|  |  |
| --- | --- |
| 과목명 | 웹프로그래밍 |
| 담당교수 | 천인국 |
| 학과 | 컴퓨터공학과 |
| 학년 | 2 |
| 학번 | 20204101 |
| 이름 | 신동욱 |
| 제출일 | 11.12 |

로고, 등록 상표, 폰트, 상징이(가) 표시된 사진

자동 생성된 설명

**웹프로그래밍**

**벽돌깨기게임 기능 추가하기**

1. 벽돌게임 추가한 기능 설명
2. 소스코드의 설명
3. 전체 소스코드
4. **벽돌게임 및 추가한 기능 설명**



**공**: 공은 일정한 속도로 움직이며 벽이나 벽돌에 충돌하면 이동 방향을 바꾼다

텍스트, 스크린샷, 폰트, 화이트이(가) 표시된 사진

자동 생성된 설명텍스트, 스크린샷, 폰트, 화이트이(가) 표시된 사진

자동 생성된 설명그래픽, 폰트, 원, 일러스트레이션이(가) 표시된 사진

자동 생성된 설명

**Restart 버튼**: 게임이 종료되면, 이 버튼을 클릭하여 다시 시작한다.

**점수와 생명**: 사용자에게 현재 점수, 생명을 표시한다.

**운석**: 운석은 위에서 아래로 떨어진다. 운석이 패들과 충돌하면 생명이 깎인다.  
운석이 바닥과 충돌하면 랜덤한 x 위치에서 다시 떨어진다.

**패들**: 패들은 사용자의 마우스를 따라 움직인다. 공이 패들에 닿으면 반사되고, 공이 패들에 닿지 않고 바닥과 충돌하면 생명이 깎인다.

**벽돌**: 왼쪽부터 경도 3, 2, 1으로 공이 벽돌에 충돌하면 벽돌의 경도가 1씩 줄어든다. 경도가 0이 된다면 부숴진다. 벽돌이 부숴지면 점수가 100 오른다.

텍스트, 스크린샷, 디스플레이, 직사각형이(가) 표시된 사진

자동 생성된 설명텍스트, 스크린샷, 폰트, 디스플레이이(가) 표시된 사진

자동 생성된 설명텍스트, 스크린샷, 폰트, 그래픽이(가) 표시된 사진

자동 생성된 설명

**게임 전체 화면**

게임이 종료되면 공, 운석, 패들을 정지시킨다

벽돌을 다 부수면 You Win! 텍스트를 출력하고   
생명이 0이 되면 You Lose.. 텍스트를 출력한다

1. **소스코드의 설명**

<!DOCTYPE html>

<html lang="ko">

<head>

    <style>

        canvas {

캔버스 스타일  
색과 테두리를 지정  
레이아웃을 block으로 지정

            background: rgb(243, 245, 117);

            border: 1px solid black;

            display: block;

        }

        .numberbox {

            display: flex;

numbox 클래스의 스타일 지정  
디스플레이를 flex로 지정하고 수평으로 넓게 펼친다  
패딩, 마진 지정  
크기, 테두리 지정

            justify-content: space-between;

            background-color: #ccc;

            padding: 10px;

            margin: 10px;

            width: 550px;

            border: 2px solid rgb(138, 138, 138);

        }

        .numberbox h1 {

마진 제거

            margin: 0;

        }

        button {

버튼 스타일 지정  
색, 테두리 지정  
패딩 지정  
커서를 포인터로 설정

            background-color: #007bff;

            color: #fff;

            border: none;

            padding: 10px 20px;

            cursor: pointer;

        }

        button:hover {

배경색 설정

            background-color: #0056b3;

        }

        .gameover {

gameover 클래스 스타일 지정  
위치를 절대 위치로 지정 및 설정  
폰트 사이즈 설정

            position: absolute;

            top: 100px;

            left: 160px;

            font-size: 2em;

        }

    </style>

</head>

<body>

    <canvas id="myCanvas" width="600" height="400"></canvas>

    <script>

        // 캔버스 객체를 찾아서 변수 canvas에 저장한다.

        let canvas = document.getElementById("myCanvas");

        // 컨텍스트 객체를 얻는다.

        let context = canvas.getContext("2d");

        let brickRows = 3; // 벽돌의 총 행 수

        let brickColumns = 5; // 벽돌의 총 열 수

        let brickWidth = 75; // 벽돌 1개의 폭

        let brickHeight = 20; // 벽돌 1개의 높이

        let brickPadding = 10; // 벽돌 사이의 간격

        let brickStartX = 30; // 화면에서 벽돌이 시작되는 위치

        let brickStartY = 30; // 화면에서 벽돌이 시작되는 위치

        let lives = 3; //생명

        let score = 0; //점수

        let iswin = brickColumns \* brickRows; //이길 조건

        let bricks = new Array(brickRows); // 크기가 brickRow인 배열을 생성한다.

