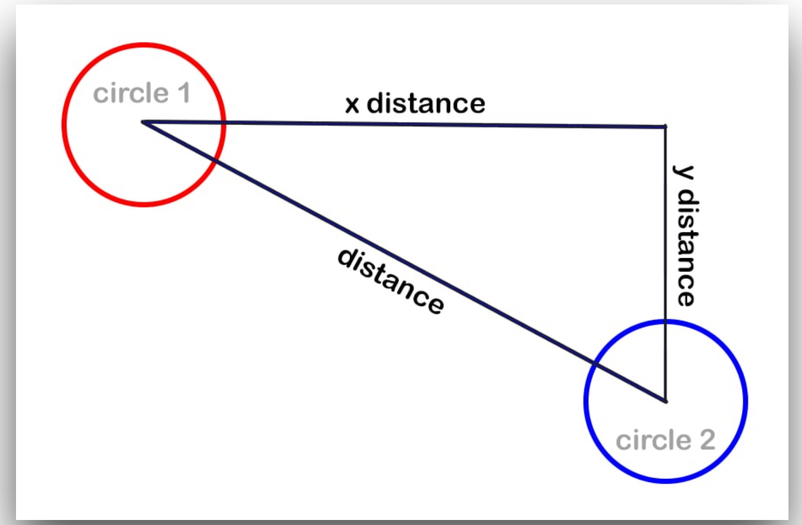
# Tutorial 2

Author : Kia Kalani
Contributors : Ali Lezzaik, Alireza Teimoori

## Introduction

In this tutorial the main focus would be on the idea behind two circles colliding with each other. By the end of tutorial you can expect to be able to detect collisions between two circles and in case of collision, change their color to a different color.



## Logic

The main components of the circles are their x position, y position and their diameter which are being used for drawing the circles. First of all, the most important thing to calculate is the distance between the x and y position of two circles. This is possible by subtracting the x position of the second circle from the first circle.

The same idea applies to the y position. By using Pythagorean theorem the distance between two x and y positions can be found by the formula: $z^2 = x^2 + y^2$

Now we have the distance between two circles in terms of x position and y position; however we are unaware of the value as a straight line distance. For calculating that, we would need to calculate for the value of z. Therefore finding the distance between the given two points can be achievable by the following formula: $|z| = \sqrt{x^2 + y^2}$
Keep in mind that the distance is always positive; therefore we are only paying attention to the positive possibility.

Now that we know how to calculate the distance shown above, we can find out if two circles have collided. We currently have the diameter of the circles. For finding out if the circles have collided we just need to see if the distance of two circles is less than or equal to the sum of the radius of two circles. Therefore in theory collision would take place if the following happens:
$z \leq (diameter1)/2 + (diameter2)/2$

## In code:

Let's start by setting up the index.html file as we did in the first tutorial. For further clarification for doing so, please refer to the first tutorial. Through our sketch.js file we would create a method called:

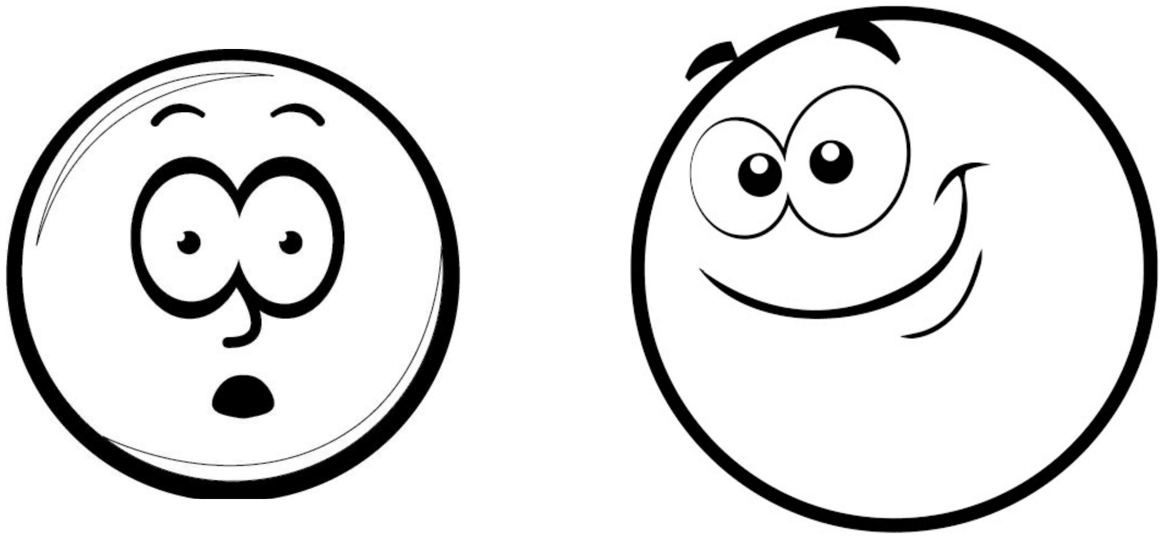
circleCollidedWithCircle(x: ,y: ,diameter: ,x2: ,y2: ,diameter2: )

