

گروه مهندسی کامپیوتر دانشگاه علم و فرهنگ

موضوع :

پیاده سازی بازی Pacman

پیاده سازی توسط :

نرگس تبریزی

استاد راهنما :

استاد مینوفام

دی 98

* نحوه انجام بازی:

در این بازی شما می‌بایست برای خروج از صفحه بازی مسیری را انتخاب کنید که در آن هیولاهای سبز رنگ مقابل شما قرار نگیرند چرا که در صورت برخورد با آن‌ها صدمه می‌بینید.

* صحنه:

برای اینکه صحنه‌ای سه بعدی ایجاد شود از skybox استفاده شد.

همچنین زاویه دوربین را برای آن تنظیم شد.

**var** reflectionCube = **new *THREE***.CubeTextureLoader()  
 .setPath(**'texture/cube/skybox/'**)  
 .load([**'px.jpg'**, **'nx.jpg'**, **'py.jpg'**, **'ny.jpg'**, **'pz.jpg'**, **'nz.jpg'**]);  
reflectionCube.**format** = ***THREE***.RGBFormat;  
***scene***.**background** = reflectionCube;

* دوربین:

به جهت آنکه مدلسازی به دنیای واقعی نزدیکتر شود از دید پرسپکتیو (perspective) استفاده شده است.

امکان lookAt نیز عامل ایجاد نگاه دنیا به صحنه است، همانند عملکرد دوربین.

**var** SCREEN\_WIDTH = ***window***.**innerWidth**, SCREEN\_HEIGHT = ***window***.**innerHeight**;  
**var** VIEW\_ANGLE = 45, ASPECT = SCREEN\_WIDTH / SCREEN\_HEIGHT, NEAR = 0.1, FAR = 20000;  
***camera*** = **new *THREE***.PerspectiveCamera(VIEW\_ANGLE, ASPECT, NEAR, FAR);  
***scene***.add(***camera***);  
***camera***.**position**.set(-200, 200, 400);  
***camera***.**lookAt**(***scene***.**position**);

* محاسبه گر تصویر:

بازی در قالب WebGL روی مرورگر وب نمایش داده می‌شود.

همچنین antialias ، true در نظر گرفته شده تا بازی حالت واقع گرایانه تری به خود بگیرد.

همچنین برای دوربین OrbitControls فعال شده است و به آن این امکان رل می‌دهد که در حالی که جهت بالای دوربین در راستای مثبت محور y ثابت است، دور هدف گردش کند.

***renderer*** = **new *THREE***.WebGLRenderer({**antialias**: **true**});  
***renderer***.setClearColor(**new *THREE***.**Color**(0xfccccc));  
***renderer***.setSize(***window***.**innerWidth**, ***window***.**innerHeight**);  
***document***.**body**.appendChild(***renderer***.**domElement**);  
  
***controls*** = **new *THREE***.OrbitControls(***camera***, ***renderer***.**domElement**);

* صفحه بازی:

صفحه‌ای که بازی روی آن انجام می‌شود با هندسه صفحه مسطح

(Plane Geometry) و به صورت ماده ساده (پایه) و رنگ 0x961153 تعریف شده و به صحنه اضافه شده است.

**var** surfaceGeometry = **new *THREE***.PlaneGeometry(350, 250);  
**var** surfaceMaterial = **new *THREE***.MeshBasicMaterial({**color**:0x961153, **side**: ***THREE***.DoubleSide});  
**var** surfaceMesh = **new *THREE***.**Mesh**(surfaceGeometry, surfaceMaterial);  
surfaceMesh.**rotateX**(***Math***.**PI** / 2);  
surfaceMesh.**position**.**y** = -50;  
***scene***.add(surfaceMesh);

* موانع:

بیست و چهار عدد مانع به رنگ 0x37D9D9 با هندسه مکعب و به صورت ماده ساده (پایه) در صفحه بازی تعریف شده است و هر کدام به صورت جداگانه تعریف و به صحنه اضافه شدند و تنها در ابعاد و مکان تفاوت دارند.

**var** planeGeometry11 = **new *THREE***.CubeGeometry(3, 1, 60);  
**var** planeMaterial11 = **new *THREE***.MeshBasicMaterial({**color**: 0x37D9D9});  
**cube11** = **new *THREE***.**Mesh**(planeGeometry11, planeMaterial11);  
**cube11**.**position**.**x** = 175;  
**cube11**.**position**.**z** = 0;  
**cube11**.**position**.**y** = -49;  
***collidableMeshList***.push(**cube11**);  
  
***scene***.add(**cube11**);

* پرنده قرمز:

شیء که با آن بازی می‌کنیم.