자바스크립트에서 사용할 변수들을 선언한다.

        let ball = { // 공을 리터럴 객체로 정의한다.

            x: canvas.width / 2, // 공의 x좌표

            y: canvas.height - 30, // 공의 y좌표

            dx: 2, // 공의 x 방향 속도

            dy: -2, // 공의 y 방향 속도

            radius: 10, // 공의 반지름

            draw: function () { // 공을 화면에 그리는 메소드

                context.beginPath();

                context.arc(this.x, this.y, this.radius, 0, Math.PI \* 2);

                context.fillStyle = "#000000";

                context.fill();

                context.closePath();

            }

        };

공을 리터럴 객체로 정의한다. 공의 좌표와 속도를 지정하고 공을 그리는 draw 함수를 정의한다

        let paddle = { // 패들을 객체로 정의한다.

            height: 10, // 패들의 높이

            width: 90, // 패들의 폭

            x: 300, // 패들의 x좌표

            draw: function () { // 패들을 화면에 그리는 함수

                context.beginPath();

                context.rect(this.x, canvas.height - this.height, this.width, this.height);

                context.fillStyle = "#2f92fb";

                context.fill();

                context.closePath();

            } };

패들을 리터럴 객체로 정의한다. 패들의 너비와 높이, 위치를 지정한다  
패들을 그리는 draw 함수를 정의한다

        let meteorimg = new Image();

        meteorimg.src = 'meteor.png';

        let meteor = {

            x: 50,

            y: 50,

            dy: 1,

            draw: function() {

                context.drawImage(meteorimg, this.x, this.y, meteorimg.width, meteorimg.height);

            }

        }

운석으로 사용할 이미지 객체를 생성한다.  
운석을 리터럴 객체를 정의한다. 운석의 위치와 속도를 지정한다.  
운석을 그리는 draw 함수를 정의한다. 이 때 이미지 개체를 인수로 전달한다.

        class Brick { // 벽돌을 클래스로 정의한다.

            constructor(x, y, status) { // 생성자 함수

                this.x = x;

                this.y = y;

                this.status = status;

                this.strength = Math.floor(Math.random() \* 3) + 1; //벽돌 강도 랜덤

            }

벽돌을 class로 객체를 정의한다.  
생성자를 정의한다. 벽돌의 위치, 상태(1이면 표시, 0이면 안보임), 강도(1~3 정수 랜덤)

            draw() { // 벽돌의 화면에 그리는 함수

                if (this.status == 1) {// 아직도 살아 있는 벽돌이면

                    context.beginPath();

                    context.rect(this.x, this.y, brickWidth, brickHeight);

                    switch(this.strength) {

                    case 1:

                        context.fillStyle = "#A9A9A9";

                        break;

                    case 2:

                        context.fillStyle = "#808080";

                        break;

                    case 3:

                        context.fillStyle = "#505050";

                        break;

                    }

                    context.fill();

                    context.closePath();

                } }

벽돌을 그리는 draw 함수를 정의한다  
만약 살아있는 벽돌이라면(if status == 1) 컨텍스트 객체를 이용하여 화면에 그린다  
이 때 랜덤으로 설정된 강도에 따라 그리는 색을 달리 한다

            check() { // 공과 벽돌의 충돌 여부를 검사하는 함수

                if (this.status == 1) { // 아직도 살아 있는 벽돌이면

                    if (ball.x > this.x && ball.x < this.x + brickWidth // 충돌 조건

                        && ball.y > this.y && ball.y < this.y + brickHeight) {

                        ball.dy = -ball.dy; // 볼의 방향을 반전한다.

                        this.strength--; // 강도를 1 감소시킨다

                        if (this.strength == 0) { // 강도가 0이면

                            score += 100; // 벽돌 부수면 점수+

                            this.status = 0; // 벽돌을 보이지 않게 한다.

