게임프로그래밍

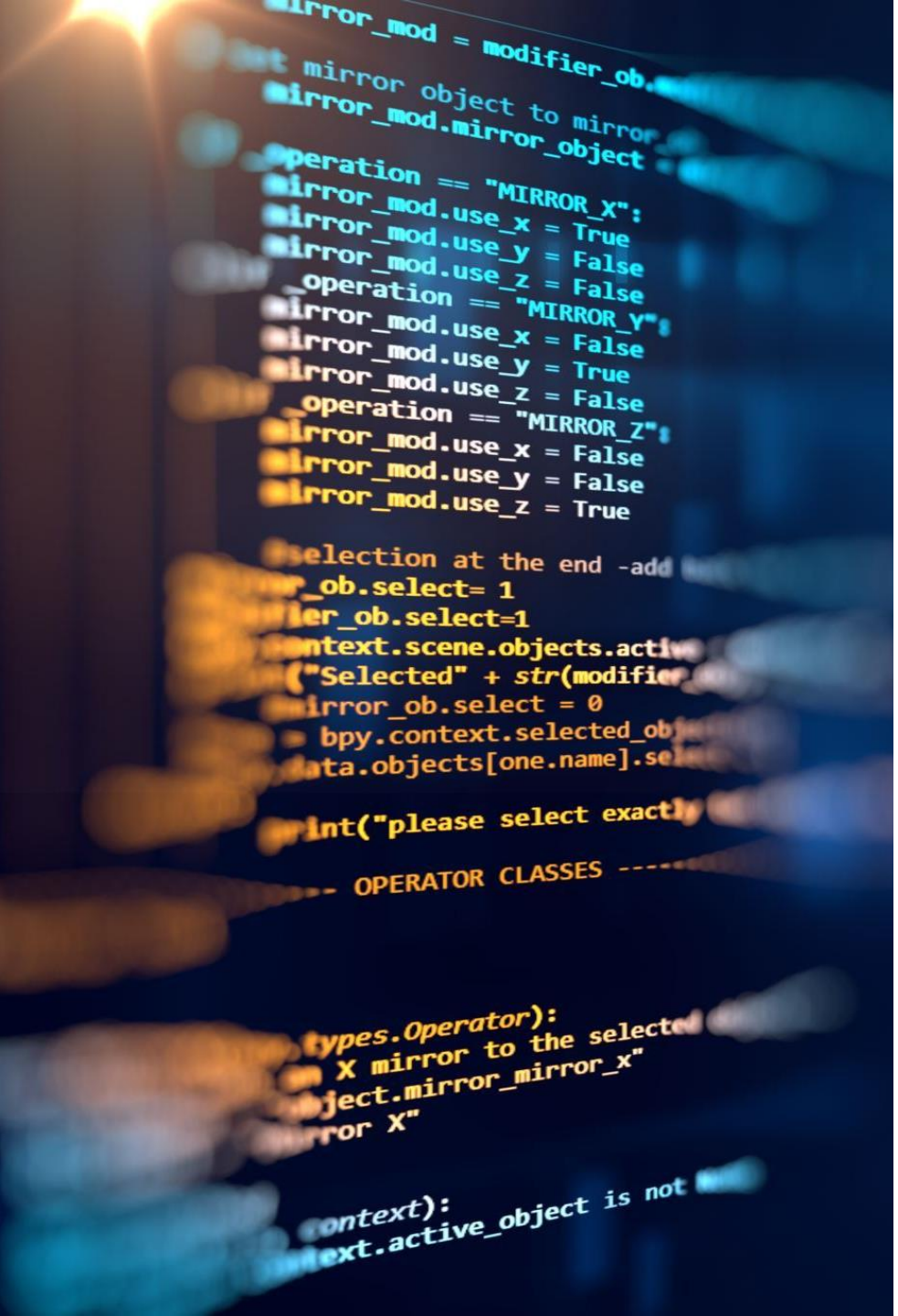
(html game)

