Vector and Affine Math

Don Fussell
Computer Science Department
The University of Texas at Austin
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 $\mathbf{F}$
  - 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(bv) = (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: \(xi + yj + zk\)
  - Coordinate representation: \([x \ y \ z]\)

\[
a[x_1 \ y_1 \ z_1] + b[x_2 \ y_2 \ z_2] = [ax_1 + bx_2 \ ay_1 + by_2 \ az_1 + bz_2]
\]
Row and Column Vectors

- We can represent a vector, \( \mathbf{v} = (x,y) \), 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

$$f(a_1v_1 + \ldots + a_m v_m) = a_1f(v_1) + \ldots + a_m f(v_m)$$
Transformation Representation

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

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

- If $v$ is a column vector, $M$ goes on the left: $v' = Mv$

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

- If $v$ is a row vector, $M^T$ goes on the right: \( v' = vM^T \)

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

- 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 $y$:
  - $x' = ax$
  - $y' = dy$


University of Texas at Austin  CS354 - Computer Graphics  Don Fussell
Reflection

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

Examples:

\[
\begin{bmatrix}
-1 & 0 \\
0 & 1
\end{bmatrix}
\]

\[
\begin{bmatrix}
1 & 0 \\
0 & -1
\end{bmatrix}
\]
Shear

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

This gives:  
\[ x' = x + by \]
\[ y' = y \]
Effect on unit square

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

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

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

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} 0 & 0 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 0 & c & c+d & d \end{bmatrix}$$
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
From our observations of the effect on the unit square, it should be easy to write down a matrix for “rotation about the origin”:

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:

\[ \mathbf{v}' = \mathbf{Mv} \]

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

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

\[ = x \cdot \mathbf{u} + y \cdot \mathbf{w} \]

- 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 $u, w, t$ (basis and origin) a frame for an affine space.
- Then, we can represent a change of frame as:

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

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

\[
\mathbf{u} = \begin{bmatrix} r \\ s \\ t \end{bmatrix} \quad \mathbf{v} = \begin{bmatrix} x \\ y \\ z \end{bmatrix} \quad \mathbf{u} + \mathbf{v} = \begin{bmatrix} r + x \\ s + y \\ t + z \end{bmatrix} \quad a\mathbf{v} = \begin{bmatrix} ax \\ ay \\ az \end{bmatrix}
\]

\[
\|\mathbf{v}\| = \sqrt{x^2 + y^2 + z^2} \quad \text{norm}(\mathbf{v}) = \frac{\mathbf{v}}{\|\mathbf{v}\|}
\]
Parametric line segment

Or line, or ray, or just linear interpolation

\[
p = p_0 + t(p_1 - p_0) = (1 - t)p_0 + tp_1
\]

\[
\begin{bmatrix}
  x \\
  y \\
  z
\end{bmatrix} = \begin{bmatrix}
  x_0 \\
  y_0 \\
  z_0
\end{bmatrix} + t \begin{bmatrix}
  x_1 - x_0 \\
  y_1 - y_0 \\
  z_1 - z_0
\end{bmatrix} = \begin{bmatrix}
  (1 - t)x_0 + tx_1 \\
  (1 - t)y_0 + ty_1 \\
  (1 - t)z_0 + tz_1
\end{bmatrix}
\]

- Line segment  \( 0 \leq t \leq 1 \)
- Ray  \( 0 \leq t \leq \infty \)
- Line  \( -\infty \leq t \leq \infty \)
The vector dot product is defined by:

\[ \mathbf{u} \cdot \mathbf{v} = rx + sy + tz = ||\mathbf{u}|| ||\mathbf{v}|| \cos(\phi) \]
Projection

- **Projection** \((u\) component parallel to \(v\)) \[ w = \frac{u \cdot v}{v \cdot v} \]
- **Rejection** \((u\) component orthogonal to \(v\)) \[ u - w \]
- Particularly useful when vectors are normalized
Cross product

- If \( U \) & \( V \) point along the same line, \( W = 0 \)
- Useful for constructing local coordinate frames
- Length of cross product is area of parallelogram spanned by \( U \) and \( V \) (divide by 2 for area of the triangle)

\[
\mathbf{w} = \mathbf{u} \times \mathbf{v} = \begin{vmatrix}
\mathbf{i} & \mathbf{j} & \mathbf{k} \\
r & s & t \\
x & y & z
\end{vmatrix} = \begin{bmatrix}
sz - ty \\
tx - rz \\
ry - sx
\end{bmatrix}
\]

