Trigonometry Notes

Trigonometry is one of those subjects that leaves students asking "What's in it for me?" After all, knowing a lot of stuff about ancient Greeks and how to prove theorems isn't much use to a lawyer or a musician. Graphic arts is another story, at least if you want to write your own programs. There are no theorems to prove, but some of the basic ideas are necessary all the time. This paper covers the concepts you are likely to encounter when doing graphics in Max or Processing.

Cartesian coordinates.

Trigonometry lives on graph paper:

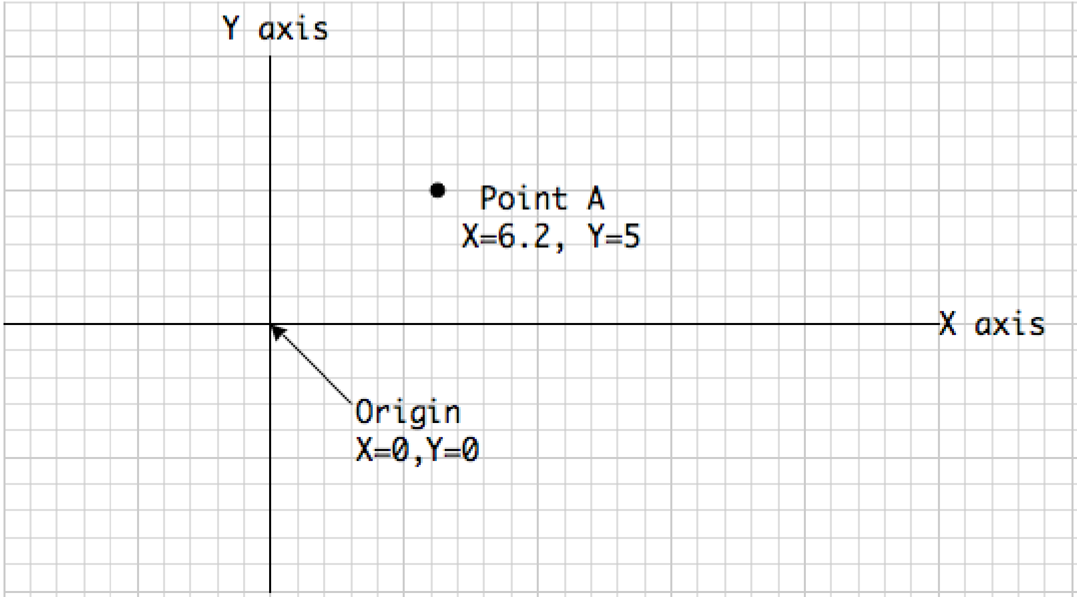


Figure 1. Cartesian space is measured with grid lines.

Figure 1 shows the usual layout. Each line of the grid has a number, and locations on the grid are defined by the intersection of the two nearest grid lines. It's a lot like my saying I live at the corner of Laurel and Cleveland, or I'll meet you at third and Beach. Since this is math, we count to name the lines, and we count starting with 0. The intersection of 0 and 0 is the origin, and lines left of the origin or below it have negative numbers. Also since this is math, we assign letters to indicate horizontal or vertical lines: X for counting to the left and Y for counting up. The horizontal numbered 0 is called the X axis, and the vertical numbered 0 is the Y axis. Any point on the paper can then be defined with an X and Y value. If the point is between lines, it can have fractional values. We always put X before Y, so it makes sense to say point A is at (6.2,5).
Further points about Cartesian graphs:

- There can be three dimensions-- the Z axis points out of the paper, and is counted with positive numbers in front of the page. More dimensions are possible in math, but hard to draw in our universe.
- You are free to move the origin around, as long as you adjust all of the other numbers to compensate.
- This system is named after Rene Descartes, who lived in the early 17th century. He didn't invent it, but his writings about it are so important that other mathematicians copied his notation.

The application of this to computer graphics should be obvious: it's how we describe the pixels on a screen. The origin is the upper left corner, and pixels are counted down and left. Since Y is positive going up and pixel addresses are positive going down, graphics have to be turned upside down before they are displayed. Some programs do this for you, some do not[1].

