# Affine transformations

**Brian Curless**
**CSE 457**
**Spring 2017**

## Reading

Optional reading:

- Angel 3.1, 3.7-3.11
- Angel, the rest of Chapter 3
- Foley, et al, Chapter 5.1-5.5.
- David F. Rogers and J. Alan Adams, *Mathematical Elements for Computer Graphics*, 2nd Ed., McGraw-Hill, New York, 1990, Chapter 2.

# Geometric transformations

Geometric transformations will map points in one space to points in another: $(x', y', z') = \mathbf{f}(x, y, z)$.

These transformations can be very simple, such as scaling each coordinate, or complex, such as non-linear twists and bends.

We'll focus on transformations that can be represented easily with matrix operations.

# Vector representation

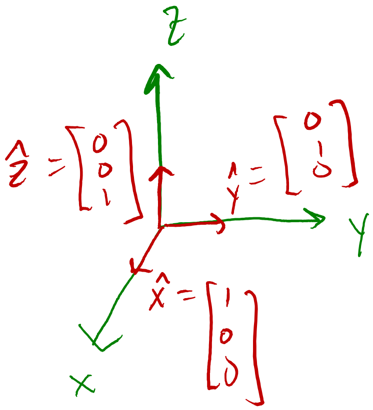
We can represent a **point**, **p** = ($x$, $y$), in the plane or **p** = ($x$, $y$, $z$) in 3D space

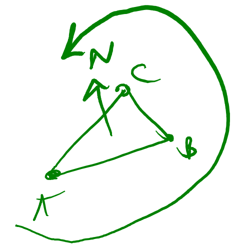
- as column vectors $\begin{bmatrix} x \\ y \end{bmatrix}$ $\begin{bmatrix} x \\ y \\ z \end{bmatrix}$

- ~~as row vectors~~ $\cancel{\begin{bmatrix} x & y \end{bmatrix}}$ $\cancel{\begin{bmatrix} x & y & z \end{bmatrix}}$

# Canonical axes

right-handed coord. systems



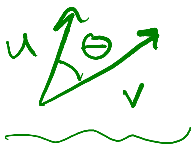

A, B, C

$$V = \begin{bmatrix} v_x \\ v_y \\ v_z \end{bmatrix} = \underbrace{v_x \hat{x}} + v_y \hat{y} + v_z \hat{z}$$

$$v_x \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} v_x \\ 0 \\ 0 \end{bmatrix}$$

$$\hat{x} \times \hat{y} = \hat{z}$$

# Vector length and dot products




$$v = \begin{bmatrix} v_x \\ v_y \\ v_z \end{bmatrix} \qquad \|v\| = \sqrt{v_x^2 + v_y^2 + v_z^2}$$

$$u \cdot v \equiv u_x v_x + u_y v_y + u_z v_z$$

$$\|v\|^2 = v \cdot v$$

$$u \cdot v = u^T v = \begin{bmatrix} u_x & u_y & u_z \end{bmatrix} \begin{bmatrix} v_x \\ v_y \\ v_z \end{bmatrix}$$

$$v \cdot u = v^T u$$

$$u \cdot v \overset{?}{=} v \cdot u \ \checkmark$$

$$\hat{v} = \frac{v}{\|v\|}$$

$$\|\hat{v}\| = 1$$

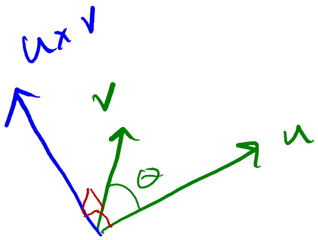
$$u \cdot v = \|u\|\|v\| \cos\theta$$

$$u \cdot v = 0 \Rightarrow u \perp v \quad (\theta = 90^\circ)$$

$$\{u \neq 0, v \neq 0\}$$

$$\hat{u} \cdot \hat{v} = \cos\theta$$

# Vector cross products

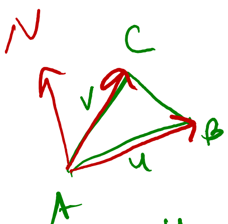



$$u \times v \overset{\text{det}}{=} \begin{bmatrix} \hat{x} & \hat{y} & \hat{z} \\ u_x & u_y & u_z \\ v_x & v_y & v_z \end{bmatrix} = (u_y v_z - u_z v_y)\hat{x} + (u_z v_x - u_x v_z)\hat{y} + (u_x v_y - u_y v_x)\hat{z} = \begin{bmatrix} u_y v_z - u_z v_y \\ \cdot \\ \cdot \end{bmatrix}$$

