Circle Game

Basic JS
By Ricardo Lopez

Basic Needs

First I made the canvas in js and the background with CSS

```
var canvas = document.createElement("canvas");
var ctx = canvas.getContext("2d");
document.body.appendChild(canvas);
canvas.width = window.innerWidth;
canvas.height = window.innerHeight;
```

```
body {
  overflow: hidden;
}

canvas {
  position: absolute;
  top: 0;
  left: 0;
  z-index: 0;
  background-color:cornflowerblue
}
```

Add the coordinates for your 2 players

Make two arrays containing the x and y coordinates, the radius of the circle/players, and the color.

```
const thing = {
  x: 0,
  y: 0,
  r:50,
  color:"yellow"
};
const thing2 ={
  x:0,
  y:0,
  r:50,
  color:"orange"
}
```

Add WASD and Arrow keys variables for the 2 players.

Make 8 variables with **let** instead of **var**, name them left, right, up, down, W, A, S, and D. Make them all equal to false.

```
let left = false;
let right = false;
let up = false;
let down = false;

let a=false;
let d=false;
let w=false;
let s=false;
```

Continued

Then add this code that checks if the keys are pressed or not and to trigger the command. I don't really know how to explain this

```
window.addEventListener("keyup", function (event) {
  switch (event.code) {
    case "ArrowRight":
      right = false;
      break;
    case "ArrowLeft":
      left = false;
      break;
    case "ArrowUp":
      up = false;
      break;
    case "ArrowDown":
      down = false;
      break;
  }
});
```

```
window.addEventListener("keyup", function (event) {
  switch (event.code) {
    case "KeyA":
      d = false;
      break;
    case "KeyD":
      a = false;
      break;
    case "KeyW":
      w = false;
      break;
    case "KeyS":
      s = false;
      break;
  }
});
```

```
window.addEventListener("keydown", function (event) {
  switch (event.code) {
    case "ArrowRight":
      right = true;
      break;
    case "ArrowLeft":
      left = true;
      break;
    case "ArrowUp":
      up = true;
      break;
    case "ArrowDown":
      down = true;
      break;
  }
});
```

```
window.addEventListener("keydown", function (event) {
  switch (event.code) {
    case "KeyA":
      d = true;
      break;
    case "KeyD":
      a = true;
      break;
    case "KeyW":
      w = true;
      break;
    case "KeyS":
      s = true;
      break;
  }
});
```

Continued

Make a function titled cycle and add if statements to state: If the “up” is pressed player 1 will move up. Do it for all 8 keys.

```
function cycle() {
  if (left) {
    thing.x -= 5;
  }
  if (right) {
    thing.x += 5;
  }
  if (up) {
    thing.y -= 5;
  }
  if (down) {
    thing.y += 5;
  }
if (a){
  thing2.x +=5
}
```

```
}
if(d){
  thing2.x -=5
}
  if(w){
    thing2.y-=5
  }
  if(s){
    thing2.y+=5
  }
```

Animation Cycle

Do not close the function just yet. Add requestAnimationFrame(cycle) then close the function. After the closed curly bracket add another requestAnimationFrame(cycle) to finish it off.

```
  requestAnimationFrame(cycle);
}
requestAnimationFrame(cycle);
```

Now to draw the actual players.

Since you need to move the players we will draw the circle inside the function cycle. First add **ctx.clearRect(0,0,canvas.width,canvas.height)** to make the movement better. For player 1, add **translate(thing.x, thing.y)** and at the end of your drawing, **translate(-thing.x, -thing.y)** Between the two lines draw your players (Make it a circle). Repeat for player two, but make sure to change the variable to **thing2** because then you will be moving two of the same players.

(Picture on next slide)

```
ctx.translate(thing.x,thing.y)
ctx.fillStyle=thing.color
ctx.beginPath()
ctx.arc(100, 100, thing.r, 0, 2 * Math.PI);
ctx.fill()
ctx.beginPath()
ctx.fillStyle="black"
ctx.rect(62,70,25,15)
ctx.rect(87,75,25,5)
ctx.rect(112,70,25,15)
ctx.fill()
ctx.beginPath()
ctx.moveTo(70,110)
ctx.quadraticCurveTo(98, 134, 128, 110);
ctx.stroke();
ctx.translate(-thing.x,-thing.y)
```

```
ctx.translate(thing2.x,thing2.y)
ctx.fillStyle=thing2.color
ctx.beginPath()
ctx.arc(100, 300, thing2.r, 0, 2 * Math.PI)
ctx.fill()
ctx.beginPath()
ctx.fillStyle="black"
ctx.rect(62,270,25,15)
ctx.rect(87,275,25,5)
ctx.rect(112,270,25,15)
ctx.fill()
ctx.beginPath()
ctx.moveTo(70,310)
ctx.quadraticCurveTo(98, 334, 128, 310);
ctx.stroke();
ctx.translate(-thing2.x,-thing2.y)
```

I added extra details to the players

Add the star/small circles

First add an array containing the x and y coordinates, the radius and the color of the stars. Repeat for the amount of stars you want in your game. I added one star that is a different color to represent different amounts of points you can get.
REMEMBER TO ADD THE VARIABLES BEFORE FUNCTION CYCLE

```
// red stars
const star = {
  x:250,
  y:250,
  color:"red",
  r:15
}

const star2={
  x:300,
  y:40,
  color:"red",
  r:15
}
```

```
const star5={
  x:450,
  y:350,
  color:"red",
  r:15
}

const bluestar={
  x:450,
  y: 250,
  color:"blue",
  r:15
}
```

