RED BALL

İNTERNET PROGRAMLAMA

PROJESİ

Kubilay Dicle
202013172019

Serkan Mengücek
202213172801

Ali Ozan Güleş
202113172035

GÖREV TANIMLARI

Kubilay Dicle	Serkan Mengücek	Ali Ozan Güleş
HTML CSS	HTML CSS	HTML CSS
Player Movement	Spawner system	Foods System
Game UI	Ennemy System	Button and Voice
Optimizasyon ve raporun harızlan	Otomasyon ve sürdürülebilirlik	Component System

KOD AÇIKLAMALARI

HTML

HTML KODLARI

- `<!DOCTYPE html>`: HTMLversiyonunu ve türünü belirtir.
- `<html lang="">`: HTML belgesinin dilini belirtir
- `<head>`: HTML belgesinin başlık bölümünü içerir.
- `<meta charset="UTF-8">`: Belgenin karakter setini UTF-8 olarak belirtir.
- `<title>`: Web sayfasının başlığını belirtir.
- `<link rel="stylesheet" href="./style.css">`: Sayfanın stillerini içeren harici bir CSS dosyasına bağlantı sağlar.
- `<body>`: Bu kısımda sayfa içeriği ve kullanıcı arayüzü yer alır.
- `<div>`: içerik gruplaması sağlar.
- `id` özniteliği: Öğelere benzersiz tanımlayıcılar sağlar.
- `<svg>`: Vektör tabanlı grafikler oluşturmak için kullanılan bir XML tabanlı dosya formatıdır.
- `<audio>`: Ses dosyalarını çalmak için kullanılan bir HTML5 öğesidir.
- Başlık, düğmeler, skor tablosu ve ses efektlerinin yanı sıra SVG ve ses dosyalarıdır

index.html

```
<!DOCTYPE html>
<html lang="" >
<head>
  <meta charset="UTF-8">
  <title>RED BALL</title>
  <link rel="stylesheet" href="./style.css">

</head>
<body>

<!doctype html>
<html>
    <head>
        <title>aye</title>
        <link type="text/css" href="css/styles.css" rel="stylesheet"/>


    </head>
    <body>

<div id = "container">

        <div id="field">


        </div>
        <div id = "control">
            <div id= "title">KIRMIZI TOPU <br> KORU</div>
            <div id = "start-button"></div>
            <div id = "quit-button"></div>
            <div id="points">Skor: <br> <span id = "pointn">0</span></div>
            <div id="high-points">En Yüksek Skor: 0</div>
            <audio id="player" src="sounds/ice_cream_machine.mp3" loop autoplay></audio>

    <div id = "music">
<svg id="soundoff" class = "soundcolor"

   width="400pt"
   height="400pt"
   viewBox="0 0 75 75"

        <g id="g1"><polygon
   id="polygon1"
   points="39.389,13.769 22.235,28.606 6,28.606 6,47.699 21.989,47.699 39.389,62.75 39.389,13.769"
   style="stroke-width:5;stroke-linejoin:round;" />

```

KOD AÇIKLAMALARI

HTML

HTML KODLARI

- `<path>` : Scalable Vector Graphics (SVG) formatında yolları tanımlar. iki farklı yolu temsil eden iki ayrı `<path>` etiketi mevcuttur. Bu yolların görünümü, belirlenen noktalar arasındaki bağlantılar ve çizgi kalınlığı gibi özelliklerle belirlenir.
- `<svg>` : SVG formatı, web sayfalarında görsel öğeler oluşturmaya ve stil vermeye yarar. İlk `<svg>` etiketi "soundoff" adını taşır ve sesin kapalı olduğunu simgeler. İkinci `<svg>` etiketi "soundon" adını taşır ve sesin açık olduğunu simgeler
- `<script>` : JavaScript kodlarını içerir. Bu kodlar, butonlara tıklama gibi kullanıcı etkileşimlerini algılar ve buna göre sayfanın davranışını değiştirir.
- `<link>` : Harici bir CSS dosyasına bağlantı sağlar. CSS ,HTML elementlerine stili uygulamak için kullanılır. Bu web sayfalarının görünümü ve düzeni belirler.

```
<path
   id="path3003"
   d="M 48.651772,50.269646 69.395223,25.971024"
   style="fill:none;stroke-width:5;stroke-linecap:round"
   inkscape:connector-curvature="0"
   sodipodi:nodetypes="cc" />
<path
   id="path3003-1"
   d="M 69.395223,50.269646 48.651772,25.971024"
   style="fill:none;stroke-width:5;stroke-linecap:round"
   inkscape:connector-curvature="0"
   sodipodi:nodetypes="cc" /></g>
</svg>

<svg id="soundon" class = "soundcolor"
xmlns:dc="http://purl.org/dc/elements/1.1/"
xmlns:cc="http://web.resource.org/cc/"
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:svg="http://www.w3.org/2000/svg"
xmlns="http://www.w3.org/2000/svg"
xml:space="preserve"
version="1.0"
id="layer1"
width="400pt" height="400pt"
viewBox="0 0 75 75"><metadata
id="metadata1"><rdf:RDF><cc:Work
rdf:about=""><dc:format>image/svg+xml</dc:format><dc:type
rdf:resource="http://purl.org/dc/dcmitype/StillImage" /></cc:Work></rdf:RDF></metadata><g
id="g1"><polygon
id="polygon1"
points="39.389,13.769 22.235,28.606 6,28.606 6,47.699 21.989,47.699 39.389,62.75 39.389,13.769"
style="stroke-width:5;stroke-linejoin:round;;"
/><path id="path1"
d="M 48.128,49.03 C 50.057,45.934 51.19,42.291 51.19,38.377 C 51.19,34.399 50.026,30.703 48.043,27.577"
style="fill:none;stroke-width:5;stroke-linecap:round"/>
<path id="path2"
d="M 55.082,20.537 C 58.777,25.523 60.966,31.694 60.966,38.377 C 60.966,44.998 58.815,51.115 55.178,56.076"
style="fill:none;stroke-width:5;stroke-linecap:round"/>
<path id="path1"
d="M 61.71,62.611 C 66.977,55.945 70.128,47.531 70.128,38.378 C 70.128,29.161 66.936,20.696 61.609,14.01"
style="fill:none;stroke-width:5;stroke-linecap:round"/>
</g>
</svg>
  </div>

      </div>
    </div>
      <script src="js/script.js"></script>
    </body>


</html>
  <script  src="./script.js"></script>
```

```css
body {
  color: red;
  background-color: lavdenderblush;
  z-index: 0;
  /*overflow: hidden;*/
}

#field {
  z-index: 5;
  cursor: default;
  overflow: hidden;
  display: inline-block;
  background: linear-gradient(to bottom, salmon, pink, lavender, azure);
  border-radius: 10px;
}

/*--------player---*/
#player1 {
  width: 0px;
  height: 0px;
  position: absolute;
  background-color: red;
  cursor: none;
  border-radius: 50%;
  z-index: 4;
  background-image: url("../img/Ayyy_pepe.png");
}

.foe {
  background-color: #000;
  position: absolute;
  height: 1px;
  width: 1px;
  border-radius: 50%;
  z-index: 4;
  box-shadow: 0 0 20px #fff;
}

.food {
  background-color: teal;
  position: absolute;
  top: 300px;
  left: 45px;
  height: 1px;
  width: 1px;
  border-radius: 50%;
  z-index: 4;
}
```

