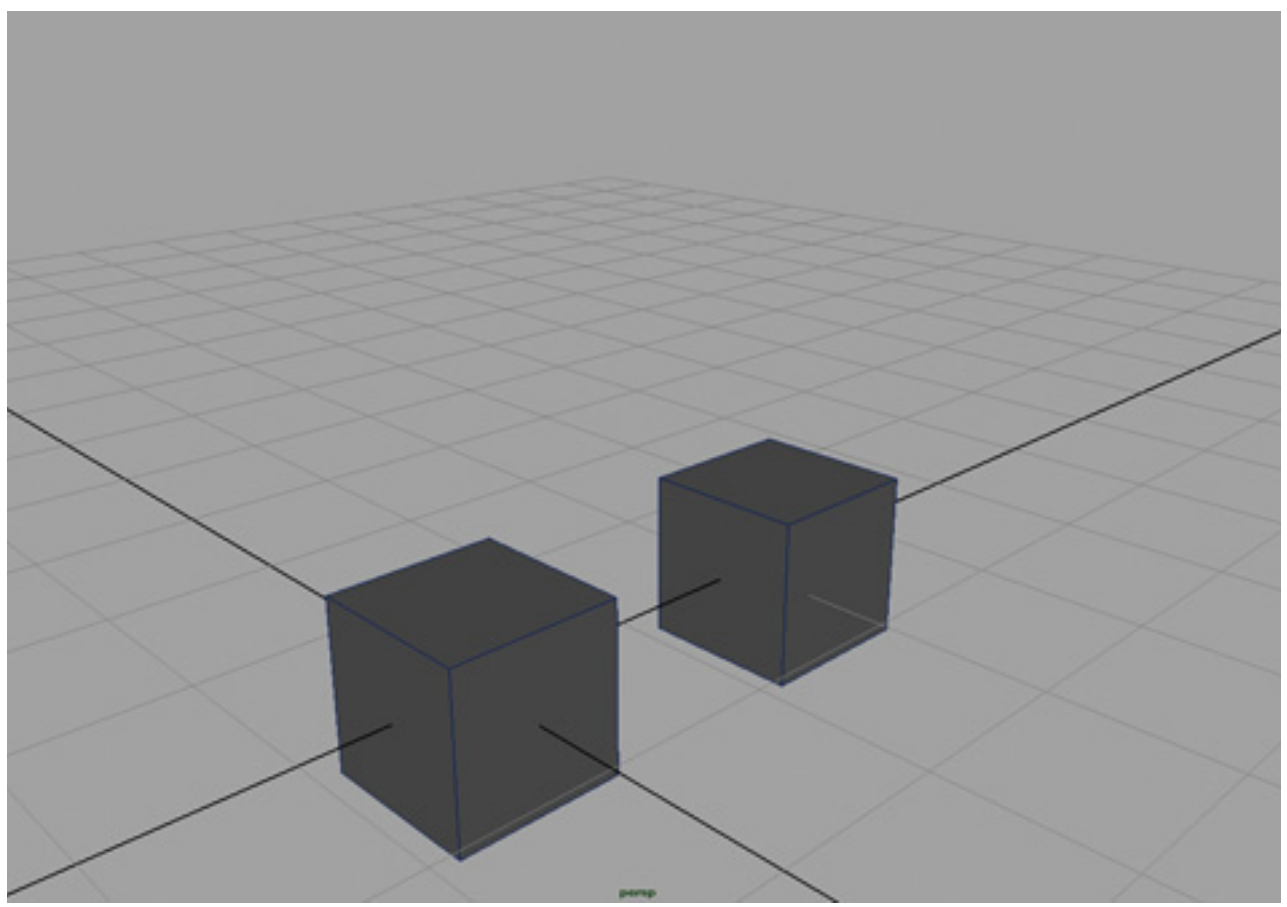
Translation

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



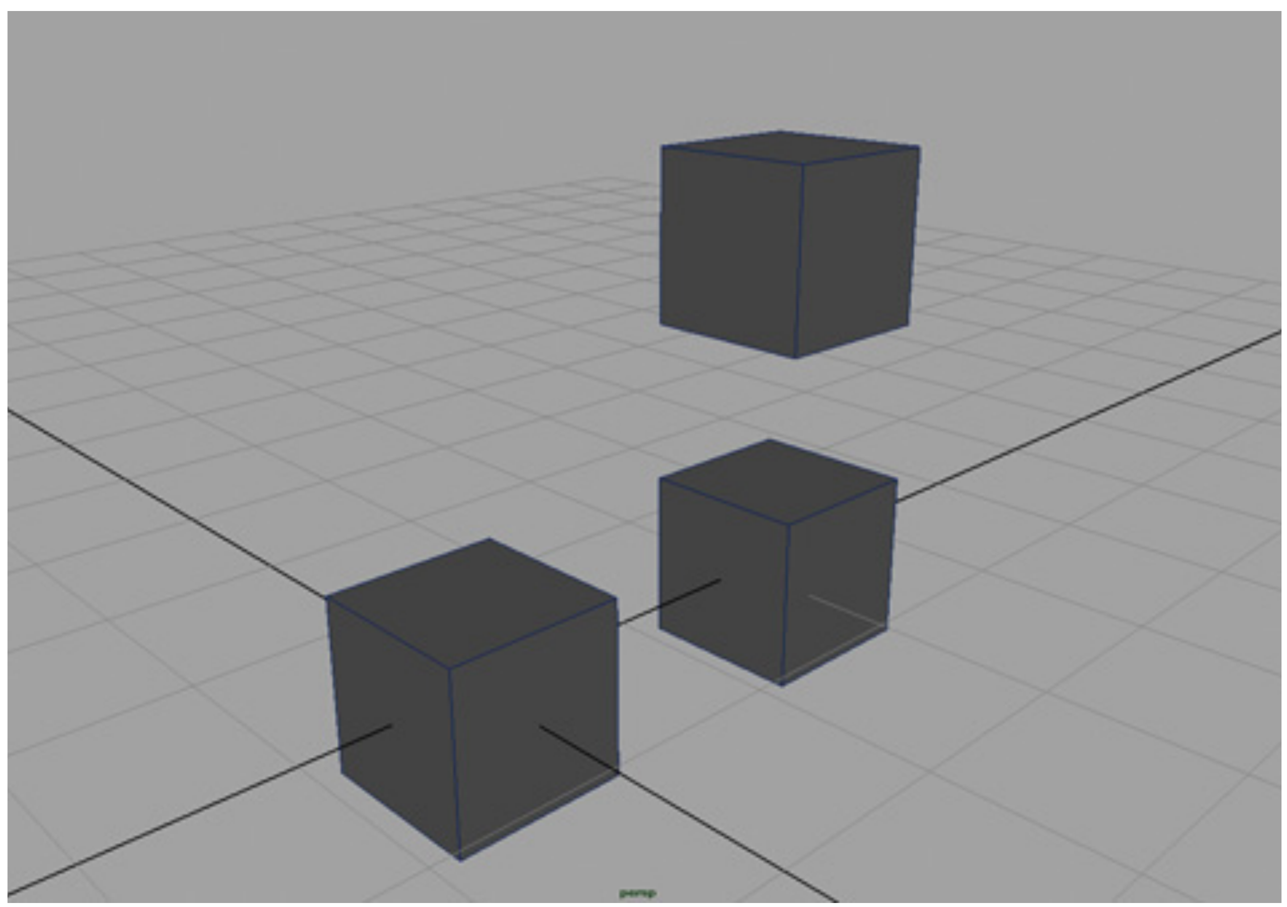
Translation

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



