BÀI TẬP THỰC HÀNH 6

Họ và tên: Phạm Thành Lam

MSSV: 22520743

Lớp: IE104.P11.1

4.1 Tạo đối tượng mới

- Hàm dựng Shape()

```
// define Shape constructor
function Shape(x, y, velX, velY, exists) {
  this.x = x;
  this.y = y;
  this.velX = velX;
  this.velY = velY;
  this.exists = exists;
}
```

- Hàm dựng Ball()

```
// define Ball constructor
function Ball(x, y, velX, velY, exists, color, size) {
  Shape.call(this, x, y, velX, velY, exists);
  this.color = color;
  this.size = size;
}
```

- Thiết lập prototype và hàm dựng của Ball()

```
// define ball draw method
Ball.prototype = Object.create(Shape.prototype);
Ball.prototype.constructor = Ball;

Ball.prototype.draw = function () {
  ctx.beginPath();
  ctx.fillStyle = this.color;
  ctx.arc(this.x, this.y, this.size, 0, 2 * Math.PI);
  ctx.fill();
};

// define ball update method

Ball.prototype.update = function () {
  if ((this.x + this.size) >= width) {
    this.velX = -(this.velX);
  }

  if ((this.x - this.size) <= 0) {
    this.velX = -(this.velX);
  }

  if ((this.y + this.size) >= height) {
    this.velY = -(this.velY);
  }

  if ((this.y - this.size) <= 0) {
    this.velY = -(this.velY);
  }

  this.x += this.velX;
  this.y += this.velY;
};
```

- Phương thức collisionDetect() của prototype Ball

```
// define ball collision detection

Ball.prototype.collisionDetect = function () {
  for (let j = 0; j < balls.length; j++) {
    if (!(this === balls[j]) && balls[j].exists) {
      const dx = this.x - balls[j].x;
      const dy = this.y - balls[j].y;
      const distance = Math.sqrt(dx * dx + dy * dy);
      if (distance < this.size + balls[j].size) {
        balls[j].color = this.color =
          "rgb(" +
          random(0, 255) +
          "," +
          random(0, 255) +
          "," +
          random(0, 255) +
          ")";
      }
    }
  }
};
```

4.2. Định nghĩa "vòng tròn ma quỷ" EvilCircle():

```
// define EvilCircle constructor
function EvilCircle(x, y, exists) {
  Shape.call(this, x, y, 20, 20, exists);
  this.color = 'white';
  this.size = 10;
}
```

4.3. Định nghĩa các phương thức của EvilCircle():

```
// define EvilCircle draw method
EvilCircle.prototype = Object.create(Shape.prototype);
EvilCircle.prototype.constructor = EvilCircle;

EvilCircle.prototype.draw = function () {
  ctx.beginPath();
  ctx.lineWidth = 3;
  ctx.strokeStyle = this.color;
  ctx.arc(this.x, this.y, this.size, 0, 2 * Math.PI);
  ctx.stroke();
}

// define EvilCircle checkBounds method
EvilCircle.prototype.checkBounds = function () {
  if ((this.x + this.size) >= width) {
    this.x -= this.size;
  }

  if ((this.x - this.size) <= 0) {
    this.x += this.size;
  }

  if ((this.y + this.size) >= height) {
    this.y -= this.size;
  }

  if ((this.y - this.size) <= 0) {
    this.y += this.size;
  }
};
```

