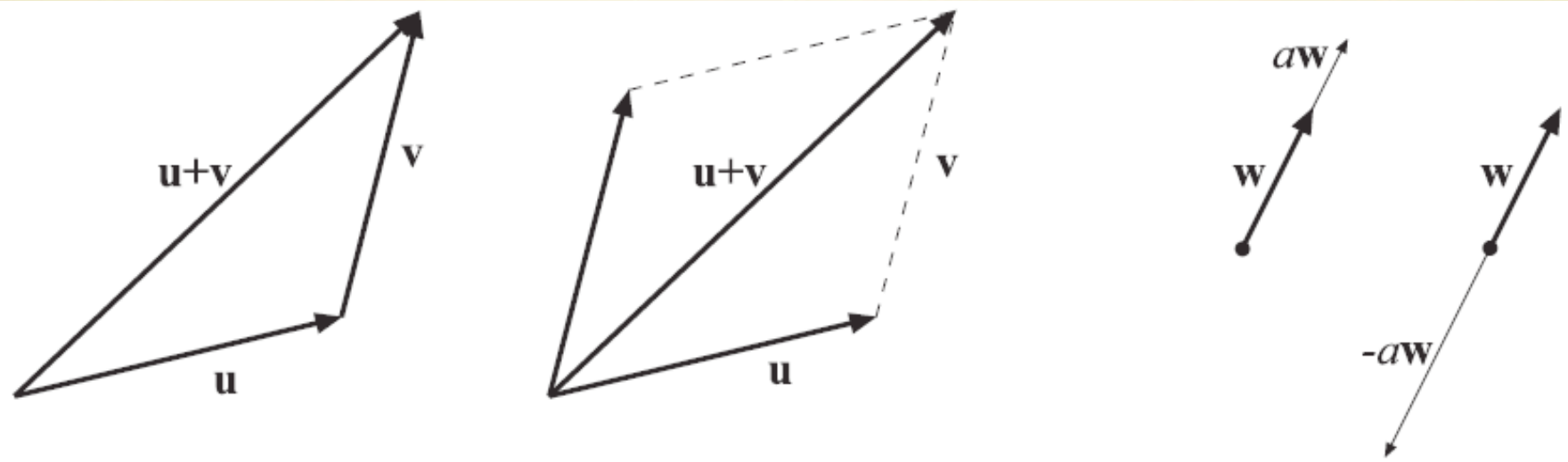
Vector and Affine Math

Don Fussell
Computer Science Department
The University of Texas at Austin

Vectors

- A vector is a direction and a magnitude
- Does NOT include a point of reference
- Usually thought of as an arrow in space
- Vectors can be added together and multiplied by scalars
- Zero vector has no length or direction



Vector Spaces

- Set of vectors
- Closed under the following operations
  - Vector addition: $\mathbf{v}_1 + \mathbf{v}_2 = \mathbf{v}_3$
  - Scalar multiplication: $s\,\mathbf{v}_1 = \mathbf{v}_2$
  - Linear combinations: $\sum_{i=1}^{n} a_i \mathbf{v}_i = \mathbf{v}$
- Scalars come from some field **F**
  - e.g. real or complex numbers
- Linear independence
- Basis
- Dimension

Vector Space Axioms

- Vector addition is associative and commutative
- Vector addition has a (unique) identity element (the **0** vector)
- Each vector has an additive inverse
  - So we can define vector subtraction as adding an inverse
- Scalar multiplication has an identity element (1)
- Scalar multiplication distributes over vector addition and field addition
- Multiplications are compatible (a(b**v**)=(ab)**v**)

Coordinate Representation

- Pick a basis, order the vectors in it, then all vectors in the space can be represented as sequences of coordinates, i.e. coefficients of the basis vectors, in order.
- Example:
  - Cartesian 3-space
  - Basis: [**i** **j** **k**]
  - Linear combination: $x\mathbf{i} + y\mathbf{j} + z\mathbf{k}$
  - Coordinate representation: $[x \;\; y \;\; z]$

$$a[x_1 \quad y_1 \quad z_1] + b[x_2 \quad y_2 \quad z_2] = [ax_1 + bx_2 \quad ay_1 + by_2 \quad az_1 + bz_2]$$