날짜 : 2023. 11. 07

학번 : 2019775045

이름 : 이상배

경성대학교
KYUNGSUNG UNIVERSITY



목 차

1. 코드 소개 및 설명
2. 실행 결과

코드 소개

```
<!DOCTYPE html>
<html>
<head>
<meta name="viewport" content="width=device-width, initial-scale=1.0"/>
<style>
canvas {
    border:1px solid #d3d3d3;
    background-color: #f1f1f1;
}
</style>
</head>
<body onload="startGame()">
<script>

var myGamePiece;
var myObstacle;

function startGame() {
    myGamePiece = new component(30, 30, "red", 10, 120);
    myObstacle  = new component(10, 200, "green", 300, 120);
    myGameArea.start();
}

var myGameArea = {
    canvas : document.createElement("canvas"),
    start : function() {
        this.canvas.width = 480;
        this.canvas.height = 270;
        this.context = this.canvas.getContext("2d");
        document.body.insertBefore(this.canvas, document.body.childNodes[0]);
        this.interval = setInterval(updateGameArea, 20);
    },
    clear : function() {
        this.context.clearRect(0, 0, this.canvas.width, this.canvas.height);
    }
}
```

```
function component(width, height, color, x, y, type) {
    this.type = type;
    this.score = 0;
    this.width = width;
    this.height = height;
    this.speedX = 0;
    this.speedY = 0;
    this.x = x;
    this.y = y;
    this.gravity = 0;
    this.gravitySpeed = 0;
    this.update = function() {
        ctx = myGameArea.context;
        if (this.type == "text") {
            ctx.font = this.width + " " + this.height;
            ctx.fillStyle = color;
            ctx.fillText(this.text, this.x, this.y);
        } else {
            ctx.fillStyle = color;
            ctx.fillRect(this.x, this.y, this.width, thi
        }
    }
    this.newPos = function() {
        this.gravitySpeed += this.gravity;
        this.x += this.speedX;
        this.y += this.speedY + this.gravitySpeed;
        this.hitBottom();
    }
    this.hitBottom = function() {
        var rockbottom = myGameArea.canvas.height - this.height;
        if (this.y > rockbottom) {
            this.y = rockbottom;
            this.gravitySpeed = 0;
        }
    }
    this.crashWith = function(otherobj) {
        var myleft = this.x;
        var myright = this.x + (this.width);
        var mytop = this.y;
        var mybottom = this.y + (this.height);
        var otherleft = otherobj.x;
        var otherright = otherobj.x + (otherobj.width);
        var othertop = otherobj.y;
        var otherbottom = otherobj.y + (otherobj.height);
        var crash = true;
        if ((mybottom < othertop) || (mytop > otherbottom) || (myright < otherleft) || (myleft > otherright))
{
            crash = false;
        }
        return crash;
    }
}

function updateGameArea() {
    var x, height, gap, minHeight, maxHeight, minGap, maxGap;
    for (i = 0; i < myObstacles.length; i += 1) {
        if (myGamePiece.crashWith(myObstacles[i])) {
            return;
        }
    }
    myGameArea.clear();
    myGameArea.frameNo += 1;
    if (myGameArea.frameNo == 1 || everyinterval(150)) {
        x = myGameArea.canvas.width;
        minHeight = 20;
        maxHeight = 200;
        height = Math.floor(Math.random()*(maxHeight-minHeight+1)+minHeight);
        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));
    }
    for (i = 0; i < myObstacles.length; i += 1) {
        myObstacles[i].x += -1;
        myObstacles[i].update();
    }
    myScore.text="SCORE: " + myGameArea.frameNo;
    myScore.update();
    myGamePiece.newPos();
    myGamePiece.update();
}

function everyinterval(n) {
    if ((myGameArea.frameNo / n) % 1 == 0) {return true;}
    return false;
}

function accelerate(n) {
    myGamePiece.gravity = n;
}
</script>
<br>
<button onmousedown="accelerate(-0.2)" onmouseup="accelerate(0.05)">ACCELERATE</button>
<p>Use the ACCELERATE button to stay in the air</p>
<p>How long can you stay alive?</p>
</body>
</html>
```