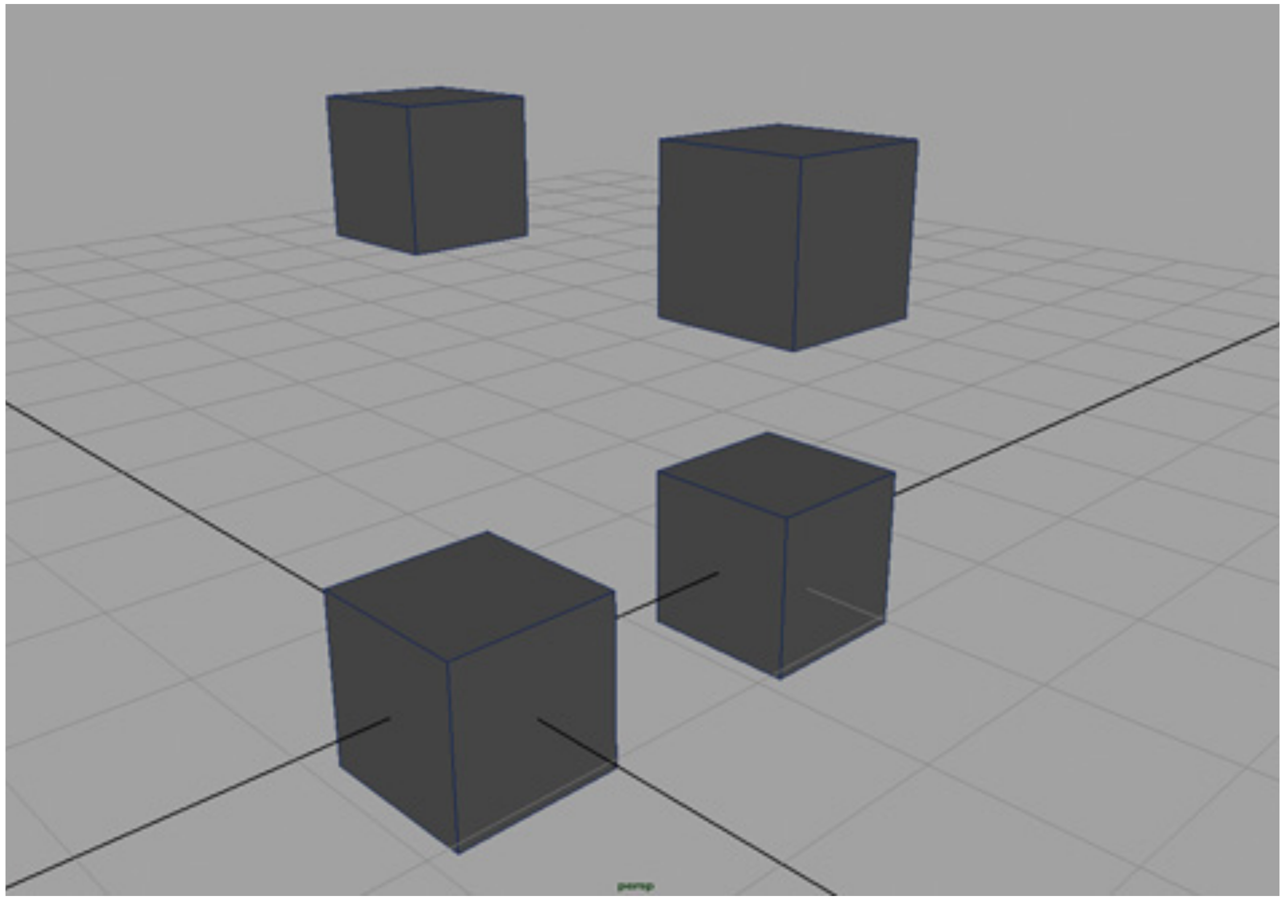
Translation

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



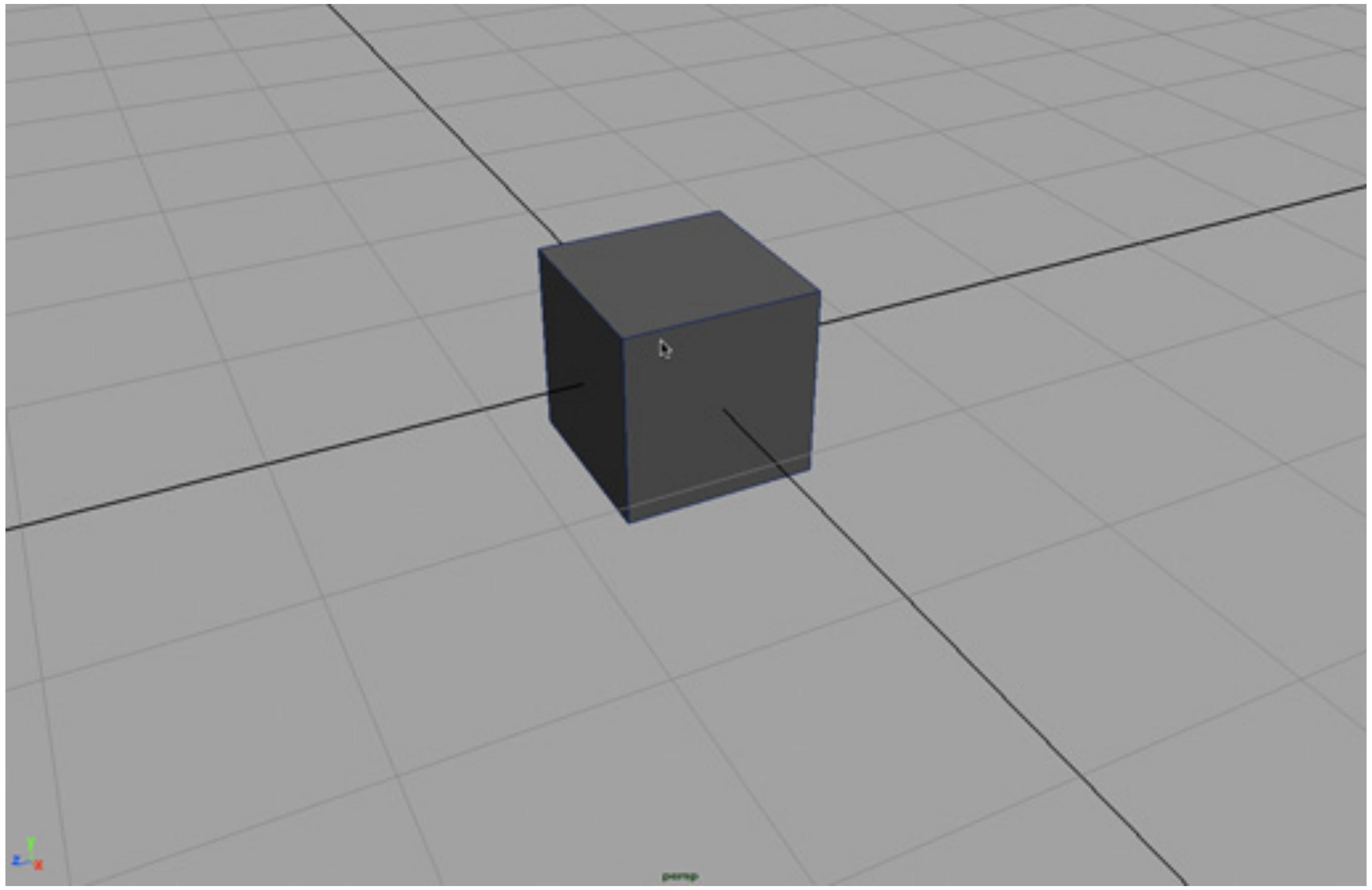
Translation

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