Polar Coordinates

There is a second way to look at location. In polar notation, a point is defined by a radius and angle, as if it were on a circle:

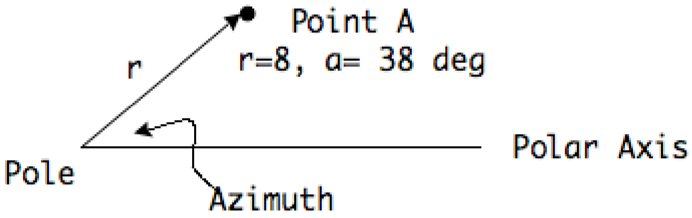


Figure 2.
The center of the circle is called the pole, and is equivalent to the origin in the Cartesian system. (Everything is measured from here). A horizontal line from the pole to the right is called the polar axis. To locate a point, draw a line from the pole. The length of this line is the radius r, and the angle the line makes with the polar axis is the azimuth, a[2]. Angles may be measured in degrees or radians. These are both defined in terms of how many are in a complete circle: 360 degrees or 2π radians.[3]

This may seem like an awkward ward way to describe locations, but polar notation makes some operations very easy. For instance, rotation around a point just requires a change to the angle. We usually work in Cartesian coordinates, but occasionally convert to polar for

[1] When we get to openGL, the origin is in the middle and Y counts up. Z is toward the viewer.

[2] Mathematicians generally use Greek letters for azimuth and angles in general. The most common is θ (theta). My graphic program doesn't do Greek, and Greek in Word doesn't print reliably, so I use roman letters.

[3] I thought radians were awful until I started doing math with them. Any formula that includes both π and angles in radians is easy to work out because the π terms will cancel out.

some operation, and immediately convert back. To see how to convert, we must revisit the bane of all trig students, the Trigonometric Functions.

Trigonometric Functions

Trigonometry originated in surveying, looking at problems like determining the height of a tree without cutting it down. Many of the problems are about angles and distance, and can be solved by the basic functions of sine, cosine or tangent.

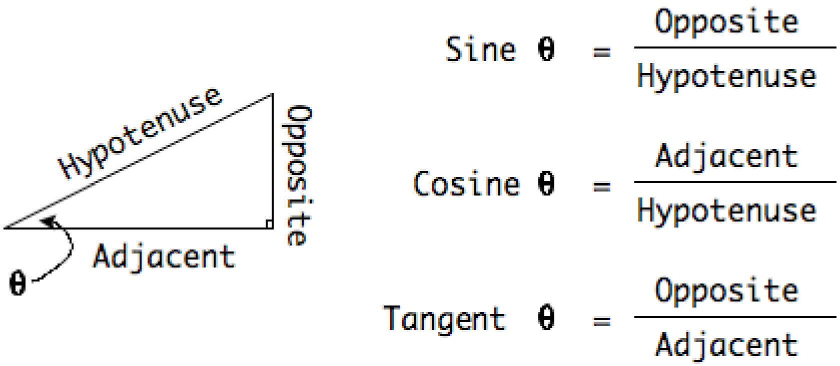

Hypotenuse
Opposite
Adjacent
θ
Sine θ = Opposite / Hypotenuse
Cosine θ = Adjacent / Hypotenuse
Tangent θ = Opposite / Adjacent

Figure 3.

As figure 3 shows, these functions are defined via a triangle with a 90° corner, or right triangle. The sides are identified as to whether they are opposite or adjacent to the angle we know. (The side opposite the right angle is the hypotenuse.)

For any angle, the sine of the angle is defined as the ratio of the length of the opposite side to the length of the hypotenuse- since the hypotenuse is always the longest side, sines will be a fraction. The sine of a particular angle is always the same- to find what it is, you can just look it up in a table.[4]

The cosine of an angle is the length of the adjacent side over the hypotenuse. If you look at the drawing, you can see that the cosine of the angle θ is the sine of the unmarked angle, and vice versa.

The tangent is the ratio of the opposite to adjacent sides. Most of the time, we know this ratio, and want to find the angle, so we look for the tangent in the table. This process is called finding an arctangent.

There are other functions, such as secant (hypotenuse/adjacent) but they are aren't used as often. The big three are all we really need to memorize. Among other things, they will allow us to convert between polar and Cartesian coordinates.

Cartesian to Polar and Back

Figure 4 shows how Cartesian and Polar coordinates are related.

[4] Many mathematicians spent their careers constructing tables of sines.