با هندسه کره و ماده ساده (پایه) تعریف شده است.

یک فایل .png به عنوان بافت روی آن اعمال شده است.

و با صفحه کلید به چهار جهت حرکت می‌کند.

**var** textureSphere = **new *THREE***.TextureLoader().load(**'texture/bOne.png'**);  
**var** planeGeometry = **new *THREE***.SphereGeometry(10, 6, 12);  
**var** planeMaterial = **new *THREE***.MeshBasicMaterial({**color**: 0xFFFF00, **map**: textureSphere, **side**: ***THREE***.DoubleSide });  
Sphere = **new *THREE***.**Mesh**(planeGeometry, planeMaterial);  
**Sphere**.**position**.**x** = -185;  
**Sphere**.**position**.**z** = 55;  
**Sphere**.**position**.**y** = -40; ***scene***.add(**Sphere**);

* هیولاها:

اشیایی که نباید با آن‌ها برخورد کنیم!

پنج شیء که با هندسه کره و ماده ساده (پایه)، هر یک به شکل مجزا، تعریف شده و به صحنه اضافه شده اند.

یک فایل .jpg به عنوان بافت روی هر یک از آن‌ها اعمال شده است.

و برای هر یک از آن‌ها یک تابع tween تعریف شده است که به وسیله آن حرکت می‌کنند.

**var** textureSphere = **new *THREE***.TextureLoader().load(**'texture/E1.jpg'**);  
**var** sphereGeometry1 = **new *THREE***.SphereGeometry(12, 8, 14);  
**var** sphereMaterial1 = **new *THREE***.MeshBasicMaterial({**color** :0xfccccc, **map**: textureSphere, **side**: ***THREE***.DoubleSide });  
**sphere1** = **new *THREE***.**Mesh**(sphereGeometry1, sphereMaterial1);  
*// sphere1.position.x = -145;  
// sphere1.position.y = -38;  
// sphere1.position.z = -90;***sphere1**.**position**.set(-120, -38, -90);  
  
***collidableMeshList***.push(**sphere1**);  
  
***scene***.add(**sphere1**);  
  
**var** target = {**x**: 145, **y**: -38, **z**: -90};  
**var** tween = **new** TWEEN.Tween(**sphere1**.**position**).to(target, 7000);  
tween.**delay**(500);  
tween.easing(TWEEN.**Easing**.**Back**.Out);  
tween.start();

* حرکت پرنده قرمز:

همان طور که گفته شد پرنده قرمز می‌تواند به چهار جهت حرکت کند.

با کمک کیبورد، با d به راست، با a به چپ، با w به بالا و با s به پایین می‌رود.

**if** (***keyboardControler***.pressed(**"a"**))  
 Sphere.**position**.**x** -= 0.3;  
**if** (***keyboardControler***.pressed(**"d"**))  
 Sphere.**position**.**x** += 0.3;  
**if** (***keyboardControler***.pressed(**"w"**))  
 Sphere.**position**.**z** -= 0.3;  
**if** (***keyboardControler***.pressed(**"s"**))  
 Sphere.**position**.**z** += 0.3;

تکه کد بالا را در تابع update می‌نویسیم.

* برخورد با استفاده از collision detection :

در صورت برخورد پرنده قرمز با هیولاها و موانع پیامی ظاهر می‌شود.

**var** originPoint = Sphere.**position**.clone();  
  
*clearText*();  
  
**for** (**var** vertexIndex = 0; vertexIndex < Sphere.**geometry**.**vertices**.**length**; vertexIndex++) {  
 **var** localVertex = Sphere.**geometry**.**vertices**[vertexIndex].clone();  
  
  
 **var** globalVertex = localVertex.applyMatrix4(Sphere.**matrix**);  
  
  
 **var** directionVector = globalVertex.sub(Sphere.**position**);  
  
  
 **var** ray = **new *THREE***.Raycaster(originPoint, directionVector.clone().normalize());  
  
 **var** collisionResults = ray.intersectObjects(***collidableMeshList***);  
  
 **if** (collisionResults.**length** > 0 && collisionResults[0].**distance** < directionVector.**length**())  
 *appendText*(**" Oooops!!! "**);  
  
}