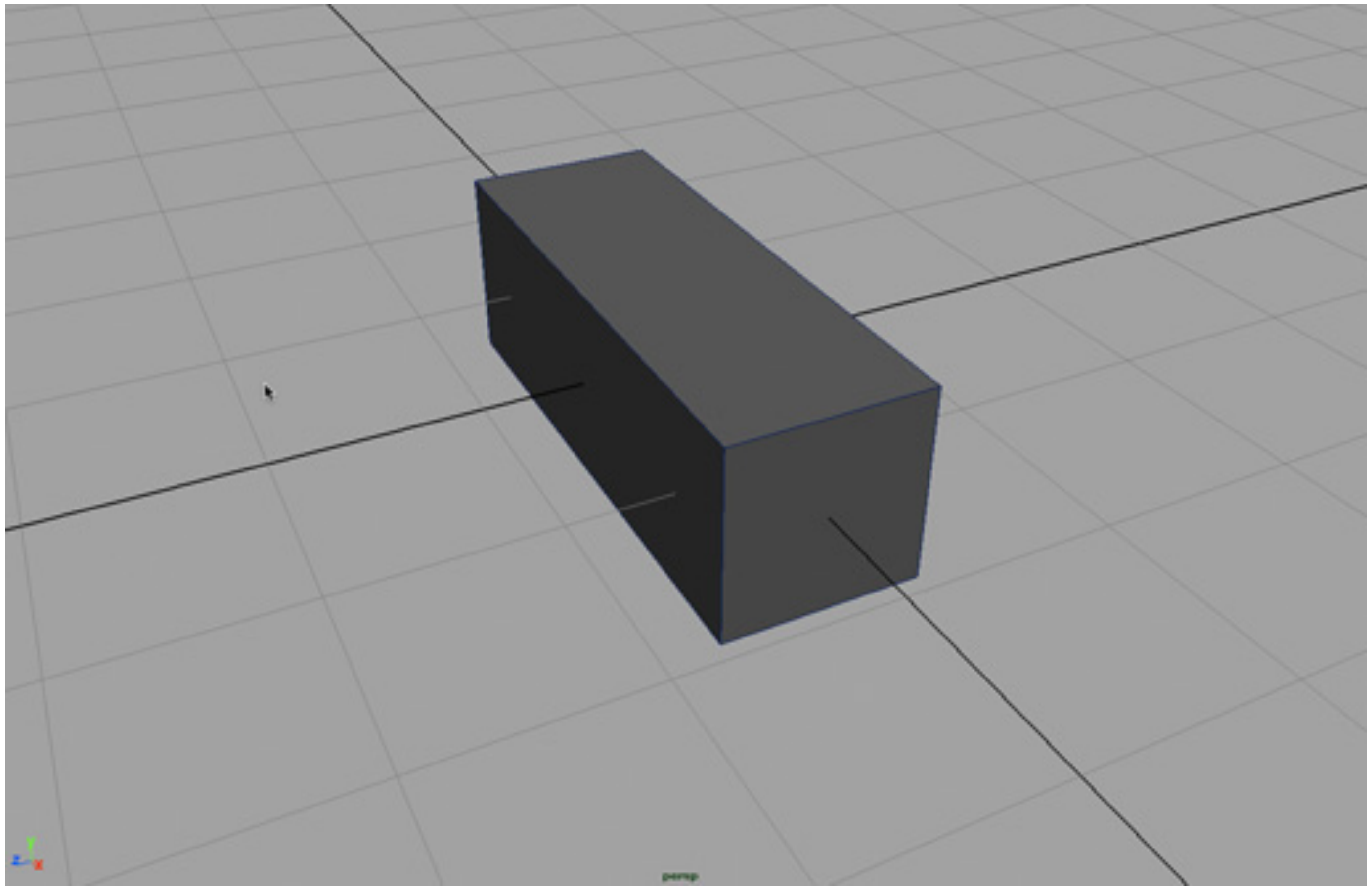
Scaling

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



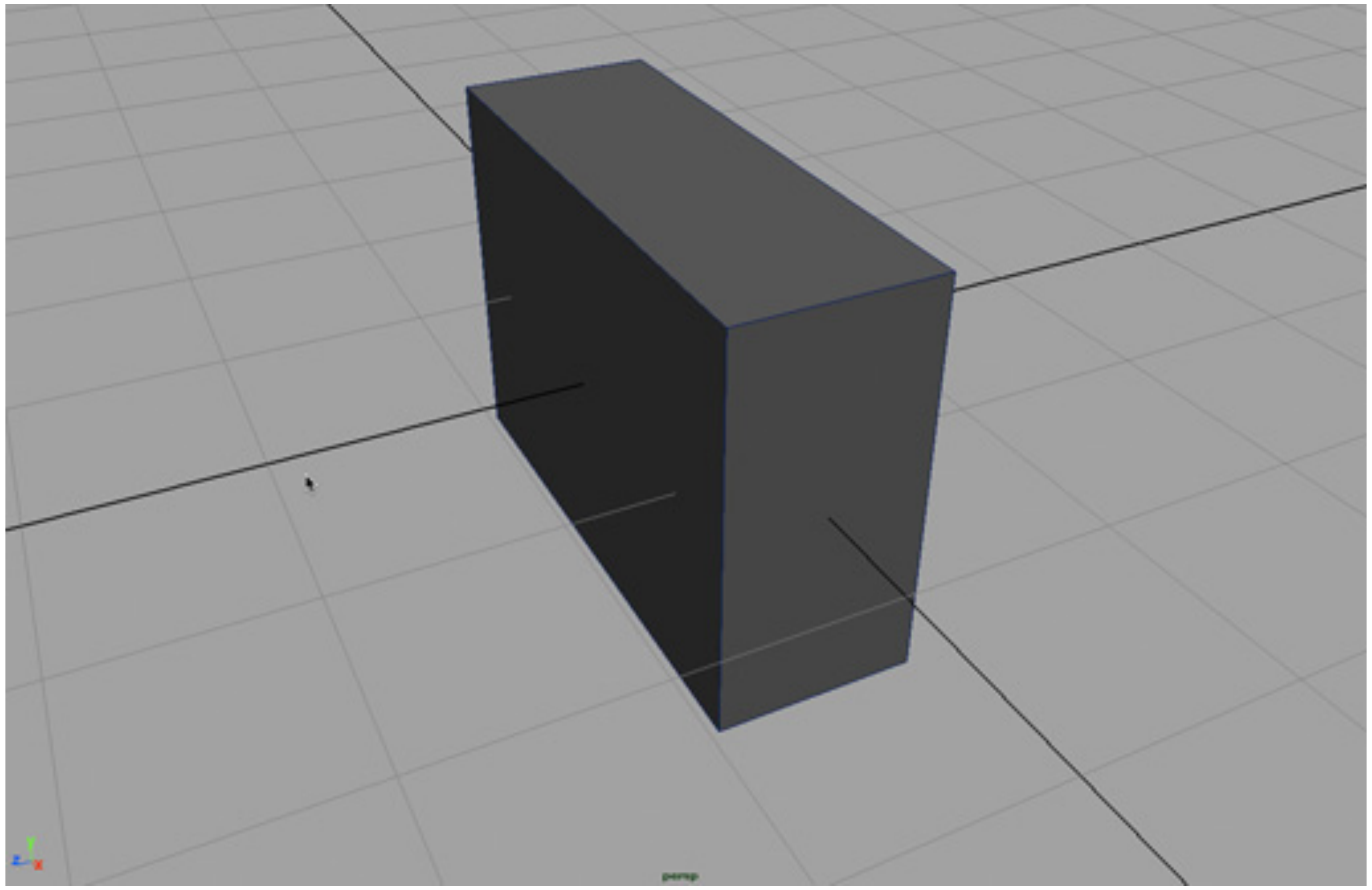
Scaling

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



