Vector Geometry for Computer Graphics

Bob Geitz
January, 2007

Contents

Part I: Basic Definitions

Coordinate Systems

Most 2D geometry is pictured with the first coordinate on the horizontal axis and the second coordinate on the vertical axis, as in
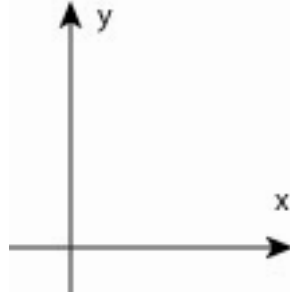
The 3D situation is somewhat more complex. Note that the following two coordinate systems are essentially the same; we can rotate one into the other:
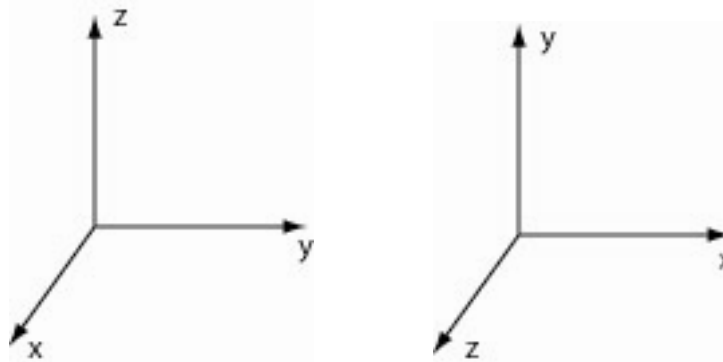
On the other hand, the following pair are different; no set of rotations will convert one axis into the other:
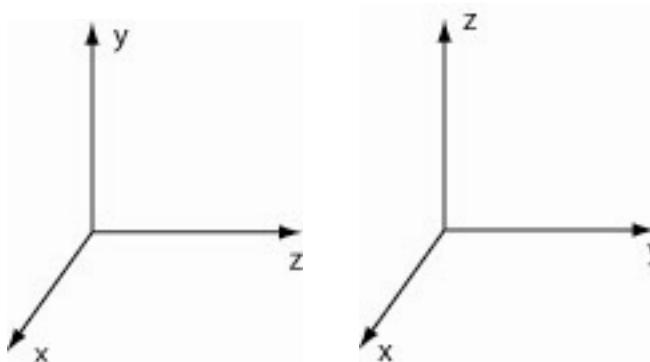
The usual way to think of this is that coordinate axes have "handedness". Coordinate axes are either right-handed or left-handed. If you label your index fingers with an "x", your middle fingers with "y" and your thumbs with "z", then you will be able to align one

of your hands with a given axis. For the two systems illustrated immediately above, you should be able to align the fingers of your left hand with the system on the left. This means it is a left-handed system. Similarly, the system on the right matches the fingers of your right hand and so is a right-handed system.

Points and Vectors

A *point* in n-dimensional space is just a collection of n values that can be considered the point's coordinates. A *vector* in n-dimensional space is also a collectio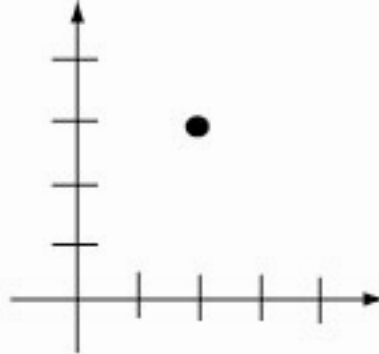n of n values. The difference between a point and a vector lies not in the coordinates, but rather in the way we interpret these coordinates. The coordinates of a point give it a position in space. For example, the point (2, 3) might be drawn:

A vector, on the other hand, provides a direction and a magnitude rather than a position. You can think of a vector <a, b> as extending from any point (x, y) to the point (x+a, y+b). All of the vectors in the following picture are <2, 3>, just given different starting points.
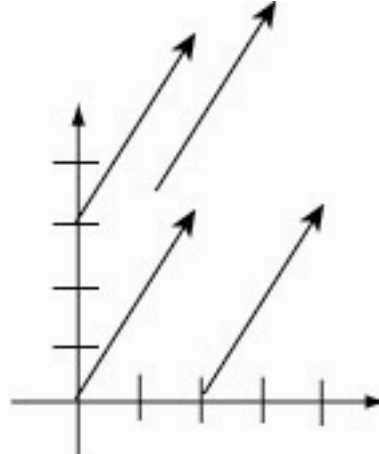
Alternatively, the vector extending from point (x_1, y_1) to the point (x_2, y_2) is $<x_2-x_1, y_2-y_1>$. This is an important fact that we will use in many situations.

The length or magnitude of a vector is the square root of the sums of the squares of its coordinates. The length of vector v is often written $|v|$. For example, the length of the vector <2, 3> is $\sqrt{2^2 + 3^2} = \sqrt{13}$. An easy way to make a vector of length 1 in a given direction is to first find any vector in this direction, then to divide both coordinates of

vector by its length. Thus, a vector in the same direction of <a, b> but having length 1 is $\left\langle \dfrac{a}{\sqrt{a^2+b^2}}, \dfrac{b}{\sqrt{a^2+b^2}} \right\rangle$. Vectors with length 1 are commonly called *unit* vectors.

For 3-dimensional geometry there are standard names for the unit vectors that point along the three axes: **i** is the vector <1, 0, 0>, **j** is <0, 1, 0> and **k** is <0, 0, 1>.