                        }

                        let sc = document.getElementById('score');

                        sc.textContent = score;

                    }

                }

            }

        }

공과 벽돌의 충돌 여부를 검사하는 check 함수를 정의한다  
이 함수가 호출되고, 아직 살아있는 벽돌이면(if status == 1)  
공과 현재 자기 자신 객체(벽돌)의 좌표를 비교하여 충돌을 검사한다   
만약 위의 조건에서 참이라면 공의 진행 방향을 바꾸고 강도를 1 감소시킨다  
만약 강도가 0이 된다면 점수를 100 더하고 벽돌을 안보이게 한다(status = 0)  
HTML에서 id가 score 인 요소를 찾아 점수를 표시한다

        function init() { // 초기화 함수

            // 마우스 이동 이벤트 처리 함수 등록

            document.addEventListener("mousemove", mouseHandler, false);

            for (let r = 0; r < brickRows; r++) { // 전체 벽돌 행에 대하여

                bricks[r] = new Array(brickColumns); // 1차원 배열을 생성하여 대입한다.

                for (let c = 0; c < brickColumns; c++) { // 벽돌의 위치를 계산한다.

                    let brickX = (c \* (brickWidth + brickPadding)) + brickStartX;

                    let brickY = (r \* (brickHeight + brickPadding)) + brickStartY;

                    // 벽돌 객체를 생성하여서 배열에 저장한다.

                    bricks[r][c] = new Brick(brickX, brickY, 1);

                }

            }

        }

초기화 함수를 정의한다  
이 함수가 호출되면 마우스 이동 이벤트 처리 함수(mouseHandler)를 등록한다  
전채 벽돌 행에 대하여 1차원 배열을 생성하여 대입한다  
이후 반복문을 또 돌아서 벽돌의 위치를 계산하고  
벽돌 객체를 생성하여 2차원 배열에 저장한다

        function mouseHandler(e) { // 마우스 이벤트 처리기

            let x = e.clientX - canvas.offsetLeft; // 마우스의 위치를 계산한다.

            if (x > 0 && x < canvas.width) {

                paddle.x = x - paddle.width / 2; // 패들의 x좌표에 저장한다.

            }

        }

마우스 이벤트 처리기 함수를 정의한다  
마우스의 위치를 계산하여 패들의 x 좌표에 저장한다

        function checkwin() {

            for (let r = 0; r < brickRows; r++) {

                for (let c = 0; c < brickColumns; c++) {

                    if (bricks[r][c].status === 1) {

                        // 아직 살아있는 벽돌이 있다면 승리 상태가 아님

                        return false;

                    }

                }

            }

            return true; // 모든 벽돌이 파괴되면 true, 그렇지 않으면 false 반환

        }

승리 조건을 검사하는 함수를 정의한다  
모든 벽돌에 대하여 검사하고, 아직 살아있는 상태(status == 1)의 벽돌이 있다면 false를 반환한다.  
만약 모든 벽돌의 상태가 0이라면 true를 반환한다

        function gameLoop() { // 게임 루프 함수

            context.clearRect(0, 0, canvas.width, canvas.height); // 화면을 지운다.

            let life = document.getElementById('life');

            life.textContent = lives;

            let sc = document.getElementById('score');

            sc.textContent = score;

            ball.draw(); // 화면에 공을 그린다.

            paddle.draw(); // 화면에 패들을 그린다.

            meteor.draw();

게임 루프 함수를 정의한다. 이 함수는 주기적으로 호출되어 갱신된다  
먼저 콘텍스트 객체의 화면을 지운다  
id가 life인 HTML 요소를 찾아서 현재 생명을 표시한다  
id가 score인 HTML 요소를 찾아서 현재 점수를 표시한다  
화면에 공과 패들과 운석을 그린다

            for (let r = 0; r < brickRows; r++) {

                for (let c = 0; c < brickColumns; c++) { // 각 벽돌에 대하여

                    bricks[r][c].check(); // 충돌 검사를 한다.

                    bricks[r][c].draw(); // 벽돌을 화면에 그린다.

