به نام خدا

داکیومنتیشن پروژه گرافیک کامپیوتری

حمید قدیریان، علی زمانی

پروژه به صورت رانر که برخلاف تصور که توپ به سمت جلو حرکت میکند توپ در جای خود ثابت ایستاده و مخروط های ترافیکی به سمت توپ میایند با انجام عملیات پرش و چپ و راست رفتن باید از برخورد با انها اجتناب کرد.

```
function World()
```

دنیایی که در آن بازی رخ میدهد.و لوپ جریان بازی در ان وجود دارد.

```
function init()
```

تابع که در ان به ساختن ارکان اصلی بازی مثل صحنه، رندرر، نور، دوربین و مِین پلیر در ان بوجود می اید و چرخه اجرای بازی رخ میدهد

```
element = document.getElementById('world');
```

مشخص کردن اینکه دنیای بازی در کجای صفحه رخ بدهد.

```
renderer = new THREE.WebGLRenderer({
   alpha: true,
   antialias: true
});
renderer.setSize(element.clientWidth, element.clientHeight);
renderer.shadowMap.enabled = true;
element.appendChild(renderer.domElement);
```

مقدار دهی و ساخت رندرر.

```
scene = new THREE.Scene();
fogDistance = 40000;
scene.fog = new THREE.Fog(0xbadbe4, 1, fogDistance);
```

ساخت صحنه و تخصیص مه به آن برای طبیعی تر جلوه دادن.

```
camera = new THREE.PerspectiveCamera(60, element.clientWidth /
element.clientHeight, 1, 120000);
camera.position.set(0, 1500, -2000);
camera.lookAt(new THREE.Vector3(0, 600, -5000));
window.camera = camera;
```

ساخت دوربین پرسپکتیو و ست کردن موقعیت دوربین، مشخص کردن جهت نگاه دوربین

```
window.addEventListener('resize', handleWindowResize, false);
```

اضافه کردن لیسنر برای قابلیت ریسپانسیو بودن وب پیج به تغیرات اندازه صفحه

```
light = new THREE.DirectionalLight(0xffffff, 1);
light.position.set(1, 2, 0);
light.castShadow = true;
scene.add(light);
```

ساخت و قرار دهی موقعیت نو از نوع دایرکشنال برای روشن کردن محیط

```
character = new Character();
scene.add(character.element);
```

اینستنس و ساخت کاراکتر و اضافه کردن آن به صحنه بازی

```
var ground = createBox(3000, 20, 120000, Colors.sand, 0, -400, -60000);
scene.add(ground);
```

استفاده از تابع createbox برای ساخت زمین و سطح بازی

```
objects = [];
obstaclePresenceProb = 0.2;
for (var i = 10; i < 40; i++) {
   createRowOfTrees(i * -3000, obstaclePresenceProb, 0.5, maxTreeSize);
}
```

ساختن ارایه ابجکت ها و صدا کردن تابع ساخت موانع

```
gameOver = false;
paused = true;
```

در ابتدا اجرای بازی بازی تمام نشده ولی در حالت سکون یا پاز است

```
var left = 37;
var up = 38;
var right = 39;
var p = 80;
```

استفاده از left right up و p به جای استفاده مستمر از کد های کلید های کیبورد برای خوانایی بهتر

```
keysAllowed = {};
document.addEventListener(
   'keydown',
   function(e) {
      if (!gameOver) {
         var key = e.keyCode;
         if (keysAllowed[key] === false) return;
         keysAllowed[key] = false;
         if (paused && !collisionsDetected() && key > 18) {
            paused = false;
            character.onUnpause();
            document.getElementById(
               "variable-content").style.visibility = "hidden";
            document.getElementById(
               "controls").style.display = "none";
         } else {
            if (key == p) {
               paused = true;
               character.onPause();
               document.getElementById(
                  "variable-content").style.visibility = "visible";
               document.getElementById(
                  "variable-content").innerHTML =
                  "کن کلیک دکمه یه ادامه برای شده، استاپ بازی";
            }
            if (key == up && !paused) {
               character.onUpKeyPressed();
            }
            if (key == left && !paused) {
               character.onLeftKeyPressed();
            }
            if (key == right && !paused) {
               character.onRightKeyPressed();
            }
         }
      }
   }
);
```