KOD AÇIKLAMALARI

CSS

CSS KODLARI

- `body`: Sayfanın genel arka plan rengini (lavenderblush) ve metin rengini (kırmızı) belirler. Ayrıca `z-index` özelliği, sayfanın diğer elemanlarının üstüne çıkma sırasını belirler.
- `#field`: Belirli bir HTML öğesi olan `#field`in (alan) stilini tanımlar. `z-index` özelliği, bu alanın diğer öğelerin üzerine çıkma sırasını belirler. Alanın arka plan rengi, bir lineer gradyan kullanılarak belirlenir. `border-radius` özelliği, köşelerin yuvarlatılmasını sağlar.
- `#player1`: Bir oyuncu öğesinin (player1) stilini belirler. Bu oyuncu, bir oyun veya etkileşimli bir uygulama içinde kullanılır. Öğenin konumu (`position: absolute`) ve boyutları (0 piksel) burada belirlenir. Ayrıca `background-image` özelliği, oyuncunun arka plan resmini belirler.
- `.foe`: düşman (foe) öğelerinin stilini belirler. `box-shadow` özelliği, rakiplerin görünümünü oluşturur.
- `.food`: yenebilen (food) öğelerinin stilini belirler. `top` ve `left` özellikleri, öğenin dikey ve yatay konumunu belirler.

```css
.eaten {
  background-color: white;
}

#rarePepe {
  background-color: grey;
  background-image: url("../img/rainy_pepe.gif");
  background-repeat: no-repeat;
}

#points {
  position: relative;
  font-family: sans-serif;
  color: #693030;
  margin-top: 40px;
  font-size: 20px;
  text-align: center;
}

#pointn {
  font-size: 30px;
  font-family: "Comic", cursive;
  color: LightSeaGreen;
}

#high-points {
  color: #693030;
  text-align: center;
  margin-top: 15px;
  font-size: 15px;
}
.gameOver {
  background-image: url("../img/sadpepe.png");
  background-size: 500px 500px;
  background-repeat: no-repeat;
  font-family: "Comic Sans MS", sans-serif;
  text-align: right;
  position: relative;
  margin: 0 auto;
  z-index: 100;
  color: white;
  text-shadow: 2px 2px darkgray;
  font-size: 50px;
  width: 900px;
  height: 500px;
  display: inline-block;
  line-height: 500px;
}
```

