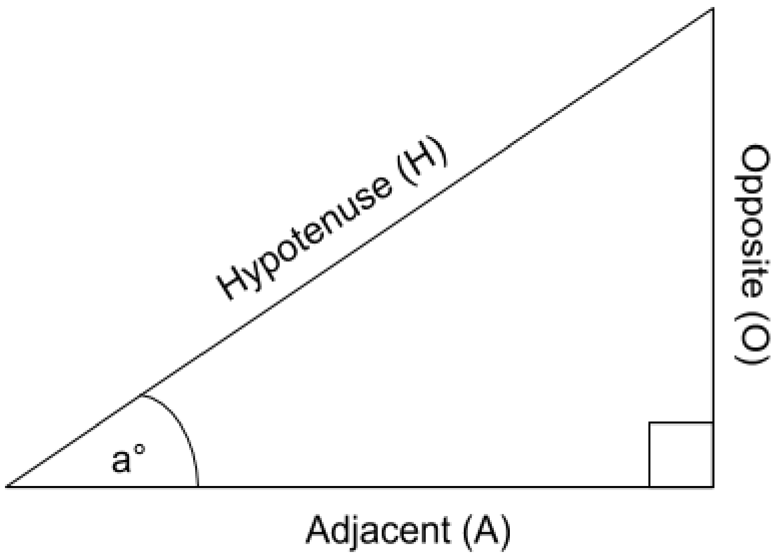
Basic Trigonometry for Games Programming

Overview
Trigonometry is the study of the relationship between angles and triangles.
Greek: [*trigōnon* -> triangle and *metron* -> measure]

Trigonometry is used everywhere. I'll be looking at it, mainly, from a games programming point of view.



On a standard right angled triangle you have 3 sides, and 2 angles other than the 90° right angle.

The sides are the **Hypotenuse**, the **Opposite** and the **Adjacent** sides.

Hypotenuse - The longest side of the right angled triangle.
Opposite - The side directly across from the angle you are working with. In the example above that would be angle (a).
Adjacent - The side that is neither the Hypotenuse or the Opposite. Situated next to (adjacent to) angle (a) or the angle you are working with. So the Opposite side in the diagram above could be swapped with the Adjacent side, provided we were working with the other angle in the triangle.

Trigonometric Functions

There are three trigonometric functions; **Sin**, **Cos** and **Tan**. Each standing for Sine, Cosine and Tangent respectively and their values, when worked out, will be between -1 and 1. These functions all take an angle as you are using them to find the Sine, Cosine or Tangent of said angle.

In most programming languages, you can simply input an angle into the Sin, Cos and Tan functions, although this is the case, it is still useful to know how to find them yourself. The functions return a value but you can also find the Sine, Cosine or Tangent of an angle by dividing the length of 2 sides together, these sides change depending upon which function you are trying to find.

Let’s use angle (a°) as an example:

Sin: To find the Sine of angle (a°) you would either use the **Sin** function *Sin(a°)* or you would divide the length of the Opposite side (O) by the length of the Hypotenuse side (H). This would mean: *Sin(a°) = O / H;*

Cos: To find the Cosine of angle (a°) you would either use the **Cos** function *Cos(a°)* or you would divide the length of the Adjacent side (A) by the length of the Hypotenuse side (H). This would mean: *Cos(a°) = A / H;*

Tan: To find the Tangent of angle (a°) you would either use the **Tan** function *Tan(a°)* or you would divide the length of the Opposite side (O) by the length of the Adjacent side (A). This would mean: *Tan(a°) = O / A;*