Matrices and Determinants

A 2-dimensional *matrix* is a rectangular array of numbers, as in $\begin{bmatrix} 3 & 8 & 1 \\ 9 & -2 & 5 \end{bmatrix}$. The size of such a matrix is $n \times m$ where n is the number of rows and m is the number of columns. Thus, the matrix above has size 2×3. We use matrices in computer graphics to represent transformations, spline curves and surfaces, textures, and many other things.

A *determinant* is an operation that can be applied to a $n \times n$ matrix to produce a single value. The determinant of matrix A is often written $|A|$. The definition of the determinant is recursive. First, the determinant of a 2×2 matrix is $\begin{vmatrix} a & b \\ c & d \end{vmatrix} = ad - bc$.

Thus, $\begin{vmatrix} 4 & 2 \\ 5 & 3 \end{vmatrix} = 12 - 10 = 2$. Then, the determinant of a 3×3 matrix is defined in terms of 3 2×2 determinants:

$$\begin{vmatrix} a & b & c \\ d & e & f \\ g & h & i \end{vmatrix} = a\begin{vmatrix} e & f \\ h & i \end{vmatrix} - b\begin{vmatrix} d & f \\ g & i \end{vmatrix} + c\begin{vmatrix} d & e \\ g & h \end{vmatrix} = a(ei - fh) - b(di - gf) + c(dh - ge)$$

Note that this multiplies each element of the top row times the 2×2 determinant that results from removing that element's row and column from the grid. The results of these products are put into an alternating sum.

For example, we compute

$$\begin{vmatrix} 3 & 1 & 2 \\ -2 & 4 & 0 \\ 6 & 2 & -1 \end{vmatrix} = 3\begin{vmatrix} 4 & 0 \\ 2 & -1 \end{vmatrix} - 1\begin{vmatrix} -2 & 0 \\ 6 & -1 \end{vmatrix} + 2\begin{vmatrix} -2 & 4 \\ 6 & 2 \end{vmatrix}$$
$$= 3(-4 - 0) - 1(2 - 0) + 2(-4 - 24)$$
$$= -12 - 2 - 56$$
$$= -70$$

There are many ways to interpret the meaning of a determinant. One simple interpretation concerns 2D parallelograms and 3D parallelpipeds. In the 2D case, consider the vectors $u = <u_1, u_2>$ and $v = <v_1, v_2>$. Then the area of the parallelogram formed by u and v,
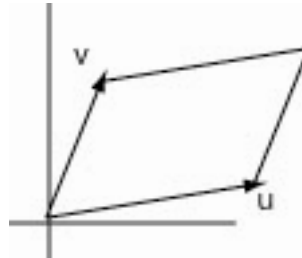
is

$$Area = \begin{vmatrix} u_1 & u_2 \\ v_1 & v_2 \end{vmatrix} = u_1 v_2 - v_1 u_2$$

Similarly, with three 3D vectors $<u_1, u_2, u_3>$ $<v_1, v_2, v_3>$ and $<w_1, w_2, w_3>$ we can define a parallelpiped, whose volume is

$$Volume = \begin{vmatrix} u_1 & u_2 & u_3 \\ v_1 & v_2 & v_3 \\ w_1 & w_2 & w \end{vmatrix}$$

Determinants have many uses, but we will primarily use them to define and evaluate the cross product of two 3D vectors.

Part II: Operations

Vector addition and scalar multiplication

We can add or subtract two vectors of the same size by summing corresponding coordinates:
$$<2, 4, -1> + <5, 1, 6> = <7, 5, 5>$$
We can multiply a vector by a scalar by multiplying each coordinate of the vector by the scalar; the result is a vector:
$$<2, 5, 1>*3 = <6, 15, 3>$$

Both of these operations have a simple geometrical interpretation. Recall that we can think of a vector as providing a direction and a distance. The sum of two vectors

represents moving in the direction of the first vector for its distance, then in the direction of the second for its distance. For example, the picture below on the left shows two vectors **A** and **B**. The picture on the right shows the sum **A+B**:
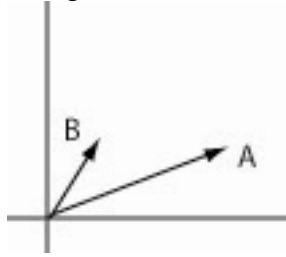
Finally, the picture below shows the sum **A+B+B+B = A+3B**:
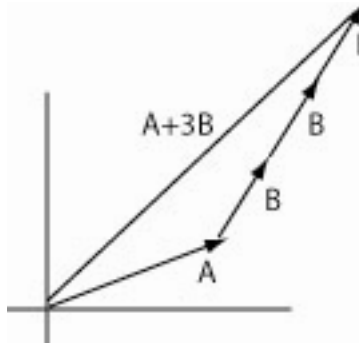
Any scalar multiple of a vector results in a parallel vector. In fact, two vectors are parallel if and only if each is a scalar multiple of the other.

The Dot Product of Two Vectors

There are several ways to multiply two vectors together. While they are all important for graphics, the one most frequently used is the dot-product, whose name comes from the fact that it is written with a dot between the two vectors. The dot-product can be applied to any two vectors of the same size; it is computed by multliplying the corresponding coordinates of the vectors and summing all of these products.   Thus
<a, b> • <c, d> = ac+bd. and <1, 2, 3> • <4, 5, 6> = 4 + 10 + 18 = 32
.

There is a very useful geometrical interpretatioin of the dot product: If u and v are two vectors then $u \bullet v = |u||v|\cos(\theta)$ where θ is the angle between u and v. For example, since $\cos(0) = 1$, if two vectors are pointing in the same direction then their dot product is the product of their lengths.   In particular, for any vector u, $u \bullet u = |u|^2$. Moreover**,** since $\cos(90°) = \cos(270°) = 0$, **two vectors are perpendicular if and only if their dot product is 0**. This is one of the most important facts from vector geometry and is used throughout computer graphics.

