지진효과 예제 따라 하기

계정 비쥬얼 시뮬레이션 연구실

이번 예제에서는 ‘e’버튼을 누르면 지진이 일어나는 듯한 효과를 줄 것이다.   
이번 예제는 다음과 같은 순서로 진행할 것이다.

1. 기본적인 html페이지 구성하기.
2. 기본적인 Three.js 요소 설정하기.
3. 땅, 안개, 글자 생성하기.
4. 말이 달리게 하기.
5. 지진효과 부여하기.

**Step 1. 기본적인 html 페이지 구성하기**

|  |  |
| --- | --- |
| <code1> | |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29 | <!DOCTYPE html>  <html lang="en">  <head>      <title>three.js webgl - shadow map</title>      <meta charset="utf-8">      <meta name="viewport" content="width=device-width, user-scalable=no, minimum-scale=1.0, maximum-scale=1.0">      <style>          #info {              position: absolute;              top: 20px;              width: 100%;              text-align: center;              z-index: 100;              display:block;              font-size: 30px;          }      </style>  </head>  <body>  <div id="info">      Press KEY<br>      earthquake : e  </div>  <script src="./lib/three/build/three1.min.js"></script>  <script src="./lib/three/examples/js/controls/OrbitControls.js"></script>  <script src="./lib/three/examples/js/shaders/UnpackDepthRGBAShader.js"></script>  <script src="./PostProcessor.js"></script>  </body>  </html> |

<code1>과 같이 기본적인 html페이지를 구성했다. <div>태그를 사용해서 ‘e’버튼을 누르면 지진효과가 일어난다는 것을 안내하고, 이 안내문을 스타일을 결정하기 위해서 css를 추가했다.

24번째 줄에서부터 27번째 줄까지에는 이번 예제를 구현하는데 필요한 js파일들을 미리 연결시켜놨다.

**Step 2. 기본적인 Three.js의 요소 설정하기**

|  |  |
| --- | --- |
| <code2> | |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90  91  92  93 | <!DOCTYPE html>  <html lang="en">  <head>      <title>three.js webgl - shadow map</title>      <meta charset="utf-8">      <meta name="viewport" content="width=device-width, user-scalable=no, minimum-scale=1.0, maximum-scale=1.0">      <style>          #info {              position: absolute;              top: 20px;              width: 100%;              text-align: center;              z-index: 100;              display:block;              font-size: 30px;          }      </style>  </head>  <body>  <div id="info">      Press KEY<br>      earthquake : e  </div>  <script src="./lib/three/build/three1.min.js"></script>  <script src="./lib/three/examples/js/controls/OrbitControls.js"></script>  <script src="./lib/three/examples/js/shaders/UnpackDepthRGBAShader.js"></script>  <script src="./PostProcessor.js"></script>  <script>      var camera, controls, scene, renderer;      var container;        var quake = false;        var SHADOW\_MAP\_WIDTH = 2048, SHADOW\_MAP\_HEIGHT = 1024;        function init() {          container = document.createElement( 'div' );          document.body.appendChild( container );            camera = new THREE.PerspectiveCamera( 23, window.innerWidth / window.innerHeight, 5, 3000 );          camera.position.set( 600, -100, 1900 );            controls = new THREE.OrbitControls( camera );            scene = new THREE.Scene();            var ambient = new THREE.AmbientLight( 0x444444 );          scene.add( ambient );            var light = new THREE.SpotLight( 0xffffff, 1, 0, Math.PI/2, 1 );          light.position.set( 0, 1500, 1000 );          light.target.position.set( 0, 0, 0 );          light.castShadow = true;          light.shadow.camera.near = 700;          light.shadow.camera.far = camera.far;          light.shadow.camera.fov = 50;          light.shadow.bias = 0.0001;          light.shadow.mapSize.width = SHADOW\_MAP\_WIDTH;          light.shadow.mapSize.height = SHADOW\_MAP\_HEIGHT;            scene.add(light);            renderer = new THREE.WebGLRenderer( { antialias: false } );          renderer.setPixelRatio(window.devicePixelRatio);          renderer.setSize( window.innerWidth, window.innerHeight );          renderer.autoClear = false;            container.appendChild( renderer.domElement );            window.addEventListener( 'resize', onWindowResize, false );          window.addEventListener( 'keydown', onKeyDown, false );      }        function onKeyDown(e) {          if(e.keyCode == "69"){              if(quake){                  quake = false;              }              else {                  quake = true;              }          }      }        function onWindowResize() {          camera.aspect = window.innerWidth / window.innerHeight;          camera.updateProjectionMatrix();            renderer.setSize( window.innerWidth, window.innerHeight );      }  </script>  </body>  </html> |

기본적으로 사용해야 하는 요소들을 생성하고 설정했다. Camera, controls, scene, renderer, light가 그것이다. 특히 light의 자세한 설정들은 그림자효과를 자연스럽게 주기 위한 설정들이다. 그리고 창의 크기변화에 대응할 함수를 만들어 연결하고, 키보드의 버튼이 눌렸을 때 그것을 처리할 함수도 만들어 연결했다. <code2>의 73번째 줄에 “69”는 ‘e’버튼의 고유번호를 나타낸다. 이 함수에서의 동작은 ‘e’버튼으로 quake값의 Boolean값을 변경하는 것인데 Step5에서 quake값에 따라 지진효과를 부여할 것이기 때문에 이런 동작을 하도록 설정했다.

