게임프로그래밍 과제
〔장애물 피하기〕

소프트웨어학과
4학년
2018843103
정현준

코드 소개

HTML5와 자바스크립트를 사용하여 웹 브라우저에서 실행되는 간단한 게임을 만드는 예제입니다.

사용자는 화면에 표시되는 사각형을 조종하여 장애물을 피해야 합니다.

장애물은 무작위로 생성되며 플레이어는 버튼을 클릭하여 캐릭터를 상, 하, 좌, 우로 이동시킬 수 있습니다.

게임의 목표는 장애물에 부딪히지 않고 가능한 오래 살아남는 것입니다.

각 업데이트 프레임마다 점수가 1점씩 증가합니다.

주요 개념

- **HTML5 Canvas**

  그래픽을 그리기 위해 사용되며, 게임의 시각적 요소를 렌더링하는데 중요한 역할을 합니다.

- **JavaScript**

  게임 로직을 구현하고 사용자의 입력을 처리하며, 게임의 상태를 업데이트하는 데 사용됩니다.

- **CSS**

  게임 캔버스의 스타일을 지정하는 데 사용됩니다.

구성 요소

- 게임 영역 : 플레이어가 게임을 진행하는 캔버스 영역입니다.
- 게임 캐릭터 :사용자가 조종하는 주체로, 장애물을 피해야 하는 사각형입니다.
- 장애물: 게임 캐릭터가 피해야 할 무작위로 생성되는 사각형입니다.
- 점수: 사용자가 게임을 진행하면서 획득하는 점수로, 화면에 표시됩니다.

코드 리뷰

```
1   <!DOCTYPE html>
2   <html>
3   <head>
4   <!-- 모바일 장치에서 반응형 디자인을 위한 뷰포트 설정 -->
5   <meta name="viewport" content="width=device-width, initial-scale=1.0"/>
6   <style>
7   /* 캔버스 요소에 테두리와 배경색을 추가하는 스타일 */
8   canvas {
9       border:1px solid #d3d3d3;
10      background-color: #f1f1f1;
11  }
12  </style>
13  </head>
14  <body onload="startGame()">
15  <script>
16
```

코드 리뷰

```
1   <!DOCTYPE html>
2   <html>
3   <head>
4   <!-- 모바일 장치에서 반응형 디자인을 위한 뷰포트 설정 -->
5   <meta name="viewport" content="width=device-width, initial-scale=1.0"/>
6   <style>
7   /* 캔버스 요소에 테두리와 배경색을 추가하는 스타일 */
8   canvas {
9       border:1px solid #d3d3d3;
10      background-color: #f1f1f1;
11  }
12  </style>
13  </head>
14  <body onload="startGame()">
15  <script>
16
```

코드 리뷰

```
17    // 게임 변수 초기화
18    var myGamePiece;
19    var myObstacles = [];
20    var myScore;
21
22    // 게임을 시작하는 함수
23    function startGame() {
24        myGamePiece = new component(30, 30, "red", 10, 120); // 게임 캐릭터 생성
25        myScore = new component("30px", "Consolas", "black", 280, 40, "text"); // 점수 표시 생성
26        myGameArea.start(); // 게임 영역 시작
27    }
28
```

코드 리뷰

```
29  // 게임 영역을 나타내는 객체
30  var myGameArea = {
31      canvas : document.createElement("canvas"),
32      start : function() {
33          this.canvas.width = 480; // 캔버스 너비 설정
34          this.canvas.height = 270; // 캔버스 높이 설정
35          this.context = this.canvas.getContext("2d"); // 2D 컨텍스트 획득
36          document.body.insertBefore(this.canvas, document.body.childNodes[0]); // 캔버스를 문서에 삽입
37          this.frameNo = 0; // 프레임 번호 초기화
38          this.interval = setInterval(updateGameArea, 20); // 게임 업데이트 간격 설정
39      },
40      clear : function() {
41          this.context.clearRect(0, 0, this.canvas.width, this.canvas.height); // 캔버스 지우기
42      },
43      stop : function() {
44          clearInterval(this.interval); // 게임 업데이트 간격 중지
45      }
46  }
47
```

코드 리뷰

```
48  // 게임 구성 요소를 생성하는 생성자 함수
49  function component(width, height, color, x, y, type) {
50      // 구성 요소의 속성들
51      this.type = type;
52      this.width = width;
53      this.height = height;
54      this.speedX = 0;
55      this.speedY = 0;
56      this.x = x;
57      this.y = y;
58      this.update = function() {
59          // 캔버스 위에 구성 요소를 그림
60          ctx = myGameArea.context;
61          if (this.type == "text") {
62              ctx.font = this.width + " " + this.height;
63              ctx.fillStyle = color;
64              ctx.fillText(this.text, this.x, this.y); // 텍스트 구성 요소의 경우
65          } else {
66              ctx.fillStyle = color;
67              ctx.fillRect(this.x, this.y, this.width, this.height); // 사각형 구성 요소의 경우
68          }
69      }
```