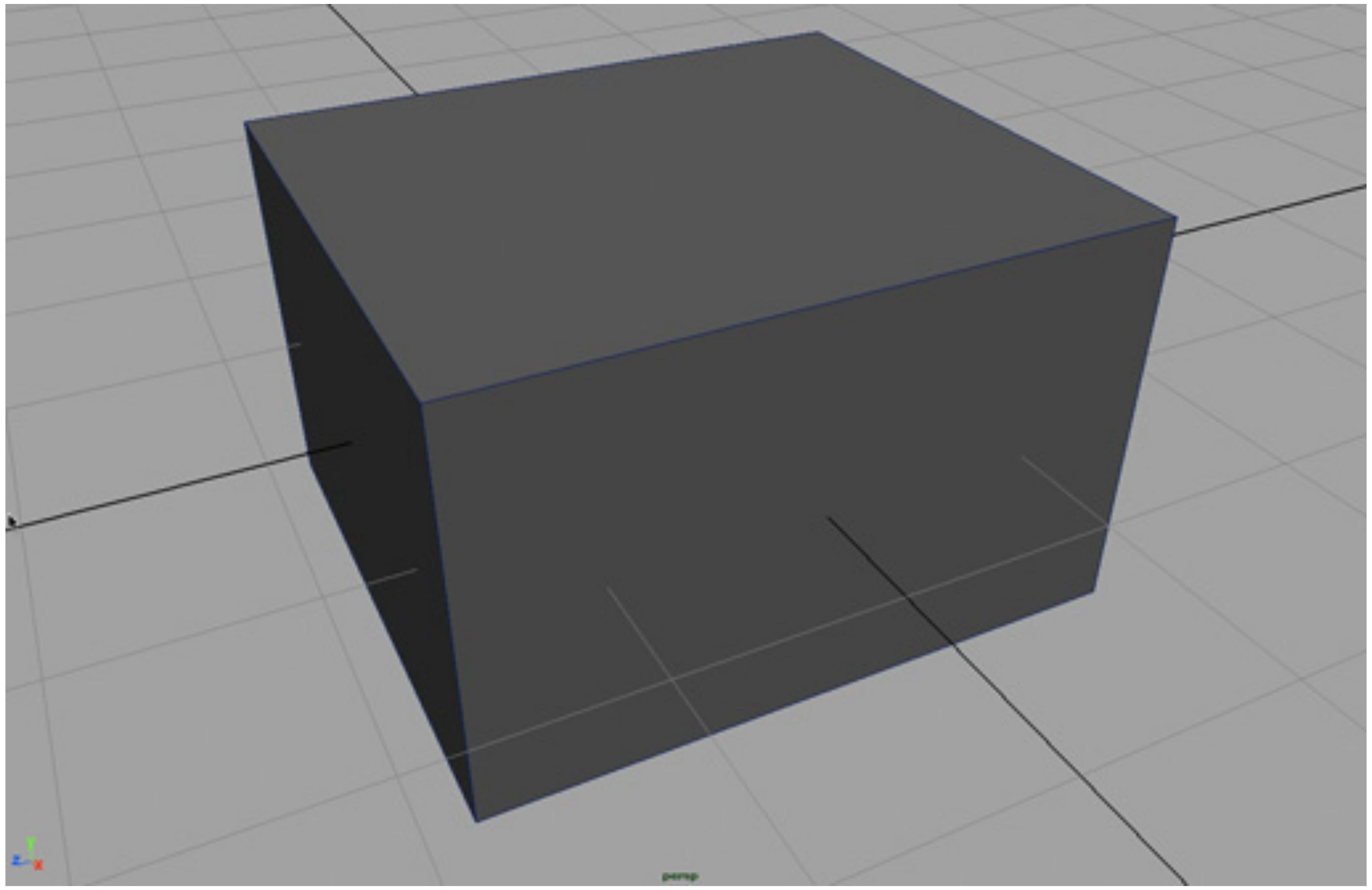
Scaling

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



Scaling

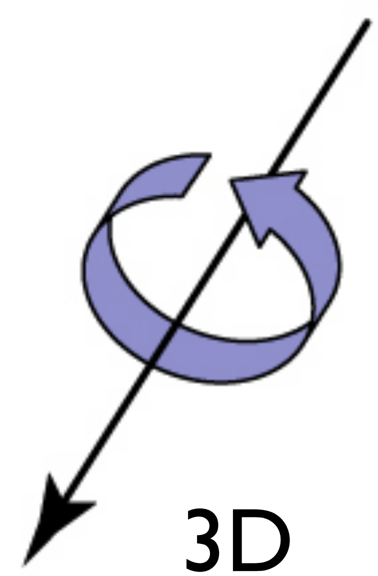
$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$