It should be obvious that dot products are commutative ($a \bullet b = b \bullet a$).

Projections

It is sometimes useful to find the projection of one vector in the direction of another. The following picture shows the setup for this. We want to project vector **B** onto vector **A**.
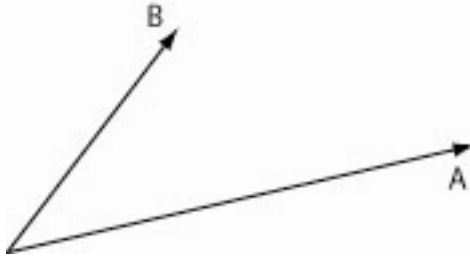
The following picture illustrates the derivation. We want to find vector p, which is the projection of **B** onto **A**. Angle Θ is the angle between **A** and **B**,
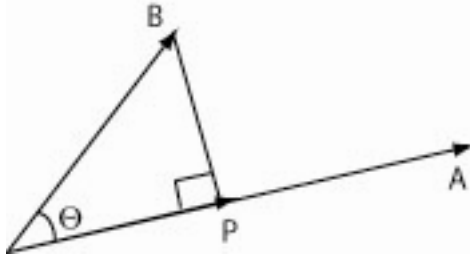
The idea is to first find the length of p, then to multiply this length times a unit vector in the direction of A: $p = |p|\dfrac{A}{|A|}$

Here is the derivation:

$$\frac{|p|}{|B|} = \cos(\theta) = \frac{A \bullet B}{|A||B|} \quad \text{so} \quad |p| = \frac{A \bullet B}{|A|}$$

$$p = |p|\frac{A}{|A|} = \frac{A \bullet B}{|A|^2} A$$

Altogether, the projection of **B** onto **A** is $\dfrac{A \bullet B}{|A|^2} A$

We can also find the projection of a vector onto a plane. Here is the setup: we start with the plane and vector **B**; we want to find the projection of B onto the plane, which is vector **p**;
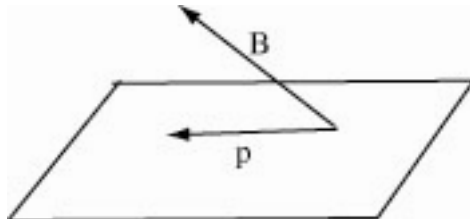
To solve this we introduce vector **N**, the normal to the plane, and vector **r**, the projection of **B** onto **N**:
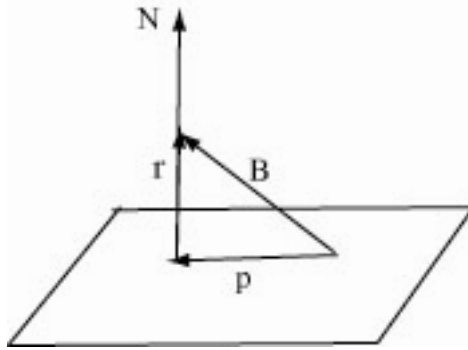
We know how to calculate **r**; this is just the projection of one vector onto another:

$$r = \frac{B \bullet N}{|N|^2} N$$

Now note that **p**+**r** = **B**, so **p** = **B**-**r.**

Altogether, the projection of **B** onto the plane with normal **N** is

$$B - \frac{B \bullet N}{|N|^2} N$$

The Cross-Product of two 3D vectors

The cross-product is another way to multiply two vectors. This time the result is a vector. Cross products are only defined for 3D vectors.

There are many equivalent definitions for the cross-product. Perhaps the most common definition is

$$\langle a_1, a_2, a_3 \rangle \times \langle b_1, b_2, b_3 \rangle = \begin{vmatrix} \mathbf{i} & \mathbf{j} & \mathbf{k} \\ a_1 & a_2 & a_3 \\ b_1 & b_2 & b_3 \end{vmatrix}$$ where **i**, **j**, and **k** are the unit vectors parallel

to the coordinate axes.

An equivalent formulation using matrix multiplication (see below if you don't recall how to multiply matrices) is

$$\langle a_1, a_2, a_3 \rangle \times \langle b_1, b_2, b_3 \rangle = \begin{bmatrix} 0 & -a_3 & a_2 \\ a_3 & 0 & -a_1 \\ -a_2 & a_1 & 0 \end{bmatrix} \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix}$$

Finally, there is a geometrical interpretation: $\mathbf{u} \times \mathbf{v} = \mathbf{n}|\mathbf{u}||\mathbf{v}|\sin(\theta)$, where θ is the angle between *u* and *v* and *n* is the unit vector normal to both *u* and *v* given by the *right-hand*

rule: if you put the index finger of your right hand in the direction of **_u_** and your middle finger in the direction of **_v_**, then your thumb will point in the direction of **_n_**. This interpretation makes clear the most essential facts about the cross-product: it gives a vector normal to the two input vectors, and the cross-product of two parallel vectors is 0.

Some people restate the geometrical interpretation as: the cross-product $\mathbf{u} \times \mathbf{v}$ is a vector normal to both **u** and **v** in the direction given by the right-hand rule whose magnitude is the area of the parallelogram determined by **u** and **v**.

The main use of cross-products is in finding normal vectors. For example, if you know three points in a plane and you want to find the $n \times m$ normal vector to the plane, find two non-parallel vectors between the three points and take the cross-product of these vectors.