코드 리뷰

```
70      this.newPos = function() {
71          // 구성 요소의 위치를 업데이트함
72          this.x += this.speedX;
73          this.y += this.speedY;
74      }
75      this.crashWith = function(otherobj) {
76          // 다른 구성 요소와의 충돌을 확인함
77          var myleft = this.x;
78          var myright = this.x + (this.width);
79          var mytop = this.y;
80          var mybottom = this.y + (this.height);
81          var otherleft = otherobj.x;
82          var otherright = otherobj.x + (otherobj.width);
83          var othertop = otherobj.y;
84          var otherbottom = otherobj.y + (otherobj.height);
85          var crash = true;
86          if ((mybottom < othertop) || (mytop > otherbottom) || (myright < otherleft) || (myleft > otherright)) {
87              crash = false; // A의 어느 면도 B를 벗어나지 않으면
88          }
89          return crash;
90      }
91  }
92
```

코드 리뷰

```
93  // 게임 영역을 업데이트하는 메인 함수
94  function updateGameArea() {
95      var x, height, gap, minHeight, maxHeight, minGap, maxGap;
96      // 충돌을 확인함
97      for (i = 0; i < myObstacles.length; i += 1) {
98          if (myGamePiece.crashWith(myObstacles[i])) {
99              myGameArea.stop();
100             return;
101         }
102     }
103     myGameArea.clear(); // 캔버스를 지움
104     myGameArea.frameNo += 1; // 프레임 번호 증가
105     // 일정 간격으로 장애물 생성
106     if (myGameArea.frameNo == 1 || everyinterval(150)) {
107         x = myGameArea.canvas.width;
108         minHeight = 20; // 아래쪽 장애물의 최소 높이
109         maxHeight = 200; // 아래쪽 장애물의 최대 높이
110         height = Math.floor(Math.random()*(maxHeight-minHeight+1)+minHeight);
111         minGap = 50; // 장애물 사이의 최소 간격
112         maxGap = 200; // 장애물 사이의 최대 간격
113         gap = Math.floor(Math.random()*(maxGap-minGap+1)+minGap);
114         // 플레이어가 통과할 수 있는 간격을 만들기 위해 두 개의 장애물 생성
115         myObstacles.push(new component(10, height, "green", x, 0));
116         myObstacles.push(new component(10, x - height - gap, "green", x, height + gap));
117     }
```

코드 리뷰

```
118         // 장애물을 이동시킴
119         for (i = 0; i < myObstacles.length; i += 1) {
120             myObstacles[i].speedX = -1; // 왼쪽으로 이동
121             myObstacles[i].newPos(); // 위치 업데이트
122             myObstacles[i].update(); // 업데이트된 위치를 그림
123         }
124         // 점수 텍스트를 업데이트함
125         myScore.text="SCORE: " + myGameArea.frameNo;
126         myScore.update(); // 점수를 표시함
127         myGamePiece.newPos(); // 게임 캐릭터의 위치를 업데이트함
128         myGamePiece.update(); // 게임 캐릭터를 그림
129     }
130
131     // 현재 프레임 번호가 특정 간격에 해당하는지 확인하는 함수
132     function everyinterval(n) {
133         if ((myGameArea.frameNo / n) % 1 == 0) {return true;}
134         return false;
135     }
136
```

코드 리뷰

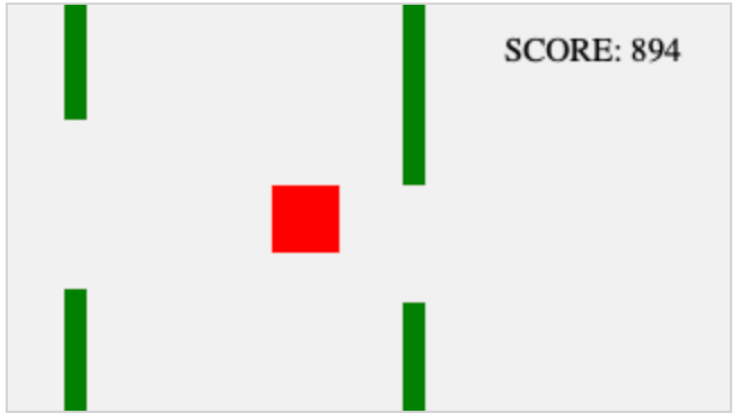
```
137  // 게임 캐릭터의 속도를 업데이트하여 움직임을 제어하는 함수들
138  function moveup() {
139      myGamePiece.speedY = -1;
140  }
141
142  function movedown() {
143      myGamePiece.speedY = 1;
144  }
145
146  function moveleft() {
147      myGamePiece.speedX = -1;
148  }
149
150  function moveright() {
151      myGamePiece.speedX = 1;
152  }
153
```