استفاده از یک لیسنر برای جرای یوزر اینپوت ها که بدین صورت وارد میشود که اگر بازی تمام نشده بود وورودی دکمه درست داشتیم وارد شرط شده و در صورت اینپوت دکمه p بازی با فراخوانی تابع

onpause به حالت توقف یا استاپ میرود، اگر به ترتیب دکمه های بالا و چپ و راست زده شد با فراخوانی توابع onupkeypressed و onleftkeypressed و onrightkeypressed کاراکتر ما روی سطح به سمت چپ و راست حرکت یا به بالا میپرد.

```
document.addEventListener(
   'keyup',
   function(e) {
      keysAllowed[e.keyCode] = true;
   }
);
```

در صورت برداشتن دست از روی دکمه ها دوباره تابع صدا شده تا دستورات بعدی اجرا شوند

```
loop();
```

اجرای تابع لوپ که در ادامه به توضیح ان میپردازیم

```
if ((objects[objects.length - 1].mesh.position.z) % 3000 == 0) {

   createRowOfObstacles(-120000, 0.5 , 0.5);
   scene.fog.far = fogDistance;
}
```

شرطی که فاصله پشت هم بودن موانع را چک میکند و در صورت بودن فاصله مناسب در راستای محور z با استفاده از createrowofobstacle موانع جدید را بوجود میاورد

```
objects.forEach(function(object) {
   object.mesh.position.z += 100;
});
```

به ازای هر یک از آبجکت های مانع انها را در راستای z به سمت توپ حرکت میدهد به این صورت که در واقعیت این تصور میشود که توپ در حال جلو رفتن است در صورتی که این موانع هستند که به سمت ما می آیند

```
objects = objects.filter(function(object) {
   return object.mesh.position.z < 0;
});
```

در این قسمت موانع اضافی که از بردار مثبت z یا همان میدان دوربین خارج میشوند را برای بهینه سازی پردازش ها حذف میکنیم

```
character.update();
```

فراخوانی تابعupdate برای بروزرسانی و حرکت توپ با اینپوتها

```
if (collisionsDetected()) {
   gameOver = true;
   paused = true;
   document.addEventListener(
            'keydown',
            function(e) {
               if (e.keyCode == 40)
                document.location.reload(true);
            }
         );
         var variableContent = document.getElementById("variable-content");
         variableContent.style.visibility = "visible";
         variableContent.innerHTML =
            "بده فشار رو پایین ی دکمه دوباره شروع برای .لوزری خیلی";
```

```
        var table = document.getElementById("ranks");
        var rankIndex = Math.floor(score / 15000);
}
```

با استفاده از تابع collisionsdetected در صورت بروز برخورد بازی را به حالت اتمام برده صحنه را پاز میکنیم و با استفاده از یک لیسنر منتظر کلید پایین برای لود کردن دوباره و اغاز دوباره بازی میمانیم. و با دسترسی به المنت اچ تی ام ال و پیام مورد نظر را نمایش میدهیم

```
score += 10;
document.getElementById("score").innerHTML = score;
```

برای بروز رسانی امتیاز بر روی صفحه

```
renderer.render(scene, camera);
requestAnimationFrame(loop);
```

رندر کردن و تکرار لوپ بازی

```
function handleWindowResize() {
   renderer.setSize(element.clientWidth, element.clientHeight);
   camera.aspect = element.clientWidth / element.clientHeight;
   camera.updateProjectionMatrix();
}
```

تابع مورد نظر که با استفاده از محاسبات در صورت بروز تغیر اندازه صفحه اندازه جدید انرا به رندرر و کمرا میدهد تا صفحه را تا حدودی ریسپانسیو کنیم

```
function createRowOfObstacles(position, probability, scale) {
   for (var lane = -1; lane < 2; lane++) {
      var randomNumber = Math.random();
      if (randomNumber < probability) {
         var scale = scale;
         var obstacle = new Obstacle(lane * 800, -400, position, scale);
         objects.push(obstacle);
         scene.add(obstacle.mesh);
      }
   }
```

تابع ساخت موانع که با ساخت یک عدد رندوم و چک کردن در صورت کمتر بودن مقدار ان از احتمال وجود مانع در هر خط یک ردیف با فراخوانی تابع obstacle که در پایین به آن میپردازیم یک مانع جدید ساخته و انرا در ارایه ابجکت ها قرار میدهد و به صحنه اضافه میکند

```
function collisionsDetected() {
   var charMinX = character.element.position.x - 115;
   var charMaxX = character.element.position.x + 115;
   var charMinY = character.element.position.y - 310;
   var charMaxY = character.element.position.y + 320;
   var charMinZ = character.element.position.z - 40;
   var charMaxZ = character.element.position.z + 40;
   for (var i = 0; i < objects.length; i++) {
      if (objects[i].collides(charMinX, charMaxX, charMinY,
            charMaxY, charMinZ, charMaxZ)) {
         return true;
      }
   }
   return false;
}
```