Matrix multiplication

Suppose A and B are matrices. It is possible to multiply $A \times B$ if matrix A has size $n \times m$ and matrix B has size $m \times p$ (i.e., if the number of columns of A is the same as the number of rows of B.) The resulting matrix has size $n \times p$. The entry in the ith row and j^{th} column of the result is the dot-product of the ith row of A with the jth column of B.

For example, consider the product $\begin{bmatrix} 2 & 1 & 3 \\ 4 & 0 & -1 \end{bmatrix} \begin{bmatrix} 1 & 2 & 1 & 3 \\ -1 & 0 & 2 & 1 \\ 1 & 1 & 0 & 1 \end{bmatrix}$. This multiplication can be performed because the left matrix has 3 columns and the right matrix 3 rows. The entry in the first row and first column of the result is the dot product of the first row of the left matrix, $\begin{bmatrix} 2 & 1 & 3 \end{bmatrix}$ with the first column of the right matrix, $\begin{bmatrix} 1 \\ -1 \\ 1 \end{bmatrix}$. This dot product is 4. The entry in the first row, second column of the result is the dot product of $\begin{bmatrix} 2 & 1 & 3 \end{bmatrix}$ with $\begin{bmatrix} 2 \\ 0 \\ 1 \end{bmatrix}$, which is 7. Continuing on, we see that

$$\begin{bmatrix} 2 & 1 & 3 \\ 4 & 0 & -1 \end{bmatrix} \begin{bmatrix} 1 & 2 & 1 & 3 \\ -1 & 0 & 2 & 1 \\ 1 & 1 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 4 & 7 & 4 & 10 \\ 3 & 7 & 4 & 11 \end{bmatrix}$$

It is easy to see that matrix multiplication is not commutative. Indeed, *BA* might not even be defined when *AB* is. Matrix multiplication is, however, associative.

Matrices are used for many things in graphics. Perhaps the most typical application of matrices is to represent transformations. Matrix *T* might represent a particular transformation; to apply this transformation to a point *x*, we multiply *xT*. To first apply transformation T_1 to *x*, then apply T_2 to the result, we multiply xT_1T_2. Because matrix multiplication is associate, it makes no difference if we group this as $(xT_1)T_2$ or as $x(T_1T_2)$. The latter formulation is convenient if there are a number of points to which this sequence of transformations must be applied because it allows us to multiply T_1T_2 once, and then apply the result to each point in turn with one matrix multiplication. Because of this, some old texts refer to matrix multiplication as "concatenating" matrices to represent the composition of transformations.

Part III: Applications

The equation of a line or a ray

The y=mx+b equation for a line that you learned in high school is not very useful for graphics because it is limited to 2 dimensions. We more commonly use the vector form of a line, which can be applied to any number of dimensions. The equation of the line through point *a* parallel to vector *v* is

$$p = a + tv$$

Here t is a scalar parameter and *p* is an arbitrary point on the line.

For example, suppose we need the equation of the line through points <2, 1, 3> and <4, 0, 5>. A vector parallel to this line is the vector from <2, 1, 3> to <4, 0, 5>, which is <4-2, 0-1, 5-3> = <2, -1, 2>. The line is then

$$p = <2, 1, 3> + t<2, -1, 2>$$

We get points on the line by assigning values to t. For instance, if t=2 we get <2, 1, 3> + <4, -2, 4> = <4, -1, 7>. Alternatively, we can use this equation for the line to get three parametric equations for x, y, and z in terms of t:

$$<x, y, z> = <2, 1, 3> + t<2, -1, 2>$$

so

$$x = 2 + 2t$$
$$y = 1 - t$$
$$z = 3 + 2t$$

In ray tracing we need the equation of a *ray*, or half-line. A ray has exactly the same equation as a line: $p = a + tv$, only for rays the parameter t cannot have negative values.

The equation of a plane

Suppose we need to find the equation of the plane that has normal vector N=<A,B,C> and that passes through the point (a, b, c):
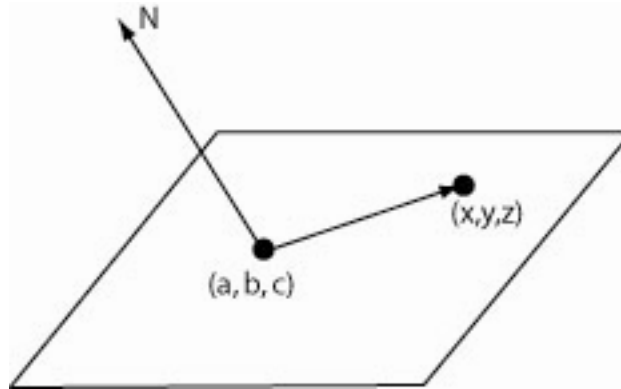
Let (x,y,z) be any point in the plane. Since N is normal to the plane it is normal to each vector lying in the plane. In particular, it is normal to the vector from (a,b,c) to (x,y,z); this vector is <x-a, y-b, z-c>. In other words $N \bullet \langle x-a \quad y-b \quad z-c \rangle = 0$. This equation can be rewritten as $\langle A \quad B \quad C \rangle \bullet \langle x-a \quad y-b \quad z-c \rangle = 0$, or as $Ax + By + Cz = Aa + Bb + Cc$.

In other words, the equation of a plane with normal vector <A, B, C> is $Ax+By+Cz$=constant. You can find the constant by plugging any point that is known to be in the plane into $Ax+By+Cz$.

If you need to find the equation of the plane containing 3 known points, take the cross-product of two non-parallel vectors connecting these points to get the normal; then proceed as above. For example, to find the plane containing (1, 1, 1), (1, 0, 1) and (2, 1, 2), we first need two vectors. The vector from (1, 0, 1) to (1, 1, 1) is <0, 1, 0>. The vector from (1, 1, 1) to (2, 1, 2) is <1, 0, 1>. The cross-product of <0, 1, 0> with <1, 0, 1> is <1, 0, -1>. Thus, the equation of the plane is x-z = constant. We can put any of the given points into this to see that the constant is 0. The plane is x-z = 0.

