دی
1398

به نام او

# Air Hockey

اساتید : سید امیر هادی مینوفام و فراز سامعی

دانشجویان : علی احمدی و امین ملک لو

پروژه گروه ما به اسم Air Hockey است که به صورتی تولید شده که ۲ بازیکن دارد که میتوانند در حالات Average و Expert با یکدیگر بازی کنند.

در این بازی از توابع setRenderer ، setCamera ، setScene ، setWorld ، onWindowsResize و setLights استفاده شده است.

که در هر کدام از موارد بالا تنظیمات خاص خود پیاده سازی شد.

به عنوان مثال در تابع setWorld به تعریف اجزا صحنه پرداخته شده که از جمله آنها میتوان موارد زیر را نام برد :

۱- اضافه کردن زمین بازی

۲- اضافه کردن بازیکن ۱

۳- اضافه کردن بازیکن ۲

٤- اضافه کردن توپ

```
function setWorld() {
    x_map = 5;
    y_map = 3;
    x_rocket = 0.1;
    y_rocket = 0.5;

    /* Add kardane map */
    var geometry = new THREE.BoxGeometry(x_map, y_map, 0.01);
    var material = new THREE.MeshPhongMaterial({color: 0x7CC7FF, side: THREE.DoubleSide});
    map = new THREE.Mesh(geometry, material);
    scene.add(map);


    /* Add kardane  player 1 */
    var geometry = new THREE.BoxGeometry(x_rocket, y_rocket, 0.1);
    var material = new THREE.MeshPhongMaterial({color: 0x005000});
    player_1 = new THREE.Mesh(geometry, material);
    player_1.position.x = -x_map / 2;
    scene.add(player_1);

    /* Add kardane player 2 */
    var geometry = new THREE.BoxGeometry(x_rocket, y_rocket, 0.1);
    var material = new THREE.MeshPhongMaterial({color: 0xff0000});
    player_2 = new THREE.Mesh(geometry, material);
    player_2.position.x = x_map / 2;
    scene.add(player_2);


    /* Add kardane ball */
    ball_radius = 0.1;
    var geometry = new THREE.SphereGeometry(ball_radius, 32, 32);
    var material = new THREE.MeshPhongMaterial({color: 0xFF8C00});
    ball = new THREE.Mesh(geometry, material);
    ball.position.z += 0.1;
    scene.add(ball);


}
```

در ادامه توابعی نوشته شد که بوسیله آنها بازیکنان بعد از بیرون رفتن توپ، با توجه به شرایط به بازی برگردند. همچنین نیاز بود تا توپ را بعد از گل شدن، مدتی از دور بازی خارج کنیم تا بازیکنان دچار سر در گمی نشوند. که این مدت زمان را ۱ ثانیه لحاظ کردیم.

در حین انجام این بازی توپ به راکت (قرمز یا آبی) برخورد میکند که برای سخت تر شدن بازی تصمیم بر این شد تا از تابع رندوم برای محاسبه زاویه برگشت توپ استفاده شود.

```
var hold = 0;

function respawn_on_player1(recover_speed) {
    ball.position.copy(player_1.position);
    console.log(recover_speed)
    ball_speed = -recover_speed;
    hold = 0;
}

function respawn_on_player2(recover_speed) {
    ball.position.copy(player_2.position);
    console.log(recover_speed)
    console.log(player_1.position)
    console.log(player_2.position)
    console.log(ball.position)
    ball_speed = -recover_speed;
    hold = 0;
}

function get_random_angle(minimum, maximum) {

    var randomnumber = Math.random() * ( maximum - minimum ) + minimum;

    return randomnumber;
}

var ball_angle = Math.PI;
var player2_speed = 0.05;

```

☑ Flag ای تعریف شد تحت عنوان hold تا وضعیت توپ در آن ذخیره شود (۰ : رها شده و ۱ : نگه داشته شده)

در این بازی برای راحتی در کدنویسی و موقعیت سنجی توپ و بازیکنان از محور های x و y کمک گرفته شد.