Row and Column Vectors

- We can represent a **vector**, **v** = (x,y), in the plane
  - as a column vector $\begin{bmatrix} x \\ y \end{bmatrix}$
  - as a row vector $\begin{bmatrix} x & y \end{bmatrix}$

Linear Transformations

- Given vector spaces V and W
- A function $f : V \rightarrow W$ is a *linear map* or *linear transformation* if

$$f(a_1\mathbf{v}_1 + \ldots + a_m\mathbf{v}_m) = a_1 f(\mathbf{v}_1) + \ldots + a_m f(\mathbf{v}_m)$$

Transformation Representation

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

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

- If **v** is a column vector, M goes on the left: $\mathbf{v}' = \mathbf{M}\mathbf{v}$

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

- If **v** is a row vector, M^T goes on the right: $\mathbf{v}' = \mathbf{v}\mathbf{M}^{\mathbf{T}}$

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

- 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: $x' = ax + by$

  $y' = cx + dy$

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

Identity

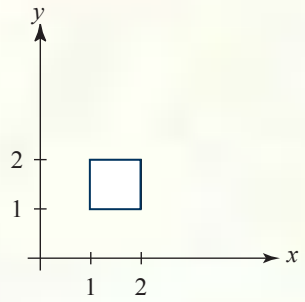
- Suppose we choose *a=d=1, b=c=0:*
  - Gives the **identity** matrix:

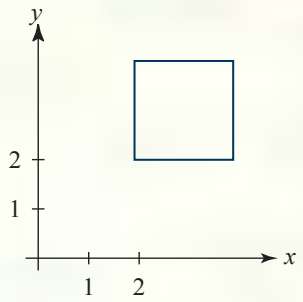
$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

  - Doesn't change anything

Scaling

- Suppose $b=c=0$, but let a and d take on any *positive* value:
  - Gives a **scaling** matrix: $\begin{bmatrix} a & 0 \\ 0 & d \end{bmatrix}$
  - Provides **differential (non-uniform) scaling** in x and y:

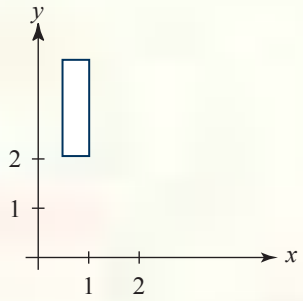




$$\begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix}$$

$$x' = ax$$

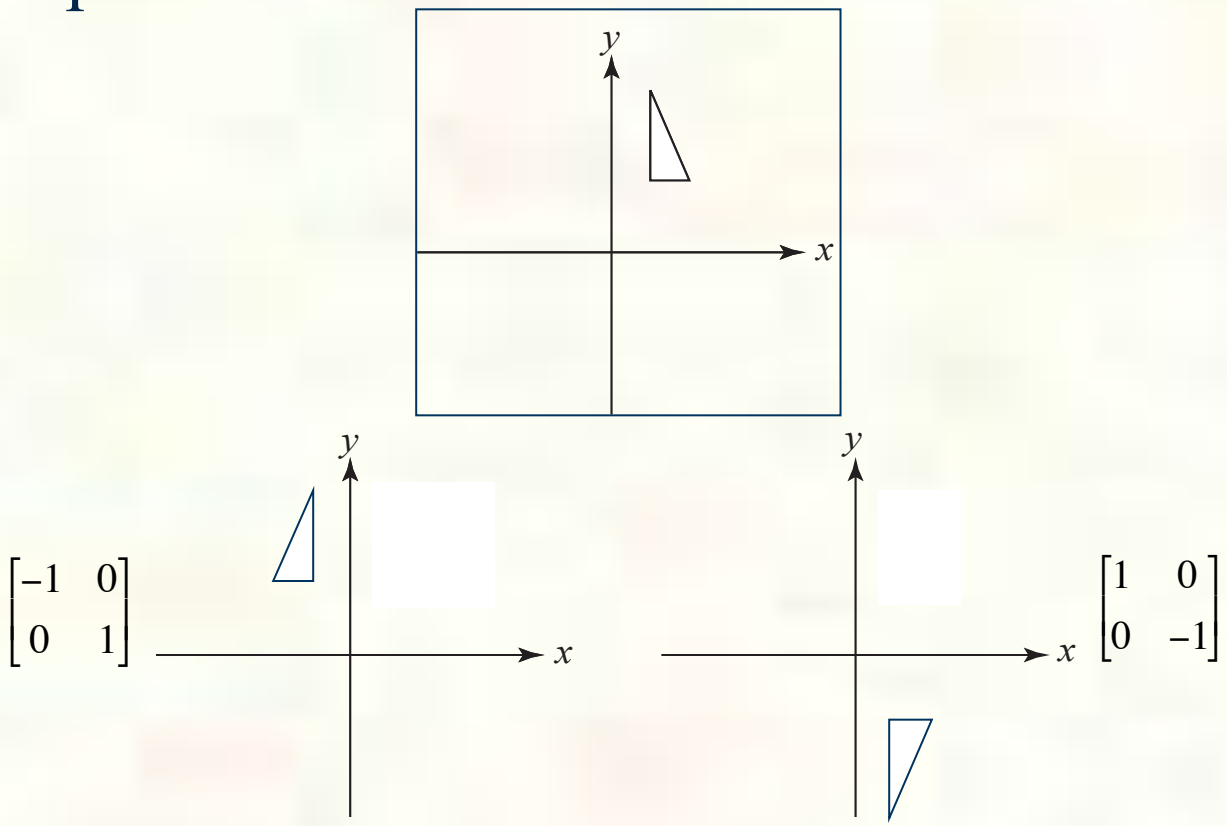
$$y' = dy$$



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

Reflection

- Suppose $b=c=0$, but let either a or d go negative.
- Examples:

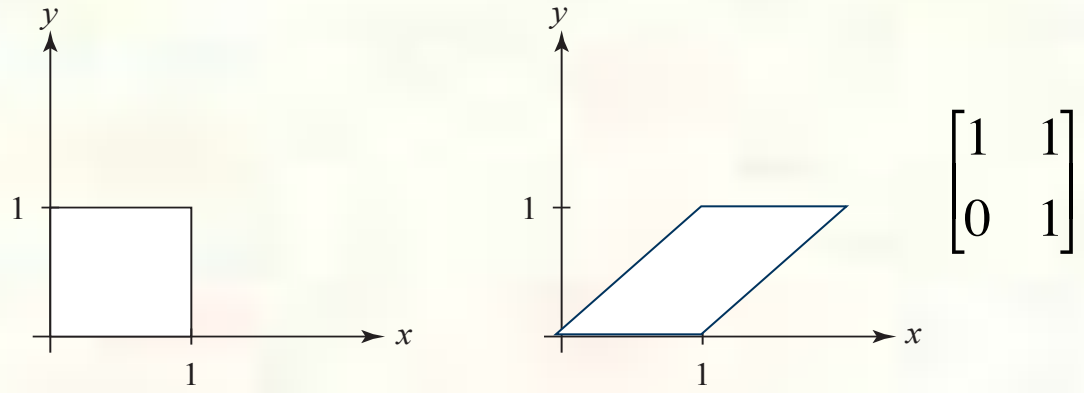


Shear

- Now leave $a=d=1$ and experiment with b
- The matrix $\begin{bmatrix} 1 & b \\ 0 & 1 \end{bmatrix}$

gives: $x' = x + by$

$y' = y$



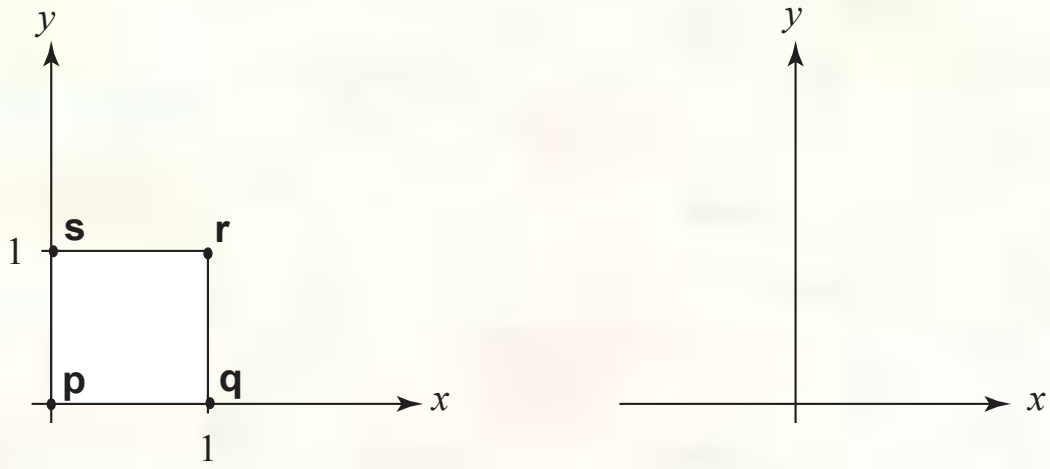
$\begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}$