Outward-pointing Normals

A *polygon* is a body that lies in a plane and is bounded by edges that are line segments, like a triangle or an octagon. A *polyhedron* is a 3D body bounded by faces that are polygons, such as a cube or a pyramid. Sometimes it is helpful to find an *outward-pointing normal* to a polygon or a polyhedron. In both cases this is a vector that is normal to a given face or edge and points towards the outside of the body.

Consider first the case of a polyhedron:
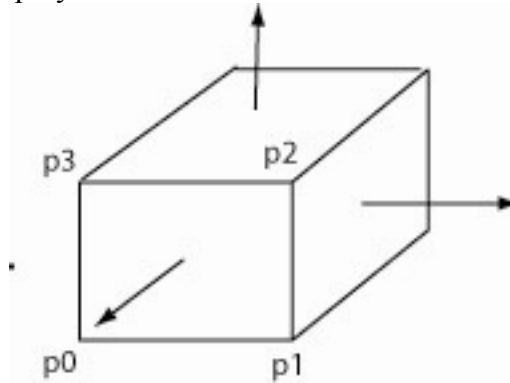
If the vertices of the face are numbered p_0, p_1, ... in a counter-clockwise order then we can compute the outward-pointing normal from the cross-product of two of the edges. For example, it is given by $(p_1 - p_0) \times (p_2 - p_1)$, or starting from any point p_i, $(p_{i+1} + p_i) \times (p_{i+2} - p_{i+1})$. The right-hand rule for cross-products guarantees that this is the outward-pointing normal and not the inward-pointing normal.

The case of a polygon is similar:
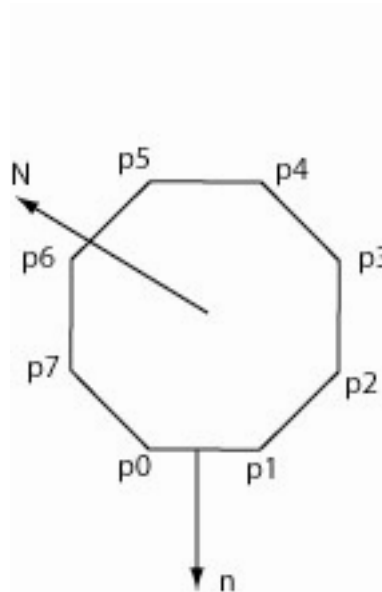
This time we want to find the normal $\boldsymbol{n}$ that lies in the same plane as the polygon and is normal to a given edge. If N is the normal to this entire plane and if the edge in consideration is the edge from p_i to p_{i+1} (where again the vertices have been numbered counterclockwise starting from p_0), then the outward pointing normal is $\mathbf{n} = (p_{i+1} - p_i) \times \mathbf{N}$. Of course, if N is not given it can be computed from the cross-product of any two edges of the polygon.

Finding the viewer coordinate axes

Viewer coordinates are frequently specified in terms of a left-handed coordinate system with origin at the viewer's position and z-axis pointing in the direction the viewer is looking.
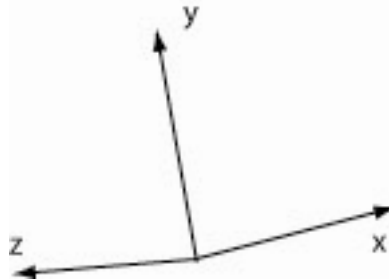
Rather than directly giving the three coordinate axes, many system prefer to determine this system from the viewer's position, a *lookat* point in the direction the viewer is looking, and an *up* vector that specifies the vertical direction. Unless there is additional information, we assume that the viewer's x-axis is orthogonal to the *up* vector. If the viewer is standing vertically her y-axis will point in the direction of the *up* vector, but the viewer can lean over and still have her x-axis orthogonal to this *up* vector.

We will find vectors V_x, V_y and V_z parallel to the three axes. First, V_z points from the viwer to the *lookat* point: $V_z = lookat\text{-}viewer$. Next, V_x is normal to both V_z and the *up* vector. From the right-hand rule for cross-products we see that $V_x = V_z \times up$. Finally, $V_y = V_x \times V_z$

The distance from a point to a line

Suppose we have the line $x = tv + p_1$ (a line through point p_1 parallel to vector v) and a separate point p. We want to find the distance from p to the line. Let w be the vector from p_1 to p. The following picture shows the setup:
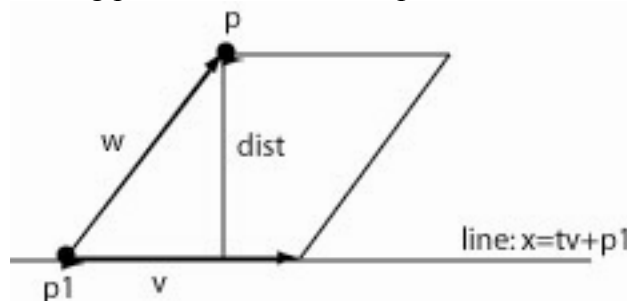
We know that the cross-product of v and w is a vector whose length is the area of the parallelogram determined by v and w. This area is also given by the length of v multiplied times the distance from p to the line that forms the base of the parallelogram. Thus $|v \times w| = |v| dist$, or $dist = \dfrac{|v \times w|}{|v|}$. An even simpler formation is $dist = |u \times w|$, where u is a unit vector parallel to the line and w is an vector from the line to p.