- \( \mathbf{w} \) is orthogonal to \( \mathbf{u} \) and \( \mathbf{v} \)
- \( \|\mathbf{w}\| = \|\mathbf{u}\|\|\mathbf{v}\|\sin(\phi) \)
- \( \|\mathbf{w}\| \) area of parallelogram
- Use right-hand rule
- \( \mathbf{u} \times \mathbf{v} = - (\mathbf{v} \times \mathbf{u}) \)
- \( (\mathbf{u} \times \mathbf{v}) \times \mathbf{w} \neq \mathbf{u} \times (\mathbf{v} \times \mathbf{w}) \)
**Determinants**

\[
\begin{vmatrix} a & b \\ c & d \end{vmatrix} = ad - bc
\]

\[
\begin{vmatrix} a & b & c \\ d & e & f \\ g & h & i \end{vmatrix} = a \begin{vmatrix} e & f \\ h & i \end{vmatrix} - b \begin{vmatrix} d & f \\ g & i \end{vmatrix} + c \begin{vmatrix} d & e \\ g & h \end{vmatrix} = aei - afh + bfg - bdi + cdh - ceg
\]

- \( \det(M^T) = \det(M) \)
- \( \det(AB) = \det(A)\det(B) \)
- if \( \det(M) = 0 \), \( M \) is singular, has no inverse
Plane equation

- Given normal vector $\mathbf{N}$ orthogonal to the plane and any point $\mathbf{p}$ in the plane $\mathbf{N} \cdot \mathbf{p} + d = 0$

$$\begin{bmatrix} a & b & c \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} + d = ax + by + cz + d = 0$$

- For a triangle
  $$\mathbf{N} = norm((\mathbf{v}_1 - \mathbf{v}_0) \times (\mathbf{v}_2 - \mathbf{v}_0))$$

- Order matters, usually CCW
Homogeneous Coordinates

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

\( \mathbf{p}' = \mathbf{Mp} \)

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

\[
= \begin{bmatrix}
u & w & t
\end{bmatrix}
\begin{bmatrix}
x \\
y \\
1
\end{bmatrix}
\]

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

- 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:

\[
p' = M_T p
\]

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

\[
x' = x + t_x
\]

\[
y' = y + t_y
\]
Rotation about arbitrary points

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

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

- vector + vector $\rightarrow$
- vector · vector $\rightarrow$
- point – point $\rightarrow$
- point + vector $\rightarrow$
- point + point $\rightarrow$

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

**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:

$$u = A - C \quad v = B - C \quad t = C$$

We can then write $P$ in this coordinate frame

$$p = \alpha u + \beta v + t$$

The coordinates $(\alpha, \beta, \gamma)$ 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 \( P \):

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

Cramer’s rule gives the solution:

\[
\begin{align*}
\alpha &= \begin{vmatrix} p_x & B_x & C_x \\ p_y & B_y & C_y \\ A_x & B_x & C_x \end{vmatrix} \\
\beta &= \begin{vmatrix} A_x & p_x & C_x \\ A_y & p_y & C_y \\ 1 & 1 & 1 \end{vmatrix} \\
\gamma &= \begin{vmatrix} A_x & B_x & p_x \\ A_y & B_y & p_y \\ 1 & 1 & 1 \end{vmatrix}
\end{align*}
\]

Computing the determinant of the denominator gives:

\[
B_x C_y - B_y C_x + A_y C_x - A_x C_y + A_x B_y - A_y B_x
\]
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 \((B-A) \times (C-A)\), which is two times the area of triangle \(ABC\)!

Thus, we find:

\[
\alpha = \frac{\text{SArea}(pBC)}{\text{SArea}(ABC)} \quad \beta = \frac{\text{SArea}(ApC)}{\text{SArea}(ABC)} \quad \gamma = \frac{\text{SArea}(ABp)}{\text{SArea}(ABC)}
\]

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 $\alpha_i$ ‘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:

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

$$F(\alpha_1 p_1 + \ldots + \alpha_n p_n) = \alpha_1 F(p_1) + \ldots + \alpha_n F(p_n)$$

where the $p_i$ 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)

\[
\begin{align*}
\text{ratio} &= \frac{\|pq\|}{\|qr\|} = \frac{s}{t} = \frac{\|p'q'\|}{\|q'r'\|}
\end{align*}
\]
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??

University of Texas at Austin    CS354 - Computer Graphics    Don Fussell
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 glGetInteger or glGetFloat 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