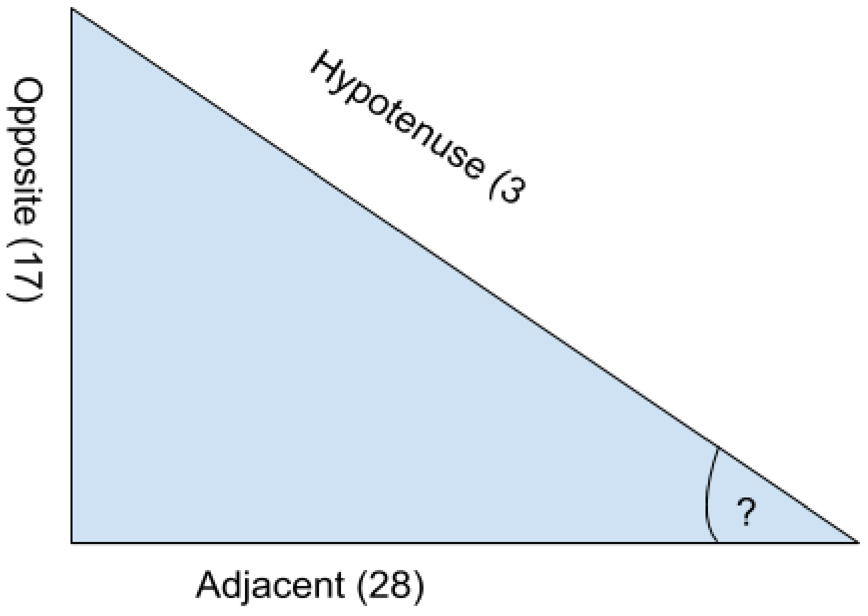
Trigonometry: Atan and Atan2

What is Atan?
Atan is the inverse of the Tangent, just as Asin and Acos are their respective inverses. Putting a value into the Atan (or Asin, Acos) function when we know 2 sides of a triangle will yield an angle.

So, to put that into perspective. If we had the following triangle:



We know all of the sides but the hypotenuse, and so how would we get the missing angle?

Well we know that the Sine of an angle is (Opposite/Hypotenuse), so we can't use that, and we know that the Cosine of an angle is (Adjacent/Hypotenuse), so we can't use that either; This leaves us with the Tangent of an angle (Opposite/Adjacent).

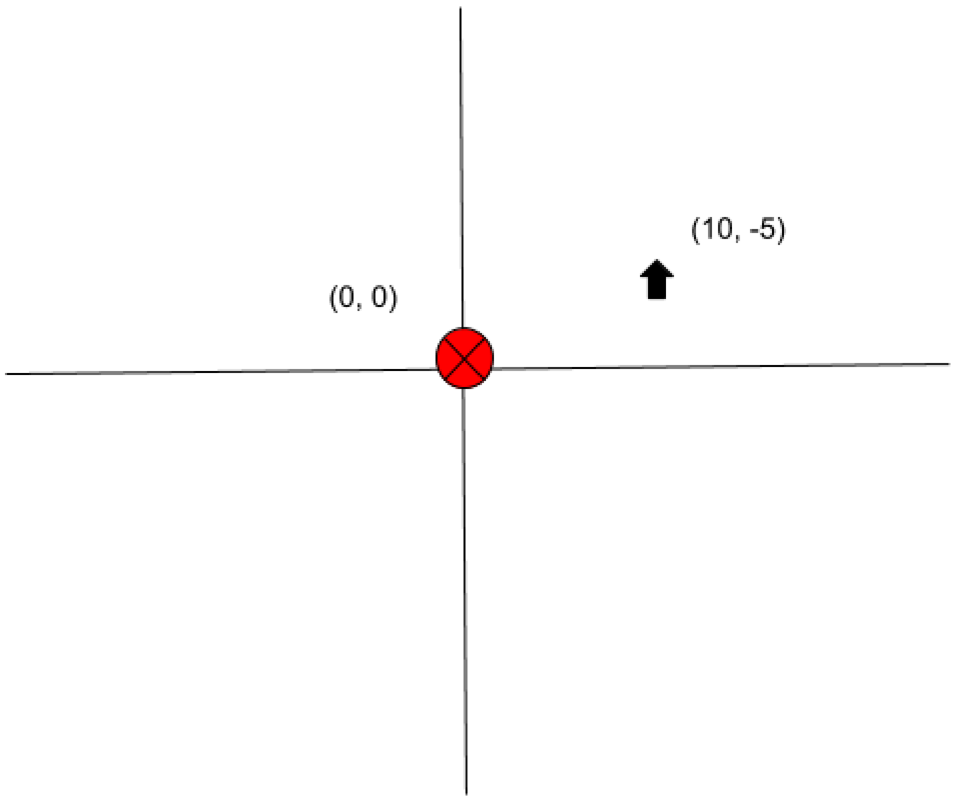
Knowing this, the Tangent of the angle is (17 / 28). Which gets us 0.607. Now if you remember back to the previous document, 0.607 is the Tangent, and to find the angle from this, we need to use the inverse Tangent, exactly the same as we did with the inverse sine and inverse cosine.

