Angles and units

In the vectors and forces sections, we carefully worked out an object-oriented structure to make something move on the screen, using the concept of a vector to represent location, velocity, and acceleration driven by forces in the environment. We could move straight from here into topics such as particle systems, steering forces, group behaviours, etc. If we did that, however, we'd skip an important area of mathematics that we're going to need: ***trigonometry***, or the mathematics of triangles, specifically right triangles.

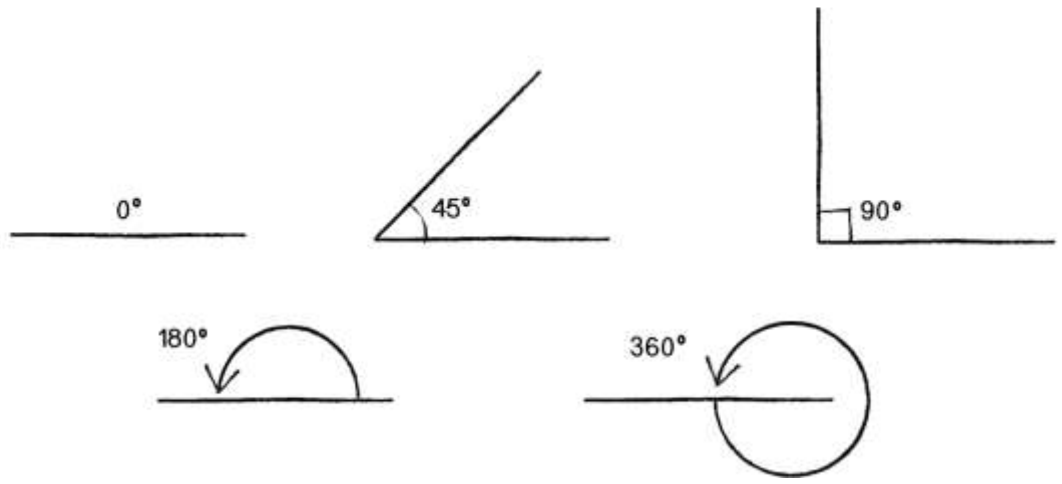
Trigonometry is going to give us a lot of tools. We'll get to think about angles and angular velocity and acceleration. Trig will teach us about the sine and cosine functions, which when used properly can yield an nice ease-in, ease-out wave pattern. It's going to allow us to calculate more complex forces in an environment that involves angles, such as a pendulum swinging or a box sliding down an incline.

So this section is a bit of a mishmash. We'll start with the basics of angles in ProcessingJS and cover many trigonometric topics, tying it all into forces at the end. And by taking this break now, we'll also pave the way for more advanced examples that require trig later in this course.

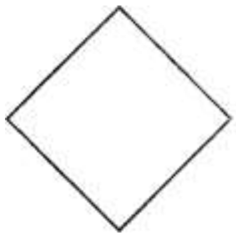
Angles

OK. Before we can do any of this stuff, we need to make sure we understand what it means to be an angle in ProcessingJS. If you have experience with ProcessingJS, you've undoubtedly encountered this issue while using the `rotate()` function to rotate and spin objects.

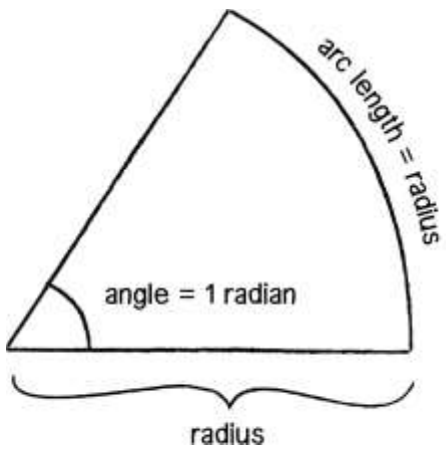
The first order of business is to cover ***radians*** and ***degrees***. You're probably most familiar with the concept of an angle in degrees. A full rotation goes from 0 to 360 degrees. 90 degrees (a right angle) is 1/4th of 360, shown below as two perpendicular lines.



It’s fairly intuitive for us to think of angles in terms of degrees. For example, the square in the diagram below is rotated 45 degrees around its centre.



But sometimes we may find it better to specify our angles in ***radians***. A radian is a unit of measurement for angles defined by the ratio of the length of the arc of a circle to the radius of that circle. One radian is the angle at which that ratio equals one (see the first diagram). 180 degrees = PI radians, 360 degrees = 2*PI radians, 90 degrees = PI/2 radians, etc.



The formula to convert from degrees to radians is:

radians = 2 * PI * (degrees / 360)$radians=2*PI*(degrees/360)$r, a, d, i, a, n, s, equals, 2, times, P, I, times, left parenthesis, d, e, g, r, e, e, s, slash, 360, right parenthesis

Thankfully, ProcessingJS makes it easy to decide which unit we want to use, radians or degrees, when using the functions that deal with angles, like `sin()`and `atan()`. In the Khan Academy environment, the default is degrees, but it can be changed to radians like so:

```
angleMode = "radians";
```

In addition, ProcessingJS also provides functions to make it easy to switch between the two units. The `radians()` function will automatically convert values from degrees to radians, and the constants PI and TWO_PI provide convenient access to these commonly used numbers (equivalent to 180 and 360 degrees, respectively). The following code, for example, will rotate shapes by 60 degrees.

```
angleMode = "radians";

var angle = radians(60);

rotate(angle);
```

If you are not familiar with how rotation is implemented in ProcessingJS, you could try this tutorial: Processing - Transform 2D, but note there are some differences between Processing and ProcessingJS.

<div class="callout">

What is PI?

The mathematical constant pi (or π) is a real number defined as the ratio of a circle’s circumference (the distance around the perimeter) to its diameter (a straight line that passes through the circle’s centre and has endpoints on its perimeter). It is equal to approximately 3.14159 and can be accessed in ProcessingJS with the built-in variable `PI`.

</div>