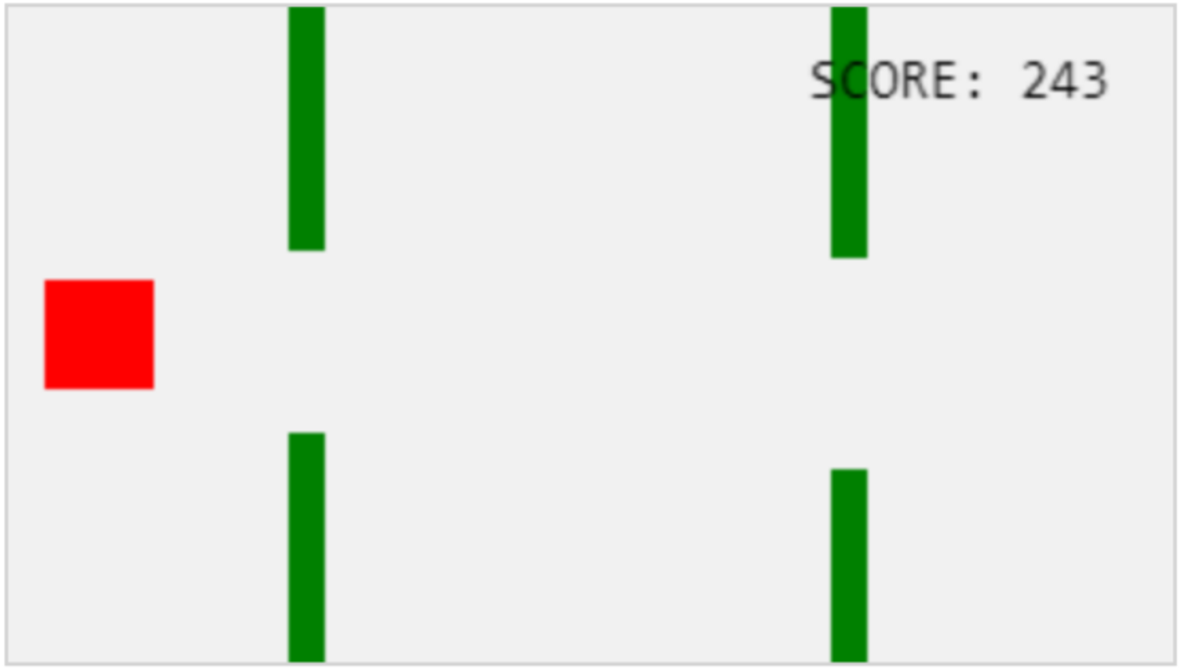
게임프로그래밍

HTML Game Example 분석

2019775009
소프트웨어학과
김수민

게임 소개

장애물 피하는 게임



코드 분석 (1)


```
<meta name="viewport" content="width=device-width, initial-scale=1.0"/>
<!--  메타태그에는 모바일 환경(viewport)에서 보여질 화면을 제어합니다
      width를 휴대폰의 화면 너비와 동일하게
      스케일은 확대 비율입니다(1.0은 100%) -->
<style>
    /* canvas 요소를 만들고 회색의 1px 두께와 , 하얀 배경색을 통해 게임이 진행될 공간을 만듭니다 */
canvas {
    border:1px solid #d3d3d3;
    background-color: #f1f1f1;
}
</style>
</head>
<!-- onload의 의미는 페이지가 랜더링이 완전히 된 후에 함수를 불러오는 의미입니다 -->
<body onload="startGame()">
```

```
<script>

var myGamePiece;        // 플레이어가 조작하는 객체(사각형)를 참조하는 변수
var myObstacles = [];   // 장애물을 저장하는 배열
var myScore;            // 점수를 저장하는 변수

function startGame() {
    myGameArea.start();  // myGameArea 객체의 start메소드 호출
    // component 라는 생성자 함수를 사용하여  객체를 생성하고 변수에 저장합니다.
    myGamePiece = new component(30, 30, "red", 10, 120);
    myScore = new component("30px", "Consolas", "black", 280, 40, "text"); // 점수를 표시할 객체를 저장
}
```

코드 분석 (2)


```
var myGameArea = { // 객체를 저장
    canvas : document.createElement("canvas"), // 동적으로 canvas html 요소를 생성후 canvas 속성할당
    start : function() { // 게임 진행되는 공간 넓이,높이
        this.canvas.width = 480;
        this.canvas.height = 270;
        this.context = this.canvas.getContext("2d"); // 2D 그래픽요소를 불러옵니다
        document.body.insertBefore(this.canvas, document.body.childNodes[0]);
            // document.body의 첫 번재 자식 노드 앞에 canvas 노드가 삽입됩니다.
        this.frameNo = 0; // 프레임 수(score값)
        this.interval = setInterval(updateGameArea, 20);
        //  20밀리초마다(초당 50회) updateGameArea함수 실행
        },
    clear : function() {
        this.context.clearRect(0, 0, this.canvas.width, this.canvas.height);
        //  전체 캔버스를 지우는 clear() 함수를 추가합니다.
    },
    stop : function() {
        // 인터벌 중지
        clearInterval(this.interval);
    }
}
```