                }

            }

모든 벽돌에 대하여 check 함수와 draw 함수를 호출하여 상태를 체크하고 그린다

            // 공이 화면의 왼쪽이나 오른쪽에 도달하였으면 방향을 바꾼다.

            if (ball.x + ball.dx > canvas.width - ball.radius

                || ball.x + ball.dx < ball.radius) {

                ball.dx = -ball.dx;

            }

공이 화면 왼쪽이나 오른쪽에 도달하면 방향을 바꾼다

            // 공이 화면의 위쪽에 도달하였으면 방향을 바꾼다.

            if (ball.y + ball.dy < ball.radius) {

                ball.dy = -ball.dy;

            }

공이 화면 위쪽에 도달하면 방향을 바꾼다

            // 공이 화면의 아래쪽에 도달하였으면 패들과 충돌을 검사한다.

            else if (ball.y + ball.dy > canvas.height - ball.radius) {

                if (ball.x > paddle.x && ball.x < paddle.x + paddle.width) {

                    ball.dy = -ball.dy;

                }

                else {

                    lives--; // 생명 감소

                    ball.dy = -ball.dy;

                }

            }

공이 화면 아래쪽에 도달하면 공의 방향을 바꾼다  
패들과 충돌을 검사하여 패들과 충돌하지 않았다면 생명을 감소시킨다

            if (meteor.y + meteor.dy + meteorimg.height > canvas.height) {

                if (meteor.x > paddle.x && meteor.x < paddle.x + paddle.width) {

                    lives--;

                }

                meteor.x = Math.floor(Math.random() \* 570) + 1

                meteor.y = -meteorimg.height;

            }

운석이 화면 아래쪽에 도달하는지 검사한다. 화면 아래에 도달하면 운석은 랜덤한 x, 화면 맨 위 y 좌표로 이동한다  
만약 운석이 패들과 충돌했다면 생명을 감소시킨다

            if (lives <= 0) { //패배했는지 검사

                let life = document.getElementById('life');

                life.textContent = 0;

                let winorlose = document.getElementById('winorlose');

                winorlose.textContent = "You Lose..";

                document.removeEventListener("mousemove", mouseHandler, false);

                clearInterval(interval);

            }

생명이 0 이하(패배조건)인지 검사한다. 만약 0이라면 id가 winorlose인 HTML 요소를 찾아서   
You Lose.. 텍스트를 출력한다. 이벤트 핸들러를 제거하고 interval을 제거한다

            if (checkwin()) { //승리했는지 검사

                let winorlose = document.getElementById('winorlose');

                winorlose.textContent = "You Win!";

                document.removeEventListener("mousemove", mouseHandler, false);

                clearInterval(interval);

            }

checkwin() 함수를 호출하여 승패를 검사한다  
만약 조건문이 hit면 id가 winorlose인 HTML 요소를 찾아서 You Win 텍스트를 출력한다  
이벤트 핸들러를 제거하고 interval을 제거한다

            ball.x += ball.dx; // 공의 위치를 변경한다.

            ball.y += ball.dy;

            meteor.y += meteor.dy;

        }

공의 위치와 운석의 위치를 변경한다.

        function restartGame() {

            location.reload(); // 게임 다시 시작

        }

        init();

        setInterval(gameLoop, 5); // 5밀리초마다 gameLoop()을 호출한다.

        </script>

게임을 다시 실행하는 함수를 정희아한다  
init 함수를 호출하고 5밀리초마다 gameLoop 함수를 호출한다

    <div class="numberbox">

        <h1>Life:</h1>

        <h1 id="life"></h1>

    </div>

div 요소를 하나 만든다 이것은 생명을 표시한다

    <div class="numberbox">

        <h1>Score:</h1>

        <h1 id="score"></h1>

    </div>

div 요소를 하나 만든다 이것은 점수를 표시한다

    <div class="gameover">

        <h1 id="winorlose"></h1>

    </div>

div 요소를 하나 만든다 이는 사용자에게 승패를 표시한다

    <button onclick="restartGame()">게임 다시 시작</button>

</body>

</html>

버튼을 하나 생성한다 이 버튼을 누르면 게임을 다시 시작한다

1. **전체 소스 코드 (설명이 없습니다)**

<!DOCTYPE html>

<html lang="ko">

<head>

    <style>

        canvas {

            background: rgb(243, 245, 117);

            border: 1px solid black;

            display: block;

        }

        .numberbox {

            display: flex;

            justify-content: space-between;

            background-color: #ccc;

            padding: 10px;

            margin: 10px;

            width: 550px;

            border: 2px solid rgb(138, 138, 138);

        }

        .numberbox h1 {

            margin: 0;

        }

        button {

            background-color: #007bff;

            color: #fff;

            border: none;

            padding: 10px 20px;

            cursor: pointer;