코드 설명 <head>

- name="viewport": 이 <meta> 태그가 뷰포트에 대한 설정을 제공한다는 것을 나타낸다.
- content="width=device-width, initial-scale=1.0": 뷰포트의 너비를 기기의 너비로 설정하고, 초기 확대/축소 비율을 1.0으로 설정한다.

```
<head>
<meta name="viewport" content="width=device-width, initial-scale=1.0"/>
```

코드 설명 <style>

- border:1px solid #d3d3d3;: 캔버스 요소의 테두리를 1픽셀 두께로 설정하고, 색상을 #d3d3d3(회색)으로 지정한다.
- background-color: #f1f1f1;: 캔버스 요소의 배경색을 #f1f1f1(밝은 회색)로 설정한다.

```
<style>
canvas {
    border:1px solid #d3d3d3;
    background-color: #f1f1f1;
}
</style>
```

코드 설명 <script> startGame()

- var myGamePiece;: myGamePiece라는 변수를 선언
- var myObstacles = [];: myObstacles라는 변수를 선언하고 빈 배열로 초기화
- var myScore;: myScore라는 변수를 선언
- MyGamePiece :가로, 세로 30px의 크기를 가지며, 빨간색으로 채워진 게임 캐릭터를 나타낸다. 시작 위치는 x좌표 10, y좌표 120이다.
- MyGamePiece.gravity : 중력의 크기를 0.05로 설정한다.
- myScore : 폰트 크기가 30px이며, "Consolas" 폰트를 사용하고, 검은색으로 표시되는 텍스트를 나타낸다. 시작 위치는 x좌표 280, y좌표 40이다.
- myGameArea.start();: myGameArea 객체의 start() 메서드를 호출하여 게임 영역을 초기화하고 게임을 시작한다.

```
<script>

var myGamePiece;
var myObstacles = [];
var myScore;

function startGame() {
    myGamePiece = new component(30, 30, "red", 10, 120);
    myGamePiece.gravity = 0.05;
    myScore = new component("30px", "Consolas", "black", 280, 40, "text");
    myGameArea.start();
}
```

코드 설명 <script> myGameArea

- this.context : 2D 그래픽 컨텍스트를 가져온다.
- document.body.insertBefore: <canvas> 요소를 문서의 첫 번째 자식 노드로 삽입합니다. 이를 통해 게임 영역이 문서에 표시됩니다.
- this.frameNo: frameNo라는 속성을 0으로 초기화한다. 이 속성은 게임이 진행된 프레임 수를 나타낸다.
- this.interval: updateGameArea 함수를 20밀리초마다 반복 호출한다. 이를 통해 게임 영역이 지속적으로 업데이트된다.
- clear : 게임 영역을 지우는 역할을 한다다. (재시작)

```
var myGameArea = {
    canvas : document.createElement("canvas"),
    start : function() {
        this.canvas.width = 480;
        this.canvas.height = 270;
        this.context = this.canvas.getContext("2d");
        document.body.insertBefore(this.canvas, document.body.childNodes[0]);
        this.frameNo = 0;
        this.interval = setInterval(updateGameArea, 20);
        },
    clear : function() {
        this.context.clearRect(0, 0, this.canvas.width, this.canvas.height);
    }
}
```