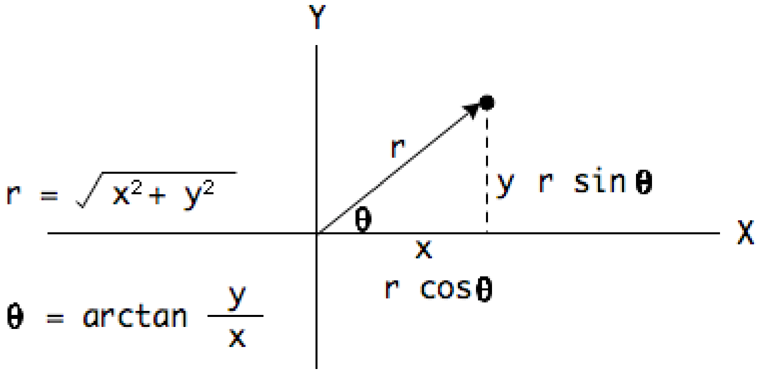


Figure 4.

To go from Cartesian to Polar, the radius is the square root of the sum of the squares of the other two sides. The angle is the arctangent of the ratio of opposite to adjacent. In C or Java code that would be:

```
r = sqrt(x*x + y*y);
theta = atan(y/x);
```

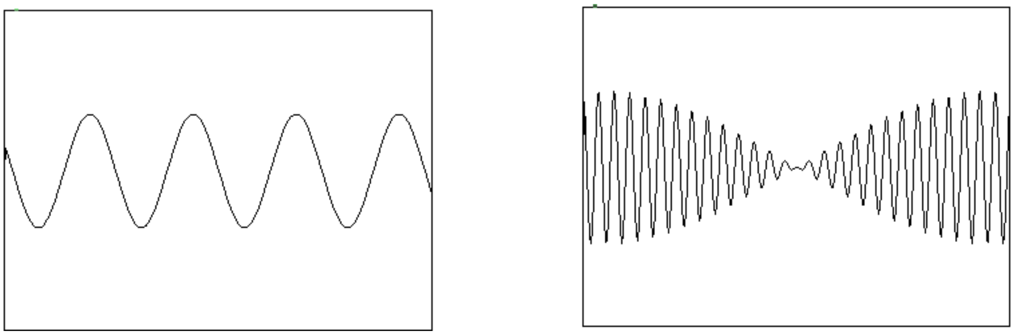
Of course in Max the whole thing is calculated in cartopol. The angle is reported in radians. To go back after the process:

```
y = r*sin(theta);
x = r*cos(theta);
```

Or the poltocar object. One other interesting use of these formulas is that if you plot them with a steady radius and a theta that hits most of the numbers between 0 and 2π, you will get a circle.

Plotting Sine Functions

We don't really need to bother ourselves with the numbers in the sine table, the computer looks them up for us. However I often do play with these numbers, because they are beautiful. Figure 5 shows a few examples of sine based drawing. In the later cases, lines are drawn between points on two different sine waves.