$u, v \perp u \times v$

$(u \times v) \cdot u = 0$

$(u \times v) \cdot v = 0$




$$N = \overset{u}{(B-A)} \times \overset{v}{(C-A)}$$

$$\hat{N} = \frac{N}{\|N\|}$$

$u \times v = -v \times u$

$u \times v = \|u\| \|v\| \sin\theta \, \hat{N}$

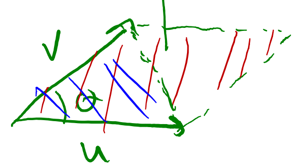
$\|u \times v\| = \|u\| \|v\| \, |\sin\theta|$

$u \times v = 0 \Rightarrow u \parallel v$

$u \neq 0, v \neq 0$ $\quad$ $u = \alpha v$

$\|u \times v\| = \text{Area}(\text{▱}_{u,v})$

$\frac{1}{2}\|u \times v\| = \text{Area}(\Delta_{u,v})$




$\Delta_{u,v}$

# Representation, cont.

$$(AB)^T = B^T A^T$$

We can represent a **2-D transformation** $M$ by a matrix

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

$$(AB)^{-1} =$$

$$A \cdot A^{-1} = I$$

If **p** is a column vector, $M$ goes on the left:

$$\left.\begin{aligned} (\mathbf{p}' &= M\mathbf{p})^T \\ \begin{bmatrix} x' \\ y' \end{bmatrix} &= \begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} ax + by \\ cx + dy \end{bmatrix} \end{aligned}\right\} \text{us}$$

If **p** is a row vector, $M^T$ goes on the right:

$$\mathbf{p}' = \mathbf{p}M^T$$

$$\begin{bmatrix} x' & y' \end{bmatrix} = \begin{bmatrix} x & y \end{bmatrix} \begin{bmatrix} a & c \\ b & d \end{bmatrix}$$

$$(AB)^{-1}(AB) = I$$

$$(AB)^{-1} AB = I$$

$$(AB)^{-1} A = B^{-1}$$

$$(AB)^{-1} = B^{-1} A^{-1}$$

We will use **column vectors**.

## Two-dimensional transformations

Here's all you get with a 2 x 2 transformation matrix $M$:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

So:

$$\begin{aligned} x' &= ax + by \\ y' &= cx + dy \end{aligned}$$

We will develop some intimacy with the elements *a, b, c, d*…

# Identity

Suppose we choose $a = d = 1$, $b = c = 0$:

- Gives the **identity** matrix:

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

- Doesn't move the points at all

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} x \\ y \end{bmatrix}$$

# Scaling

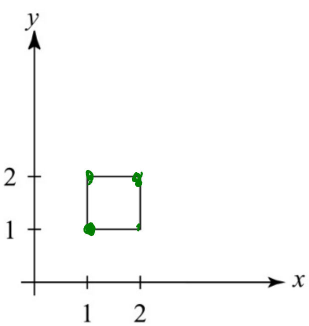
Suppose we set $b = c = 0$, but let $a$ and $d$ take on any *positive* value:

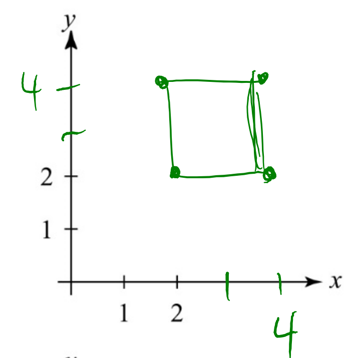
- Gives a **scaling** matrix:

$$\begin{bmatrix} a & 0 \\ 0 & d \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

- Provides **differential (non-uniform) scaling** in $x$ and $y$:

