경성대학교
KYUNGSUNG UNIVERSITY

# 게임프로그래밍

학번 : 2019775054

이름 : 전상훈

날짜 : 2023-11-08

과목 : 게임프로그래밍

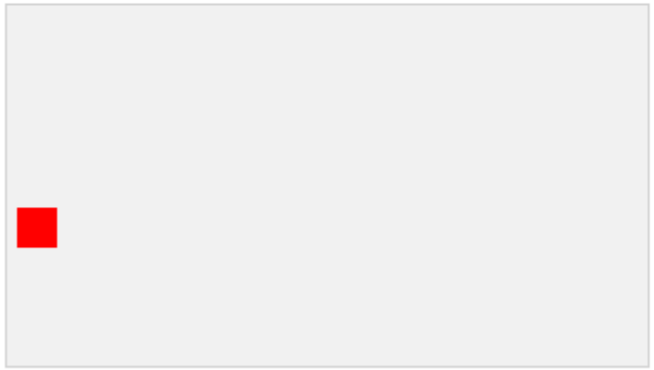
# 목차

# 1. 코드분석

## Game Movement

‹ Previous

With the new way of drawing components, explained in the Game Rotation chapter, the movements are more flexible.



Play again

Use the Keyboard

# 1. 코드분석-전체코드

```
<!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;

function startGame() {
    myGamePiece = new component(30, 30, "red", 225, 225);
    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.frameNo = 0;
        this.interval = setInterval(updateGameArea, 20);
        window.addEventListener('keydown', function (e) {
            e.preventDefault();
            myGameArea.keys = (myGameArea.keys || []);
            myGameArea.keys[e.keyCode] = (e.type == "keydown");
        })
        window.addEventListener('keyup', function (e) {
            myGameArea.keys[e.keyCode] = (e.type == "keydown");
        })
    },
    stop : function() {
        clearInterval(this.interval);
    },
    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.width = width;
    this.height = height;
    this.speed = 0;
    this.angle = 0;
    this.moveAngle = 0;
    this.x = x;
    this.y = y;
    this.update = function() {
        ctx = myGameArea.context;
        ctx.save();
        ctx.translate(this.x, this.y);
        ctx.rotate(this.angle);
        ctx.fillStyle = color;
        ctx.fillRect(this.width / -2, this.height / -2, this.width,
this.height);
        ctx.restore();
    }
    this.newPos = function() {
        this.angle += this.moveAngle * Math.PI / 180;
        this.x += this.speed * Math.sin(this.angle);
        this.y -= this.speed * Math.cos(this.angle);
    }
}

function updateGameArea() {
    myGameArea.clear();
    myGamePiece.moveAngle = 0;
    myGamePiece.speed = 0;
    if (myGameArea.keys && myGameArea.keys[37]) {myGamePiece.moveAngle = -1; }
    if (myGameArea.keys && myGameArea.keys[39]) {myGamePiece.moveAngle = 1; }
    if (myGameArea.keys && myGameArea.keys[38]) {myGamePiece.speed= 1; }
    if (myGameArea.keys && myGameArea.keys[40]) {myGamePiece.speed= -1; }
    myGamePiece.newPos();
    myGamePiece.update();
}
</script>

<p>Make sure the gamearea has focus, and use the arrow keys to move the red square around.</p>
</body>
</html>
```

# 1. 코드분석-myGameArea

```
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);
        window.addEventListener('keydown', function (e) {
            e.preventDefault();
            myGameArea.keys = (myGameArea.keys || []);
            myGameArea.keys[e.keyCode] = (e.type == "keydown");
        })
        window.addEventListener('keyup', function (e) {
            myGameArea.keys[e.keyCode] = (e.type == "keydown");
        })
    },
    stop : function() {
        clearInterval(this.interval);
    },
    clear : function() {
        this.context.clearRect(0, 0, this.canvas.width, this.canvas.height);
    }
}
```

canvas : 브라우저에서 제공하는
Canvas API를 이용하여 캔버스를 생성합니다.

start : 게임을 시작하는 함수입니다.

this.context = this.canvas.getContext("2d");
context라는 속성에, 2D 그래픽 컨텍스트를 할당합니다.
이 컨텍스트는 캔버스에 그림을 그리는 데 사용됩니다.

document.body.insertBefore(this.canvas,document.body.childNodes[0]) :
생성한 canvas를 HTML body의 맨 앞에 삽입합니다.