KOD AÇIKLAMALARI

CSS

CSS KODLARI

- `.eaten`: Bir öğe yenildiğinde olacağı eventi belirler. Arka plan rengi beyaz olarak ayarlanır.
- `#rarePepe`: Arka plan rengi gri ve arka plan resmi bu öğeye ekler. `background-repeat` özelliği, arka plan resminin yinelenmemesini sağlar.
- `#points`: puanların stilini belirler. Metin rengi koyu kırmızı, yazı tipi sans-serif olarak ayarlanır. `margin-top` özelliği, öğenin üstündeki boşluk miktarını belirler. Yazı tipi boyutu ve hizalaması da burada belirlenir.
- `#pointn`: puan numarası stilini belirler. Yazı tipi boyutu büyük ve yazı tipi ailesi `"Comic", cursive` olarak ayarlanır. Metin rengi LightSeaGreen olarak belirlenir.
- `#high-points`: en yüksek puanlar stilini belirler. Metin rengi koyu kırmızı `ve hizalama merkezi olarak ayarlamaya yarar.
- `.gameOver`: oyunun bitiş ekranının stilini belirler. Arka plan resmi ve boyutu burada ayarlanır. .

```css
#angryPepe {
  background-image: url("../img/Angry_Pepe.jpg");
}
#quit-button {
  z-index: 1000;
  color: white;
  text-align: center;
  width: 100px;
  height: 50px;
  background-color: firebrick;
  line-height: 50px;
  border-radius: 5px;
  margin: auto;
  margin-bottom: 2px;
}
#quit-button:hover {
  z-index: 1000;
  color: white;
  text-align: center;
  width: 100px;
  height: 50px;
  background-color: #971d1d;
  line-height: 50px;
  border-radius: 5px;
  margin-bottom: 2px;
}
#start-button {
  z-index: 1000;
  color: white;
  text-align: center;

  width: 100px;
  height: 50px;
  line-height: 50px;
  background-color: lightgreen;
  border-radius: 5px;
  margin: auto;
  margin-bottom: 2px;
}
#start-button:hover {
  z-index: 1000;
  color: white;
  text-align: center;
  margin-bottom: 2px;
  width: 100px;
  height: 50px;
  line-height: 50px;
  background-color: #7aca7a;
  border-radius: 5px;
}
```

KOD AÇIKLAMALARI

CSS

CSS KODLARI

- `#angryPepe`: arka plan resmini belirler.
- `#quit-button`: çıkış butonu stilini belirler. Metin rengi beyaz, arka plan rengi firebrick olarak ayarlanır. Butonun genişliği, yüksekliği ve kenar yarıçapı gibi özellikler belirlenir. Butonun alt kenarı `margin-bottom` özelliğiyle 2 piksel olarak ayarlanır.
- `#quit-button:hover`: Bu stil tanımı, `#quit-button`ın üzerine gelindiğinde uygulanacak durumu belirler. Butonun arka plan rengi `koyu kırmızı olarak değişir.
- `#start-button`: başlat butonu durumunu belirler. Metin rengi beyaz, arka plan rengi lightgreen olarak ayarlanır. Diğer özellikler, `#quit-button` ile benzer şekilde belirlenir.
- `#start-button:hover`: `#start-button`ın üzerine gelindiğinde hangi durumu uygulanacak durumları belirler. Butonun arka plan rengi açık yeşil olarak değişir.

```css
#control {
  display: inline-block;
  vertical-align: top;
  font-family: sans-serif;
  margin-left: 20px;
}

#container {
  /*padding: 100px;*/
}

#title {
  margin: 0 auto;
  margin-bottom: 50px;
  display: block;
  color: #594b76;
  width: 100%;
  text-align: center;
  font-size: 30px;
  font-family: helvetica, sans-serif;
}

#music {
  margin-top: 2px;
  color: black;
  height: 100px;
  width: 100px;
  background-size: 50px 50px;
  background-repeat: no-repeat;
  background-position: center;
  text-align: center;
}

#soundon,
#soundoff {
  width: 40px;
  height: 40px;
  margin: auto;
  margin-top: 50px;
}

.soundcolor {
  fill: salmon;
  stroke: salmon;
}
```

KOD AÇIKLAMALARI

CSS

CSS KODLARI

- `#control`: kontrol stilini belirler. Öğe `inline-block` olarak ayarlanır ve `vertical-align` özelliği ile üst hizalanır. Metin fontu `sans-serif` olarak belirlenir, sol kenardan (`margin-left`) 20 piksel uzaklıkta olacak şekilde hizalanır.
- `#title`: başlık stilini belirler. Başlık metni merkezlenir ve renk mor olarak ayarlanır. Font büyüklüğü 30 piksel ve font türü `helvetica, sans-serif` olarak belirlenir. Başlık, altında 50 piksel boşluk bırakacak şekilde ayarlanır.
- `#music`: müzik stilini belirler. Öğenin üst kenarından 2 piksel uzaklıkta (`margin-top`) olacak şekilde ayarlanır. Renk siyah, genişlik ve yükseklik 100 piksel olarak belirlenir. Arka plan resminin boyutu 50x50 piksel ve yinelenmeme özelliği ile ayarlanır.
- `#soundon`, `#soundoff`: ses açma ve kapatma simgeleri stilini belirler. Genişlik ve yükseklikleri 40 piksel olarak ayarlanır ve dikey ve yatay olarak ortalanır. Simgelerin üst kenarından 50 piksel uzaklıkta olacak şekilde ayarlanır.
- `.soundcolor`: ses rengi stilini belirler. Dolgu ve kenar rengi `salmon` olarak ayarlanır.