تابعی که درآن به اصطلاح باکس کلایدر توپ ساخته شده و با استفاده از تابع collides که در پایین به ان میپردازیم در صورت بروز برخورد میان باکس کلایدر های توپ و مانع مقدار true را برمیگرداند.

```
this.sphereColor = Colors.red;
this.jumpDuration = 0.6;
this.jumpHeight = 2000;
```

تعین ثابت های رنگ توپ، مدت زمان پرش که بعدا به ان میپردازیم که به چه درد میخورد و میزان پرش.

```
function init() {


   self.sphere = createSphere(150, 150, 150, self.sphereColor, 50, -180, 0);

   self.element = createGroup(0, 0, -4000);
   self.element.add(self.sphere);

   self.isJumping = false;
   self.isSwitchingLeft = false;
   self.isSwitchingRight = false;
   self.currentLane = 0;
   self.runningStartTime = new Date() / 1000;
   self.pauseStartTime = new Date() / 1000;
   self.stepFreq = 2;
   self.queuedActions = [];

}
```

ساخت توپ و معین کردن مقدار های دیفالت اولیه ان شامل ایا درحال پرش است یا خیر، به چپ/راست میرود یا خیر و خط فعلی که در ان است و خالی بودن لیست دستورات موجود برای اجرا

```
this.update = function() {

   // Obtain the curren time for future calculations.
   var currentTime = new Date() / 1000;

   // Apply actions to the ball if none are currently being
   // carried out.
   if (!self.isJumping &&
      !self.isSwitchingLeft &&
      !self.isSwitchingRight &&
      self.queuedActions.length > 0) {
      switch(self.queuedActions.shift()) {
         case "up":
            self.isJumping = true;
            self.jumpStartTime = new Date() / 1000;
            break;
         case "left":
            if (self.currentLane != -1) {
               self.isSwitchingLeft = true;
            }
            break;
         case "right":
            if (self.currentLane != 1) {
               self.isSwitchingRight = true;
            }
```

```
            break;
        }
    }
```

متدی که برای اپدیت حرکت توپ با گذر زمان مورد استفاده قرار میگیرد، ابتدا زمان فعلی گرفته شده سپس در صورتی که توپ اکشن خاصی در حال انجام(پرش،حرکت به چپ/راست) ندارد و فضای کافی بر روی باند را بررسی و در صورت خارج نشدن از باند توپ را به چپ و راست انتقال میدهد.

```
if (self.isJumping) {
   var jumpClock = currentTime - self.jumpStartTime;
   self.element.position.y = self.jumpHeight * Math.sin(
      (1 / self.jumpDuration) * Math.PI * jumpClock) +
      sinusoid(2 * self.stepFreq, 0, 20, 0,
         self.jumpStartTime - self.runningStartTime);
   if (jumpClock > self.jumpDuration) {
      self.isJumping = false;
      self.runningStartTime += self.jumpDuration;
   }
} else {
   var runningClock = currentTime - self.runningStartTime;
   self.element.position.y = sinusoid(2 * self.stepFreq, 01, 20, 0, runningClock);
   // If the ball is not jumping, it may be switching lanes.
   if (self.isSwitchingLeft) {
      self.element.position.x -= 200;
      var offset = self.currentLane * 800 - self.element.position.x;
      if (offset > 800) {
         self.currentLane -= 1;
         self.element.position.x = self.currentLane * 800;
         self.isSwitchingLeft = false;
      }
   }
   if (self.isSwitchingRight) {
      self.element.position.x += 200;
      var offset = self.element.position.x - self.currentLane * 800;
      if (offset > 800) {
         self.currentLane += 1;
         self.element.position.x = self.currentLane * 800;
         self.isSwitchingRight = false;
      }
   }
}
```

با استفاده از شروط پیش رو اگر توپ در حال پرش باشد موقعیت y ان را به روز رسانی میکنیمدر غیر اینصورت در حال حرکت میماند،و اگر در حال پرش نیست دارد به چپ و راست میرود که باید موقعیتx ان بروز رسانی شود و در صورتی که در منتها علیه های چپ و راست قرار دارد امکان حرکت دوباره به خارج از باند را از ان میگیریم

```
this.onLeftKeyPressed = function() {
   self.queuedActions.push("left");
}
```

حرکت توپ به سمت چپ را انجام میدهد و دستور را برای اجرا به داخل کیو میبرد تا تک به تک بررسی و اجرا شوند

```
this.onUpKeyPressed = function() {
   self.queuedActions.push("up");
}
```