Effect on unit square

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

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

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

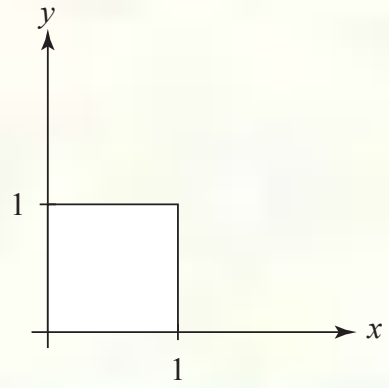


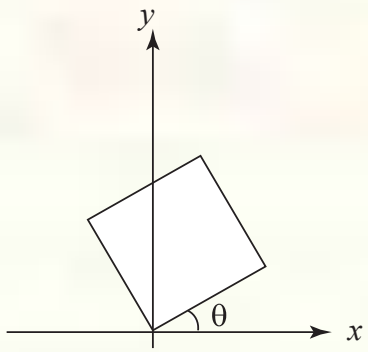
Effect on unit square, cont.

- 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

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

Linear transformations

- The unit square observations also tell us the 2x2 matrix transformation implies that we are representing a vector in a new coordinate system:

$$\mathbf{v}' = \mathbf{M}\mathbf{v}$$

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

$$= \begin{bmatrix} \mathbf{u} & \mathbf{w} \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

$$= x \cdot \mathbf{u} + y \cdot \mathbf{w}$$

- where $\mathbf{u}=[a\ c]^T$ and $\mathbf{w}=[b\ d]^T$ are vectors that define a new **basis** for a **linear space**.
- The transformation to this new basis (a.k.a., change of basis) is a **linear transformation**.

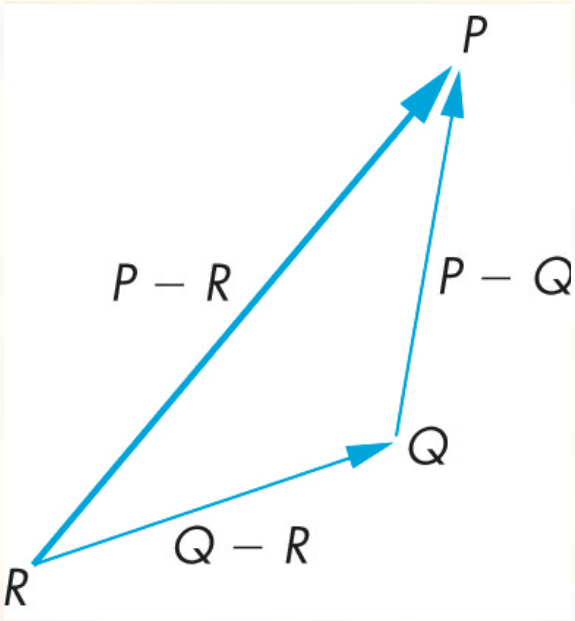
Limitations of the 2 x 2 matrix

- A 2 x 2 linear transformation matrix allows
  - Scaling
  - Rotation
  - Reflection
  - Shearing

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

Points

- A point is a location in space
- Cannot be added or multiplied together
- Subtract two points to get the vector between them
- Points are not vectors



Affine transformations

- In order to incorporate the idea that both the basis and the origin can change, we augment the linear space **u**, **w** with an origin **t**.
- Note that while **u** and **w** are **basis vectors**, the origin **t** is a **point**.
- We call **u**, **w**, and **t** (basis and origin) a **frame** for an **affine space**.
- Then, we can represent a change of frame as:

$$\mathbf{p}' = x \cdot \mathbf{u} + y \cdot \mathbf{w} + \mathbf{t}$$

- This change of frame is also known as an **affine transformation**.
- How do we write an affine transformation with matrices?

