CSE3002 - Internet and web programming

Internet and web programming theory digital assessment – 1

Name: jeevaa

faculty : Lydia jane G

Reg no: 20BCE2414

slot: C1

Date: 09-11-2022

Question:

1) Create a small game using HTML and Javascript

CODE:

INDEX1.HTML:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
    <link rel="stylesheet" href="index1.css">
</head>
<body>
    <canvas id="canvas"></canvas>
<div class="minScore">
    <div>
        <span>Score</span>
        <span id="minScore">0</span>
    </div>
    <div>
        <span>High Score</span>
        <span id="highScore">0</span>
    </div>
</div>
<div class="name">
    <div>
        <span>jeevaa</span>
    </div>
    <div>
```

```
        <span>20BCE2414</span>
    </div>
</div>
<div class="bigScore shown">
    <h1 id="bigScore">0</h1>
    <p>Points:</p>
    <button>Play</button>
    <script src="index1.js"></script>

</div>
</body>
</html>
```

INDEX1.CSS:

```
@import
url("https://fonts.googleapis.com/css2?family=Montserrat:ital,wght@0,400;0,500
;0,600;0,700;0,800;0,900;1,400;1,500;1,600;1,700;1,800;1,900&display=swap");

html {
    scroll-behavior: smooth;
}

body {
    margin: 0;
    padding: 0;
    box-sizing: border-box;
    font-family: "Montserrat", "Gill Sans", "Gill Sans MT", Calibri,
        "Trebuchet MS", sans-serif;
    background: #000;
}

canvas {
    position: absolute;
    top: 0;
    left: 0;
}

.minScore {
    position: relative;
    user-select: none;
}
.name {
    position: relative;
    user-select: none;
}
div span {
    color: white;
```

```css
}

.bigScore {
    user-select: none;
    position: fixed;
    top: 50%;
    left: 50%;
    width: 50%;
    height: 250px;
    border-radius:50px;
    border: 5px solid #ccc;
    background: white;
    font-size: 2em;
    text-align: center;
    transform: translate(-50%, -50%) scale(0);
    z-index: -1000;
    transition: 0.5s ease-in-out;
    opacity: 0;
}

.bigScore.shown {
    opacity: 1;
    transform: translate(-50%, -50%) scale(1);
    z-index: 1000;
}

.bigScore h1 {
    margin: 0%;
}

.bigScore button {
    width: 70%;
    height: 45px;
    border-radius: 45px;
    outline: none;
    border: none;
    background: aqua;
    font-size: 30px;
    font-weight: 900;
    /* font-style: italic; */
    transition: 0.25s;
}

.bigScore button:hover {
    background: dodgerblue;
}
```

Index1.js :

```
document.getElementById("canvas");
const ctx = canvas.getContext("2d");
canvas.width = window.innerWidth;
canvas.height = window.innerHeight;
if (!localStorage.getItem("maxScore")) {
    localStorage.setItem("maxScore", 0);
}
let score = 0;
let maxScore;
const minScore = document.getElementById("minScore"),
    bigScore = document.getElementById("bigScore"),
    button = document.querySelector("button"),
    highScore = document.getElementById("highScore");
let projectiles = [];
let enemies = [];
let particles = [];
class Player {
    constructor(x, y, radius, color) {
        this.x = x;
        this.y = y;
        this.radius = radius;
        this.color = color;
    }
    draw() {
        ctx.beginPath();
        ctx.arc(this.x, this.y, this.radius, 0, Math.PI * 2, false);
        ctx.fillStyle = this.color;
        ctx.fill();
    }
}
class Projectile {
    constructor(x, y, radius, color, velocity) {
        this.x = x;
        this.y = y;
        this.radius = radius;
        this.color = color;
        this.velocity = velocity;
    }
    draw() {
        ctx.beginPath();
        ctx.arc(this.x, this.y, this.radius, 0, Math.PI * 2, false);
        ctx.fillStyle = this.color;
        ctx.fill();
    }
    update() {
        this.x += this.velocity.x;
        this.y += this.velocity.y;
```