The distance from a point to a plane.

A similar problem is finding the distance from a point $p=(p_1, p_2, p_3)$ to the plane $Ax+By+Cz=D$. Let $a=(a_1, a_2, a_3)$ be any point in the plane.
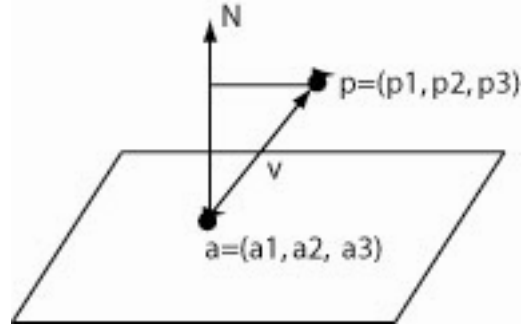
Let vector v be the vector from a to p. As you can see from the picture, the distance from p to the plane is the length of the projection of v onto the normal vector $N = <A, B, C>$. The projection is $(v \bullet N)\dfrac{N}{|N|^2}$. This vector has length $\dfrac{v \bullet N}{|N|}$.

We know that $N=<A, B, C>$, $v=<p_1-a_1, p_2-a_2, p_3-a_3>$, and since a is a point in the plane, $Aa_1+Ba_2+Ca_3=D$. we can simplify the distance from p to the plane to

$$\frac{|Ap_1 + Bp_2 + Cp_3 - D|}{\sqrt{A^2 + B^2 + C^2}}$$

Determining if a point is inside a convex polygon

Given a polygon (which by definition lies in a plane) and a single point in the same plane, we would like an algorithm for determining whether the point lies inside the polygon. The following algorithm only applies to *convex* polygons – those with the property that given any two points inside the polygon, the entire line segment connecting those two points must lie inside the polygon. Such polygons are the intersection of half-planes, which means that if you extend the boundary edges into infinite lines the polygon will always lie completely on one side of each edge line. If a point is on the same side of each edge as the polygon, then it must lie inside the polygon.

For this algorithm we assume that the vertices of the polygon are numbered $p_0, p_1, p_2, \ldots$ in counterclockwise order. Edge e_i extends from p_i to p_{i+1}. Vector N is normal to the plane containing the polygon; this is easy to construct as $e_0 \times e_1$, or as $e_i \times e_{i+1}$ for any i.

Vector n_i is an outward-pointing normal for edge e_i; we can construct this as $e_i \times N$.

Here is an illustration of all of these items:
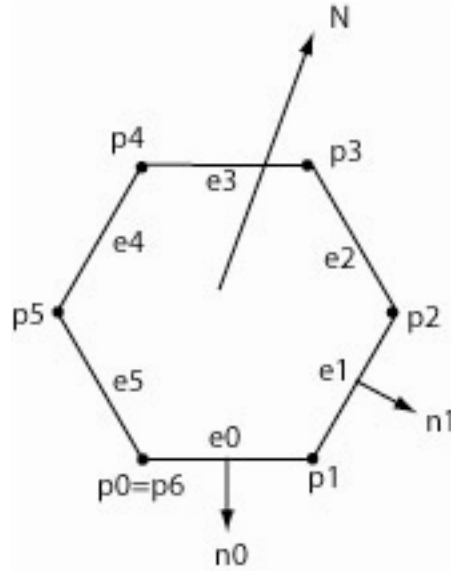
Note that $e_i = p_{i+1} - p_i$, and n_i is normal to e_i, so $n_i \bullet (p_{i+1} - p_i) = 0$. This means that $n_i \bullet p_{i+1} = n_i \bullet p_i$. It is easy to extend this to show that $n_i \bullet p = n_i \bullet p_i$ for any point p on the line defined by edge e_i. This lets us split the plane into two regions: points p for which $n_i \bullet p \leq n_i \bullet p_i$ and points p for which $n_i \bullet p \geq n_i \bullet p_i$. Which side is the polygon on? Consider point $p_i + n_i$, which is outside the polygon because n_i is an outward-pointing normal. $n_i \bullet (p_i + n_i) = n_i \bullet p_i + n_i \bullet n_i = n_i \bullet p_i + |n_i|^2 > n_i \bullet p_i$. Thus the half-plane containing points p where $n_i \bullet p \geq n_i \bullet p_i$ is on the side of the edge pointed to by the outward-pointing normal; it does not contain the polygon. The half-plane defined by $n_i \bullet p \leq n_i \bullet p_i$ does contain the polygon.

Our algorithm for determining if point p is inside the polygon is thus:

p is inside the polygon if $n_i \bullet p \leq n_i \bullet p_i$ for each edge e_i, where n_i is the outward-pointing normal to edge e_i.

Determining if a point is inside a convex polyhedron

The argument above for convex polygons also applies to convex polyhedron:

p is inside the polyhedron if $n_i \bullet p \leq n_i \bullet p_i$ for each face f_i, where n_i is the outward-pointing normal to face f_i.

Algorithm for finding reflection rays

The Phong model for specular reflections has light reflecting from a surface in such as way that angle of incidence equals angle of reflection. To use this model we need to find the reflection direction, given the surface normal and the direction of the light source:
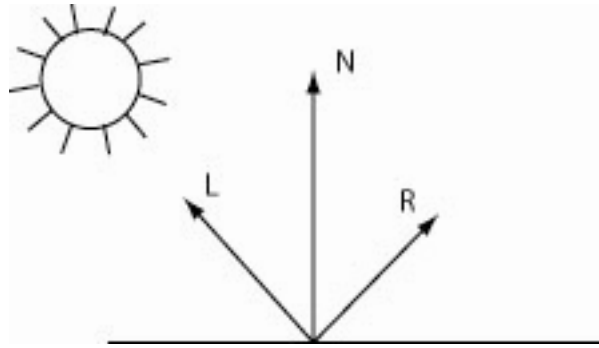
In this picture **L** is the direction of the light source and **R** is the direction we want to calculate. **N** is the surface normal. The angle between **N** and **L** should be the same as the angle between **N** and **R**.