코드 설명 component

- 이 component 생성자 함수는 게임에서 사용되는 다양한 객체를 생성하고, 객체의 상태를 업데이트하며, 그래픽을 그리는 데 사용된다. 이를 통해 게임의 캐릭터, 장애물 등을 표현하고 조작할 수 있다.

```
function component(width, height, color, x, y, type) {
    this.type = type;
    this.score = 0;
    this.width = width;
    this.height = height;
    this.speedX = 0;
    this.speedY = 0;
    this.x = x;
    this.y = y;
    this.gravity = 0;
    this.gravitySpeed = 0;
    this.update = function() {
        ctx = myGameArea.context;
        if (this.type == "text") {
            ctx.font = this.width + " " + this.height;
            ctx.fillStyle = color;
            ctx.fillText(this.text, this.x, this.y);
        } else {
            ctx.fillStyle = color;
            ctx.fillRect(this.x, this.y, this.width, this.height);
        }
    }
    this.newPos = function() {
        this.gravitySpeed += this.gravity;
        this.x += this.speedX;
        this.y += this.speedY + this.gravitySpeed;
        this.hitBottom();
    }
    this.hitBottom = function() {
        var rockbottom = myGameArea.canvas.height - this.height;
        if (this.y > rockbottom) {
            this.y = rockbottom;
            this.gravitySpeed = 0;
        }
    }
    this.crashWith = function(otherobj) {
        var myleft = this.x;
        var myright = this.x + (this.width);
        var mytop = this.y;
        var mybottom = this.y + (this.height);
        var otherleft = otherobj.x;
        var otherright = otherobj.x + (otherobj.width);
        var othertop = otherobj.y;
        var otherbottom = otherobj.y + (otherobj.height);
        var crash = true;
        if ((mybottom < othertop) || (mytop > otherbottom) || (myright < otherleft) || (myleft > otherright)) {
            crash = false;
        }
        return crash;
    }
}
```

코드 설명 updateGameArea()

- 이 함수는 게임 영역을 업데이트하고, 장애물을 생성하고 이동시키며, 충돌 검사를 수행하고, 점수를 업데이트하는 등의 게임 로직을 담당한다. 이 함수는 게임이 진행될 때마다 호출되며, 게임의 상태를 반영하여 화면에 그래픽을 그린다.

```
function updateGameArea() {
    var x, height, gap, minHeight, maxHeight, minGap, maxGap;
    for (i = 0; i < myObstacles.length; i += 1) {
        if (myGamePiece.crashWith(myObstacles[i])) {
            return;
        }
    }
    myGameArea.clear();
    myGameArea.frameNo += 1;
    if (myGameArea.frameNo == 1 || everyinterval(150)) {
        x = myGameArea.canvas.width;
        minHeight = 20;
        maxHeight = 200;
        height = Math.floor(Math.random()*(maxHeight-minHeight+1)+minHeight);
        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));
    }
    for (i = 0; i < myObstacles.length; i += 1) {
        myObstacles[i].x += -1;
        myObstacles[i].update();
    }
    myScore.text="SCORE: " + myGameArea.frameNo;
    myScore.update();
    myGamePiece.newPos();
    myGamePiece.update();
}
```

코드 설명 everyinterval, accelerate

- everyinterval 함수는 게임 영역의 프레임 간격에 따라 true 또는 false를 반환하는 함수이다. 이를 통해 일정한 간격으로 특정 동작을 수행하거나 조건을 체크할 수 있다.
- accelerate 함수는 중력에 의한 객체의 움직임을 조정하는 함수이다. 매개변수 n을 통해 중력의 강도를 조절할 수 있으며, 더 큰 값은 더 강한 중력을 의미한다. 이를 통해 게임 캐릭터의 점프나 낙하 속도 등을 조정할 수 있다.

```
function everyinterval(n) {
    if ((myGameArea.frameNo / n) % 1 == 0) {return true;}
    return false;
}

function accelerate(n) {
    myGamePiece.gravity = n;
}
```

코드 설명 <button>

- 이 버튼은 마우스를 누르면 중력을 감소시키고, 마우스를 뗄 때 중력을 증가시키는 기능을 수행한다. 이를 통해 게임 캐릭터의 움직임을 조작할 수 있다.

```
<br>
<button onmousedown="accelerate(-0.2)" onmouseup="accelerate(0.05)">ACCELERATE</button>
<p>Use the ACCELERATE button to stay in the air</p>
```

출처

https://ko.wikipedia.org/wiki/%EC%BA%94%EB%B2%84%EC%8A%A4_%EC%9A%94%EC%86%8C

https://developer.mozilla.org/ko/docs/Web/API/Canvas_API/Tutorial/Basic_usage

교수님의 강의자료