```
        this.draw();
    }
}
class Enemy {
    constructor(x, y, radius, color, velocity) {
        this.x = x;
        this.y = y;
        this.radius = radius;
        this.color = color;
        this.velocity = velocity;
    }
    draw() {
        ctx.beginPath();
        ctx.arc(this.x, this.y, this.radius, 0, Math.PI * 2, false);
        ctx.fillStyle = this.color;
        ctx.fill();
    }
    update() {
        this.x += this.velocity.x;
        this.y += this.velocity.y;
        this.draw();
    }
}
class Particle {
    constructor(x, y, radius, color, velocity, alpha) {
        this.x = x;
        this.y = y;
        this.radius = radius;
        this.color = color;
        this.velocity = velocity;
        this.alpha = 1;
    }
    draw() {
        ctx.save();
        ctx.globalAlpha = this.alpha;
        ctx.beginPath();
        ctx.arc(this.x, this.y, this.radius, 0, Math.PI * 2, false);
        ctx.fillStyle = this.color;
        ctx.fill();
        ctx.restore();
    }
    update() {
        this.x += this.velocity.x;
        this.y += this.velocity.y;
        this.alpha -= 0.01;
        this.draw();
    }
}
```

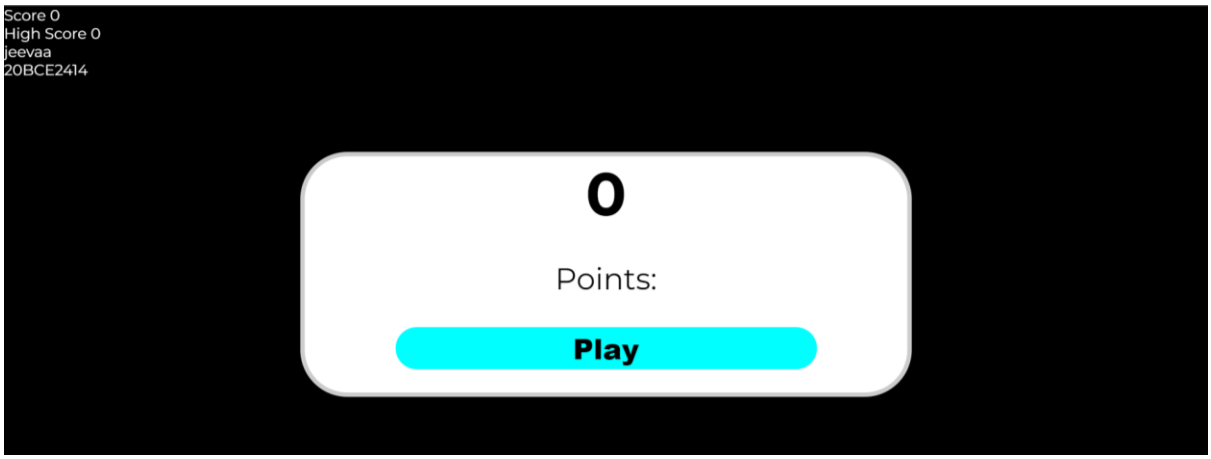
```
function spawnEnemy() {
    const radius = Math.random() * 24 + 6;
    let x, y;
    if (Math.random() < 0.5) {
        x = Math.random() < 0.5 ? 0 - radius : canvas.width + radius;
        y = Math.random() * canvas.height;
    } else {
        x = Math.random() * canvas.width;
        y = Math.random() < 0.5 ? 0 - radius : canvas.height + radius;
    }
    const color = `hsl(${Math.floor(Math.random() * 360)}, 50%, 50%)`;
    const angle = Math.atan2(canvas.height / 2 - y, canvas.width / 2 - x);

    const velocity = {
        x: Math.cos(angle),
        y: Math.sin(angle)
    };
    enemies.push(new Enemy(x, y, radius, color, velocity));
}