코드 분석 (3)


```
//  ES6 신문법 이전에는 class 객체가 없었기 때문에 함수를 이용하여 객체를 생성하였습니다.
function component(width, height, color, x, y, type) {
    this.type = type;        // 타입(text or else)
    this.width = width;      // 가로
    this.height = height;    // 세로
    this.speedX = 0;         // x축 속도
    this.speedY = 0;         // y축 속도
    this.x = x;              // x좌표
    this.y = y;              // y좌표
    this.update = function() { // 2D 그래픽 그리기
        ctx = myGameArea.context; //myGameArea 객체의 context 속성을 부여(2D그래픽)
        if (this.type == "text") { // 스코어를 표시 할 때
            ctx.font = this.width + " " + this.height;
            ctx.fillStyle = color;
            ctx.fillText(this.text, this.x, this.y);
        } else {  // 도형을 그릴 때 (유저,장애물)
            ctx.fillStyle = color;
            ctx.fillRect(this.x, this.y, this.width, this.height);
        }
    }
    this.newPos = function() { // 좌표 설정
        this.x += this.speedX;
        this.y += this.speedY;
    }
```

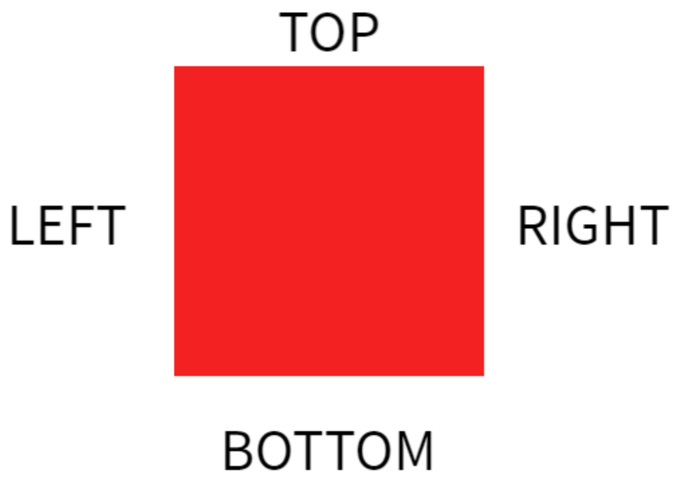
코드 분석 (4)

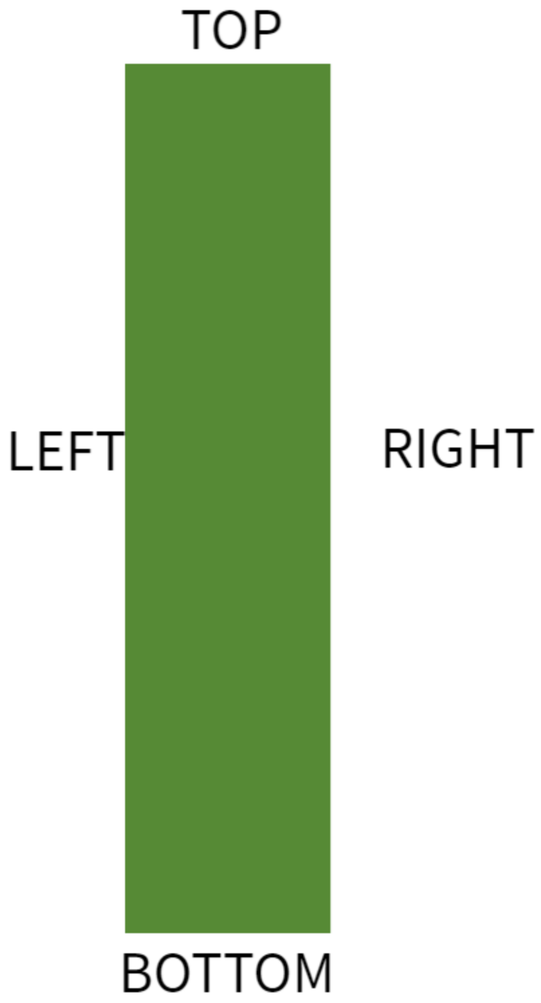

```
this.crashWith = function(otherobj) { // myObstacles[i] 즉 장애물의 객체
    var myleft = this.x;                    // 유저(사각형)의 좌측 좌표
    var myright = this.x + (this.width);  // 유저(사각형)의 우측 좌표
    var mytop = this.y;                     // 유저(사각형)의 상단 좌표
    var mybottom = this.y + (this.height);// 유저(사각형)의 하단 좌표

    var otherleft = otherobj.x;                             // 장애물 형태의 좌측 좌표
    var otherright = otherobj.x + (otherobj.width);     // 장애물 형태의 우측 좌표
    var othertop = otherobj.y;                              // 장애물 형태의 상단 좌표
    var otherbottom = otherobj.y + (otherobj.height);   // 장애물 형태의 하단 좌표
    var crash = true;   // 충돌 boolean 변수
    if ((mybottom < othertop) || (mytop > otherbottom) || (myright < otherleft) || (myleft > otherright)) {
        crash = false;  // // 충돌 상태가 아님
    }
    // 위 if문은 아래의 조건에 해당하지 않을 시 충돌이 아닙니다.
    // 사각형의 하단 좌표가 장애물의 상단 좌표보다 작다면
    // 사각형의 상단 좌표가 장애물의 하단 좌표보다 크다면
    // 사각형의 우측 좌표가 장애물의 좌측 좌표 보다 작다면
    // 사각형의 좌측 좌표가 장애물의 우측 좌표를 넘어섰다면

    // 위 조건문에 하나라도 해당하지 않는다면 충돌
    return crash;
}
```