This is where Atan comes in. Putting our Tangent value into the Atan function will gives us the missing angle. So let's try that.

If our Tangent is 0.607, then atan(0.607) is equal to 0.545 radians, which is 31 degrees. *(RADIANS * 180) / PI*

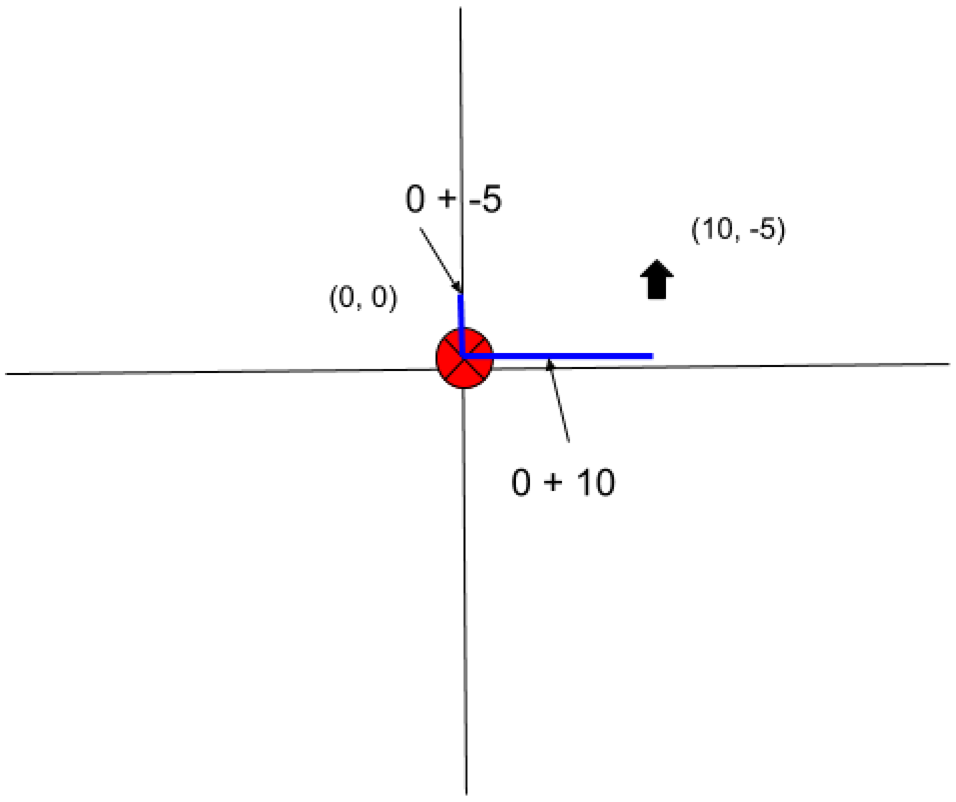
Now, to avoid seeming random, let’s put our knowledge to the test in a game-like scenario.

The Scenario
You have a player, he has a position of (0, 0), you want your player to face in the direction of the mouse cursor, which currently sits at (10, -5).

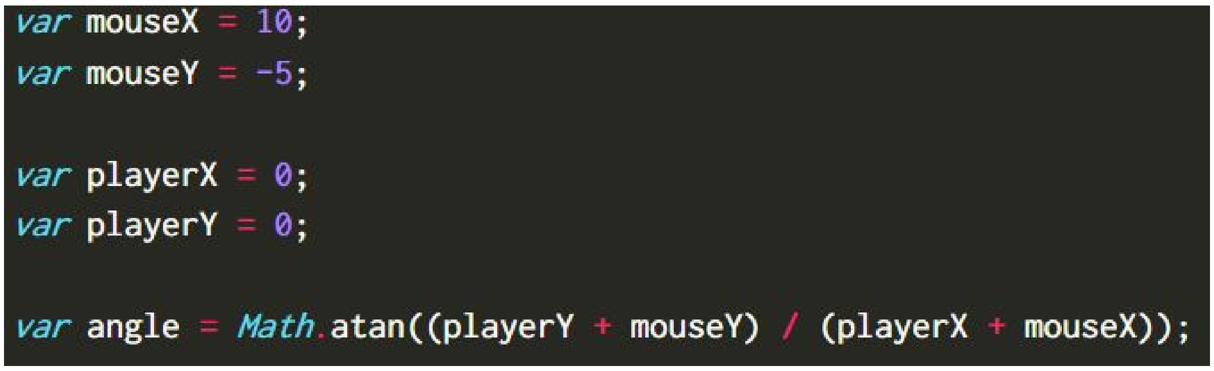


We know that in order to find the angle, we need a triangle. So let’s create an invisible one. The ‘Adjacent’ side of the triangle will be the Player’s X position (0) + the Mouse Cursor’s X position (10) and the ‘Opposite’ side of the triangle will be the Player’s Y position (0) + the Mouse Cursor’s Y position (-5).