We can solve this problem using projections. The following diagram shows the additional parts we need:

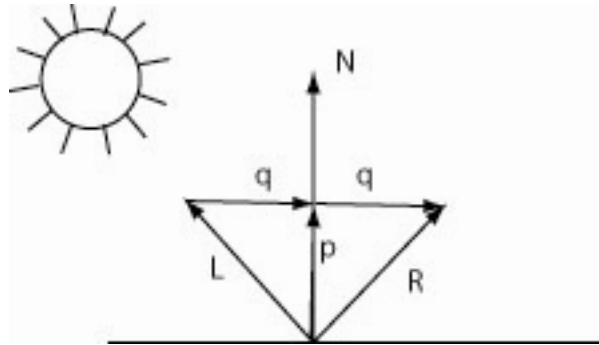Vector **p** is the projection of **L** onto **N**, so $p = \dfrac{N \bullet L}{|N|^2} N$. Vector **q** connects **L** with its

projection: **L+q=p**, so **q=p-L**. Finally, **R=p+q=2p-L**. Putting all of this together, we see that

$$R = \frac{2N \bullet L}{|N|^2} N - L$$

If we choose **N** to be a unit normal to the surface we can make this even easier:

$$R = (2N \bullet L)N - L$$

In any case this formula produces a vector **R** with the same length as **L**.

Part IV: Ray Intersection Algorithms

This is a collection of algorithms used in ray tracing for finding where a ray intersects various shapes. The ray has an equation of the form $P = P_0 + tv$, where P_0 is the starting point of the ray and **v** is a vector parallel to the ray. These algorithms generally give a value of the parameter t. If t < 0 there is no intersection; if t >=0 you can use the expression $P_0 + tv$ to get the actual point of intersection. All of these algorithms make no assumption about the length of **v**. You need to be careful about using algorithms off the internet for this; many other algorithms have an unstated assumption that $|v| = 1$.

Ray-Sphere Intersection, Algebraic Version

This uses the quadratic formula to find t. Our ray is $P = P_0 + tv$; the sphere is centered at point C and has radius r.
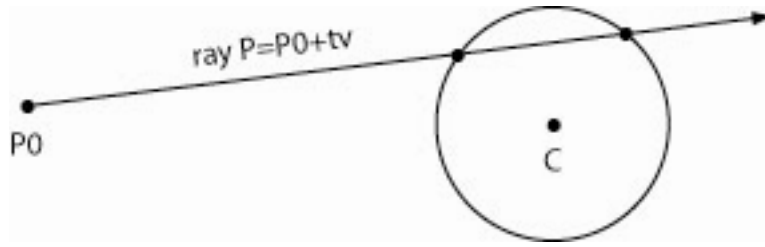
A vector formulation of the sphere is $|P - C| = r^2$, or $(P - C) \bullet (P - C) = r^2$. Inserting the ray definition of P gives $(P_0 + tv - C) \bullet (P_0 + tv - C) = r^2$. To simplify this create a new vector $A = C - P_0$. Our equation is now $(tv - A) \bullet (tv - A) = r^2$. We can multiply this out and collect terms, remembering that $v \bullet v = |v|^2$:

$$t^2 |v|^2 - 2t(v \bullet A) + |A|^2 - r^2 = 0.$$

It is now a simple matter to use the quadratic formula to find t:

$$t = \frac{2(v \bullet A) \pm \sqrt{4(v \bullet A)^2 - 4|v|^2 (|A| - r^2)}}{2|v|^2}$$

Of course, if the quantity inside the square root is negative there is no intersection.

Ray-Sphere Intersection, Geometric Version

This is an alternative way to find the intersection. It does more geometry and gives a simpler calculation, though it is a bit harder to follow. In many situations it gives a quicker exit when the ray misses the sphere.
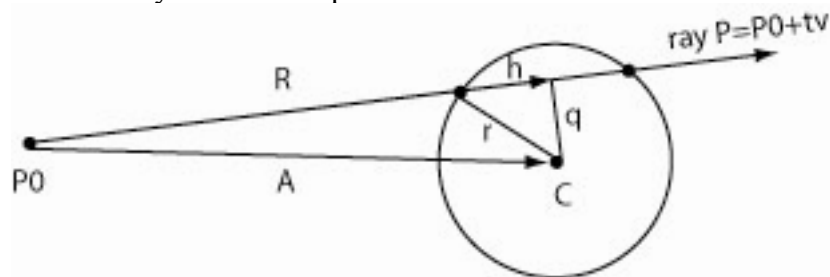
Again we let $A = C - P_0$; this is a vector from the starting point of the ray to the center of the sphere. Let R be the projection of A onto v; $R = \frac{v \bullet A}{|v|^2} v$. Now let q be the distance from the center of the sphere to the ray. Because R is the projection of A, the picture above shows that $q^2 + |R|^2 = |A|^2$. Of course, if $q > r$ then the ray does not intersect the sphere and we can stop. If there is an intersection we let h be half of the length of the chord formed by the ray inside the circle. The picture above shows that $h^2 + q^2 = r^2$, so

$h = \sqrt{r^2 - q^2}$. Finally, the distances of the two intersection points from P_0 are $|R| - h$ and $|R| + h$. We get the actual t-values by scaling these by the length of v.