محاسبات برخورد به ترتیب با : راکت بازیکن دوم، راکت بازیکن اول، لبه بالایی و لبه پایینی زمین بازی به صورت زیر با کمک محور ها انجام میشود :

```
113     /* check player_1 collision */
114     if (( ball.position.x < player_1.position.x + (x_rocket / 2) ) &&
115         ( ball.position.y < ( player_1.position.y + y_rocket / 2  ) ) &&
116         ( ball.position.y > ( player_1.position.y - y_rocket / 2  ) )) {
117
118         if (hold == 0) {
119             ball.position.x = player_1.position.x + (x_rocket / 2);
120             ball_speed = -ball_speed;
121             ball_angle = get_random_angle(-Math.PI / 4, Math.PI / 4);
122         }
123     }
124
125     /* check player_2 collision */
126     if (( ball.position.x > player_2.position.x - (x_rocket / 2) ) &&
127         ( ball.position.y < ( player_2.position.y + y_rocket / 2  ) ) &&
128         ( ball.position.y > ( player_2.position.y - y_rocket / 2  ) )) {
129
130         if (hold == 0) {
131             ball.position.x = player_2.position.x - (x_rocket / 2);
132             ball_speed = -ball_speed;
133             ball_angle = get_random_angle(-Math.PI / 4, Math.PI / 4);
134         }
135     }
136
137     /* collision ba labeye bala */
138     if (ball.position.y >= (y_map / 2)) {
139         ball_angle = -ball_angle;
140     }
141
142     /* collision ba labeye paein */
143     if (ball.position.y <= -(y_map / 2)) {
144         ball_angle = -ball_angle;
145     }
```

در ادامه، باید موقعیت جغرافیایی توپ تحت محور ها پیوسته چک شود که آیا گل اتفاق افتاده یا خیر

که در صورت درست بودن این شرط و اینکه توپ نگه داشته شده باشد موارد زیر به نسبت طرفی که توپ از آن خارج شده است انجام میشوند :

۱: سرعت توپ را صفر کن ۲: امتیاز بازیکنی که گل زده را یک شماره ارتقا بده ۳: امتیاز را نشان بده ٤: بازیکن را به بازی برگردان و بعد از یک ثانیه با سرعت قبلی توپ را رها کن

```
147
148     /* Goal on player_1 side */
149     if (ball.position.x < -x_map / 2 - 2 * ball_radius) {
150
151         if (hold == 0) {
152             player2_score += 1;
153             document.getElementById("player2_score").innerHTML = player2_score;
154             setTimeout(respawn_on_player1, 1000, ball_speed);
155             hold = 1;
156         }
157
158         ball_speed = 0;
159     }
160
161
162     /* Goal on player_2 side */
163     if (ball.position.x > x_map / 2 + 2 * ball_radius) {
164
165         if (hold == 0) {
166             player1_score += 1;
167             document.getElementById("player1_score").innerHTML = player1_score;
168             setTimeout(respawn_on_player2, 1000, ball_speed);
169             hold = 1;
170         }
171
172         ball_speed = 0;
173     }
174
175
```

سپس تعریف می کنیم که با فشردن کلید ها کاربر باید انتظار چه چیزی را روی صفحه داشته باشد. همچنین در ادامه آن باید مشخص کنیم که با کدام کلید ها هر بازیکن قادر به بازی کردن است. بدین صورت :

```
193 function setKeyboardControls() {
194
195     if (keyState[87]) {
196
197         if (player_1.position.y < ( (y_map / 2) - ( y_rocket / 2 ))) {
198             player_1.position.y += 0.075;
199         }
200     }
201
202     if (keyState[83]) {
203
204         if (player_1.position.y > ( -(y_map / 2) + ( y_rocket / 2 ))) {
205             player_1.position.y -= 0.075;
206         }
207     }
208     if (keyState[38]) {
209
210         if (player_2.position.y < ( (y_map / 2) - ( y_rocket / 2 ))) {
211             player_2.position.y += 0.075;
212         }
213     }
214     if (keyState[40]) {
215
216         if (player_2.position.y > ( -(y_map / 2) + ( y_rocket / 2 ))) {
217             player_2.position.y -= 0.075;
218         }
219     }
220
221     setTimeout(setKeyboardControls, 10);
222 }
```

برای مهیج شدن بازی از دو حالت Average و Expert اسفاده شد که پیش تر به آن اشاره شد. نحوه تعریف این دو تابع به صورت زیر میباشد :

```
250  function average() {
251      player1_score = 0;
252      player2_score = 0;
253      document.getElementById("player1_score").innerHTML = player1_score;
254      document.getElementById("player2_score").innerHTML = player2_score;
255
256      if (hold == 0) {
257          ball.position.x = 0;
258          ball.position.y = 0;
259          ball_speed = -0.05;
260          ball_angle = Math.PI;
261      }
262  }
263
264  function expert() {
265      player1_score = 0;
266      player2_score = 0;
267      document.getElementById("player1_score").innerHTML = player1_score;
268      document.getElementById("player2_score").innerHTML = player2_score;
269
270      if (hold == 0) {
271          ball.position.x = 0;
272          ball.position.y = 0;
273          ball_speed = -0.15;
274          ball_angle = Math.PI;
275      }
276  }
```

که در نهایت برای اجرایی شدن پروژه، در تابع اصلی همه توابع مورد نیازمان را صدا زدیم :

```
278  function main() {
279
280      setRenderer();
281      setCamera();
282      setEventListenerHandler();
283      setKeyboardControls();
284      setScene();
285      setLights();
286      setWorld();
287      animate();
288  }
```