Rotations: A bit different

- 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

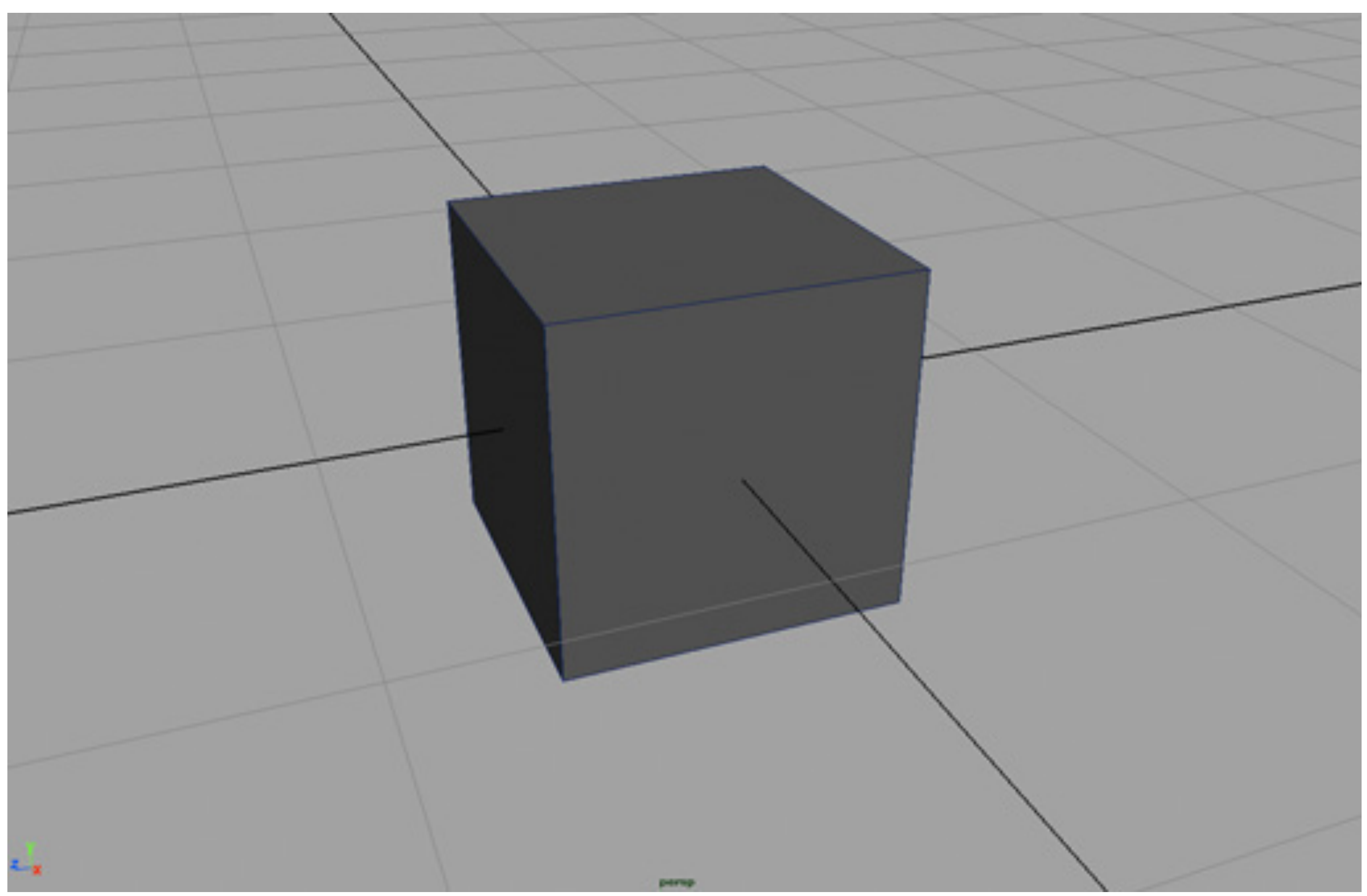
2D



3D

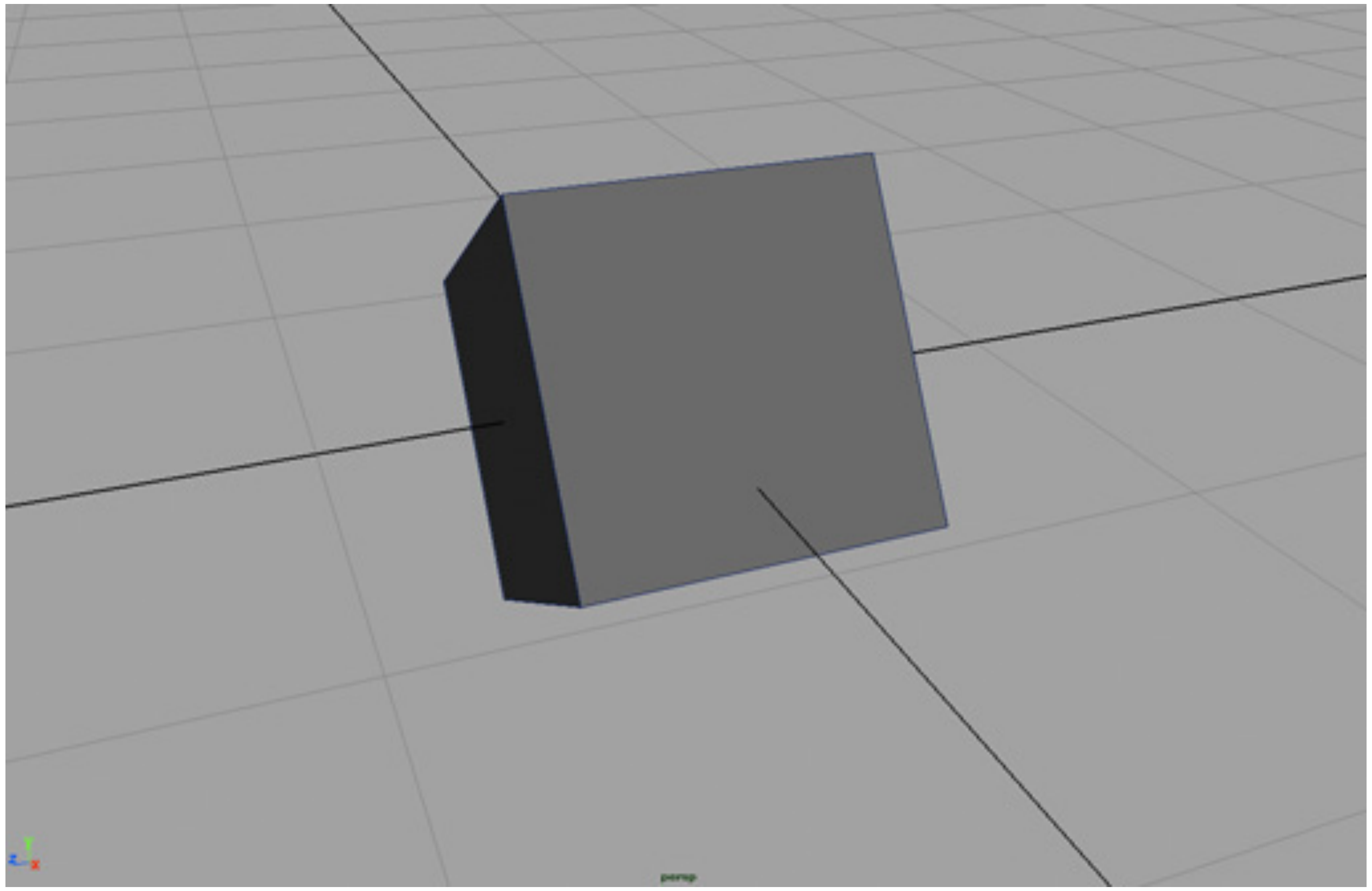
Rotation about z axis

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



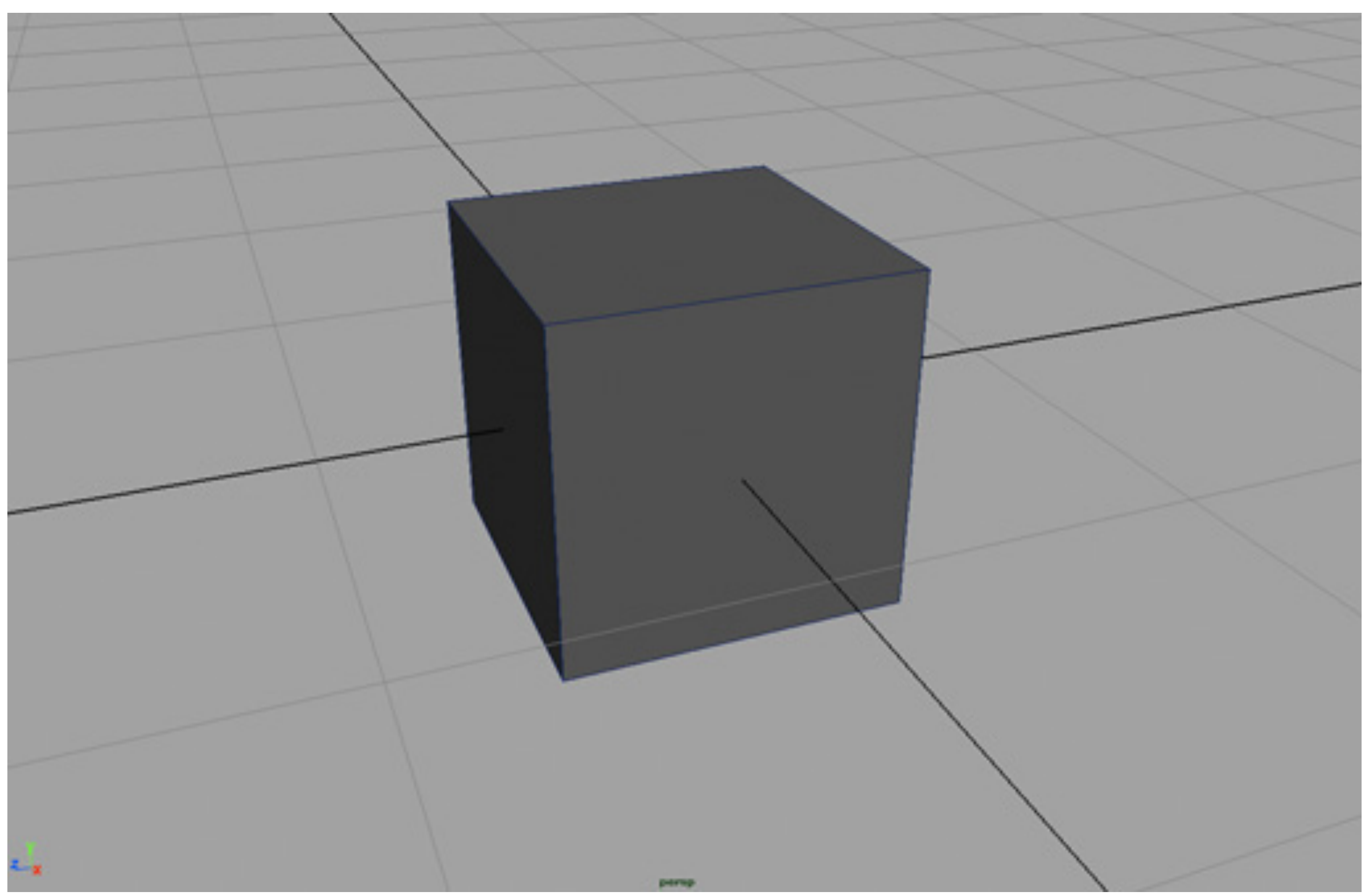
Rotation about z axis

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