# 1. 코드분석-myGameArea

```
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);
        window.addEventListener('keydown', function (e) {
            e.preventDefault();
            myGameArea.keys = (myGameArea.keys || []);
            myGameArea.keys[e.keyCode] = (e.type == "keydown");
        })
        window.addEventListener('keyup', function (e) {
            myGameArea.keys[e.keyCode] = (e.type == "keydown");
        })
    },
    stop : function() {
        clearInterval(this.interval);
    },
    clear : function() {
        this.context.clearRect(0, 0, this.canvas.width, this.canvas.height);
    }
}
```

this.interval = setInterval(updateGameArea, 20);
20밀리초마다 updateGameArea 함수를 호출하는 타이머를 시작하고, 이 타이머의 ID를 interval 속성에 저장합니다.

window.addEventListener('keydown', function (e)
키보드의 키를 누르는 이벤트에 대한 이벤트 리스너를 추가합니다

window.addEventListener('keyup', function (e)
키보드의 키를 떼는 이벤트에 대한 이벤트 리스너를 추가합니다.

# 1. 코드분석-component

```
function component(width, height, color, x, y, type) {

    this.type = type;
    this.width = width;
    this.height = height;
    this.speed = 0;
    this.angle = 0;
    this.moveAngle = 0;
    this.x = x;
    this.y = y;
    this.update = function() {
        ctx = myGameArea.context;
        ctx.save();
        ctx.translate(this.x, this.y);
        ctx.rotate(this.angle);
        ctx.fillStyle = color;
        ctx.fillRect(this.width / -2, this.height / -2, this.width, this.height);
        ctx.restore();
    }
    this.newPos = function() {
        this.angle += this.moveAngle * Math.PI / 180;
        this.x += this.speed * Math.sin(this.angle);
        this.y -= this.speed * Math.cos(this.angle);
    }
}

myGamePiece = new component(30, 30, "red", 225, 225);
```

update 메서드: 객체를 화면에 그리는 역할을 합니다.

newPos 메서드: 객체의 위치를 업데이트하는 역할을 합니다.

# 1. 코드분석-updateGameArea

```
function updateGameArea() {
    myGameArea.clear();
    myGamePiece.moveAngle = 0;
    myGamePiece.speed = 0;
    if (myGameArea.keys && myGameArea.keys[37]) {myGamePiece.moveAngle = -1; }
    if (myGameArea.keys && myGameArea.keys[39]) {myGamePiece.moveAngle = 1; }
    if (myGameArea.keys && myGameArea.keys[38]) {myGamePiece.speed= 1; }
    if (myGameArea.keys && myGameArea.keys[40]) {myGamePiece.speed= -1; }
    myGamePiece.newPos();
    myGamePiece.update();
}
```

updateGameArea 함수는 게임의 상태를 업데이트하고 화면을 갱신하는 역할을 합니다.

키 입력에 따라 myGamePiece의 이동 방향과 속도를 변경하고,

myGamePiece의 위치를 업데이트하며 myGamePiece를 화면에 그리는 작업을 수행합니다.

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

myGameArea 코드

# 1. 코드분석-전체동작

1. 웹페이지 로드시 startGame 함수 실행

2. myGamePiece 라는 빨간 사각형 게임요소 생성

3. 캔버스 생성

4. 20ms 마다 updateGameArea 함수 실행되어 지속으로 업데이트

# 1. 게임 실행

# 4. 출처

https://www.w3schools.com/

ai.com

https://wrtn.ai/