KOD AÇIKLAMALARI

JAVASCRİPT

JavaScript

- **randomw**: rastgele bir yatay konum (X koordinatı) üretir.
- **randomh**: rastgele bir dikey konum (Y koordinatı) üretir.
- **randomhRare**: düşman (rare enemy) oluşturmak için rastgele bir dikey konum üretir.
- **rareSize**: düşmanın boyutunu hesaplar.
- **randsize**: Rastgele bir noktanın boyutunu belirler.
- **randFoeSize**: Rastgele bir düşmanın boyutunu belirler.
- **Dot**: bir noktayı temsil eden bir nesne oluşturur.
- **Foe**: Bu bir düşmanları temsil eden bir nesne oluşturur. **Dot** nesnesinden türetilir.
- **makeHero**: Playerin başlangıç noktasını oluşturur.
- **makeFoods**: Yiyeceklerin konumlarını ve diğer özelliklerini oluşturur.

```
var ScreenInfo = { height: 500, width: 1000, top: 0, left: 0 };

var screenw = Math.max(
  document.documentElement.clientWidth,
  window.innerWidth || 0
);
var screenh = Math.max(
  document.documentElement.clientHeight,
  window.innerHeight || 0
);

var maxw = ScreenInfo.left + ScreenInfo.width;
var maxh = ScreenInfo.top + ScreenInfo.height;
var minh = ScreenInfo.top;
var minw = ScreenInfo.left;

var gameStarted = false;

var deathCount = 0;

var randomw = function () {
  return Math.floor(Math.random() * (maxw - minw + 1));
};

var randomh = function () {
  return Math.floor(Math.random() * (maxh - minh + 1));
};

var randomhRare = function (size) {
  return Math.floor(Math.random() * (maxh + size - minh + 1));
};

var rareSize = function () {
  return maxh / (1.1 * SCALEFACTOR);
};

var randsize = function () {
  return Math.floor(Math.random() * (MAXSIZE + 1));
};
var randFoeSize = function () {
  return Math.floor(Math.random() * (MAXFOESIZE + 1));
};

var SCALEFACTOR = 10;
var FOEARRAYSIZE = 20;
var FOODARAYSIZE = 10;
var MAXSPEED = 5;
var MAXSIZE = 10;
var MAXFOESIZE = 11;
var MAXREGENDELAY = 400;
```

```
var Dot = function (posX, posY, size, color, kind) {
  this.posX = posX;
  this.posY = posY;
  this.size = size;
  this.color = color;
  this.kind = kind;
  this.score = 0;

  this.toString = function () {
    return (
      "[" +
      this.kind +
      ", size: " +
      this.size +
      ", loc: (" +
      this.posX +
      ", " +
      this.posY +
      ")" +
      "]"
    );
  };
};

var Foe = function (posX, posY, size, color, kind, xvel, yvel) {
  Dot.call(this, posX, posY, size, color, kind);
  this.xvel = xvel;
  this.yvel = yvel;
};

Foe.prototype = Object.create(Dot.prototype);

var heroDot = new Dot();

var makeHero = function () {
  heroDot = new Dot(maxw / 2, maxh / 2, 1, "red", "player");
  heroDot.score = 0;
};

var foods = new Array();

var makeFoods = function () {
  foods = new Array();
  for (var i = 0; i < FOODARAYSIZE; i++) {
    foods.push(new Dot(randomw(), randomh(), 1, "teal", "food"));
  }
};
```

```
var makeFoes = function () {
  foes = new Array();
  for (var i = 0; i < FOEARRAYSIZE; i++) {
    addRandomFoe();
  }
  console.log("made a new set of foes");
};

Dot.prototype.movedown = function () {
  this.posX = (this.posX + MAXSPEED) % maxw;
  this.posY = (this.posY + Math.sin(2 * Math.PI * (this.posX / 5)) * 4) % maxh;
};

var moveRare = function () {
  rareDot.posX += MAXSPEED;
  if (rareDot.posY > maxh) {
    rareDot.posY -= MAXSPEED;
  }
  if (rareDot.posY < minh) {
    rareDot.posY += MAXSPEED;
  }
};

Dot.prototype.runAway = function () {
  this.posX = this.posX + MAXSPEED * 3;
  this.posY = this.posY + Math.sin(2 * Math.PI * (this.posX / 5)) * 4 * 3;
};

Dot.prototype.moveover = function () {
  this.posX = (this.posX + 0.1) % maxw;
  this.posY = (this.posY + Math.cos(2 * Math.PI * (this.posX / 5)) * 4) % maxh;
};

var isColliding = function (foecheck, foodcheck, rarecheck) {
  if (foecheck) {
    var i = 0;
    for (i = 0; i < foes.length; i++) {
      var adot = foes[i];

      var dist = Math.sqrt(
        (heroDot.posX - adot.posX) * (heroDot.posX - adot.posX) +
          (heroDot.posY - adot.posY) * (heroDot.posY - adot.posY)
      );
      if (
        dist <=
        (adot.size * SCALEFACTOR) / 2 + (heroDot.size * SCALEFACTOR) / 2
      ) {
        if (adot.size > heroDot.size) {
          console.log(
            "Hero " +
```