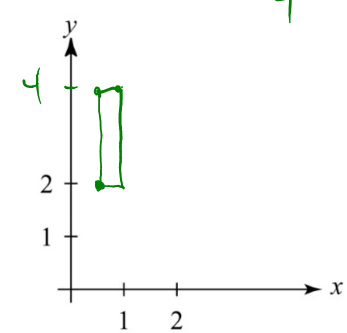
$$x' = ax$$
$$y' = dy$$




$$\underbrace{\begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix}}$$

$x' = 2x$
$y' = 2y$




$$\begin{bmatrix} 1/2 & 0 \\ 0 & 2 \end{bmatrix}$$

$x' = \frac{1}{2}x$
$y' = 2\cancel{x}y$

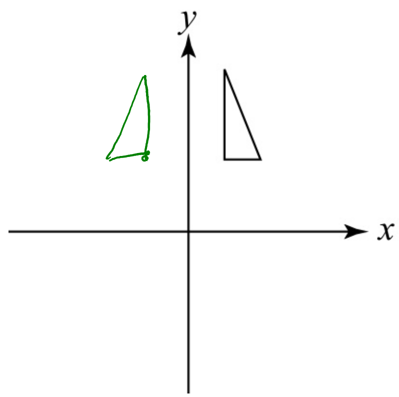
## Mirror or reflection

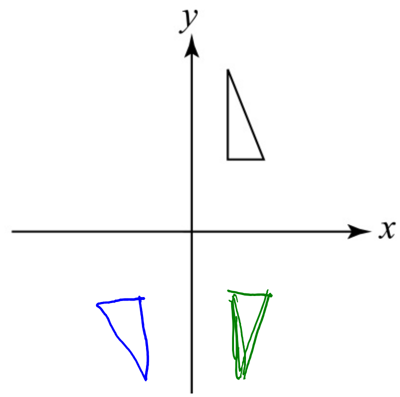
Suppose we keep $b=c=0$, but let either $a$ or $d$ go negative.

Examples:

$$\begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix}\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} -x \\ y \end{bmatrix} \qquad \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$$

$$\underbrace{\begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix}\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} = \begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix}}_{\text{rotation by } 180^\circ}$$



## Shear

Now let's leave $a = d = 1$ and experiment with $b$...

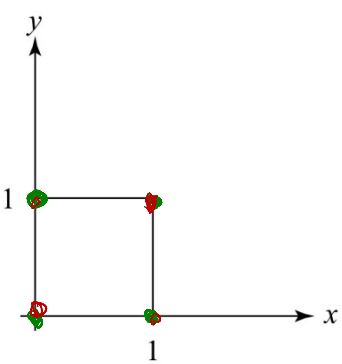
The matrix

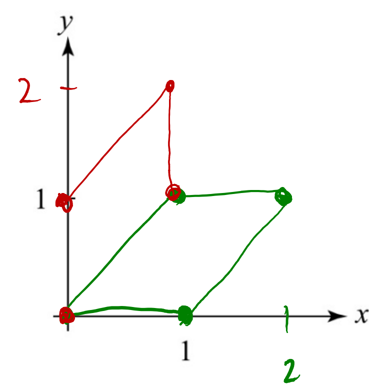
$$\begin{bmatrix} 1 & b \\ 0 & 1 \end{bmatrix}$$

gives:

$$\begin{aligned} x' &= x + by \\ y' &= y \end{aligned}$$

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




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

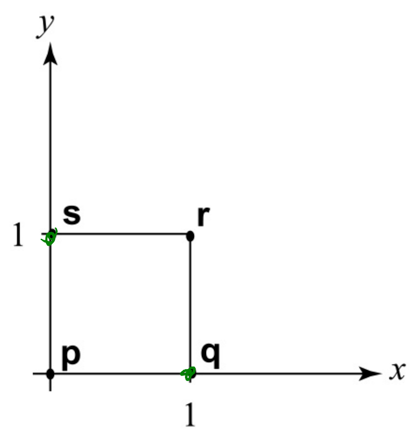
# Effect on unit square

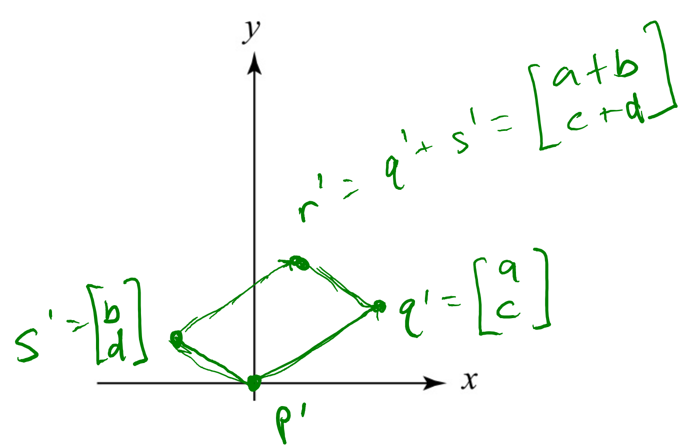
Let's see how a general 2 x 2 transformation *M* affects the unit square:

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} \mathbf{p} & \mathbf{q} & \mathbf{r} & \mathbf{s} \end{bmatrix} = \begin{bmatrix} \mathbf{p'} & \mathbf{q'} & \mathbf{r'} & \mathbf{s'} \end{bmatrix}$$

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 0 & a & a+b & b \\ 0 & c & c+d & d \end{bmatrix}$$

p q r s p' q' r' s'



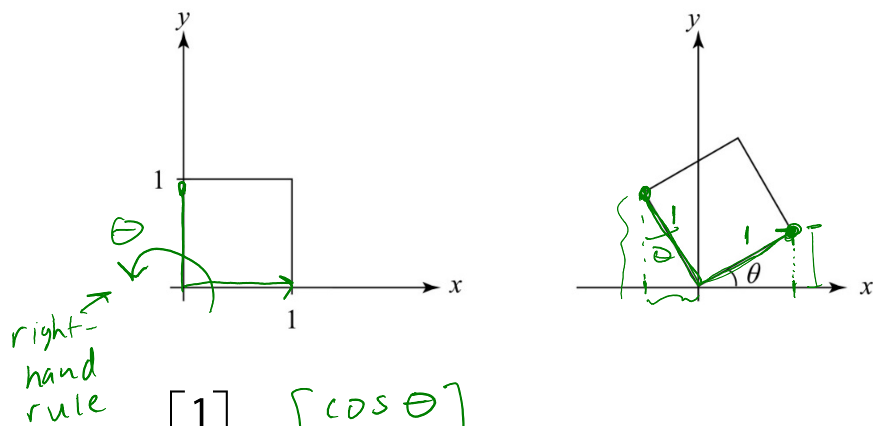
## Effect on unit square, cont.

Observe:

- Origin invariant under $M$
- $M$ can be determined just by knowing how the corners (1,0) and (0,1) are mapped
- $a$ and $d$ give $x$- and $y$-scaling
- $b$ and $c$ give $x$- and $y$-shearing

# Rotation

From our observations of the effect on the unit square, it should be easy to write down a matrix for “rotation about the origin”:




$$\begin{bmatrix} 1 \\ 0 \end{bmatrix} \rightarrow \begin{bmatrix} \cos\theta \\ \sin\theta \end{bmatrix}$$

$$\begin{bmatrix} 0 \\ 1 \end{bmatrix} \rightarrow \begin{bmatrix} -\sin\theta \\ \cos\theta \end{bmatrix}$$

Thus,

$$M = R(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$$

# Limitations of the 2 x 2 matrix

A 2 x 2 linear transformation matrix allows

- Scaling
- Rotation
- Reflection
- Shearing

**Q**: What important operation does that leave out?

translation

# Homogeneous coordinates

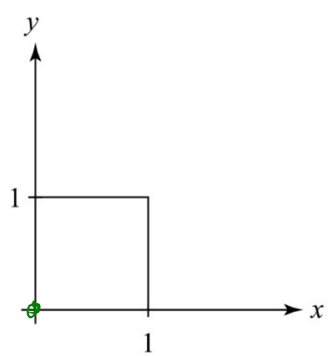
Idea is to loft the problem up into 3-space, adding a third component to every point:

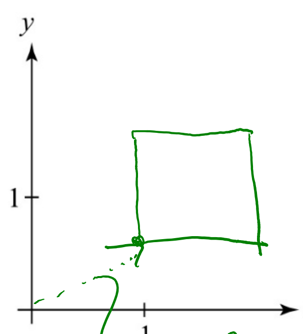
$$\begin{bmatrix} x \\ y \end{bmatrix} \rightarrow \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Adding the third "$w$" component puts us in **homogenous coordinates**.