Altogether, here is the sequence of calculations needed to find the intersection:

$$A = C - P_0$$

$$R = \frac{v \bullet A}{|v|^2} v$$

$$q = \sqrt{|A| - |R|^2}$$

If $q > r$ halt; there is no intersection.

$$h = \sqrt{r^2 - q^2} = \sqrt{r^2 + |R|^2 - |A|^2}$$

The two points are $P_0 + \dfrac{|R| - h}{|v|} v$ and $P_0 + \dfrac{|R| + h}{|v|} v$

Ray-Ellipsoid Intersection

An *ellipsoid* is the 3D analog of an ellipse. One way to write an equation for it is

$\dfrac{(x - c_1)^2}{a^2} + \dfrac{(y - c_2)^2}{b^2} + \dfrac{(z - c_3)^2}{c^2} = 1$. Such an ellipsoid has a, b, and c as half the lengths

of its three major axes. To give a vector form of this equation, let $M = \begin{bmatrix} \dfrac{1}{a} & 0 & 0 \\ 0 & \dfrac{1}{b} & 0 \\ 0 & 0 & \dfrac{1}{c} \end{bmatrix}$; this

represents a scaling transformation. Then for any point $P = (x, y, z)$ we have

$PM = \left(\dfrac{x}{a} \quad \dfrac{y}{b} \quad \dfrac{z}{c} \right)$. An ellipsoid centered at C with half-axes of lengths a, b, and c

consists of points P where $|(P - C)M|^2 = 1$. Note that this says that an ellipsoid is a body that can be scaled into the unit sphere. The rest of the derivation follows closely the algebraic form of the ray-sphere intersection algorithm. We can write the ellipsoid equation as $(PM - CM) \bullet (PM - CM) = 1$. We want to find such points where $P = P_0 + tv$. Inserting this into the equation gives $(P_0 M + tvM - CM) \bullet (P_0 M + tvM - CM) = 1$. Let $v_1 = vM$ and let $P_1 = P_0 M - CM$. Our equation is $(P_1 + tv_1) \bullet (P_1 + tv_1) = |P_1|^2 + 2tP_1 \bullet v_1 + t^2|v_1|^2 = 1$. This is easy to solve for t with the quadratic formula.

Ray-Plane Intersection

This intersects a ray with an infinite plane. Once again, we have ray $P = P_0 + tv$. A plane has equation $Ax+By+Cz=D$. We can also write this $N \bullet P = D$, where $N=<A,B,C>$ is the normal to the plane. Inserting $P = P_0 + tv$ into this, we have $N \bullet (P_0 + tv) = D$, or $tN \bullet v = D - N \bullet P_0$. This is easy to solve for t: $t = \dfrac{D - N \bullet P_0}{N \bullet v}$. Of course, this makes no sense if $N \bullet v = 0$, but in that case v is orthogonal to the plane's normal and so the ray either is completely contained in the plane or else misses the plane altogether.

Of course, once we have the appropriate value of t we can obtain the actual point of intersection from $P = P_0 + tv$.

Ray-Polygon Intersection

To intersect a ray with a polygon, first find the plane that contains the polygon (its normal can be found from the cross-product of any two non-parallel edges of the polygon) and find the point where the ray intersects that polygon. You must then decide if this point is inside the polygon. If the polygon is convex you can use the algorithm given above.

This algorithm fails in the case where the ray is orthogonal to the normal to the plane containing the polygon. In many situations it is sufficient to report "no intersection" in this case. If, however, the ray lies in the same plane as the polygon and your need to find the intersection, it can be found by intersecting the ray with the line determined by each edge of the polygon and testing whether the resulting point is inside the polygon. An algorithm for finding the intersection of two lines in a plane, or a ray with a line in the same plane, is given below. The smallest positive t-value that satisfies this will yield the desired intersection.

Finding the intersection of two lines in a plane

This algorithm can be used either to intersect two lines in a plane, or to intersect a ray with a line in the same plane.

Suppose we have two lines: $P = P_0 + tv_0$ and $P = P_1 + sv_1$. Note that it is important that the two lines have different parameter: they could intersect, for example, at a point given by t=1 on one line and s=2 on the other line. We want to find the values of t and s that result in the same point P on each line. Let N be the normal to the plane containing these lines; this can be found, for example, as $N = v_0 \times v_1$. Let n_1 be the normal in the plane to vector v_1. As we have seen before, every point on the line $P = P_1 + sv_1$ has $n_1 \bullet P = n_1 \bullet P_1$. We want to find a point on the first line that has this same property. Thus, we set $n_1 \bullet (P_0 + tv_0) = n_1 \bullet P_1$ and solve for t. This gives

$$t = \frac{n_1 \bullet P_1 - n_1 \bullet P_0}{n_1 \bullet v_0}$$

With this value of t the point of intersection is $P_0 + tv_0$

Ray-Polyhedron intersection

To intersect a ray with a polyhedron, we find the intersection of the ray with the plane containing each of the faces of the polyhedron, and then test whether the resulting points are inside the polyhedron. An algorithm is given above for determining whether a point is inside a convex polyhedron, so if the polyhedron is convex this solves the problem. Again, if the polyhedron is convex and we know the ray starts outside the polyhedron, it is sufficient to intersect the ray with the front-facing faces. These are the faces whose outward pointing normals are in the opposite direction from the ray. That is, if $\mathbf{N}$ is the outward-pointing normal and the ray is $P = P_0 + tv$, then we need to check the intersection of the ray with the face only if $N \bullet v < 0$.