Homogeneous Coordinates

- To represent transformations among affine frames, we can loft the problem up into 3-space, adding a third component to every point:

$$\mathbf{p}' = \mathbf{M}\mathbf{p}$$

$$= \begin{bmatrix} a & b & t_x \\ c & d & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

$$= \begin{bmatrix} \mathbf{u} & \mathbf{w} & \mathbf{t} \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

$$= x \cdot \mathbf{u} + y \cdot \mathbf{w} + 1 \cdot \mathbf{t}$$

- Note that $[a\ c\ 0]^T$ and $[b\ d\ 0]^T$ represent vectors and $[t_x\ t_y\ 1]^T$, $[x\ y\ 1]^T$ and $[x'\ y'\ 1]^T$ represent points.

Homogeneous coordinates

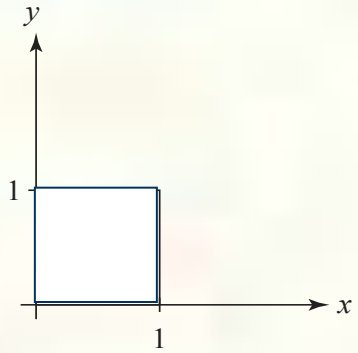
This allows us to perform translation as well as the linear transformations as a matrix operation:

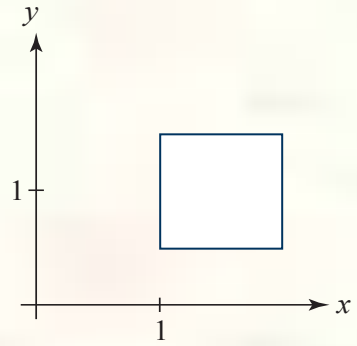
$$\mathbf{p}' = \mathbf{M}_T \mathbf{p}$$

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

$$x' = x + t_x$$

$$y' = y + t_y$$



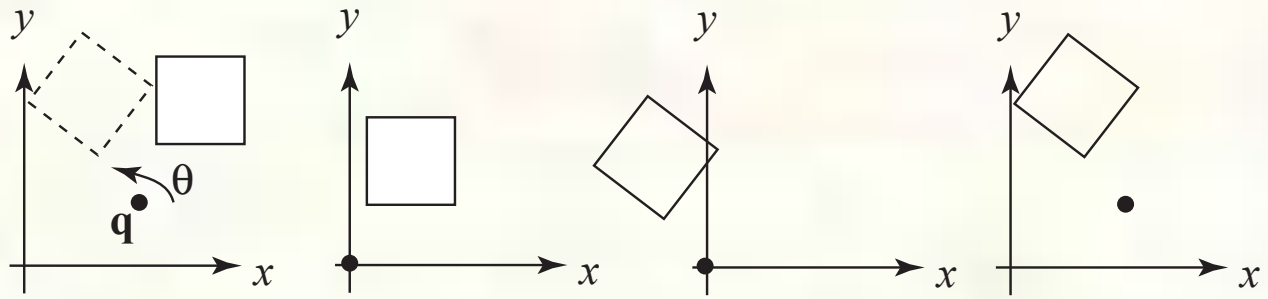


$$\begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1/2 \\ 0 & 0 & 1 \end{bmatrix}$$

Rotation about arbitrary points

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

With homogeneous coordinates, you can specify a rotation, $\mathbf{R_q}$, about any point $\mathbf{q} = [q_x\ q_y\ 1]^T$ with a matrix:



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

Line up the matrices for these step in right to left order and multiply.

Note: Transformation order is important!!

Points and vectors

From now on, we can represent points as have an additional coordinate of w=1.

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 matrix by a vector?

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

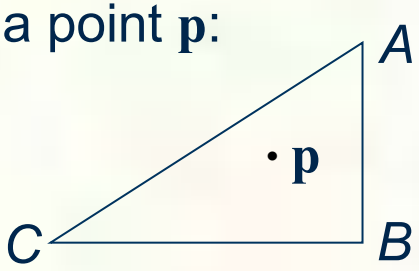
$$\text{vector} + \text{vector} \rightarrow$$
$$\text{vector} \cdot \text{vector} \rightarrow$$
$$\text{point} - \text{point} \rightarrow$$
$$\text{point} + \text{vector} \rightarrow$$
$$\text{point} + \text{point} \rightarrow$$

One useful combination of affine operations is: $\mathbf{p}(t) = \mathbf{p}_0 + t\mathbf{v}$

Q: What does this describe?