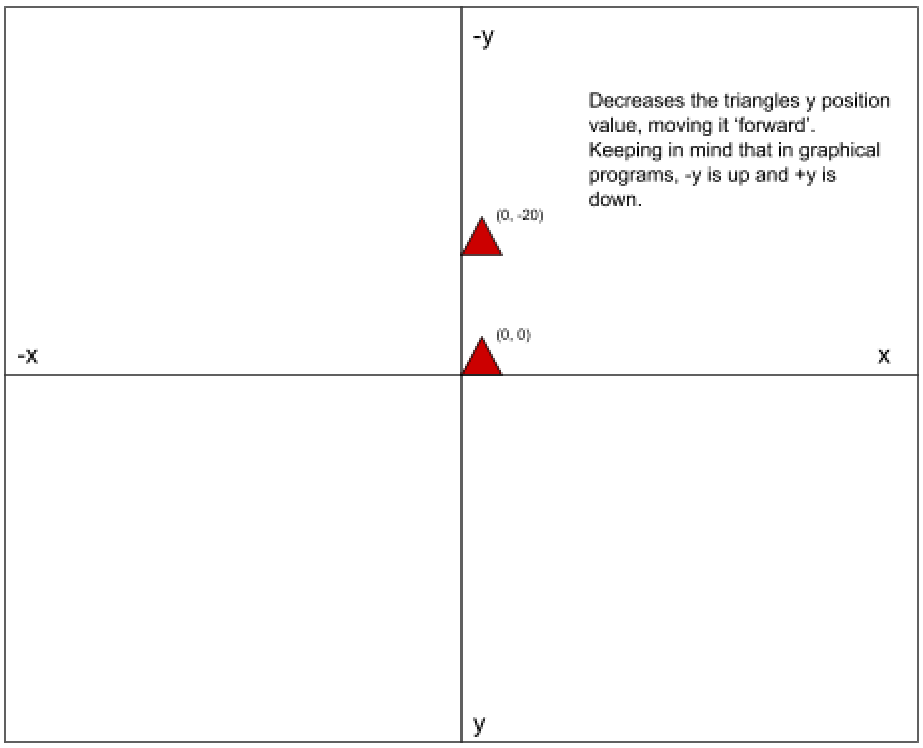
If this seems slightly confusing, don’t worry, we will demonstrate this knowledge later on using code examples and diagrams.

Why does anybody need to know the Sine or Cosine of an angle?

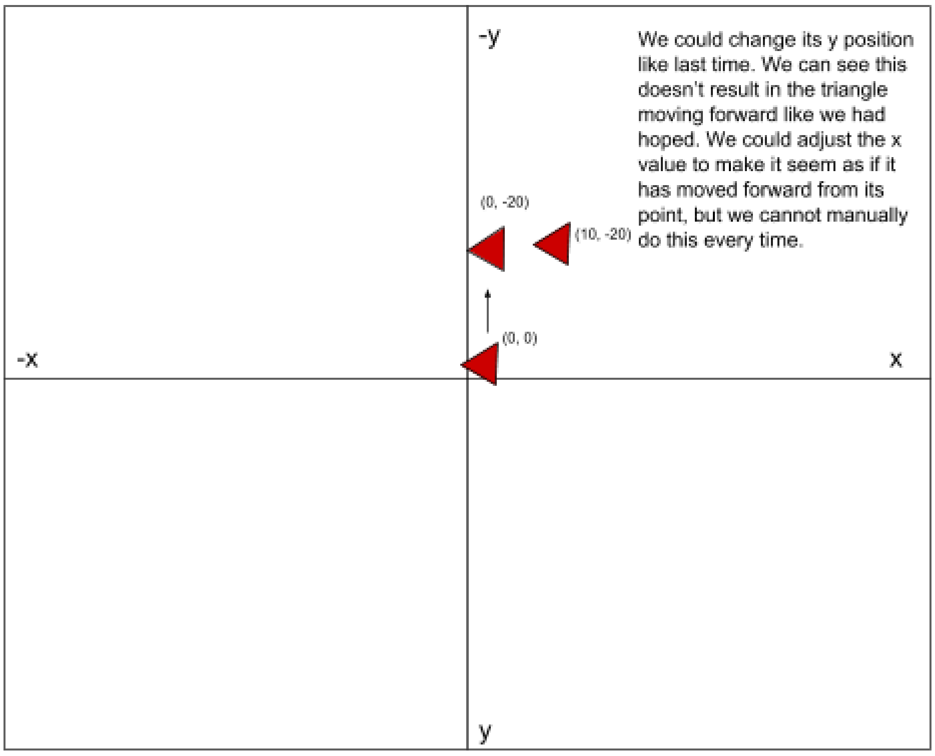
Scenario

Let’s say we have a triangle and we want that triangle to move forward. Now, it being a triangle has nothing to do with trigonometry, it could be any object, it just makes it easier to visualise rotations if it is a triangle, as opposed to something like a circle.