KOD AÇIKLAMALARI

JAVASCRİPT

JavaScript

- makeFoes: Düşmanların konumlarını ve diğer özelliklerini oluşturur.
- Dot.prototype.movedown: Playeri aşağı doğru hareket ettirir.
- moveRare: düşmanı hareket ettirir.
- Dot.prototype.runAway: Noktanın kaçmasını sağlar.
- Dot.prototype.moveover: Playerin yatay yönde hareket etmesini sağlar.
- isColliding: Çarpışma kontrolünü gerçekleştirir.
- eatFood: Yiyecekleri yiyerek oyuncunun boyutunu ve skorunu artırır.

```
            "Hero " +
              heroDot.toString() +
              " was eaten by foe " +
              adot.toString()
          );
          eatHero();
          console.log("Hero died :(");
          var audio = new Audio("sounds/bomb2.wav");
          audio.play();
        } else {
          adot.eaten = true;
          eatFoe(i);

          var audio = new Audio("sounds/Sonic_Ring.mp3");
          audio.play();
        }
      } else {
      }
    }
  }

  if (foodcheck) {
    var j = 0;
    for (j = 0; j < foods.length; j++) {
      var adot = foods[j];
      var dist = Math.sqrt(
        (heroDot.posX - adot.posX) * (heroDot.posX - adot.posX) +
          (heroDot.posY - adot.posY) * (heroDot.posY - adot.posY)
      );
      if (
        dist <=
        (adot.size * SCALEFACTOR) / 2 + (heroDot.size * SCALEFACTOR)
      ) {

        eatFood(j);
        var audio = new Audio("sounds/Sonic_Ring.mp3");
        audio.play();
      } else {
      }
    }
  }
```

```
  if (rarecheck) {
    var dist = Math.sqrt(
      (heroDot.posX - rareDot.posX) * (heroDot.posX - rareDot.posX) +
        (heroDot.posY - rareDot.posY) * (heroDot.posY - rareDot.posY)
    );
    //if intersecting with rareDot
    if (
      dist <=
      (rareDot.size * SCALEFACTOR) / 2 + (heroDot.size * SCALEFACTOR)
    ) {
      eatHero();
      rareAteHero = true;
      console.log("rareDot ate hero");
    }
  }
};

var eatFood = function (i) {
  foods.splice(i, 1);
  heroDot.size += 0.5;
  heroDot.score += 1;
};

var eatFoe = function (i) {
  foes.splice(i, 1);
  heroDot.size += 0.5;
  heroDot.score += 1;
  var delay = Math.random(0, MAXREGENDELAY * 2 + 1) - MAXREGENDELAY;
  setTimeout(addRandomFoe, delay);
};

var eatHero = function () {
  circleGame.keepPlaying = false;
};

Dot.prototype.offScreen = function () {
  var radius = (this.size * SCALEFACTOR) / 2;
  return (
    this.posX - radius > maxw ||
    this.posY - radius > maxh ||
    this.posY + radius < minh
  );
};
```

KOD AÇIKLAMALARI

JAVASCRİPT

JavaScript

- eatFood: Yiyecekler yendiğinde oyuncunun boyutunu ve skorunu artırır.
- eatFoe: Düşmanları yiyerek oyuncunun boyutunu ve skorunu artırır.
- eatHero: Oyuncunun ölümünü tetikler.
- Dot.prototype.offScreen: Playerin ekranın dışına çıkıp çıkmadığını kontrol eder.
- Foe.prototype.loop: Bir düşmanın ekrandan çıktığında yeniden oluşturulmasını sağlar.
- Foe.prototype.regenerate: düşmanların yeniden oluşturulmasını sağlar.
- addRandomFoe: Rastgele düşman ekler.

```
Foe.prototype.loop = function () {
  var radius = (this.size * SCALEFACTOR) / 2;
  if (
    this.posX - radius > maxw ||
    this.posY - radius > maxh ||
    this.posY + radius < minh
  ) {
    this.regenerate();
  }
};

Foe.prototype.regenerate = function () {
  this.posX = randomw() * -3;
  this.posY = randomh();
};

var addRandomFoe = function () {
  var xdir = Math.random(0, MAXSPEED * 2 + 1) - MAXSPEED;
  var ydir = Math.random(0, MAXSPEED * 2 + 1) - MAXSPEED;
  var size = randFoeSize();
  var color = "black";
  if (size > MAXSIZE) {
    color = "#000026";
  }
  var newfoe = new Foe(
    randomw() * -3,
    randomh(),
    size,
    color,
    "foe",
    xdir,
    ydir
  );
  foes.push(newfoe);
};

var rareDot = new Foe(
  rareSize() * -1.5,
  randomhRare(rareSize()),
  rareSize(),
  "rarePepe",
  "foe",
  Math.random(0, MAXSPEED * 2 + 1) - MAXSPEED,
  Math.random(0, MAXSPEED * 2 + 1) - MAXSPEED
);
```