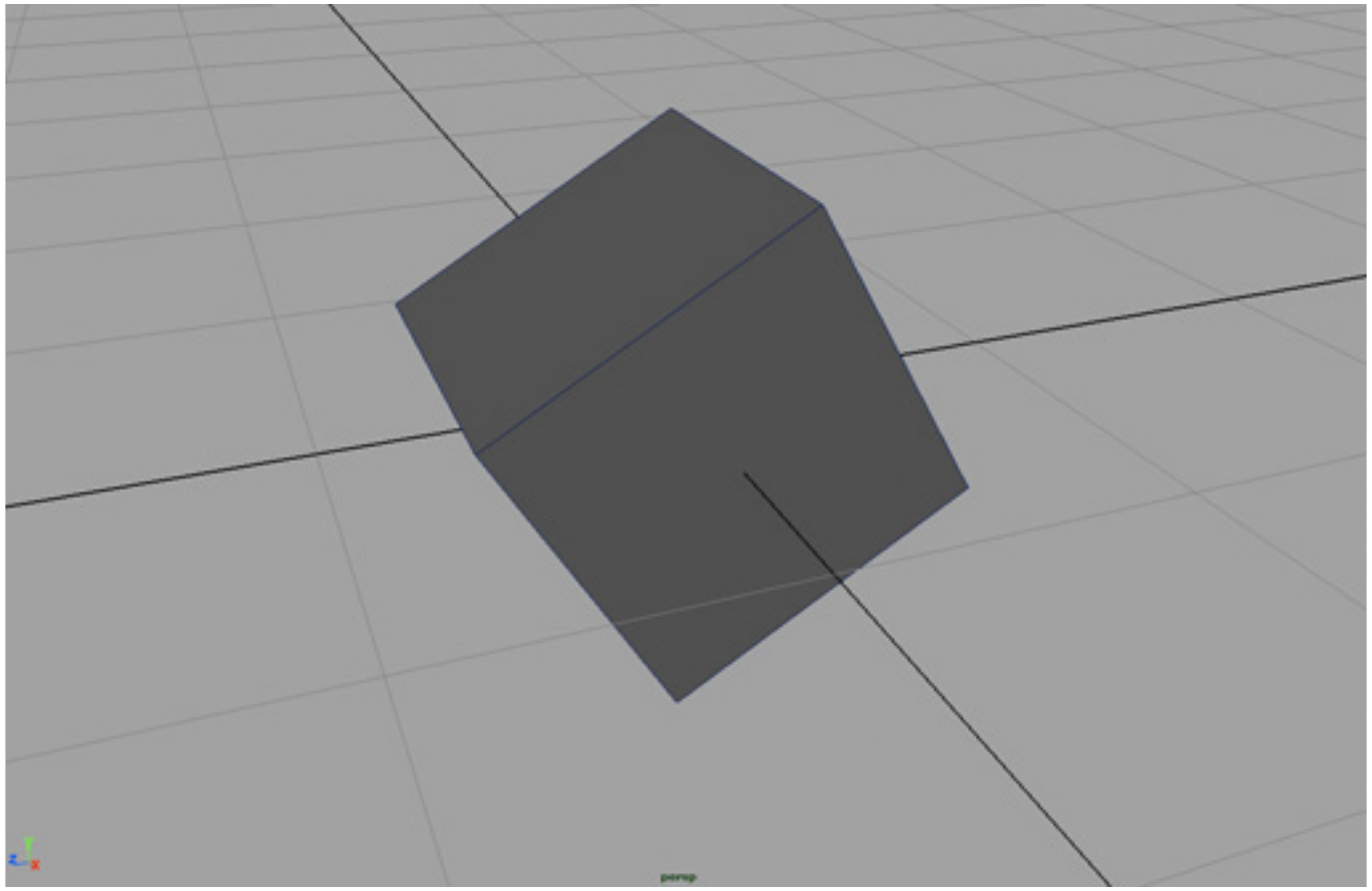
Rotation about *x* axis

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



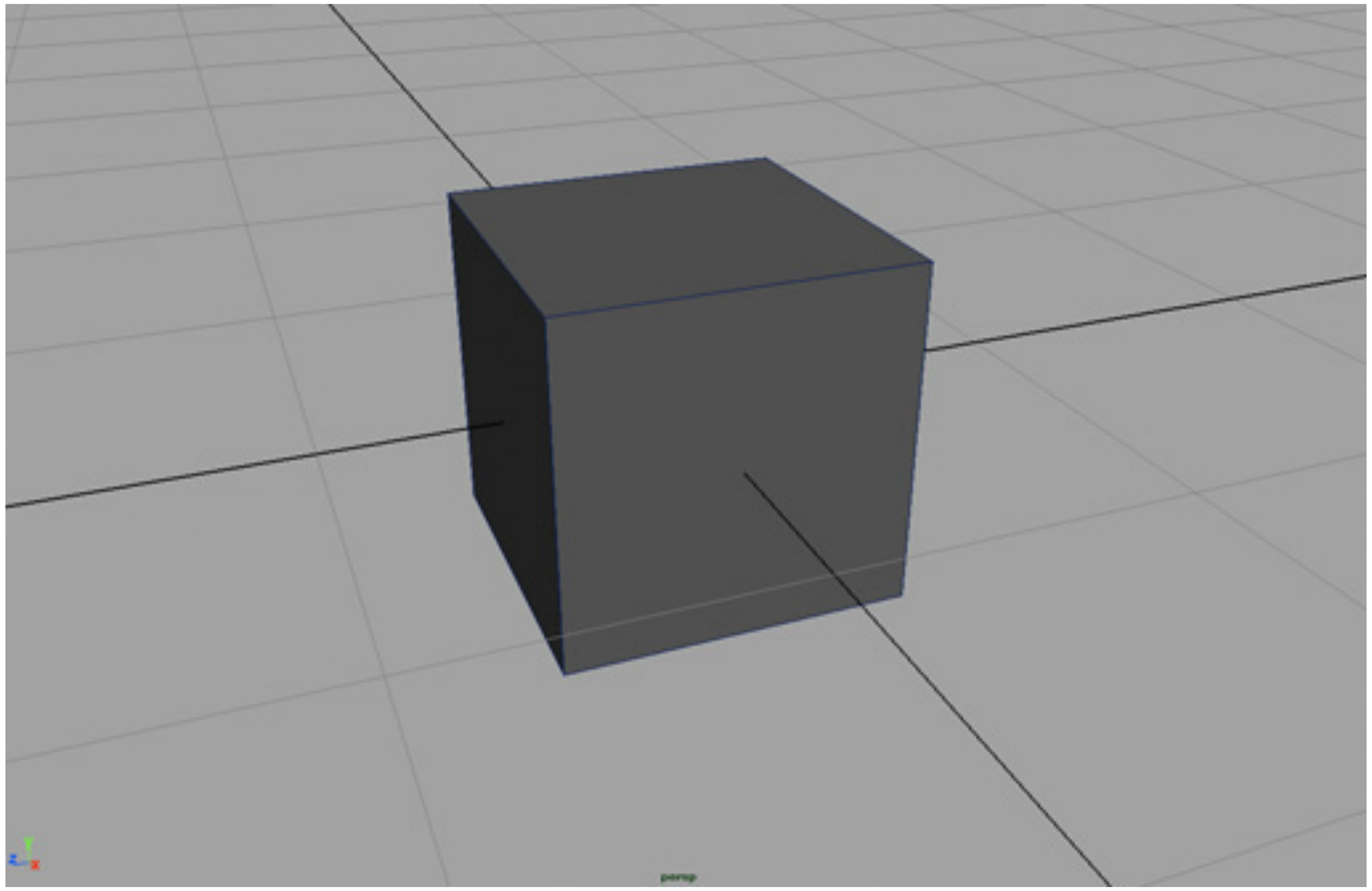
Rotation about *x* axis

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



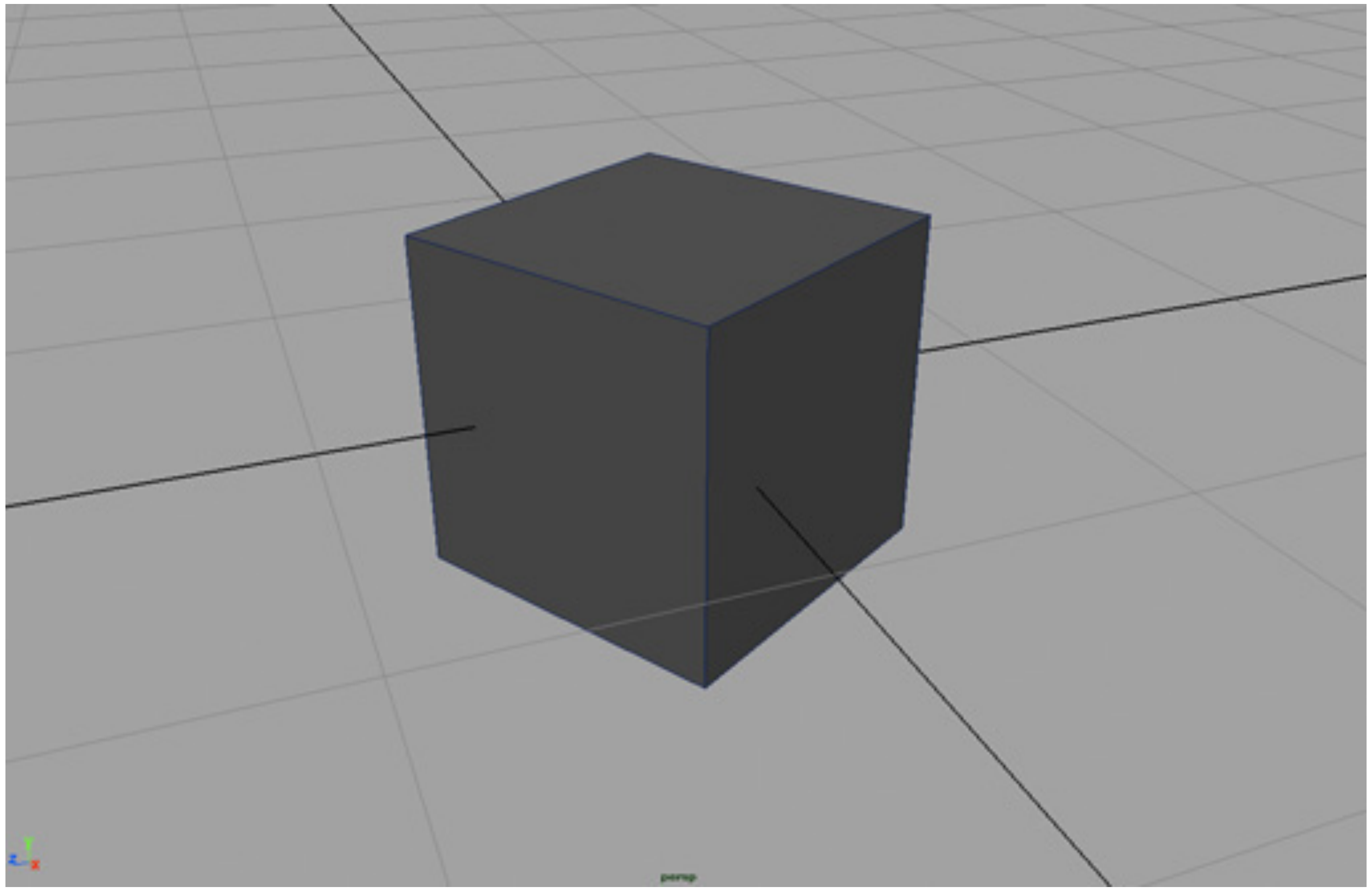
Rotation about y axis

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



Rotation about y axis

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



Rotations around an arbitrary axis

- Tricky - many ways to describe them:
  - Euler angles: 3 rotations about 3 axes
  - (Axis, angle)
  - Quaternions
- Simplest conceptually: indirectly specify via coordinate frame transformations.
  - We did this when finding a camera basis!