And then transform with a 3 x 3 matrix:

$$\begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} = T(\mathbf{t}) \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} x + t_x \\ y + t_y \\ 1 \end{bmatrix}$$




$$\begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1/2 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} x + 1 \\ y + 1/2 \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} t_x \\ t_y \end{bmatrix} = \begin{bmatrix} 1 \\ 1/2 \end{bmatrix}$$

. . . gives **translation**!

# Anatomy of an affine matrix

The addition of translation to linear transformations gives us **affine transformations**.

In matrix form, 2D affine transformations always look like this:

$$M = \begin{bmatrix} a & b & t_x \\ c & d & t_y \\ 0 & 0 & 1 \end{bmatrix} = \left[\begin{array}{cc|c} \multicolumn{2}{c|}{A} & \mathbf{t} \\ \hline 0 & 0 & 1 \end{array}\right]$$

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

2D affine transformations always have a bottom row of [0 0 1].

$$\begin{bmatrix} A\begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix} \\ 1 \end{bmatrix}$$

An "affine point" is a "linear point" with an added *w*-coordinate which is always 1:

$$p_{lin} = \begin{bmatrix} x \\ y \end{bmatrix}$$

$$p_{aff} = \begin{bmatrix} x \\ y \end{bmatrix}$$

$$\mathbf{p}_{\text{aff}} = \begin{bmatrix} \mathbf{p}_{\text{lin}} \\ 1 \end{bmatrix} = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} A p_{lin} + t \\ 1 \end{bmatrix}$$

Applying an affine transformation gives another affine point:

$$M\mathbf{p}_{\text{aff}} = \begin{bmatrix} A\mathbf{p}_{\text{lin}} + \mathbf{t} \\ 1 \end{bmatrix}$$

## Rotation about arbitrary points

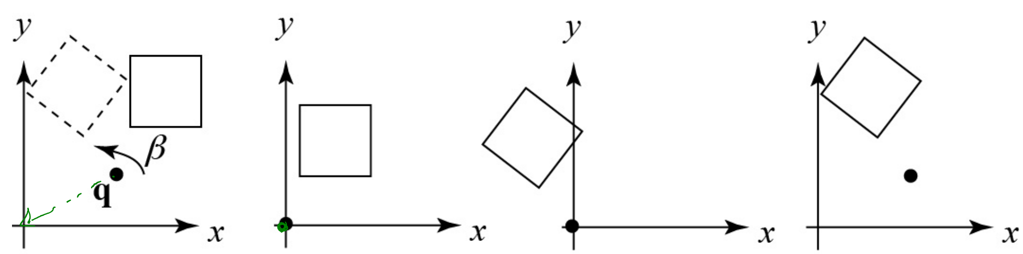
Until now, we have only considered rotation about the origin.

$$T(\mathbf{t}) = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix}$$

With homogeneous coordinates, you can specify a rotation by $\beta$, about any point $\mathbf{q} = [q_x \; q_y]^T$ with a matrix.

$$R(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Let's do this with rotation and translation matrices of the form R($\theta$) and T(**t**), respectively.



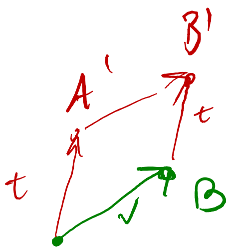

$M \neq \quad T(-q) \quad R(\beta) \quad T(q)$

1. Translate **q** to origin
2. Rotate
3. Translate back

$$\underbrace{T(q)\,R(\beta)\,T(-q)}_{M \,\checkmark} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

order is important!

## Points and vectors




$B = \begin{bmatrix} B_x \\ B_y \\ 1 \end{bmatrix}$

$A = \begin{bmatrix} A_x \\ B_y \\ 1 \end{bmatrix}$

Vectors have an additional coordinate of $w = 0$. Thus, a change of origin has no effect on vectors.