KOD AÇIKLAMALARI

JAVASCRİPT

JavaScript

- runAwayAll: tüm düşmanların haraketinin başlamasını sağlar.
- allFoesOffScreen: düşmanların ekran dışında olup olmadığını kontrol eder.
- updateGame: Oyun durumunu günceller.
- drawGame: Oyun ekranını çizer.
- Dot.prototype.sizing: Bir noktanın boyutunu ayarlar.
- Dot.prototype.drawDot: Bir noktayı ekrana çizer.
- drawPlayer: Oyuncuyu ekrana çizer.

```
var beginFlee = false;
var rareDotLaunched = false;
var rareAteHero = false;
var rareDone = false;

var runAwayAll = function () {
  beginFlee = true;
};

var allFoesOffScreen = function () {
  var allOff = true;
  foes.forEach(function (each) {
    allOff = each.offScreen() && allOff;
  });
  return allOff;
};

var updateGame = function (gameInfo) {
  if (
    !rareDone &&
    !beginFlee &&
    !rareDotLaunched &&
    deathCount == 6 &&
    circleGame.keepPlaying
  ) {
    var maxdelay = 5 * 60 * 1000;
    var mindelay = 2 * 60 * 1000;
    var delay = Math.random() * (maxdelay - mindelay + 1);
    console.log("delay: " + delay);
    setTimeout(runAwayAll, delay);
    rareDotLaunched = true;
  }

  if (allFoesOffScreen() && beginFlee && !rareDone) {
    isColliding(false, true, true);
    if (rareDot.offScreen() || rareAteHero) {
      resetFoes();
      rareDone = true;
      setTimeout(function () {
        beginFlee = false;
      }, 400);
      console.log("-----rareDot has passed");
    } else {
      rareDotLaunched = true;
      moveRare();
    }
  } else if (beginFlee && !allFoesOffScreen() && !rareDone) {
    // console.log("fleeing");
    isColliding(false, true, false);
    foes.forEach(function (each) {
      each.runAway();
    });
  } else {
    foes.forEach(function (each) {
      each.movedown();
      each.loop();
    });

    foods.forEach(function (each) {
      each.moveover();
    });

    isColliding(true, true, false);
  }
};

var drawGame = function (gameInfo) {
  $screen.innerHTML = "";
  drawScore();
  drawHighScore();
  foes.forEach(function (each) {
    each.drawDot();
  });

  foods.forEach(function (each) {
    each.drawDot();
  });
  drawPlayer();

  if (rareDotLaunched && beginFlee) {
    rareDot.drawDot();
  }
};

var $screen = document.querySelector("#field");
$screen.style.backgroundColor = "papayawhip";
$screen.style.width = ScreenInfo.width + "px";
$screen.style.height = ScreenInfo.height + "px";
$screen.style.position = "relative";
$screen.style.top = ScreenInfo.top + "px";
$screen.style.left = ScreenInfo.left + "px";
```

```
Dot.prototype.sizing = function (disp) {
  if (this.size > MAXSIZE && this.kind == "player") {
    this.size = MAXSIZE;
  }
  var dispSize = this.size * SCALEFACTOR;
  disp.style.width = dispSize + "px";
  disp.style.height = dispSize + "px";
  disp.style.transform =
    "translateX(-" + dispSize / 2 + "px) translateY(-" + dispSize / 2 + "px)";
  this.size = this.size;
};

Dot.prototype.drawDot = function () {
  var $dot = document.createElement("div");
  $dot.className = this.kind;
  $dot.style.left = this.posX + "px";
  $dot.style.top = this.posY + "px";
  if (this.color == "rarePepe") {
    $dot.id = this.color;
  } else {
    $dot.style.backgroundColor = this.color;
  }

  $screen.appendChild($dot);
  this.sizing($dot);
};

var drawPlayer = function () {
  var $p1 = document.createElement("div");
  $p1.id = "player1";
  $p1.setAttribute("id", "player1");
  $p1.style.left = heroDot.posX + "px";
  $p1.style.top = heroDot.posY + "px";
  $p1.style.backgroundColor = heroDot.color;
  heroDot.sizing($p1);

  $screen.appendChild($p1);
};

var $quitB = document.querySelector("#quit-button");
var $startB = document.querySelector("#start-button");
var drawUI = function () {
  $quitB.innerHTML = "BİTİR";
  $startB.innerHTML = "BAŞLA";
};
```

KOD AÇIKLAMALARI

JAVASCRİPT

JavaScript

- drawUI: Kullanıcı arayüzünü oluşturur.
- pressStart: Oyunu başlatır.
- pressQuit: Oyundan çıkar.
- drawScore: Oyuncunun skorunu ekrana yazar.
- drawHighScore: En yüksek skoru ekrana yazar.
- Game: oyun durumunu temsil eden bir nesne oluşturur.

```
window.addEventListener("keypress", function (ev) {
  if (ev.keyCode == "32") {
    document.body.style.backgroundColor = "lavender";
    pressStart();
  } else if (ev.keyCode == "81" || ev.keyCode == "90") {
    document.body.style.backgroundColor = "LightSeaGreen";
  } else {
    document.body.style.backgroundColor = "lavender";
    pressQuit();
  }
});

$screen.addEventListener("mousemove", function (event) {
  heroDot.posX = event.clientX;
  heroDot.posY = event.clientY;
});

var highS = 0;

var drawScore = function () {
  var score = heroDot.score;
  var $points = document.querySelector("#pointn");
  $points.innerHTML = score;
};

var drawHighScore = function () {
  var score = heroDot.score;
  var $highP = document.querySelector("#high-points");

  if (highS <= score) {
    highS = score;
  } else {
    highS = highS;
  }
  $highP.innerHTML = "En Yüksek Skor: " + highS;
};

var Game = function () {
  this.keepPlaying = true;
  var gameInfo = this;

  this.quit = function () {
    this.keepPlaying = false;
  };
};
```