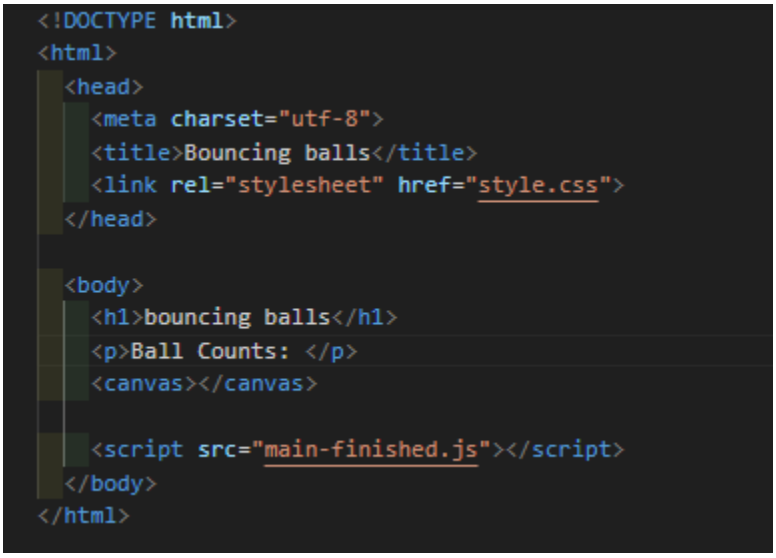
```
// define EvilCircle setControls method
EvilCircle.prototype.setControls = function () {
  let _this = this;
  window.onkeydown = function (e) {
    if (e.key === 'a') {
      _this.x -= _this.velX;
    } else if (e.key === 'd') {
      _this.x += _this.velX;
    } else if (e.key === 'w') {
      _this.y -= _this.velY;
    } else if (e.key === 's') {
      _this.y += _this.velY;
    }
  };
};

// define EvilCircle collision detection
EvilCircle.prototype.collisionDetect = function () {
  for (let j = 0; j < balls.length; j++) {
    if (balls[j].exists) {
      const dx = this.x - balls[j].x;
      const dy = this.y - balls[j].y;
      const distance = Math.sqrt(dx * dx + dy * dy);
      if (distance < this.size + balls[j].size) {
        balls[j].exists = false;
        count--;
        countText.textContent = 'Ball count: ' + count;
      }
    }
  }
};
```

4.4. Mang “vòng tròn ma quỷ” vào chương trình:

```
// create EvilCircle instance outside the loop
let evilCircle = new EvilCircle(50, 50, true);
evilCircle.setControls();

// define loop that keeps drawing the scene constantly
function loop() {
  ctx.fillStyle = 'rgba(0,0,0,0.25)';
  ctx.fillRect(0, 0, width, height);

  evilCircle.draw();
  evilCircle.checkBounds();
  evilCircle.collisionDetect();

  for (let i = 0; i < balls.length; i++) {
    if (balls[i].exists) {
      balls[i].draw();
      balls[i].update();
      balls[i].collisionDetect();
    }
  }

  requestAnimationFrame(loop);
}
```

4.5. Hiện thực chức năng đếm điểm:

- Trong nội dung tập tin HTML, thêm phần tử <p> ngay bên dưới phần tử <h1> chứa nội dung “Ball count: ”.



```html
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Bouncing balls</title>
    <link rel="stylesheet" href="style.css">
  </head>

  <body>
    <h1>bouncing balls</h1>
    <p>Ball Counts: </p>
    <canvas></canvas>

    <script src="main-finished.js"></script>
  </body>
</html>
```

- Trong nội dung CSS, thêm luật (nguyên tắc) bên dưới:

```css
html,
body {
  margin: 0;
}

html {
  font-family: 'Helvetica Neue', Helvetica, Arial, sans-serif;
  height: 100%;
}

body {
  overflow: hidden;
  height: inherit;
}

h1 {
  font-size: 2rem;
  letter-spacing: -1px;
  position: absolute;
  margin: 0;
  top: -4px;
  right: 5px;

  color: transparent;
  text-shadow: 0 0 4px white;
}

p {
  position: absolute;
  margin: 0;
  top: 35px;
  right: 5px;
  color: #aaa;
}
```

- Trong nội dung tập tin JS, cập nhật các thay đổi sau:

```
// define array to store balls and populate it

let balls = [];

while (balls.length < 25) {
  const size = random(10, 20);
  let ball = new Ball(
    // ball position always drawn at least one ball width
    // away from the edge of the canvas, to avoid drawing errors

    random(0 + size, width - size),
    random(0 + size, height - size),
    random(-7, 7),
    random(-7, 7),
    true,
    'rgb(' + random(0, 255) + ',' + random(0, 255) + ',' + random(0, 255) + ')',
    size
  );
  balls.push(ball);
}

let count = balls.length;
let countText = document.querySelector('p');
countText.textContent = 'Ball count: ' + count;

// create EvilCircle instance outside the loop
let evilCircle = new EvilCircle(50, 50, true);
evilCircle.setControls();

// define loop that keeps drawing the scene constantly
function loop() {
  ctx.fillStyle = '□rgba(0,0,0,0.25)';
  ctx.fillRect(0, 0, width, height);

  evilCircle.draw();
  evilCircle.checkBounds();
  evilCircle.collisionDetect();

  for (let i = 0; i < balls.length; i++) {
    if (balls[i].exists) {
      balls[i].draw();
      balls[i].update();
      balls[i].collisionDetect();
    }
  }

  requestAnimationFrame(loop);
}

loop();
```