Barycentric coordinates

A set of points can be used to create an affine frame. Consider a triangle *ABC* and a point **p**:



We can form a frame with an origin *C* and the vectors from *C* to the other vertices:

$$\mathbf{u} = \mathrm{A} - \mathrm{C} \quad \mathbf{v} = \mathrm{B} - \mathrm{C} \quad \mathrm{t} = \mathrm{C}$$

We can then write *P* in this coordinate frame $\mathbf{p} = \alpha\mathbf{u} + \beta\mathbf{v} + \mathbf{t}$

The coordinates (α, β, γ) are called the **barycentric coordinates** of **p** relative to *A*, *B*, and *C*.

Computing barycentric coordinates

For the triangle example we can compute the barycentric coordinates of P:

$$\alpha A + \beta B + \gamma C = \begin{bmatrix} A_x & B_x & C_x \\ A_y & B_y & C_y \\ 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \\ \gamma \end{bmatrix} = \begin{bmatrix} \mathbf{p_x} \\ \mathbf{p_y} \\ 1 \end{bmatrix}$$

Cramer's rule gives the solution:

$$\alpha = \frac{\begin{vmatrix} \mathbf{p_x} & B_x & C_x \\ \mathbf{p_y} & B_y & C_y \\ 1 & 1 & 1 \end{vmatrix}}{\begin{vmatrix} A_x & B_x & C_x \\ A_y & B_y & C_y \\ 1 & 1 & 1 \end{vmatrix}} \quad \beta = \frac{\begin{vmatrix} A_x & \mathbf{p_x} & C_x \\ A_y & \mathbf{p_y} & C_y \\ 1 & 1 & 1 \end{vmatrix}}{\begin{vmatrix} A_x & B_x & C_x \\ A_y & B_y & C_y \\ 1 & 1 & 1 \end{vmatrix}} \quad \gamma = \frac{\begin{vmatrix} A_x & B_x & \mathbf{p_x} \\ A_y & B_y & \mathbf{p_y} \\ 1 & 1 & 1 \end{vmatrix}}{\begin{vmatrix} A_x & B_x & C_x \\ A_y & B_y & C_y \\ 1 & 1 & 1 \end{vmatrix}}$$

Computing the determinant of the denominator gives:

$$B_x C_y - B_y C_x + A_y C_x - A_x C_y + A_x B_y - A_y B_x$$

Cross products

Consider the cross-product of two vectors, **u** and **v**. What is the geometric interpretation of this cross-product?

A cross-product can be computed as:

$$\mathbf{u} \times \mathbf{v} = \begin{vmatrix} \mathbf{i} & \mathbf{j} & \mathbf{k} \\ u_x & u_y & u_z \\ v_x & v_y & v_z \end{vmatrix}$$

$$= (u_y v_z - u_z v_y)\mathbf{i} + (u_z v_x - u_x v_z)\mathbf{j} + (u_x v_y - u_y v_x)\mathbf{k}$$

$$= \begin{bmatrix} u_y v_z - u_z v_y \\ u_z v_x - u_x v_z \\ u_x v_y - u_y v_x \end{bmatrix}$$

What happens when **u** and **v** lie in the *x*-*y* plane? What is the area of the triangle they span?

Barycentric coords from area ratios

Now, let's rearrange the equation from two slides ago:

$$B_x C_y - B_y C_x + A_y C_x - A_x C_y + A_x B_y - A_y B_x$$
$$= (B_x - A_x)(C_y - A_y) - (B_y - A_y)(C_x - A_x)$$

The determinant is then just the *z*-component of (B-A) × (C-A), which is two times the area of triangle *ABC*!

Thus, we find:

$$\alpha = \frac{\text{SArea}(\mathbf{p}BC)}{\text{SArea}(ABC)} \quad \beta = \frac{\text{SArea}(A\mathbf{p}C)}{\text{SArea}(ABC)} \quad \gamma = \frac{\text{SArea}(AB\mathbf{p})}{\text{SArea}(ABC)}$$

Where SArea(RST) is the signed area of a triangle, which can be computed with cross-products.

Affine and convex combinations

Note that we seem to have added points together, which we said was illegal, but as long as they have coefficients that sum to one, it's ok.