, where x and y and diameter represent the components of the first circle and x2, y2, and diameter2 represent the components of the second circle. Let's start by deploying the logic step by step. First step is to find the distance between x and y positions. Therefore inside method we can do so by typing:

```
let distanceX = x - x2;
let distanceY = y - y2;
```

After finding the x and y distances, we need to find the distance as a straight line:

```
let totalDistance = Math.sqrt(Math.pow(distanceX, 2) + Math.pow(distanceY, 2));
```



Now that we have that we can finally check to see if two circles have collided:

```
if (totalDistance <= diameter/2 + diamtere2/2) {
    return true;
} else return false;
```

Now putting everything together the method for collision detection would look similar to the following:

```
function circleCollidedWithCircle(x, y, diameter, x2, y2, diameter2) {
    let distanceX = x - x2;
    let distanceY = y - y2;
    let totalDistance = Math.sqrt(Math.pow(distanceX, 2) +
Math.pow(distanceY, 2));
    if (totalDistance <= diameter / 2 + diamtere2 / 2) {
        return true;
    } else return false;
}
```

Now although this code is functional and would detect the collision, it is inefficient since doing the square root operation is a memory consuming operation.

The alternative for detecting circle collisions would be to avoid doing the square root operation. Instead keeping the totalDistance without square root and raise the power for the radius of the circles since doing a power operation is more efficient. The code would look something like so:

```
function circleCollidedWithCircle(x, y, diameter, x2, y2, diameter2) {
    let distanceX = x - x2;
    let distanceY = y - y2;
    let totalDistance = Math.pow(distanceX, 2) + Math.pow(distanceY, 2);
    if (totalDistance <= Math.pow(diameter / 2 + diamtere2 / 2, 2)) {
        return true;
    } else return false;
}
```

If the efficiency is key, the code can be also written as follows:

```
function circleCollidedWithCircle(x,y, diameter, x2,y2, diameter2) {
    return Math.pow(x-x2, 2)+ Math.pow(y-y2, 2) <= Math.pow(diameter/2 +
diameter2/2, 2);
}
```

Now that the code for collision detection is done, let's use it for changing the color of the circles if they collide. We first setup the sketch.js file like the first tutorial.

This time around, we would draw a circle at mouseX and mouseY position and draw a circle at x position 200 and y position 200.

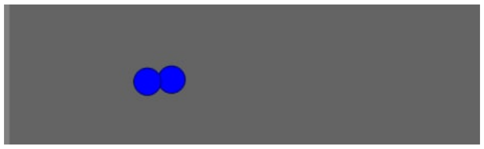
We define the diameter of the circles as 40. Now we can write the draw function with all of the necessary components.

In case there is a collision, we set the fill() to 'blue' so when circles collide they both turn blue. At the end we draw the circles.

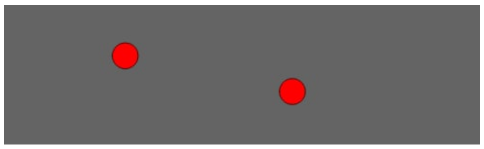
By putting everything together the final result should look something similar to this:

```
function circleCollidedWithCircle(x,y, diameter, x2,y2, diameter2) {
    return Math.pow(x-x2, 2)+ Math.pow(y-y2, 2) <= Math.pow(diameter/2 +
diameter2/2, 2);
}

const WIDTH = 1024;

const HEIGHT = 768;

let circleX = 200;

let circleY = 200;

let circleDiameter = 40;

function setup() {
    createCanvas(WIDTH, HEIGHT);
}

function draw() {
    background(100);
    if (circleCollidedWithCircle(mouseX, mouseY, circleDiameter, circleX,
circleY, circleDiameter)) {
        fill('blue');
    } else{
        fill('red');
    }
    circle(mouseX, mouseY, circleDiameter);
    circle(circleX, circleY, circleDiameter);
}
```

By opening the index.html you would notice that when the circle surrounding the mouse collides with the second circle they both turn blue:



Additionally when they stop colliding, the color goes back to red as it was before:



This sums up the second tutorial!
For getting access to the source code, please refer to: https://github.com/kiakalani/P5JSTutorial/tree/main/tutorial2/code