코드 분석 (4)-부연설명





코드 분석 (5)


```
function updateGameArea() { // 지우고 그리기를 반복합니다.초당50회 불러오던 함수
    var x, height, gap, minHeight, maxHeight, minGap, maxGap;
    for (i = 0; i < myObstacles.length; i += 1) {  // 장애물의 수 만큼 반복
        if (myGamePiece.crashWith(myObstacles[i])) { // 장애물에 하나라도 충돌 시 멈춤
            myGameArea.stop();
            return;
        }
    }
    // 충돌이 없다면 화면을 지우고 프레임을 증가
    myGameArea.clear();
    myGameArea.frameNo += 1; // 프레임==스코어 값
    if (myGameArea.frameNo == 1 || everyinterval(150)) { // 첫 프레임 혹은 150 프레임마다 장애물 생성
        x = myGameArea.canvas.width;
        // 장애물의 높이 Height 최소값,최대값을 기준으로
        minHeight = 20;
        maxHeight = 200;
        height = Math.floor(Math.random()*(maxHeight-minHeight+1)+minHeight);
        // 장애물의 간격 Gap 최소값,최대값을 기준으로
        minGap = 50;
        maxGap = 200;
        gap = Math.floor(Math.random()*(maxGap-minGap+1)+minGap);
        myObstacles.push(new component(10, height, "green", x, 0));
        myObstacles.push(new component(10, x - height - gap, "green", x, height + gap));
        // 서로 다른 크기와 좌표값의 장애물들 생성
    }
     // 장애물들의 수 만큼 좌측으로 이동시키며 그리는 것을 반복
    for (i = 0; i < myObstacles.length; i += 1) {
        myObstacles[i].speedX = -1;
        myObstacles[i].newPos();
        myObstacles[i].update();
    }
    // 사각형(유저)도 좌표값 바꾸며 다시 그려줍니다. + 스코어 값
    myScore.text="SCORE: " + myGameArea.frameNo;
    myScore.update();
    myGamePiece.newPos();
    myGamePiece.update();
}
```

myGameArea.start 내용의 일부

```
this.interval = setInterval(updateGameArea, 20);
```

코드 분석 (6)


```
// 프레임 번호가 n의 배수일 때 장애물 생성
// 즉 일정한 간격으로 장애물을 생성해주기 위해
function everyinterval(n) {
    if ((myGameArea.frameNo / n) % 1 == 0) {return true;}
    return false;
}

function moveup() {
    myGamePiece.speedY = -1;
}

function movedown() {
    myGamePiece.speedY = 1;
}

function moveleft() {
    myGamePiece.speedX = -1;
}

function moveright() {
    myGamePiece.speedX = 1;
}

function clearmove() {
    myGamePiece.speedX = 0;
    myGamePiece.speedY = 0;
}
</script>
```

```
<div style="text-align:center;width:480px;">
  <!-- 마우스 클릭 했을 때,마우스 버튼을 뗄 때 , 버튼 터치 할 때 (모바일) -->
  <button onmousedown="moveup()" onmouseup="clearmove()" ontouchstart="moveup()">UP</button><br><br>
  <button onmousedown="moveleft()" onmouseup="clearmove()" ontouchstart="moveleft()">LEFT</button>
  <button onmousedown="moveright()" onmouseup="clearmove()" ontouchstart="moveright()">RIGHT</button><br><br>
  <button onmousedown="movedown()" onmouseup="clearmove()" ontouchstart="movedown()">DOWN</button>
</div>
```

출처

https://www.w3schools.com/

감사합니다