حرکت توپ به سمت بالا یا همان پرش را بررسی میکند و انجام میدهد

```
this.onRightKeyPressed = function() {
   self.queuedActions.push("right");
}
```

حرکت توپ به سمت راست را موقع فشردن کلید جهت نمای راست بررسی میکند

```
this.onPause = function() {
   self.pauseStartTime = new Date() / 1000;
}
```

برقراری استاپ یا همان پاز هنگام فشردن p

```
this.onUnpause = function() {
   var currentTime = new Date() / 1000;
   var pauseDuration = currentTime - self.pauseStartTime;
   self.runningStartTime += pauseDuration;
   if (self.isJumping) {
      self.jumpStartTime += pauseDuration;
   }
}
```

از سر گیری دوباره بازی پس از پاز که اگر در هنگام پرش پاز رخ داده بود دقت میکنیم تا زمان پرش به کاملی رعایت شود

```
function Obstacle(x, y, z, s) {

   // Explicit binding.
   var self = this;

   // The object portrayed in the scene.
   this.mesh = new THREE.Object3D();
    var trunk = createCylinder(300, 300, 1500, 32, Colors.orange, 0, 125, 0);
    this.mesh.add(trunk);
    this.mesh.position.set(x, y, z);
   this.mesh.scale.set(s, s, s);
   this.scale = s;
```

متد استفاده از تابع ساخت سیلندر و ساخت مانع و ست کردن موقعیت و اندازه ان

```
this.collides = function(minX, maxX, minY, maxY, minZ, maxZ) {
   var obstacleMinX = self.mesh.position.x - this.scale * 250;
   var obstacleMaxX = self.mesh.position.x + this.scale * 250;
   var obstacleMinY = self.mesh.position.y;
   var obstacleMaxY = self.mesh.position.y + this.scale * 1150;
   var obstacleMinZ = self.mesh.position.z - this.scale * 250;
   var obstacleMaxZ = self.mesh.position.z + this.scale * 250;
   return obstacleMinX <= maxX && obstacleMaxX >= minX
      && obstacleMinY <= maxY && obstacleMaxY >= minY
      && obstacleMinZ <= maxZ && obstacleMaxZ >= minZ;
}
```

متدی که در ان ابتدا باکس کلایدر مانع ساخته شده را مشخص میکنیم و تشخیص میدهد ایا این مانع در حال برخورد با توپ هست یا خیر

```
function createBox(dx, dy, dz, color, x, y, z, notFlatShading) {
    var geom = new THREE.BoxGeometry(dx, dy, dz);
    var mat = new THREE.MeshPhongMaterial({
        map: new THREE.TextureLoader().load('img/1.jpg'),
        // color: Colors.yellow,
```

```
        flatShading: notFlatShading != true
    });
    var box = new THREE.Mesh(geom, mat);
    box.castShadow = true;
    box.receiveShadow = true;
    box.position.set(x, y, z);
    return box;
}
```

متد برای ساخت زمینه یا همان باند زمین که به ان تکسچر خیابان به صورت عکس نیز داده ایم و پوزیشن ان را ست کرده و سایه پذیر کرده ایم تا سایه بر روی ان بیوفتد

```
function createSphere(radius, widthSegments, heightSegments, color, x, y, z,
notFlatShading) {
   var geometry = new THREE.SphereGeometry( radius, widthSegments, heightSegments
);
   var material = new THREE.MeshPhongMaterial( {
      color: color,
      flatShading: notFlatShading != true
   });
   var sphere = new THREE.Mesh( geometry, material );
   sphere.receiveShadow = true;
   sphere.position.set(x, y, z);
   return sphere;
}
```

متد ساخت کره یا همان توپ کاراکتر بازی که قابلیت ساختن سایه و سایه پذیری دارد

```
function createCylinder(radiusTop, radiusBottom, height, radialSegments,
                  color, x, y, z) {
    var geom = new THREE.CylinderGeometry(
       radiusTop, radiusBottom, height, radialSegments);
    var mat = new THREE.MeshPhongMaterial({
       color: color,
       flatShading: true
    });
    var cylinder = new THREE.Mesh(geom, mat);
    cylinder.castShadow = true;
    cylinder.receiveShadow = true;
    cylinder.position.set(x, y, z);
    return cylinder;
}
```

متد ساخت سیلندر برای استفاده در بالا برای قرار دادن موانع با قابلیت سایه سازی و سایه پذیری و قرار دادن مکان آن