Let’s visualise this on the diagram:



We now have an ‘invisible’ triangle, or the parts we need anyway. We have enough data to pass into our Atan function to get ourselves an angle. The code would look something like this:



```
var mouseX = 10;
var mouseY = -5;

var playerX = 0;
var playerY = 0;

var angle = Math.atan((playerY + mouseY) / (playerX + mouseX));
```

Using the code above, the variable *angle* is equal to atan(Opposite / Adjacent) because Opposite = playerY + mouseY (0 + -5) and Adjacent = playerX + mouseX (0 + 10).

The result will be 26.565 degrees. This is our angle, the angle at which our player must be rotated to be facing the mouse cursor.

Why you shouldn't use Atan
Okay, so if you got through the first section, you'd be right to think this title sounds silly, after all, atan helped us find our angle incredibly easily, in only a few lines of code, why shouldn't we use it?

Well, it's a bit more complicated than that. There are a few problems with Atan when it comes to games programming, the first, and quite large, problem is that when using Atan we could potentially be dividing by 0.

If our mouse cursor's X or Y position was the same as our player's, it would mean that either the Opposite or Adjacent value used in the equation would be equal to 0 + 0; This is a very bad, dividing by zero, due to the fact that it is impossible, will not work, and will, in most cases, cause the game or program to stop working.

Now if that isn't reason enough, there is another big problem with Atan and that is that it will only give you an angle within either the first or fourth quadrants of a coordinate system. This will sometimes result in the wrong angle being given back to you as atan cannot give you angle in the second or third quadrant, which is needed if we, in our example above, wanted the player to face downwards.

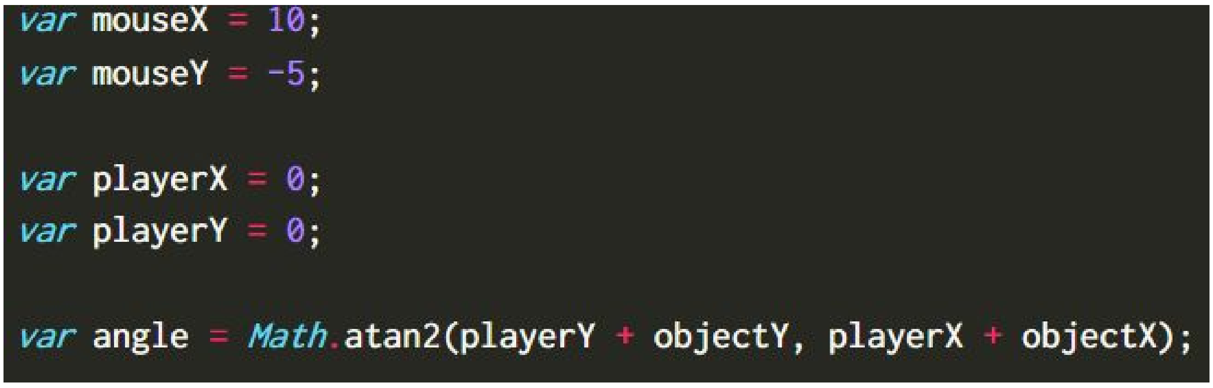
An alternative to Atan?
Now we've got the reasons out of the way, is there an alternative?

Yes. **Atan2**. Atan2 essentially gets you the same result as Atan but without all of those pesky issues Atan has as it was specifically brought about to combat those issues within computational maths.

Atan2, instead of taking the result of Opposite/Adjacent, takes 2 separate values. A Y and an X value. The Y being the Opposite side of our invisible triangle and the X being our Adjacent.

So instead of us doing the division like we did with Atan *atan(opposite/adjacent)*, we can use Atan2 like so *atan2(opposite, adjacent)*.

This is how it would look in code:



```
var mouseX = 10;
var mouseY = -5;

var playerX = 0;
var playerY = 0;

var angle = Math.atan2(playerY + objectY, playerX + objectX);
```

As you can see, it is very similar to the previous code snippet, as they are doing the same thing. We can see how we'd avoid the division by zero issue, as we are not dividing anything.

Now this is meant to be a brief overview of what the two functions are, and how they can be used to find an angle, but it is by no means comprehensive as I am not really qualified to be telling you how these functions work, just how to use them in context.

If you want to read more about how the functions work, check out some of the links below, they explain it better than I ever could.

https://gamedev.stackexchange.com/a/14603

https://stackoverflow.com/a/12011762

https://en.wikipedia.org/wiki/Atan2