let player = new Player(canvas.width / 2, canvas.height / 2, 15, "white");
player.draw();
let gameSpeed = 0;
let spawnTimer = 60;
let animateId;
function animate() {
    spawnTimer--;
    if (spawnTimer < 0) {
        spawnEnemy();
        gameSpeed += 1;
        console.log(gameSpeed);
        if (gameSpeed >= 30) {
            gameSpeed = 30;
        }
        spawnTimer = 60 - gameSpeed;
    }
    animateId = requestAnimationFrame(animate);
    ctx.fillRect(0, 0, canvas.width, canvas.height);
    player.draw();
    projectiles.forEach((projectile, index) => {
        projectile.update();
        if (
            projectile.x - projectile.radius < 0 ||
            projectile.x + projectile.radius > canvas.width ||
            projectile.y - projectile.radius < 0 ||
            projectile.y - projectile.radius > canvas.height
        ) {
            projectiles.splice(index, 1);
```

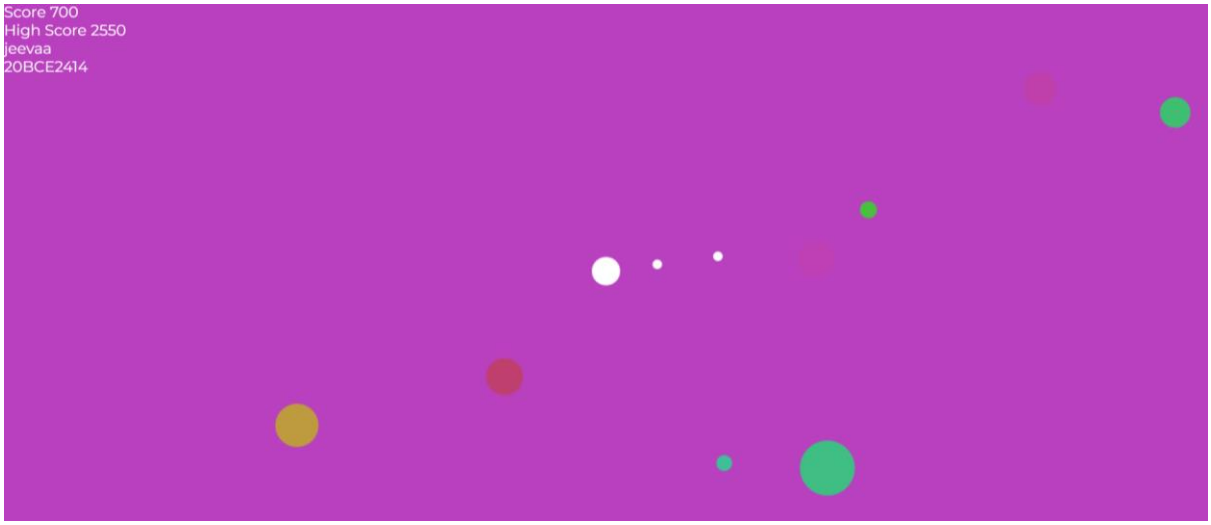
```
        }
    });
    particles.forEach((particle, index) => {
        if (particle.alpha < 0) {
            particles.splice(index, 1);
        } else {
            particle.update();
        }
    });
    enemies.forEach((enemy, index) => {
        enemy.update();
        let distance = Math.hypot(player.x - enemy.x, player.y - enemy.y);
        if (distance - player.radius - enemy.radius < 1) {
            bigScore.parentNode.classList.toggle("shown");
            bigScore.textContent = score;
            maxScore =
                localStorage.getItem("maxScore") != "0"
                    ? localStorage.getItem("maxScore")
                    : 0;
            highScore.textContent = maxScore;
            if (maxScore < score) {
                maxScore = score;
                localStorage.setItem("maxScore", maxScore);
            }
            cancelAnimationFrame(animateId);
        }
        projectiles.forEach((projectile, projectileIndex) => {
            let distance = Math.hypot(
                projectile.x - enemy.x,
                projectile.y - enemy.y
            );
            if (distance - enemy.radius - projectile.radius < 1) {
                for (let i = 0; i < enemy.radius * 2; i++) {
                    particles.push(
                        new Particle(
                            projectile.x,
                            projectile.y,
                            Math.random() * 2,
                            enemy.color,
                            {
                                x:
                                    (Math.random() * 1 - 0.5) *
                                    (Math.random() * 5),
                                y:
                                    (Math.random() * 1 - 0.5) *
                                    (Math.random() * 5)
                            }
                        )
```

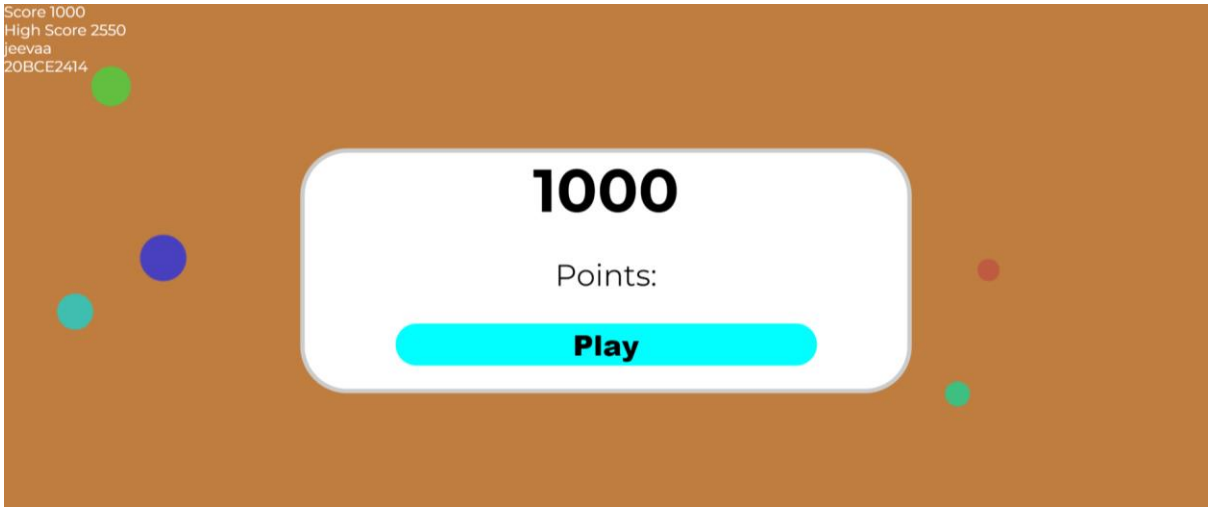
```
                    );
                }
                if (enemy.radius - 10 > 10) {
                    score += 250;
                    minScore.textContent = score;