**Q**: What happens if we multiply a vector by an affine matrix?

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

$$B - A = \begin{bmatrix} B_x - A_x \\ B_y - A_y \\ 0 \end{bmatrix}$$

$\parallel$

$$B' - A' = \begin{bmatrix} B_x + t_x \\ B_y + t_y \\ 1 \end{bmatrix} - \begin{bmatrix} A_x + t_x \\ A_y + t_y \\ 1 \end{bmatrix} = \begin{bmatrix} B_x - A_x \\ B_y - A_y \\ 0 \end{bmatrix}$$

These representations reflect some of the rules of affine operations on points and vectors:

vector + vector → vector
scalar · vector → vector
point - point → vector
point + vector → point
point + point → chaos
scalar · vector + scalar · vector → vector
scalar · point + scalar · point → it depends

$$\alpha \begin{bmatrix} A_x \\ A_y \\ 1 \end{bmatrix} + \beta \begin{bmatrix} B_x \\ B_y \\ 1 \end{bmatrix} = \begin{bmatrix} \alpha A_x + \beta B_x \\ \alpha A_y + \beta B_y \\ \alpha + \beta \end{bmatrix}$$

$\alpha + \beta = 1 \Rightarrow$ point
$\alpha + \beta = 0 \Rightarrow$ vector
else $\Rightarrow$ chaos

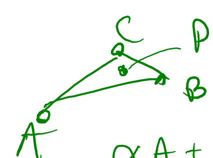

$p = p_0 + tu$

One useful combination of affine operations is:

$$\mathbf{p}(t) = \mathbf{p}_o + t\mathbf{u}$$

**Q**: What does this describe?

$t \in (-\infty, \infty) \Rightarrow$ line
$t \in [0, \infty) \Rightarrow$ half-line ray



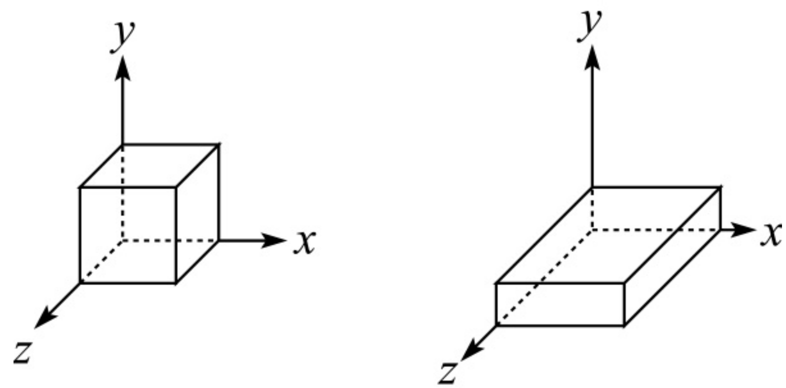

$\alpha A + \beta B + \gamma C$
$\alpha + \beta + \gamma = 1 \Rightarrow$ point

# Basic 3-D transformations: scaling

Some of the 3-D transformations are just like the 2-D ones.

For example, scaling:

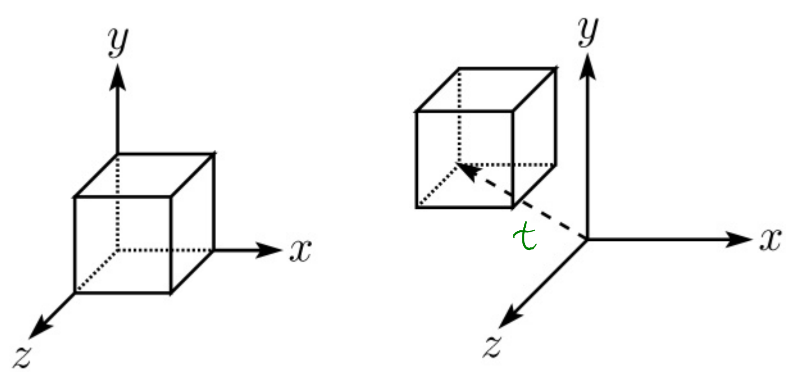
$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$