        }

        button:hover {

            background-color: #0056b3;

        }

        .gameover {

            position: absolute;

            top: 100px;

            left: 160px;

            font-size: 2em;

        }

    </style>

</head>

<body>

    <canvas id="myCanvas" width="600" height="400"></canvas>

    <script>

        // 캔버스 객체를 찾아서 변수 canvas에 저장한다.

        let canvas = document.getElementById("myCanvas");

        // 컨텍스트 객체를 얻는다.

        let context = canvas.getContext("2d");

        let brickRows = 3; // 벽돌의 총 행 수

        let brickColumns = 5; // 벽돌의 총 열 수

        let brickWidth = 75; // 벽돌 1개의 폭

        let brickHeight = 20; // 벽돌 1개의 높이

        let brickPadding = 10; // 벽돌 사이의 간격

        let brickStartX = 30; // 화면에서 벽돌이 시작되는 위치

        let brickStartY = 30; // 화면에서 벽돌이 시작되는 위치

        let lives = 3; //생명

        let score = 0; //점수

        let iswin = brickColumns \* brickRows;

        let bricks = new Array(brickRows); // 크기가 brickRow인 배열을 생성한다.

        let ball = { // 공을 리터럴 객체로 정의한다.

            x: canvas.width / 2, // 공의 x좌표

            y: canvas.height - 30, // 공의 y좌표

            dx: 2, // 공의 x 방향 속도

            dy: -2, // 공의 y 방향 속도

            radius: 10, // 공의 반지름

            draw: function () { // 공을 화면에 그리는 메소드

                context.beginPath();

                context.arc(this.x, this.y, this.radius, 0, Math.PI \* 2);

                context.fillStyle = "#000000";

                context.fill();

                context.closePath();

            }

        };

        let paddle = { // 패들을 객체로 정의한다.

            height: 10, // 패들의 높이

            width: 90, // 패들의 폭

            x: 300, // 패들의 x좌표

            draw: function () { // 패들을 화면에 그리는 함수

                context.beginPath();

                context.rect(this.x, canvas.height - this.height, this.width, this.height);

                context.fillStyle = "#2f92fb";

                context.fill();

                context.closePath();

            }

        };

        let meteorimg = new Image();

        meteorimg.src = 'meteor.png';

        let meteor = {

            x: 50,

            y: 50,

            dy: 1,

            draw: function() {

                context.drawImage(meteorimg, this.x, this.y, meteorimg.width, meteorimg.height);

            }

        }

        class Brick { // 벽돌을 클래스로 정의한다.

            constructor(x, y, status) { // 생성자 함수

                this.x = x;

                this.y = y;

                this.status = status;

                this.strength = Math.floor(Math.random() \* 3) + 1; //벽돌 강도 랜덤

            }

            draw() { // 벽돌의 화면에 그리는 함수

                if (this.status == 1) {// 아직도 살아 있는 벽돌이면

                    context.beginPath();

                    context.rect(this.x, this.y, brickWidth, brickHeight);

                    switch(this.strength) {

                    case 1:

                        context.fillStyle = "#A9A9A9";

                        break;

                    case 2:

                        context.fillStyle = "#808080";

                        break;

                    case 3:

                        context.fillStyle = "#505050";

                        break;

                    }

                    context.fill();

                    context.closePath();

                }

            }

            check() { // 공과 벽돌의 충돌 여부를 검사하는 함수

                if (this.status == 1) { // 아직도 살아 있는 벽돌이면

                    if (ball.x > this.x && ball.x < this.x + brickWidth // 충돌 조건

                        && ball.y > this.y && ball.y < this.y + brickHeight) {

                        ball.dy = -ball.dy; // 볼의 방향을 반전한다.

                        this.strength--;

                        if (this.strength == 0) {

                            score += 100; // 벽돌 부수면 점수+

                            this.status = 0; // 벽돌을 보이지 않게 한다.

