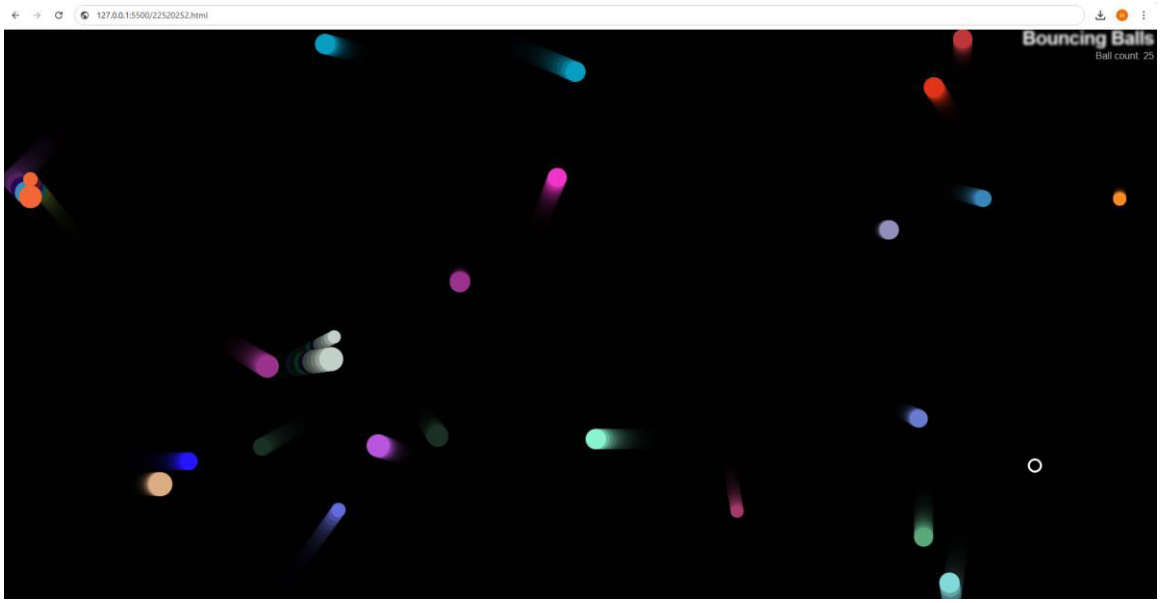
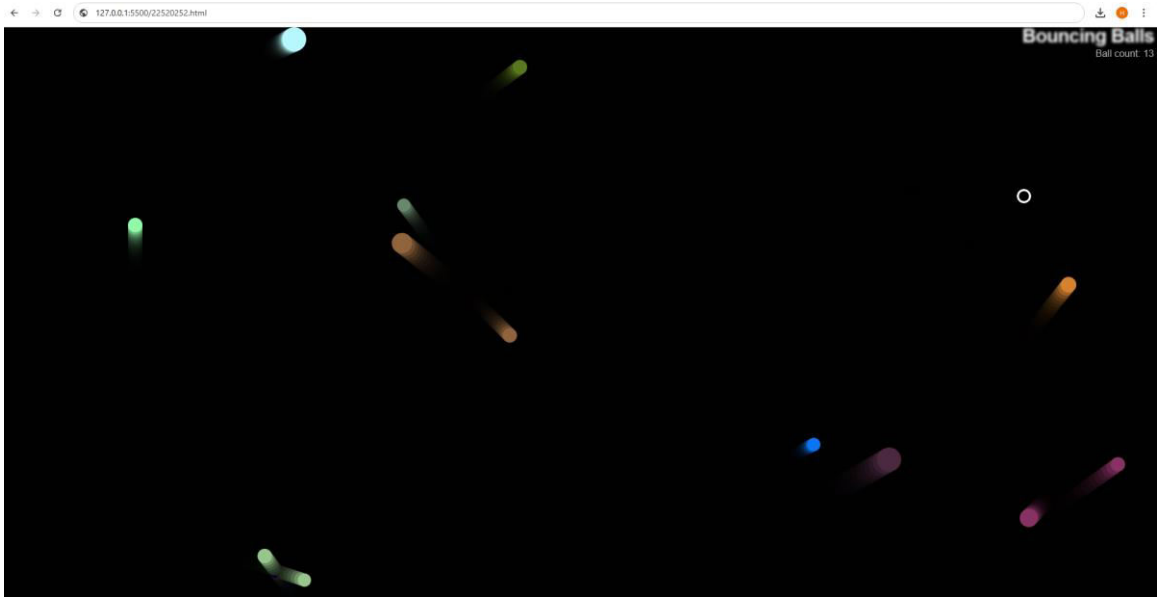
Kết quả





4.1. Tạo đối tượng mới:

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

// Class Ball
class Ball extends Shape {
  constructor(x, y, velX, velY, exists, color, size) {
    super(x, y, velX, velY, exists);
    this.color = color;
    this.size = size;
  }
```

Cập nhật phương thức collision