Now, we want this triangle to move forward, this is easy if the triangle is at an angle of 0 degrees as we can just change the x or y position depending on what you view as forward. Assuming forward moves from the top point of the triangle, we can just change the y position of the triangle like so:

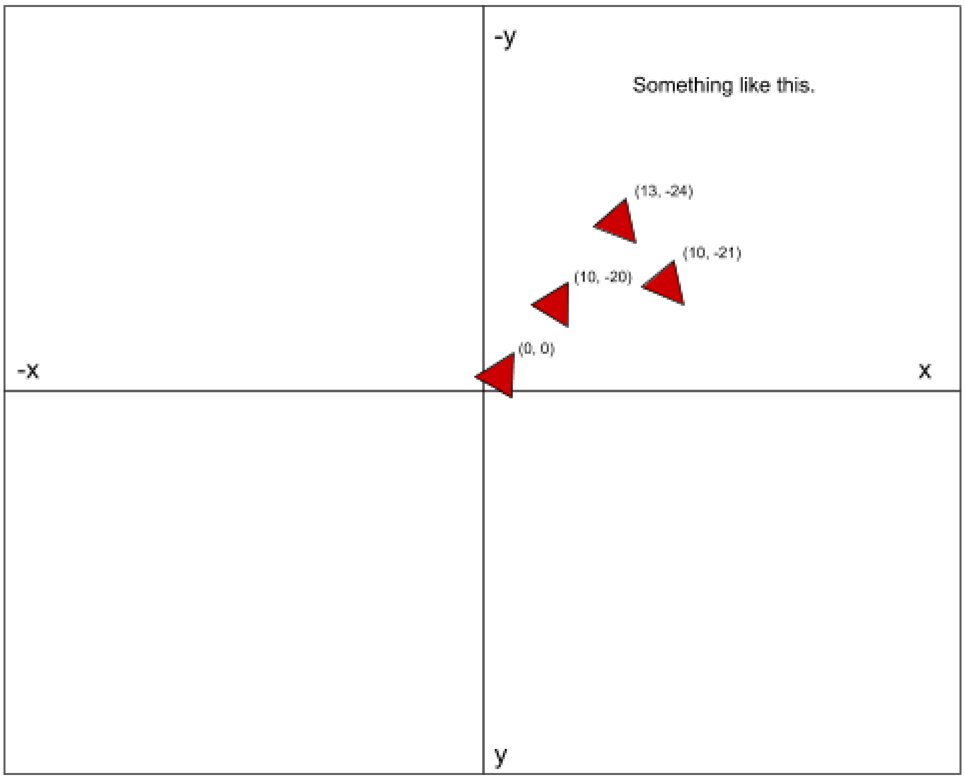

-y
Decreases the triangles y position value, moving it ‘forward’. Keeping in mind that in graphical programs, -y is up and +y is down.
(0, -20)
(0, 0)
-x
x
y

This looks fine but think about our scenario, if we want to move the triangle forward but it is rotated at an angle of 30 degrees, for example, how would we move the triangle forward by its point? You might be thinking to just change the x or y value of the triangle like we did before, but this would result in something we aren’t looking for:

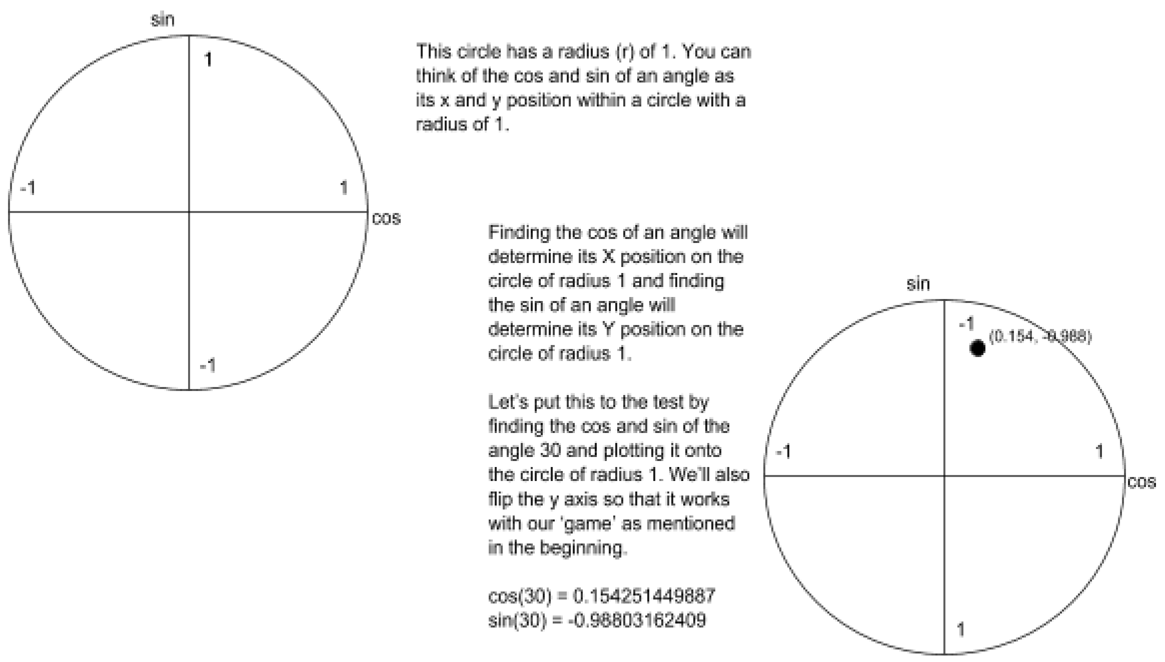

-y
We could change its y position like last time. We can see this doesn't result in the triangle moving forward like we had hoped. We could adjust the x value to make it seem as if it has moved forward from its point, but we cannot manually do this every time.
(0, -20)
(10, -20)
(0, 0)
-x
x
y

Desired Outcome

Obviously the desired outcome is that our triangle, no matter its rotation, will always be moved forward by its point.



Solution
Now obviously this wouldn't be a usage example for trigonometry section if the solution wasn't a usage example for trigonometry and so before we jump in, let's explain what sine and cosine return and why they return what they do.


This circle has a radius (r) of 1. You can think of the cos and sin of an angle as its x and y position within a circle with a radius of 1.

Finding the cos of an angle will determine its X position on the circle of radius 1 and finding the sin of an angle will determine its Y position on the circle of radius 1.

Let's put this to the test by finding the cos and sin of the angle 30 and plotting it onto the circle of radius 1. We'll also flip the y axis so that it works with our 'game' as mentioned in the beginning.

cos(30) = 0.154251449887
sin(30) = -0.98803162409

(Highly recommended) An animated representation of the explanation above can be found at https://commons.wikimedia.org/wiki/File:Mfnf-sincos.gif

Now if we look at our results we can see it gives us an x and a y value (cos and sin respectively), we could use this to move our triangle but it wouldn't move anywhere as the values are too small, what we can do is use the values returned from the cos and sin functions as a direction and multiply the values by a larger number that could act as our speed, essentially taking the x and y positions from within the circle with a radius of 1 and converting it to units that can be used in our coordinates system. We'll look at a code example below to cement this idea.

```
object.x = object.x + (speed * Math.cos(30));
object.y = object.y + (speed * Math.sin(30));
```

