" به نام خدا"

پروژه‌ی گرافیک کامپیوتری

جناب استاد مینوفام

مستندات پروژه

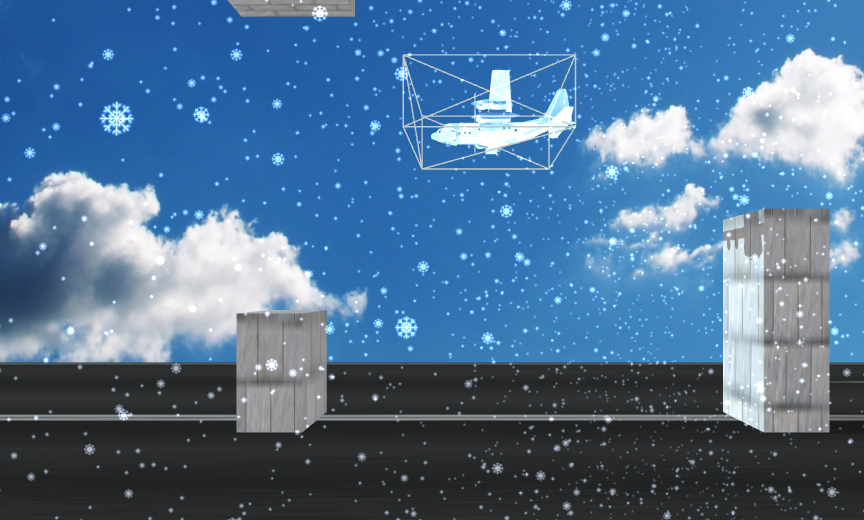
هاله سادات فاروقی

(۹۶۱۱۱۳۰۳۱)

محیا امیری

(۹۶۱۱۱۳۰۰۴)

**مقدمه:**



پروژه یک بازی هواپیما عبور از موانع است که با استفاده از فریم ورک Three.Js نوشته شده است و با استفاده از کیبورد بالا و پایین می رود.

**توضیحات کد:**

init();

animate();

تابع اصلی به صورت Prototype فراخوانی می شود.

scene = new THREE.Scene();

camera = new THREE.PerspectiveCamera(75, width / height, 0.1, 15000);

renderer = new THREE.WebGLRenderer({antialias: true});

renderer.setClearColor(new THREE.color(0x49d1ca));

renderer.setSize(width ,height);

renderer.shadowMap.enabled = true;

renderer.shadowMap.type = THREE.PCFSoftShadowMap; document.body.appendChild(renderer.domElement);

در تابع Init() مقادیر ساخت محیط برای ایجاد گرافیک قرار داده شده است. دوربین، صحنه، ترسیم گر و ایجاد سایه.

texture = THREE.ImageUtils.loadTexture('texture/road.jpg');

surface = new THREE.PlaneGeometry(15000,500 );

var surfaceMaterial = new THREE.MeshPhongMaterial({color: 0xffffff, side: THREE.DoubleSide,map:texture});

surfaceMesh = new THREE.Mesh(surface, surfaceMaterial);

surfaceMesh.rotateX(Math.PI/2);

surfaceMesh.position.x = 100;

surfaceMesh.position.y = -50;

surfaceMesh.position.z = -40;

surfaceMesh.receiveShadow = true;

scene.add(surfaceMesh)

کد ایجاد سطح است و بافت مورد انتخاب می شود. در surfaceMesh.receiveShadow = true; برای سطح تعریف می شود که این جسم سایه دریافت می کند.

directionalLight = new THREE.DirectionalLight(0x80a2ad);

directionalLight.position.set(0, 2, 2);

directionalLight.castShadow = true;

directionalLight.shadow.camera.near = 1;

directionalLight.shadow.camera.far = 500;

directionalLight.shadow.camera.right = 550;

directionalLight.shadow.camera.left = -550;

directionalLight.shadow.camera.top = 550;

directionalLight.shadow.camera.bottom = -550;

directionalLight.shadow.mapSize.width = 3048;

directionalLight.shadow.mapSize.height = 3048;

scene.add(directionalLight);

directionalLight که در ادامه به هواپیما اضافه می شود،در اینجا تعریف شده است. directionalLight باعث ایجاد سایه می شود و مقادیر سایه را تعیین می کند.