KOD AÇIKLAMALARI

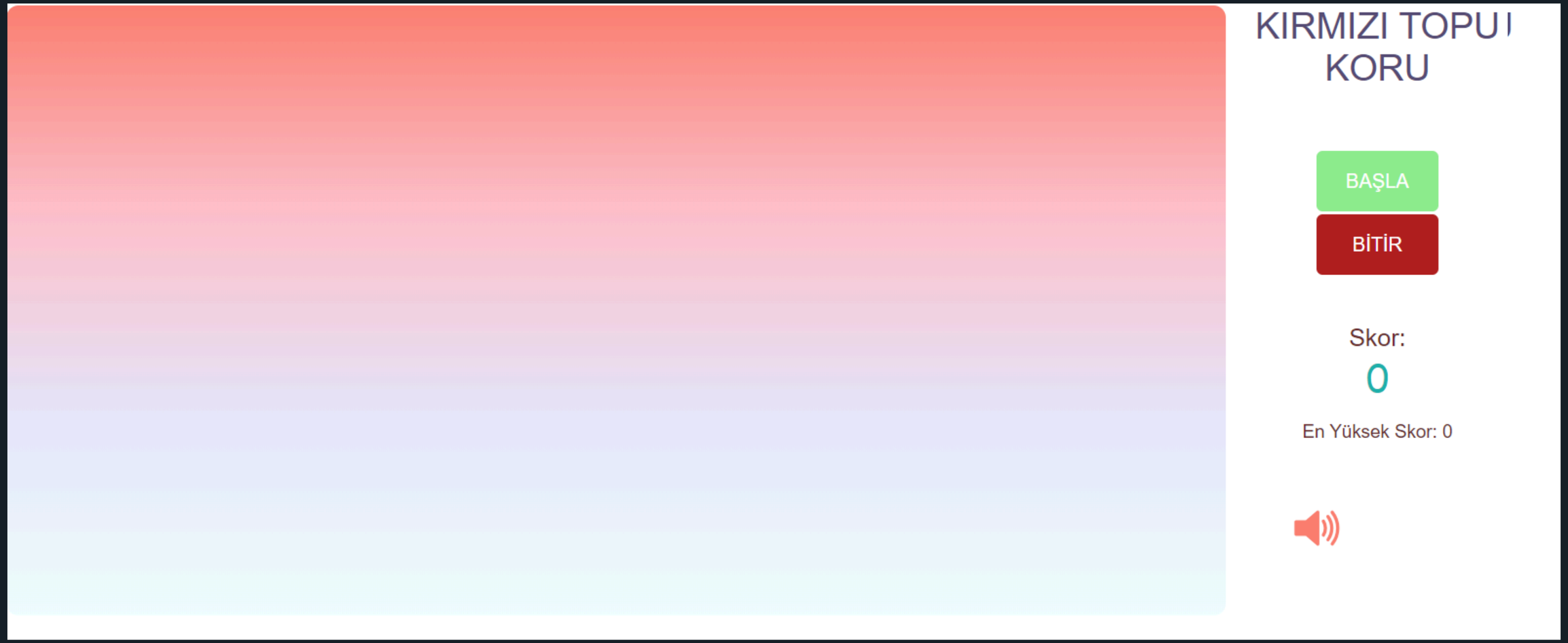
JAVASCRİPT

```
drawUI();
var circleGame;

var resetFoes = function () {
  foes.length = 0;
  setTimeout(makeFoes, 400);
};

var startGame = function () {
  circleGame = new Game();
  gameStarted = true;
  makeFoods();
  makeHero();
  resetFoes();

  var endGame = function () {
    console.log("OYUN BİTTİ");
    deathCount++;
    gameStarted = false;
    var $end = document.createElement("div");
    $end.className = "gameOver";
    $end.setAttribute("className", "gameOver");
    $end.innerHTML = "YANDIN";

    if (deathCount == 5) {
      $end.id = "angryPepe";
      $end.setAttribute("id", "angryPepe");
      $end.innerHTML = "OYUN BİTTİ";
    }

    $screen.appendChild($end);
    $screen.style.cursor = "default";
  };

  var mainLoop = function () {
    if (circleGame.keepPlaying && gameStarted) {
      updateGame(circleGame);
      drawGame(circleGame);
      window.setTimeout(mainLoop, 20);
    } else {
      endGame();
    }
  };

  mainLoop();
};
```