                    enemy.radius -= 10;
                    projectiles.splice(projectileIndex, 1);
                } else {
                    score += 100;
                    minScore.textContent = score;
                    enemies.splice(index, 1);
                    projectiles.splice(projectileIndex, 1);
                }
            }
        });
    });
}
window.addEventListener("click", (e) => {
    const angle = Math.atan2(
        e.clientY - canvas.height / 2,
        e.clientX - canvas.width / 2
    );
    const velocity = {
        x: Math.cos(angle) * 5,
        y: Math.sin(angle) * 5
    };
    projectiles.push(
        new Projectile(
            canvas.width / 2,
            canvas.height / 2,
            5,
            "white",
            velocity
        )
    );
});
button.addEventListener("click", () => {
    button.blur();
    canvas.width = window.innerWidth;
    canvas.height = window.innerHeight;
    bigScore.parentNode.classList.toggle("shown");
    particles = [];
    enemies = [];
    projectiles = [];
    score = 0;
    gameSpeed = 0;
    player.x = canvas.width / 2;
```

```
    player.y = canvas.height / 2;
    minScore.textContent = score;
    maxScore =
        localStorage.getItem("maxScore") != "0"
            ? localStorage.getItem("maxScore")
            : 0;
    highScore.textContent = maxScore;
    animate();
});
```

OUTPUT:






Score 1000
High Score 2550
jeevaa
20BCE2414
1000
Points:
Play