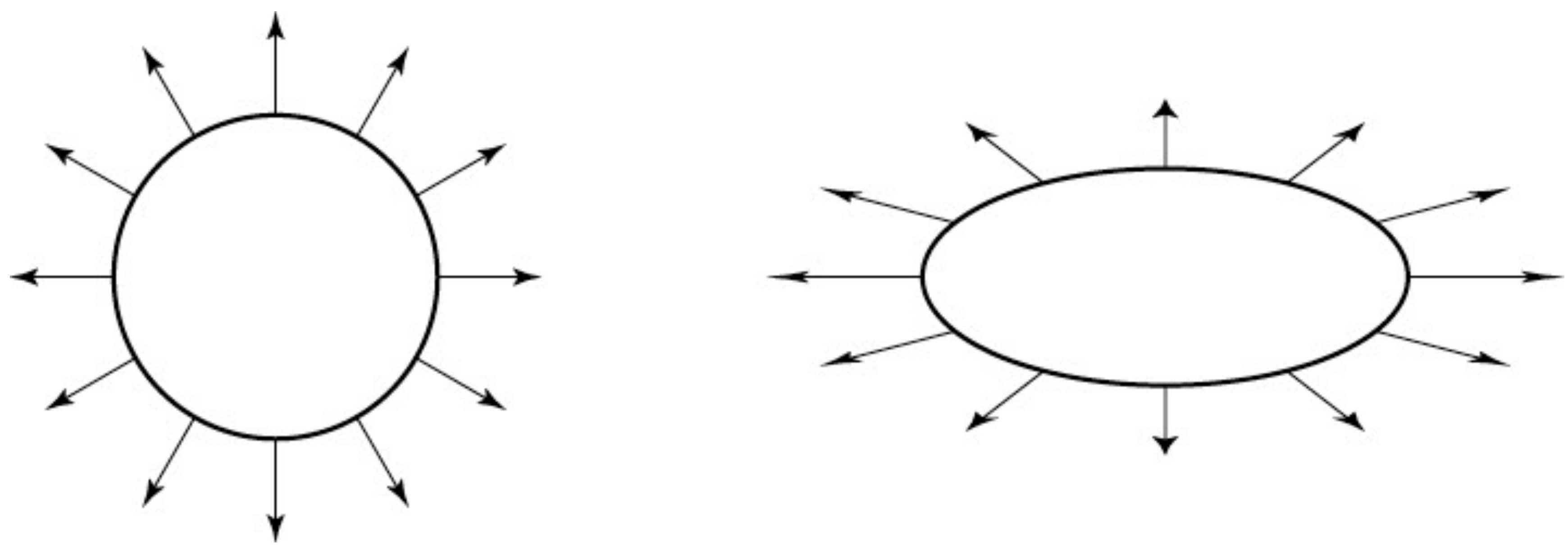
Rotations around an arbitrary axis

- Simplest conceptually: indirectly specify via coordinate frame transformations.
  - We did this when finding a camera basis!
- Simplest practically: type in a formula from wikipedia:

$$R = \begin{bmatrix} \cos\theta + u_x^2\left(1-\cos\theta\right) & u_x u_y\left(1-\cos\theta\right) - u_z\sin\theta & u_x u_z\left(1-\cos\theta\right) + u_y\sin\theta \\ u_y u_x\left(1-\cos\theta\right) + u_z\sin\theta & \cos\theta + u_y^2\left(1-\cos\theta\right) & u_y u_z\left(1-\cos\theta\right) - u_x\sin\theta \\ u_z u_x\left(1-\cos\theta\right) - u_y\sin\theta & u_z u_y\left(1-\cos\theta\right) + u_x\sin\theta & \cos\theta + u_z^2\left(1-\cos\theta\right) \end{bmatrix}$$

Transforming normal vectors

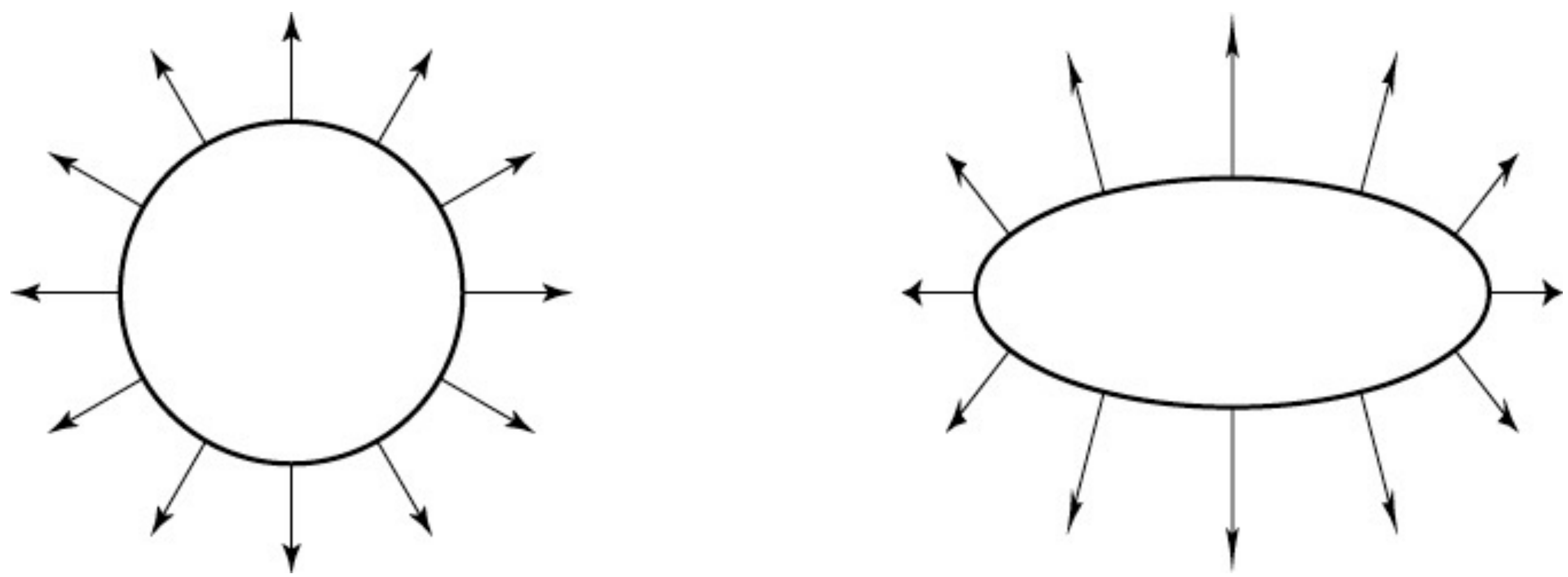
- Transforming surface normals
  - differences of points (and therefore tangents) transform OK
  - normals do not --> use inverse transpose matrix



have: $\mathbf{t} \cdot \mathbf{n} = \mathbf{t}^T \mathbf{n} = 0$
want: $M\mathbf{t} \cdot X\mathbf{n} = \mathbf{t}^T M^T X \mathbf{n} = 0$
so set $X = (M^T)^{-1}$
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 --> use inverse transpose matrix



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