We call this an **affine combination**. More generally $\mathbf{p} = \alpha_1 \mathbf{p_1} + \ldots + \alpha_n \mathbf{p_n}$

is a proper affine combination if: $\sum_{i=1}^{n} \alpha_i = 1$

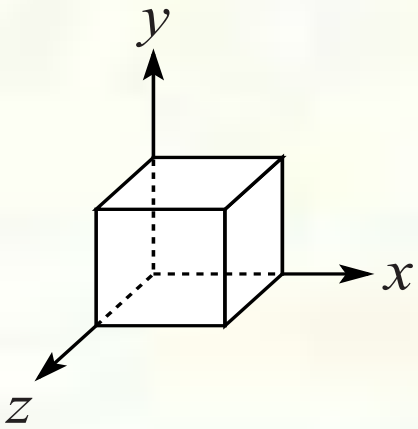
Note that if the α_i 's are all positive, the result is more specifically called a **convex combination**.

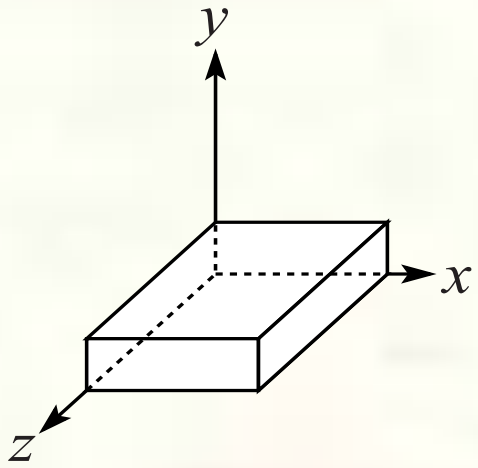
Q: Why is it called a convex combination?

Basic 3-D transformations: scaling

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

For example, scaling:

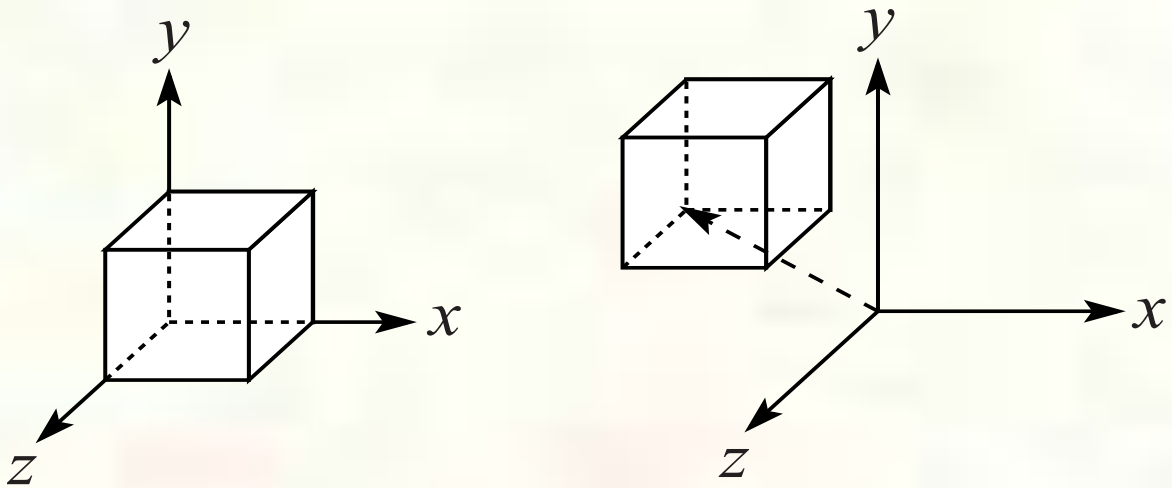




$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Translation in 3D

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



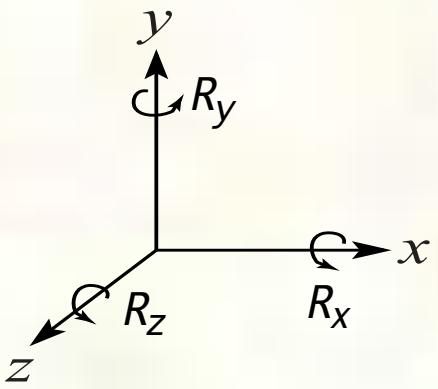
Rotation in 3D

Rotation now has more possibilities in 3D:

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

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

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

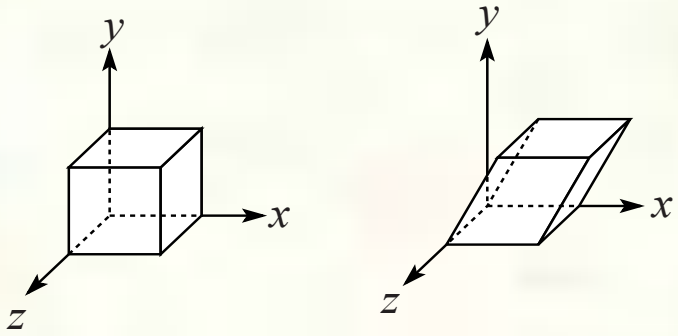


Use right hand rule

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.

Preservation of affine combinations

A transformation *F* is an affine transformation if it preserves affine combinations: $F(\alpha_1\mathbf{p_1} + \ldots + \alpha_n\mathbf{p_n}) = \alpha_1 F(\mathbf{p_1}) + \ldots + \alpha_n F(\mathbf{p_n})$

where the $\mathbf{p_i}$ are points, and: $\sum_{i=1}^{n} \alpha_i = 1$

Clearly, the matrix form of *F* has this property.

One special example is a matrix that drops a dimension. For example:

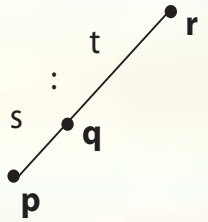
$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

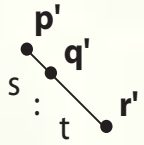
This transformation, known as an orthographic projection, is an affine transformation.

We'll use this fact later…

Properties of affine transformations

- Here are some useful properties of affine transformations:
  - Lines map to lines
  - Parallel lines remain parallel
  - Midpoints map to midpoints (in fact, ratios are always preserved)





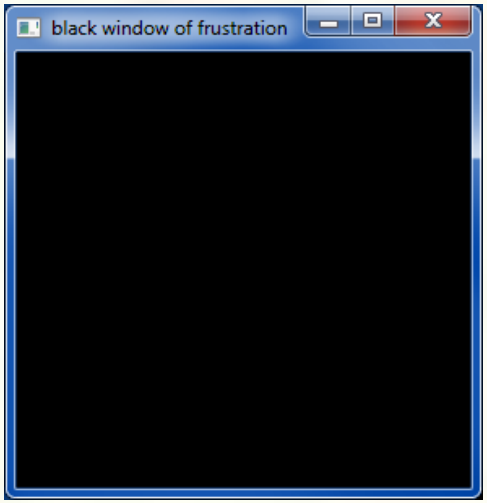
$$\text{ratio} = \frac{\|\mathbf{pq}\|}{\|\mathbf{qr}\|} = \frac{s}{t} = \frac{\|\mathbf{p'q'}\|}{\|\mathbf{q'r'}\|}$$

Next Lecture

- More Math and Transforms

Programming tips

- 3D graphics, whether OpenGL or Direct3D or any other API, can be frustrating
  - You write a bunch of code and the result is



Nothing but black window; where did your rendering go??

Things to Try

- Set your clear color to something other than black!
  - It is easy to draw things black accidentally so don't make black the clear color
  - But black is the initial clear color
- Did you draw something for one frame, but the next frame draws nothing?
  - Are you using depth buffering? Did you forget to clear the depth buffer?
- Remember there are near and far clip planes so clipping in Z, not just X & Y
- Have you checked for glGetError?
  - Call glGetError once per frame while debugging so you can see errors that occur
  - For release code, take out the glGetError calls
- Not sure what state you are in?
  - Use glGetIntegerv or glGetFloatv or other query functions to make sure that OpenGL's state is what you think it is
- Use glutSwapBuffers to flush your rendering and show to the visible window
  - Likewise glFinish makes sure all pending commands have finished
- Try reading
  - http://www.slideshare.net/Mark_Kilgard/avoiding-19-common-opengl-pitfalls
  - This is well worth the time wasted debugging a problem that could be avoided

Thanks

- Material for these slides provided by Christian Miller