//Cubes1

var textureBox = new THREE.TextureLoader().load('texture/st.jpeg');

var cubeGeometry = new THREE.BoxGeometry(50, 350, 50);

var cubeMaterial = new THREE.MeshPhongMaterial({shininess: 100, map: textureBox});

cube = new THREE.Mesh(cubeGeometry, cubeMaterial);

cube.position.x = -1450;

cube.position.y = 220;

cube.position.z = -30;

// Clone box into boxes array

Cubes();

// Randomly placed the boxes in scene

for (var j = 0; j < cubes.length; j++) {

//cubes[j].rotateX(Math.PI/2 );

cubes[j].position.x =j\*400-1000;

cubes[j].position.y =(Math.random()-1.5)\*100;

cubes[j].position.z = -30;

cubes[j].castShadow = true;

cubes[j].receiveShadow = true;

group.add(cubes[j]);

collidableMeshList.push(cubes[j]);

}

//Cubes2

var texture3Box = new THREE.TextureLoader().load("texture/st.jpeg");

var cube2Geometry = new THREE.BoxGeometry(100, 350, 50);

var cube2Material = new THREE.MeshPhongMaterial({shininess: 100, map: textureBox});

cube2 = new THREE.Mesh(cube2Geometry, cube2Material);

cube2.position.x = -1500;

cube2.position.y = 360;

cube2.position.z =-30;

// Clone box into boxes array

Cubes2();

// Randomly placed the boxes in scene

for (var k = 0; k < cubes2.length; k++) {

//cubes[k].rotateX(Math.PI/2 );

cubes2[k].position.x =k\*400-1000;

cubes2[k].position.y =((Math.random()\*2)+3.5)\*100;

cubes2[k].position.z = -30;

cubes2[k].castShadow = true;

cubes2[k].receiveShadow = true;

group.add(cubes2[k]);

collidableMeshList.push(cubes2[k]);

}

در ابتدا ستون ها را تعریف می کنیم. ستون های 1، ستون هایی اند که در پایین قرار میگیرند و ستون های 2، ستون هایی هستند که در بالا قرار می گیرند.

مکان x,z آن بدین شرح است که z=-30 و x به فاصله ی 400 تا از هم قرار گرفته اند. مکان y آنها به صورت رندوم است که در ستون اول (Math.random()-1.5)\*100

رندوم بودن آن بدین گونه ودر ستون دوم رندم بودن آن به صورت ((Math.random()\*2)+3.5)\*100 است.

هر دو سری ستون سایه ی هواپیما را دریافت می کنند. روی هر دو سری ستون، بافت ایجاد شده است.

var objectLoader = new THREE.ObjectLoader();

objectLoader.load("model/plane-fight-threejs/plane-fight.json", function (obj) {

planeFight=obj;

planeFight.position.y = 60;

planeFight.position.z = -30;

planeFight.position.x = 7500;

planeFight.scale.set(4, 4, 4);

planeFight.add(directionalLight);

planeFight.castShadow = true;

planeFight.receiveShadow = true;

scene.add(planeFight);

planeFight.rotateY(3\*Math.PI/2);

});

هواپیما را که از سایت clara.io دریافت کرده ایم، در این قسمت لود می کنیم. هواپیما سایه ایجاد می کند و همچنین مکان هواپیما که (7500, 60, -30) می باشد.

texture2 = THREE.ImageUtils.loadTexture("texture/win.jpg");

texture2.magFilter = THREE.NearestFilter;

texture2.minFilter = THREE.NearestFilter;

var sphereGeometry = new THREE.CircleGeometry(100, 64, 64);

var sphereMaterial = new THREE.MeshPhongMaterial({

map: texture2,

shininess: 150,

specular: 0xffffff,

shading: THREE.SmoothShading

});

sphere = new THREE.Mesh(sphereGeometry, sphereMaterial);

sphere.position.set(-1500, 150, -30);

sphere.rotateY(Math.PI/2);

sphere.castShadow = true;

scene.add(sphere);

collidableMeshList1.push(sphere);

در اینجا نقطه ی هدف تعریف شده که هواپیما با برخورد به آن برنده می شود که مکان آن به صورت -1500, 150, -30)) است و جنس ماده آن phong است و بر روی آن بافت قرار داده شده است.

var listener = new THREE.AudioListener();

camera.add(listener);

audioLoader = new THREE.AudioLoader();

sound = new THREE.PositionalAudio(listener);

audioLoader.load('audio/audio.mp3', function (buffer) {

sound.setBuffer(buffer);

sound.setRefDistance(2000);

sound.play();