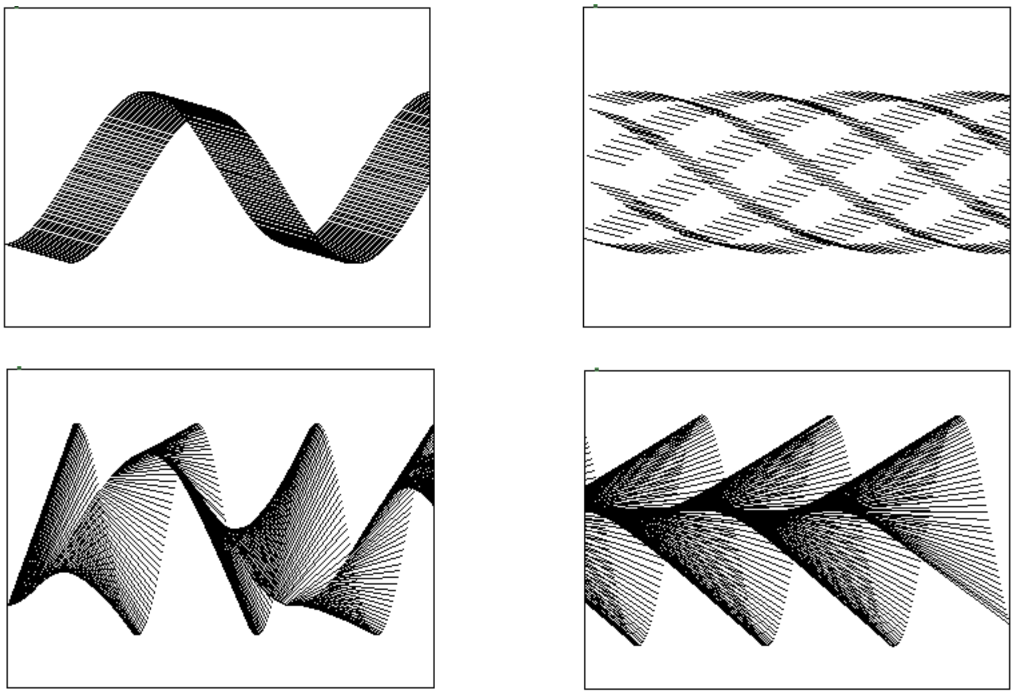


Figure 5. Sine wave art. These patterns often move slowly across the screen.

As I mentioned before, plotting x=cos(theta) and y = sin(theta) will result in circles.

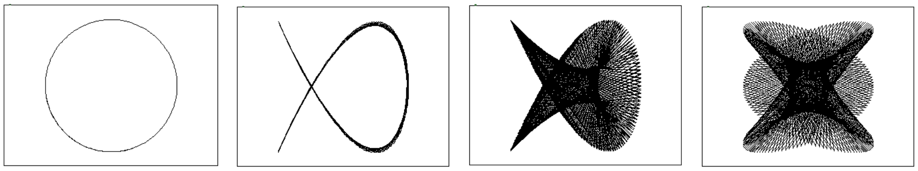


Figure 6. Lissajous patterns

If you use different angles in the sine and cosine terms you can generate many complex forms based on Lissajous patterns (See my tutorial; Lissajous art).

If you start with a polar view of things you can make spirals and flowers by manipulating the radius. The images in figure 7 are based on that technique.

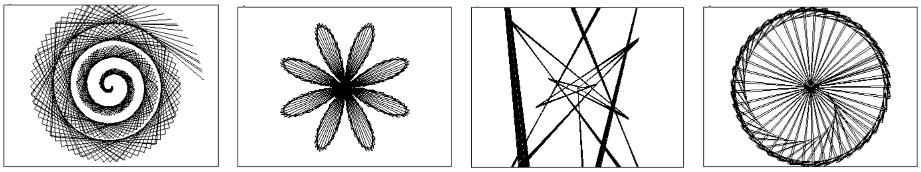


Figure 7. Polar images. These can be made to spin.

Vectors

Basics of vectors

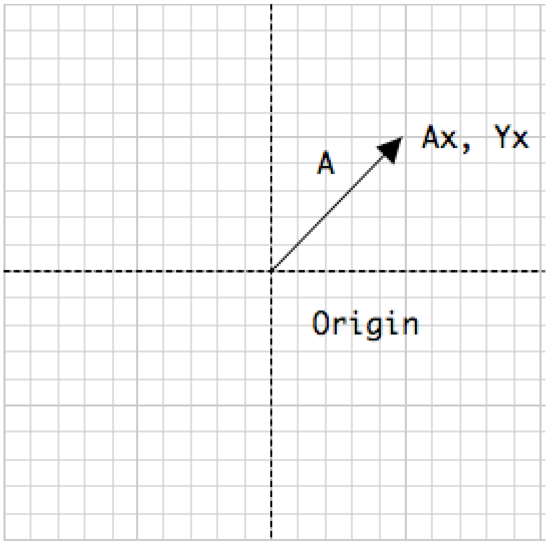


Figure 8.

A vector is a number pair that describes a spatial relationship. Vectors may be polar or Cartesian, although the latter are more common. They are usually defined by two numbers[5], which describe how the vector looks if it starts at the origin. The numbers are then the x and y coordinates of the point. Vectors may be labeled by a letter (generally upper case), A letter under an arrow, or two letters with or without an arrow- in that case the vector AB is the vector from point A to Point B. Vectors may also be indicated by the coordinates in a pair of brackets, either as a row or a column.

$$\vec{\mathrm{A}} \qquad \overrightarrow{\mathbf{AB}} \qquad [A_x, A_y] \qquad \begin{bmatrix} A_x \\ A_y \end{bmatrix}$$

You can think of the space defined by a vector as a path to travel, so a vector defines a direction and distance[6]. The distance or length of the vector is the magnitude. It can be easily calculated by the hypotenuse formula. Magnitude is often indicated by |A|. The tangent of the angle of travel is the ratio of the X and Y coordinates, so the angle can be found by the arctangent. In fact, this is just the Cartesian to polar conversion.