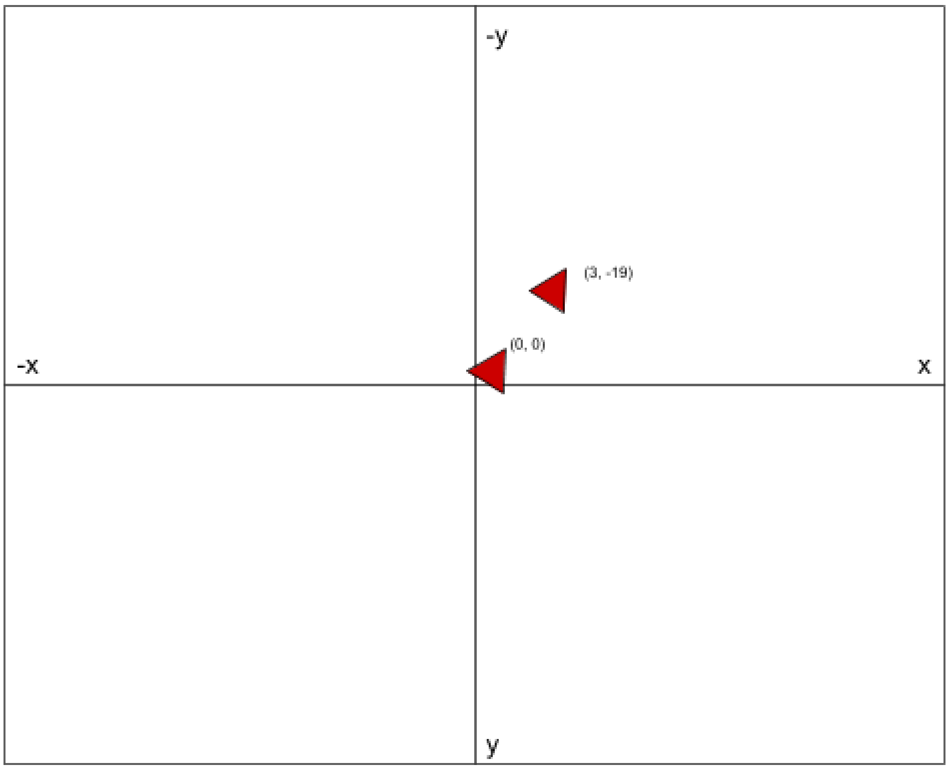
Here we can see that the object's x position is being updated to the result of *(speed * cos(30))*. We can assume that *speed* is an arbitrary value provided just to move the triangle and that *30* is the angle the triangle is currently rotated at. Let's walk through the equation:

object.x = object.x + (speed * cos(30)) and let's give *object.x* a starting x position of *0* and *speed* a value of *20*. The equation becomes ***object.x = 0 + (20 * cos(30))*** which becomes ***object.x = (20 * 0.154251440887)*** which will finally mean that *object.x* is equal to *3.085028997751*, or *3* to make it easy to write down. If you're still with me after all those numbers, great, we just figured out the triangles X position if it is at a *30* degree angle. You can substitute in any angle and you will get another result, we could also change *speed* from *20* to something higher or lower as this is only the arbitrary amount by which the triangle is actually moved and has nothing to do with the direction.

Now knowing what we know, we can look at the second line of code. ***object.y = object.y + (speed * sin(30))***. This is obviously the same as the first line, except for two small changes, we are changing the object's Y value and we are using the sin function instead of the cos function as cosine determines the X value and sine determines the Y value (see diagram above).

Running through this equation, filling in the same values as with the X position: **object.y = 0 + (20 * sin(30))** > **object.y = (20 * -0.98803162409)** which will finally mean that object.y = -19.7606324818 or -19 to make it easier to write.