Draw the circles with the variables

Just plug in the coordinates of the circle with the star coordinates you set for them.

```
 ctx.beginPath()
ctx.fillStyle=star.color
ctx.arc(star.x, star.y, star.r, 0, 2 * Math.PI);
ctx.arc(star2.x,star2.y,star2.r,0,2*Math.PI)
ctx.fill()
ctx.beginPath()
ctx.arc(star3.x,star3.y,star3.r,0,2*Math.PI)
ctx.arc(star4.x,star4.y,star4.r,0,2*Math.PI)
ctx.fill()
ctx.beginPath()
ctx.arc(star5.x,star5.y,star5.r,0,2*Math.PI)
ctx.fill()
ctx.beginPath()
ctx.fillStyle=bluestar.color
ctx.arc(bluestar.x,bluestar.y,bluestar.r,0,2*Math.PI)
ctx.fill()
```

Collisions

Now that the stars are on the canvas, we have to make them teleport to a different position after you collide with the player. We will have to use the pythagorean theorem. We will use the center point of the 2 circles as the 2 main points and the third one will be determined when it collides. $a^2+b^2=c^2$. We have to define the x and y axis of the distance between the 2 circles. To do that we will have to make a variable with the following: **const dx = star.x - thing.x-100**. We are doing minus 100 at the end because my x coordinate for my first circle is at 100, and if we don't subtract the coordinate we set for the x or y axis, we will collide with the ball 100 coordinate further than we expect. Same thing for the y axis.

```
const dx = star.x - thing.x-100
const dy = star.y - thing.y-100
const xd = star2.x - thing.x-100
const yd = star2.y - thing.y-100
const u  = star3.x - thing.x-100
const y  = star3.y - thing.y-100
const p = star4.x - thing.x-100
const o =star4.y - thing.y-100
const tt=star5.x-thing.x-100
const rr=star5.y-thing.y-100
const bx=bluestar.x-thing.x-100
const by=bluestar.y-thing.y-100
```

```
const ex= star.x - thing2.x-100
const ey= star.y - thing2.y-300
const xe= star2.x - thing2.x-100
const ye= star2.y - thing2.y-300
const l= star3.x - thing2.x-100
const f= star3.y - thing2.y-300
const n=star4.x - thing2.x-100
const m=star4.y - thing2.y-300
const rt=star5.x-thing2.x-100
const tr=star5.y-thing2.y-300
const ux=bluestar.x-thing2.x-100
const uy=bluestar.y-thing2.y-300
```

Continued

Now that we have calculated the distance, it is time to use the pythagorean theorem. We will use an if statement with: **Math.Sqrt(a^2+b^2)** is less than **star.r + 50**. We add 50 because the radius of my player is 50, and I would need to collide with the ball 50+ to make it seem as the ball is touching each other instead of the radius. Then I want to make the ball teleport to a different location, so I used Math.random to randomly generate a coordinate for the x and y axis for my stars. The coordinate should be set less than or equal to the canvas length and width. For the first ball, I would do:

```
if(Math.sqrt(dx*dx+dy*dy)<star.r+50) {

star.x=canvas.width*Math.random()

    star.y=canvas.height*Math.random()

}
```

Repeat for the other stars

```
 //first ball collision
if(Math.sqrt(dx*dx+dy*dy)<star.r+55){
  star.x=canvas.width*Math.random()
  star.y=canvas.height*Math.random()
  score=score+1
}

 if(Math.sqrt(ex*ex+ey*ey)<star.r+55){
   star.x=canvas.width*Math.random()
   star.y=canvas.height*Math.random()
  score2=score2+1
 }
```

Score

Now that you have the circles colliding and teleporting, we can use the collisions again to add the score. Make the 2 variables for the scores. I did **var score=0** and **var score2**=0. Then make 2 functions for the 2 individual player scores. Draw in the text to the canvas and for the text, I put: **fillText("Player 1 Score"+ score,8,20)**. Remember to do the font and the font color. **REMEMBER TO ADD THE FUNCTION AND VARIABLE BEFORE FUNCTION CYCLE.** Then go to the end of the whole code and call your function name. In my case it will be **drawScore()** and **drawScore2.**

```
  drawScore();
  drawScore2();

  requestAnimationFrame(cycle);
}
requestAnimationFrame(cycle);
```

```
var score=0
var score2=0

function drawScore() {
    ctx.font = "16px Arial";
    ctx.fillStyle = "black";
    ctx.fillText("Player 1 Score: "+score, 8, 20);
}

function drawScore2(){
  ctx.font="16px Arial";
  ctx.fillStyle="black"
  ctx.fillText("Player 2 Score: "+score2,440,20)
}
```

Continued.

Add **score= score+1** for the player 1 collision code and **score2=score2+1** in player 2 collision code. If you want to have a star that increases by more than 1, simply change the +1 into the number you desire When a player touches the ball, the score should increase.

```
 //first ball collision
if(Math.sqrt(dx*dx+dy*dy)<star.r+55){
  star.x=700*Math.random()
  star.y=500*Math.random()
  score=score+1
}

 if(Math.sqrt(ex*ex+ey*ey)<star.r+55){
   star.x=700*Math.random()
   star.y=500*Math.random()
  score2=score2+1
 }
```

Your done

The result —---->

Thank you.