Keeping the association of motion and vectors in mind tells us how (and why) to add vectors. The result should tell where we wind up, it is as if one vector were stuck at the point of the other:

[5] Or three or more, we'll come to that in a bit.

[6] A vector has the same semantic meaning as "three miles north". It's a concept we understand, although it's not much use until we know where to start.

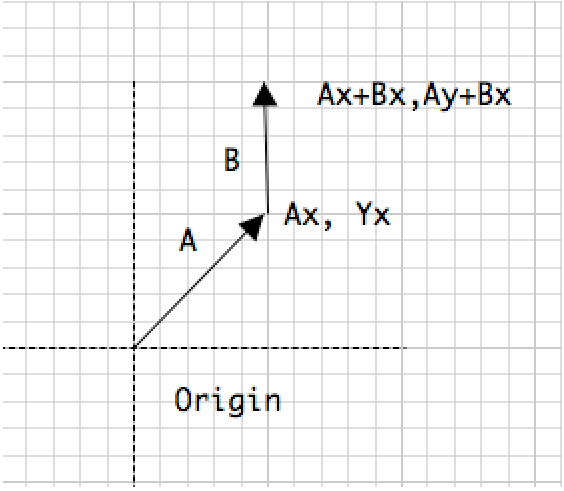


Figure 9.

The new X coordinate is the sum of the vectors' X and the new Y coordinate is the sum of the vectors' Y. These points define a new vector, of course, and you can draw it in if you like.

Simple addition accounts for 90% of our work with vectors. Vectors are used to represent anything that has magnitude and direction. Forces, velocity, stock market trends, line art in Illustrator and so on. Vectors work in three dimensions if we just add a z component. The magnitude becomes sqrt(X^2+Y^2+Z^2). Everything else is the same.

There are times when we use a vector to define a direction, such as the axis of rotation for a 3D object. In that case we think of the magnitude as infinite (defining a line) or we set it to 1. This is called a unit vector.

A line segment is similar to a vector-- it is the line between two points. The line from point A to point B is often called line AB. Both A and B are defined by coordinates. You can convert a line segment to a vector by subtracting the start from the end. A line can also be represented as a starting point and a vector. In that case the form is P+uD, where P is a point, D is a unit vector and u is the length of the line[7].

When we are dealing with objects scattered around a coordinate system, it is often handy to translate the coordinates so our starting point is 0,0. This is done by subtracting the global coordinates of the starting point from all of the coordinates we will work with. When we are done, we need to add those values back to the results.

Advanced Vectors

Sometimes we start with a vector that is not at the origin, as in B of the above drawing. Before we do any math on the vector, we must subtract the coordinates of the vector's starting point (vector origin). Sometimes we will see equations that include this operation with terms like A_x-A_0 . The following operations assume the vector origin is [0 0].

[7] This form is called parametric notation. A lot of sources use t instead of u.

Scaling

A number that is not a vector is a scalar. We can't add a scalar to a vector, but we can multiply them. 3 (a scalar) times [2 2] (a vector) is [6 6]. Each component of the vector is multiplied by the scalar. The operation is called (what else?) scaling.

Dot Product

We can multiply two vectors in a way that produces a scalar. This is the scalar product or dot product[8]. With a bit of geometry, I could show that $A\cdot B = A_xB_x+A_yB_y$. I could also show that it is equal to the product of the magnitudes of the two vectors times the cosine of the angles between them. $A\cdot B = AB\cos\theta$. This lets me find the angle between two vectors by
$\theta = acos((A_x*B_x+A_y*B_y) / (sqrt(A_x*A_x+A_y*A_y)* sqrt(B_x*B_x+B_y*B_y)))$

For three dimensional vectors, the dot product is $A\cdot B = A_xB_x + A_yB_y + A_zB_z$. The cosine formula still holds, but you need to use the z terms in calculating dot product and length.