## Translation in 3D

I t

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



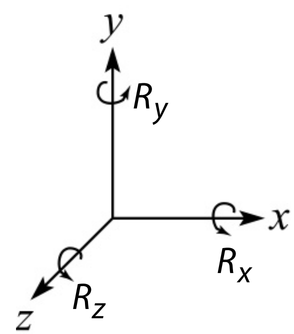
# Rotation in 3D (cont'd)

These are the rotations about the canonical axes:

$$R_x(\alpha) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha & 0 \\ 0 & \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

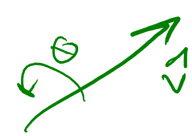
$$R_y(\beta) = \begin{bmatrix} \cos\beta & 0 & \sin\beta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\beta & 0 & \cos\beta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$R_z(\gamma) = \begin{bmatrix} \cos\gamma & -\sin\gamma & 0 & 0 \\ \sin\gamma & \cos\gamma & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$




Use right hand rule

A general rotation can be specified in terms of a product of these three matrices. How else might you specify a rotation?

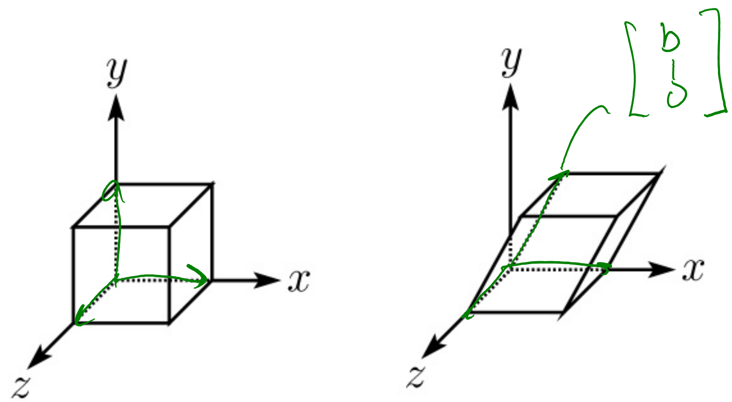


.... quaternions

# Shearing in 3D

Shearing is also more complicated. Here is one example:

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & b & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

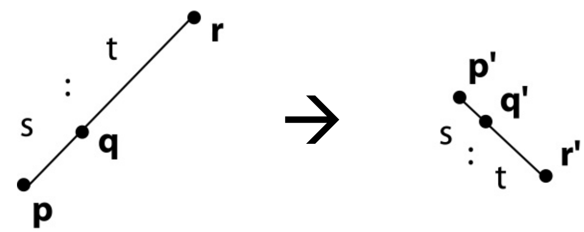



We call this a shear with respect to the x-z plane.

# Properties of affine transformations


Here are some useful properties of affine transformations:

- Lines map to lines
- Parallel lines remain parallel
    - (when transforming from N dimensions to N dimensions)
- Midpoints map to midpoints (in fact, ratios are always preserved)


$$\text{ratio} = \frac{\|\mathbf{pq}\|}{\|\mathbf{qr}\|} = \frac{s}{t} = \frac{\|\mathbf{p'q'}\|}{\|\mathbf{q'r'}\|}$$

# Affine transformations in OpenGL

OpenGL maintains a "modelview" matrix that holds the current transformation **M.**

The modelview matrix is applied to points (usually vertices of polygons) before drawing.

It is modified by commands including:

- `glLoadIdentity()` **M** ← **I**
  - set **M** to identity
- `glTranslatef(tx, ty, tz)` **M** ← **MT**
  - translate by ($t_x$, $t_y$, $t_z$)
- `glRotatef(θ, x, y, z)` **M** ← **MR**
  - rotate by angle θ about axis (x, y, z)
- `glScalef(sx, sy, sz)` **M** ← **MS**
  - scale by ($s_x$, $s_y$, $s_z$)

Note that OpenGL adds transformations by *postmultiplication* of the modelview matrix.

## Summary

What to take away from this lecture:

- All the names in boldface.
- How points and transformations are represented.
- How to compute lengths, dot products, and cross products of vectors, and what their geometrical meanings are.
- What all the elements of a 2 x 2 transformation matrix do and how these generalize to 3 x 3 transformations.
- What homogeneous coordinates are and how they work for affine transformations.
- How to concatenate transformations.
- The mathematical properties of affine transformations.