```
  collisionDetect() {
    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, 4.3 Định nghĩa EvilCircle() + Các phương thức của EvilCircle()

```
class EvilCircle extends Shape {
  constructor(x, y, exists) {
    super(x, y, 20, 20, exists);  // velX và velY luôn = 20
    this.color = "white";
    this.size = 10;
  }

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

  checkBounds() {
    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;
    }
  }

  setControls() {
    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;
      }
    };
  }

  collisionDetect() {
    for (let j = 0; j < balls.length; j++) {
      if (balls[j].exists) {
```

```
  setControls() {
    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;
      }
    };
  }
// 22520252
  collisionDetect() {
    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;
          // Update ball count
          const remainingBalls = balls.filter(ball => ball.exists).length;
          ballCountDisplay.textContent = `Ball count: ${remainingBalls}`;
        }
      }
    }
  }
}
```

4.4 Mang “vòng tròn ma quỷ” vào chương trình

```
// 22520252
const balls = [];
const ballCountDisplay = document.querySelector("p"); // Reference to <p>
while (balls.length < 25) {
  const size = random(10, 20);
  const ball = new Ball(
    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);
}
ballCountDisplay.textContent = `Ball count: ${balls.length}`; // Initialize ball count

// EvilCircle setup
const evilCircle = new EvilCircle(random(0, width), random(0, height), true);
evilCircle.setControls();

// Animation loop
function loop() {
  ctx.fillStyle = "rgba(0, 0, 0, 0.25)";
  ctx.fillRect(0, 0, width, height);

  for (let i = 0; i < balls.length; i++) {
    if (balls[i].exists) {
      balls[i].draw();
      balls[i].update();
      balls[i].collisionDetect();
    }
  }
  // 22520252
  evilCircle.draw();
  evilCircle.checkBounds();
  evilCircle.collisionDetect();

  requestAnimationFrame(loop);
}
loop();
```

4.5 Hiện thực chức năng đếm điểm

```
    <body>
      <h1>Bouncing Balls</h1>
      <p>Ball count:</p>
      <canvas></canvas>
      <script src="main-finished-es6.js"></script>

}
ballCountDisplay.textContent = `Ball count: ${balls.length}`; // Initialize ball count
```

Source code:

```
// 22520252
const canvas = document.querySelector("canvas");
const ctx = canvas.getContext("2d");

const width = (canvas.width = window.innerWidth);
const height = (canvas.height = window.innerHeight);

// Function gen random num
function random(min, max) {
  return Math.floor(Math.random() * (max - min + 1)) + min;
}

// Class Shape
class Shape {
  constructor(x, y, velX, velY, exists) {
    this.x = x;
    this.y = y;
    this.velX = velX;
    this.velY = velY;
    this.exists = exists;
  }
}

// Class Ball
class Ball extends Shape {
  constructor(x, y, velX, velY, exists, color, size) {
    super(x, y, velX, velY, exists);
    this.color = color;
    this.size = size;
  }

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

```

```
// Class Ball
class Ball extends Shape {
  constructor(x, y, velX, velY, exists, color, size) {
    super(x, y, velX, velY, exists);
    this.color = color;
    this.size = size;
  }

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

  update() {
    if (this.x + this.size >= width || this.x - this.size <= 0) {
      this.velX = -this.velX;
    }
    if (this.y + this.size >= height || this.y - this.size <= 0) {
      this.velY = -this.velY;
    }
    this.x += this.velX;
    this.y += this.velY;
  }
  // 22520252
  collisionDetect() {
    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) +
```

```
// 22520252
class EvilCircle extends Shape {
  constructor(x, y, exists) {
    super(x, y, 20, 20, exists);
    this.color = "white";
    this.size = 10;
  }

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

  checkBounds() {
    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;
    }
  }

  setControls() {
    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
```

```
  collisionDetect() {
    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;
          // Update ball count
          const remainingBalls = balls.filter((ball) => ball.exists).length;
          ballCountDisplay.textContent = `Ball count: ${remainingBalls}`;
        }
      }
    }
  }
}

// 22520252
const balls = [];
const ballCountDisplay = document.querySelector("p"); // Reference to <p>
while (balls.length < 25) {
  const size = random(10, 20);
  const ball = new Ball(
    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);
}
ballCountDisplay.textContent = `Ball count: ${balls.length}`; // Initialize ball count

// EvilCircle setup
const evilCircle = new EvilCircle(random(0, width), random(0, height), true);
evilCircle.setControls();

// Animation loop
function loop() {
```

```
// Animation loop
function loop() {
  ctx.fillStyle = "rgba(0, 0, 0, 0.25)";
  ctx.fillRect(0, 0, width, height);

  for (let i = 0; i < balls.length; i++) {
    if (balls[i].exists) {
      balls[i].draw();
      balls[i].update();
      balls[i].collisionDetect();
    }
  }
  // 22520252
  evilCircle.draw();
  evilCircle.checkBounds();
  evilCircle.collisionDetect();

  requestAnimationFrame(loop);
}
loop();
```