                        }

                        let sc = document.getElementById('score');

                        sc.textContent = score;

                    }

                }

            }

        }

        function init() { // 초기화 함수

            // 마우스 이동 이벤트 처리 함수 등록

            document.addEventListener("mousemove", mouseHandler, false);

            for (let r = 0; r < brickRows; r++) { // 전체 벽돌 행에 대하여

                bricks[r] = new Array(brickColumns); // 1차원 배열을 생성하여 대입한다.

                for (let c = 0; c < brickColumns; c++) {

                    // 벽돌의 위치를 계산한다.

                    let brickX = (c \* (brickWidth + brickPadding)) + brickStartX;

                    let brickY = (r \* (brickHeight + brickPadding)) + brickStartY;

                    // 벽돌 객체를 생성하여서 배열에 저장한다.

                    bricks[r][c] = new Brick(brickX, brickY, 1);

                }

            }

        }

        function mouseHandler(e) { // 마우스 이벤트 처리기

            let x = e.clientX - canvas.offsetLeft; // 마우스의 위치를 계산한다.

            if (x > 0 && x < canvas.width) {

                paddle.x = x - paddle.width / 2; // 패들의 x좌표에 저장한다.

            }

        }

        function checkwin() {

            for (let r = 0; r < brickRows; r++) {

                for (let c = 0; c < brickColumns; c++) {

                    if (bricks[r][c].status === 1) {

                        // 아직 살아있는 벽돌이 있다면 승리 상태가 아님

                        return false;

                    }

                }

            }

            return true; // 모든 벽돌이 파괴되면 true, 그렇지 않으면 false 반환

        }

        function gameLoop() { // 게임 루프 함수

            context.clearRect(0, 0, canvas.width, canvas.height); // 화면을 지운다.

            let life = document.getElementById('life');

            life.textContent = lives;

            let sc = document.getElementById('score');

            sc.textContent = score;

            ball.draw(); // 화면에 공을 그린다.

            paddle.draw(); // 화면에 패들을 그린다.

            meteor.draw();

            for (let r = 0; r < brickRows; r++) {

                for (let c = 0; c < brickColumns; c++) { // 각 벽돌에 대하여

                    bricks[r][c].check(); // 충돌 검사를 한다.

                    bricks[r][c].draw(); // 벽돌을 화면에 그린다.

                }

            }

            // 공이 화면의 왼쪽이나 오른쪽에 도달하였으면 방향을 바꾼다.

            if (ball.x + ball.dx > canvas.width - ball.radius

                || ball.x + ball.dx < ball.radius) {

                ball.dx = -ball.dx;

            }

            // 공이 화면의 위쪽에 도달하였으면 방향을 바꾼다.

            if (ball.y + ball.dy < ball.radius) {

                ball.dy = -ball.dy;

            }

            // 공이 화면의 아래쪽에 도달하였으면 패들과 충돌을 검사한다.

            else if (ball.y + ball.dy > canvas.height - ball.radius) {

                if (ball.x > paddle.x && ball.x < paddle.x + paddle.width) {

                    ball.dy = -ball.dy;

                }

                else {

                    lives--; // 생명 감소

                    ball.dy = -ball.dy;

                }

            }

            if (meteor.y + meteor.dy + meteorimg.height > canvas.height) { //운석이 바닥에 닿으면

                if (meteor.x > paddle.x && meteor.x < paddle.x + paddle.width) {

                    lives--;

                }

                meteor.x = Math.floor(Math.random() \* 570) + 1

                meteor.y = -meteorimg.height;

            }

            if (lives <= 0) { //패배했는지 검사

                let life = document.getElementById('life');

                life.textContent = 0;

                let winorlose = document.getElementById('winorlose');

                winorlose.textContent = "You Lose..";

                document.removeEventListener("mousemove", mouseHandler, false);

                clearInterval(interval);

            }

            if (checkwin()) { //승리했는지 검사

                let winorlose = document.getElementById('winorlose');

                winorlose.textContent = "You Win!";

                document.removeEventListener("mousemove", mouseHandler, false);

                clearInterval(interval);

            }

            ball.x += ball.dx; // 공의 위치를 변경한다.

            ball.y += ball.dy;

            meteor.y += meteor.dy;

        }

        function restartGame() {

            location.reload(); // 게임 다시 시작

        }

        init();

        setInterval(gameLoop, 5); // 5밀리초마다 gameLoop()을 호출한다.

    </script>

    <div class="numberbox">

        <h1>Life:</h1>

        <h1 id="life"></h1>

    </div>

    <div class="numberbox">

        <h1>Score:</h1>

        <h1 id="score"></h1>

    </div>

    <div class="gameover">

        <h1 id="winorlose"></h1>

    </div>

    <button onclick="restartGame()">게임 다시 시작</button>

</body>

</html>