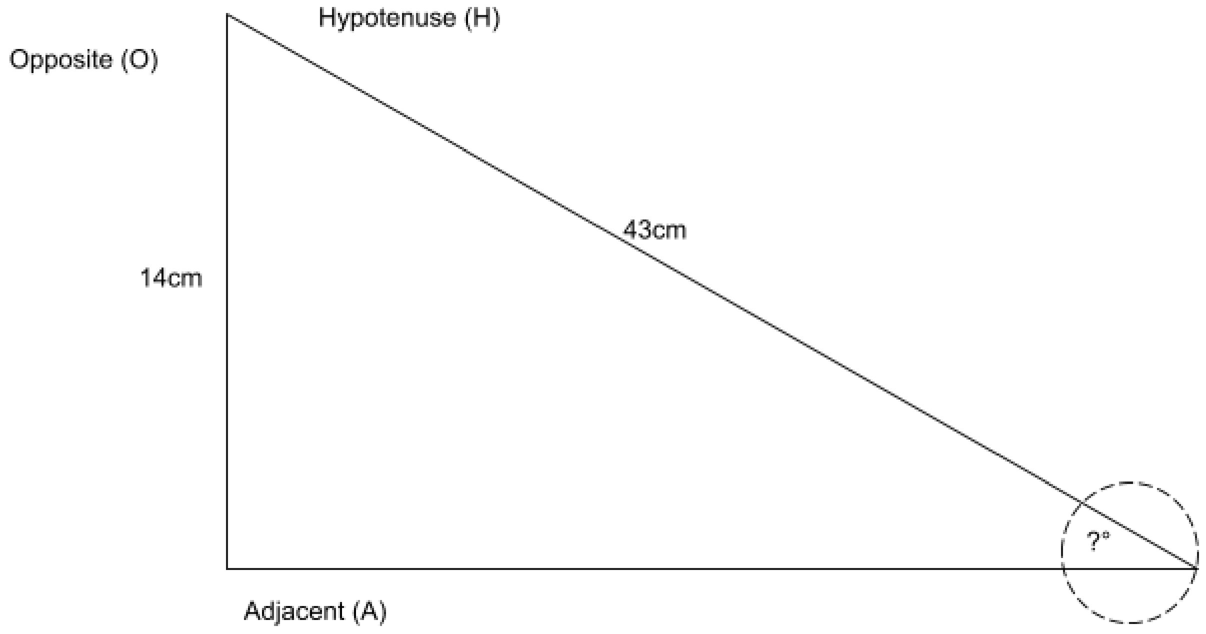
Knowing all of this, we can see that if our triangle starts at position (0, 0) at an angle of 30 degrees, to move forward from its point at a speed of 20 would mean its new position would be (3, -19). We can clearly see we have now reached our desired outcome in the diagram below.



What about if we want to find an angle?

If you've got this far, you might be wondering what the whole Hypotenuse, Opposite and Adjacent sides stuff was at the beginning, and don't worry, I haven't forgotten about it. Here's a practical use for all that stuff.

If we know the lengths of 2 sides on a triangle, we can use that information to get the angle of that triangle. Obviously using the information at the very beginning, this can be used to find any angle on that triangle.



As mentioned at the beginning, the sine of an angle is the length of the opposite side divided by the length of the hypotenuse side, the cosine of an angle is the length of the adjacent side divided by the length of the hypotenuse side and the tangent of an angle is the length of the opposite side divided by the length of adjacent side.

In the example above we know the length of the Opposite side and the length of the Hypotenuse and we need to find the angle, using our knowledge, we know that Opposite/Hypotenuse will result in the sine of an angle. This means that: 14 / 43 = sin(?°). Using this, we know that the sine of the hidden angle is 0.325581395248. We can now use something called an inverse sine which is basically the sin function backwards, this is denoted as: $\sin^{-1}$(0.325581395248), most calculators will do this and return the result, each programming language may have a different name for the function, but it is commonly known as asin (or acos, atan respectively). So in JavaScript for example, writing asin(0.325581395248) will give us the result 19°.

*Remember that a lot of programming languages return their angles as radians whereas I discuss angles as degrees, to convert these you can use this formula:
*Degrees = (Radians * 180) / PI*

Those were the basics of Trigonometry, there is obviously a lot more involved, but the knowledge covered above should be enough for simple 2D games, when dealing with rotation and angles.