Normal

A vector normal is another vector that is at right angles to the original. Some math books indicate the normal to vector A as $\widehat{A}$. In two dimensions, the normal is easy-- you just exchange the coordinate values and change the sign of either one. (Any vector has two normals, in opposite directions.) So the normal to [2 4] is [4 -2]. We use the normal to define a direction, so the magnitude is not important. Any math involving the normal is simplified if we adjust the length of the normal to be 1. This is the unit normal, found by dividing each coordinate value by the magnitude. So to find the normal to a vector A:

$N_x = A_y/ sqrt(A_x*A_x+A_y*A_y)$
and
$N_y = -A_x/ sqrt(A_x*A_x+A_y*A_y)$

The dot product of a vector and its normal is always 0. The dot product of any pair of perpendicular lines is 0.

Cross Product

We don't find the normal to a vector in three dimensions, we find the normal to a plane. Two vectors (and their origin) define a plane, and the normal to the plane is found by an operation known as the vector cross product. The cross product[9] is written with an 'x' for the multiplication sign as A x B = N

$N_x = A_yB_z - A_zB_y$
$N_y = A_zB_x - A_xB_z$
$N_z = A_xB_y - A_yB_x$

[8] Also known as inner product.
[9] Cross product may be called outer product.

This operation is essential to determining how a vector is reflected from a plane. This is involved in lighting as well as modeling dynamics like a bouncing ball.

Intersection

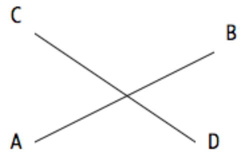


Figure 10.
Given two line segments AB and CD, find the point of intersection.
First calculate a denominator as (Dy-Cy)*(Bx-Ax) - (Dx - Cx)*(By-Ay).
If the denominator is 0, give up, because the lines are parallel.

Next
Uab = ((Dx-Cx)*(Ay - Cy) - (Dy - Cy)*(Ax - Cx))/denominator
Ucd = ((Bx-Ax)*(Ay - Cy) - (By - Ay)*(Ax - Cx))/denominator

If 0 <= Uab <= 1 and <= Ucd <= 1 the lines intersect.
The point of intersection is:
INTx = Ax + Uab*(Bx-Ax)
INTy = Ay + Uab*(By-Ay)

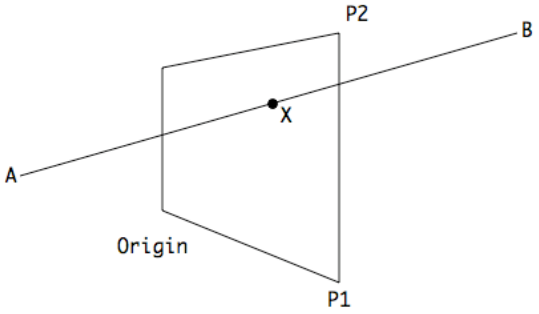


Figure 11.
The intersection of a line AB and plane is a bit more complex. A plane is defined by three points, the origin, P1 and P2.[10] The cross product of P1 and P2 is the normal, N.

The vector of the line AB is (B-A). If (B-A) ·N = 0, the line does not intersect the plane.

The intersection point X = A + (B-A)* (P1 -A) ·N/(B-A) ·N

To find out if the intersection is within a specific polygon on the plane, consider a line from the point to a point on the plane known to be outside the polygon. Test this line for intersections with the edges of the polygon. If there are an odd number of intersections, the point is within the polygon.

[10] If necessary, translate the plane and line so one of the points on the plane is the origin.

Reflection

When modeling a bouncing ball, the reflections off of the top and ends of the frame are simple, we only need change the sign of one of the components of the motion vector. When the surface is angled, the situation is a bit more complex. First we must test that the vector and surface are intersecting as above. For the most accurate model, we will also need the point of intersection.

First find the unit normal to the wall, N. The reflection R of the vector V is symmetrical about the normal. The formula that expresses that is
$R = V-2(V\cdot N)N$.

That's a dot product in there, which produces a scalar, so the component by component math is easy

$R_x = V_x-2(V\cdot N)N_x$
$R_y = V_y-2(V\cdot N)N_y$
$R_z = V_z-2(V\cdot N)N_z$

This works in either 2 or 3 dimensions.