코드 리뷰

```
154   // 게임 캐릭터의 움직임을 멈추게 하는 함수
155   function clearmove() {
156       myGamePiece.speedX = 0;
157       myGamePiece.speedY = 0;
158   }
159   </script>
160   <!-- 게임 캐릭터를 조종하는 버튼들 -->
161   <div style="text-align:center;width:480px;">
162     <button onmousedown="moveup()" onmouseup="clearmove()" ontouchstart="moveup()">UP</button><br><br>
163     <button onmousedown="moveleft()" onmouseup="clearmove()" ontouchstart="moveleft()">LEFT</button>
164     <button onmousedown="moveright()" onmouseup="clearmove()" ontouchstart="moveright()">RIGHT</button><br><br>
165     <button onmousedown="movedown()" onmouseup="clearmove()" ontouchstart="movedown()">DOWN</button>
166   </div>
167
168   <p>프레임마다 "살아남으면" 점수가 1점씩 올라갑니다.</p>
169   </body>
170   </html>
171
```

결과 리뷰

Push the buttons to move the red square:

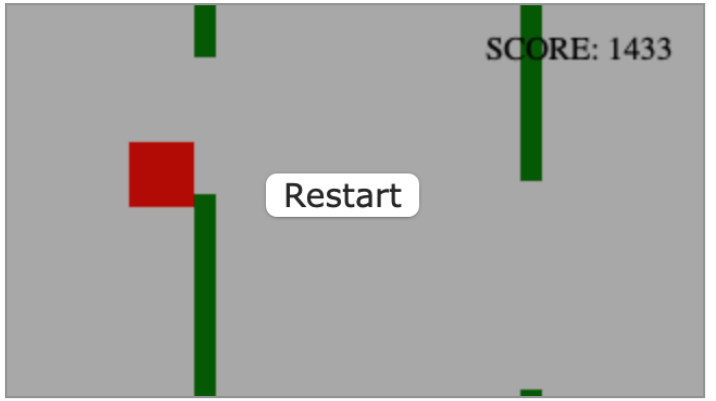


UP

LEFT RIGHT

DOWN

Push the buttons to move the red square:



UP

LEFT RIGHT

DOWN

주요 코드리뷰

게임 시작 및 초기화

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

- `startGame()` 함수는 게임을 시작할 때 호출됩니다.
- `myGamePiece`는 플레이어가 조종하는 빨간색 게임 캐릭터(사각형)를 생성합니다.
- `myScore`는 점수를 표시하는 텍스트 컴포넌트를 생성합니다.
- `myGameArea.start()`는 게임 영역을 설정하고 게임 루프를 시작합니다.

주요 코드리뷰

게임 영역 설정

```
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);
    },
    stop : function() {
        clearInterval(this.interval);
    }
}
```

- `myGameArea` 객체는 게임 캔버스와 게임의 상태를 관리합니다.
- `start` 메서드는 캔버스를 생성하고 문서에 추가한 후 `setInterval`을 사용하여 `updateGameArea` 함수를 20밀리 초마다 호출하여 게임을 업데이트합니다.
- `clear` 메서드는 캔버스를 지우는 역할을 합니다.
- `stop` 메서드는 `setInterval`을 사용하여 설정된 게임 루프를 중지합니다.

주요 코드리뷰

게임 컴포넌트 생성자

```
function component(width, height, color, x, y, type) {
    // 컴포넌트 속성 설정
    // update 메서드로 컴포넌트 그리기
    // newPos 메서드로 컴포넌트 위치 업데이트
    // crashWith 메서드로 충돌 검사
}
```

- `component` 함수는 게임 내의 객체를 생성하기 위한 생성자 함수입니다.
- 게임 캐릭터, 장애물, 점수 표시 등 모든 게임 요소를 이 생성자를 통해 만듭니다.
- `update` 메서드는 객체를 캔버스에 그리는 역할을 합니다.
- `newPos` 메서드는 객체의 위치를 업데이트합니다.
- `crashWith` 메서드는 다른 객체와의 충돌을 검사합니다.

주요 코드리뷰

게임 업데이트 함수

```
function updateGameArea() {
    // 충돌 검사
    // 캔버스 지우기
    // 장애물 생성 로직
    // 장애물 이동
    // 점수 업데이트 및 게임 캐릭터 그리기
}
```

- `updateGameArea` 함수는 게임의 메인 루프로서, 게임의 모든 업데이트를 담당합니다.
- 충돌이 발생하면 게임을 중지합니다.
- 새로운 장애물을 생성하고, 이동시키며, 점수를 업데이트하고, 게임 캐릭터의 위치를 새로 그립니다.

주요 코드리뷰

플레이어 입력 처리

```
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; }
```

- 이 함수들은 플레이어가 게임 캐릭터를 조종할 때 사용합니다.
- 각 함수는 캐릭터의 수평 또는 수직 속도를 변경하여 이동을 시작하거나 멈춥니다.

코드 깊이 보기

- 빨간 네모를 다른 이미지로 바꿀 수 있을까?
- 배경을 다른 색으로 바꿀 수 있을까?
- 배경 음악을 넣을 수 있을까?
- 컨트롤을 키보드 방향키로 할 수 있을까?
- 목숨을 여러 개로 할 수 있을까? (이어서 게임하기)

출처

- 설명 참고
- 코드
- Chat gpt

감사합니다

thank you