});

scene.add(sound);

این قسمت چگونگی لود کردن صدا را نشان می دهد. شنونده به دوربین اضافه شده است و صدا به صحنه اضافه شده است.

geometry = new THREE.Geometry();

var textureLoader = new THREE.TextureLoader();

sprite1 = textureLoader.load("texture/snow.png");

for (i = 0; i < 10000; i++) {

var vertex = new THREE.Vector3();

vertex.x = Math.random() \* 2000 - 1000;

vertex.y = Math.random() \* 2000 - 1000 ;

vertex.z = Math.random() \* 2000 - 1000 ;

geometry.vertices.push(vertex);

}

vertex.sortParticles = true;

parameters = [

// [[Color in HSL], texture, size

[[0.86, 0.04, 0.77], sprite1, 10]

];

for (i = 0; i < parameters.length; i++) {

color = parameters[i][0];

sprite = parameters[i][1];

size = parameters[i][2];

materials[i] = new THREE.PointsMaterial({

size: size,

map: sprite,

blending: THREE.AdditiveBlending,

depthTest: false,

transparent: true

});

materials[i].color.setHSL(color[0], color[1], color[2]);

particles = new THREE.Points(geometry, materials[i]);

particles.rotation.x = Math.random() \* 6;

particles.rotation.y = Math.random() \* 6;

particles.rotation.z = Math.random() \* 6;

scene.add(particles);

renderer.domElement.addEventListener('mouseMove', onDocumentMouseMove, false);

window.addEventListener('resize', onWindowResize, false);

}

}

این قسمت لود کردن سیستم ذره ای را شرح می دهد. در ابتدا یک بافت برف در آن لود شده و مقادیر X,Y,Z آن به صورت رندوم برای هر هزار دانه ی برف تعیین می شود.

سپس رنگ و سایز کلی برف دانه ها مشخص می شود.

function animate() {

var t = (Date.now() / 1000);

// render the scene

render();

requestAnimationFrame(animate);

camera.position.x-=3;

planeFight.position.x-=3;

MovingCube.position.x-=3;

update();

}

در تابعanimate(); دوربین 3 تا 3 تا در راستای منفی محور x جلو می رود.

function Cubes() {

for (var i = 0; i < 20; i++) {

cubes[i] = cube.clone();

}

}

function Cubes2() {

for (var j = 0; j < 20; j++) {

cubes2[j] = cube2.clone();

}

}

این توابع از ستونها 20 تا کپی می گیرند.

var cubeGeometry = new THREE.CubeGeometry(120,50,100,1,1,1);

var wireMaterial = new THREE.MeshBasicMaterial( { color: 0xbfc9ca, wireframe:true } );

MovingCube = new THREE.Mesh( cubeGeometry, wireMaterial );

MovingCube.position.set(7510, 72, -30);

scene.add( MovingCube );

تابع MovingCube() را برای در برگرفتن ابعاد دور هواپیما به کار می رود تا رئوس MovingCube را به عنوان برخورد در نظر بگیریم.

var originPoint = MovingCube.position.clone();

for (var vertexIndex = 0; vertexIndex < MovingCube.geometry.vertices.length; vertexIndex++)

{

var localVertex = MovingCube.geometry.vertices[vertexIndex].clone();

var globalVertex = localVertex.applyMatrix4( MovingCube.matrix );

var directionVector = globalVertex.sub( MovingCube.position );

var ray = new THREE.Raycaster( originPoint, directionVector.clone().normalize() );

var collisionResults = ray.intersectObjects( collidableMeshList );

var collisionResults1 = ray.intersectObjects( collidableMeshList1 );

if ( collisionResults.length > 0 && collisionResults[0].distance < directionVector.length() )

window.location.href = "gameover.html";

if ( collisionResults1.length > 0 && collisionResults1[0].distance < directionVector.length() )

window.location.href = "win.html";

}

این قسمت برخورد را ایجاد می کند. در آرایه ی collisionResults و collisionResults1 مختصات ستون ها و نقطه ی هدف ذخیره شده است. در این تابع، اگر مختصات ستون های ذخیره شده در آرایه collisionResults با مختصات MovingCube برابر باشد، صحنه ی GameOver ایجاد می شود.

همچنین اگر مختصات نقطه ی هدف ذخیره شده در آرایه 1collisionResults با مختصات MovingCube برابر باشد، صحنه ی Win ایجاد می شود.

**پایان**