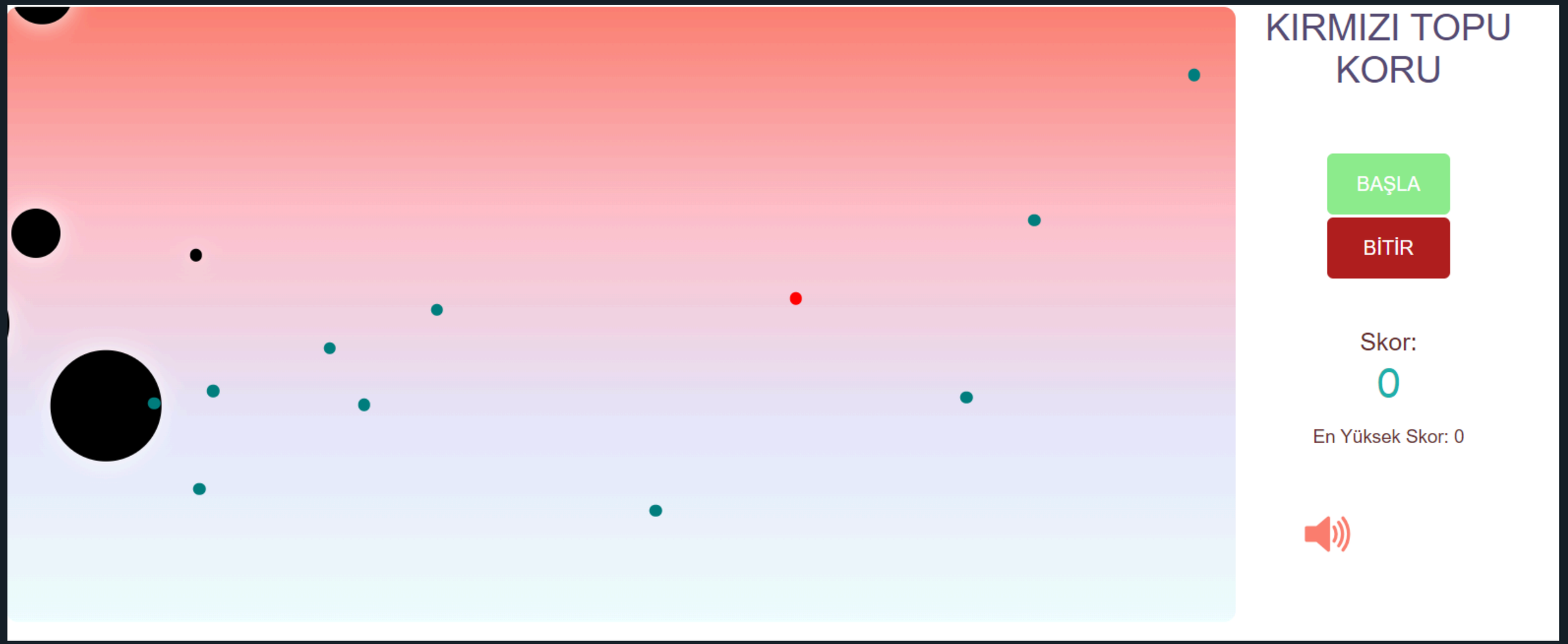
JavaScript

- resetFoes: düşmanları sıfırlar.
- startGame: oyunu başlatır.

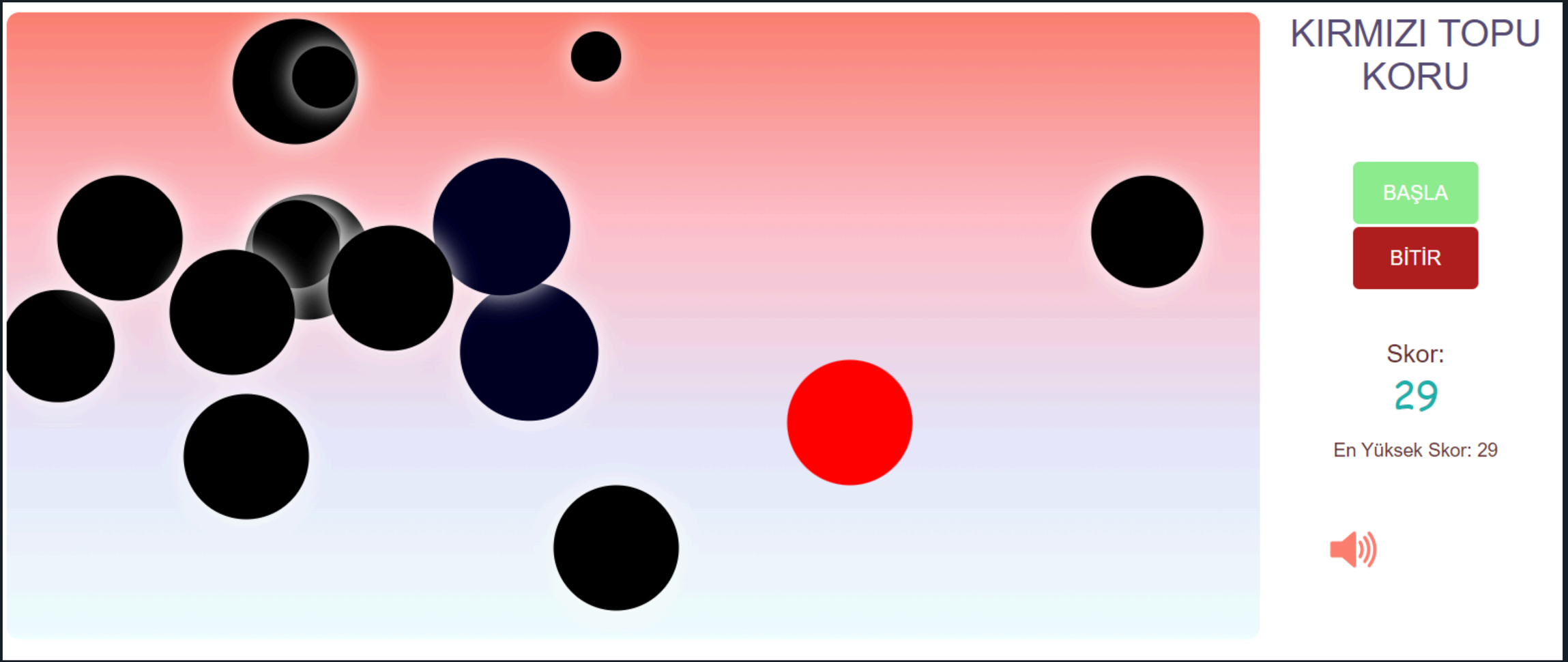
OYUN GÖRSELLERİ



OYUN GÖRSELLERİ



OYUN GÖRSELLERİ



OYUN GÖRSELLERİ

OYUN GÖRSELLERİ