**Step 3. 땅, 안개, 글자 생성하기**

|  |  |
| --- | --- |
| <code3> | |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99  100  101  102  103  104  105  106  107  108  109  110  111  112  113  114  115  116  117  118  119  120  121  122  123  124  125  126  127  128  129  130  131  132  133  134  135  136  137  138  139  140  141  142  143  144  145  146  147  148  149  150  151  152  153  154  155  156  157  158  159  160  161  162  163  164  165  166  167  168  169  170  171  172  173  174  175 | <!DOCTYPE html>  <html lang="en">  <head>      <title>three.js webgl - shadow map</title>      <meta charset="utf-8">      <meta name="viewport" content="width=device-width, user-scalable=no, minimum-scale=1.0, maximum-scale=1.0">      <style>          #info {              position: absolute;              top: 20px;              width: 100%;              text-align: center;              z-index: 100;              display:block;              font-size: 30px;          }      </style>  </head>  <body>  <div id="info">      Press KEY<br>      earthquake : e  </div>  <script src="./lib/three/build/three1.min.js"></script>  <script src="./lib/three/examples/js/controls/OrbitControls.js"></script>  <script src="./lib/three/examples/js/shaders/UnpackDepthRGBAShader.js"></script>  <script src="./PostProcessor.js"></script>  <script>      var camera, controls, scene, renderer;      var container;        var clock = new THREE.Clock();        var quake = false;        var SHADOW\_MAP\_WIDTH = 2048, SHADOW\_MAP\_HEIGHT = 1024;      var FLOOR = -250;        init();      animate();        function init() {          container = document.createElement( 'div' );          document.body.appendChild( container );            camera = new THREE.PerspectiveCamera( 23, window.innerWidth / window.innerHeight, 5, 3000 );          camera.position.set( 600, -100, 1900 );            controls = new THREE.OrbitControls( camera );            scene = new THREE.Scene();          scene.fog = new THREE.Fog( 0x59472b, 1000, 3000 );            var ambient = new THREE.AmbientLight( 0x444444 );          scene.add( ambient );            var light = new THREE.SpotLight( 0xffffff, 1, 0, Math.PI/2, 1 );          light.position.set( 0, 1500, 1000 );          light.target.position.set( 0, 0, 0 );          light.castShadow = true;          light.shadow.camera.near = 700;          light.shadow.camera.far = camera.far;          light.shadow.camera.fov = 50;          light.shadow.bias = 0.0001;          light.shadow.mapSize.width = SHADOW\_MAP\_WIDTH;          light.shadow.mapSize.height = SHADOW\_MAP\_HEIGHT;            scene.add(light);            renderer = new THREE.WebGLRenderer( { antialias: false } );          renderer.setPixelRatio(window.devicePixelRatio);          renderer.setSize( window.innerWidth, window.innerHeight );          renderer.autoClear = false;          renderer.setClearColor( scene.fog.color );      renderer.shadowMap.enabled = true;          renderer.shadowMap.type = THREE.PCFSoftShadowMap;            container.appendChild( renderer.domElement );            createScene();            window.addEventListener( 'resize', onWindowResize, false );          window.addEventListener( 'keydown', onKeyDown, false );      }        function onKeyDown(e) {          if(e.keyCode == "69"){              if(quake){                  quake = false;              }              else {                  quake = true;              }          }      }        function onWindowResize() {          camera.aspect = window.innerWidth / window.innerHeight;          camera.updateProjectionMatrix();            renderer.setSize( window.innerWidth, window.innerHeight );      }        function createScene() {          var geometry = new THREE.PlaneBufferGeometry( 100, 100 );          var planeMaterial = new THREE.MeshPhongMaterial( { color: 0xffdd99 } );            var ground = new THREE.Mesh( geometry, planeMaterial );            ground.position.set( 0, FLOOR, 0 );          ground.rotation.x = - Math.PI / 2;          ground.scale.set( 100, 100, 100 );          ground.castShadow = false;          ground.receiveShadow = true;            scene.add( ground );            var mesh = new THREE.Mesh( new THREE.BoxGeometry( 1500, 220, 150 ), planeMaterial );          mesh.position.y = FLOOR - 50;          mesh.position.z = 20;          mesh.castShadow = true;          mesh.receiveShadow = true;          scene.add( mesh );            var mesh = new THREE.Mesh( new THREE.BoxGeometry( 1600, 170, 250 ), planeMaterial );          mesh.position.y = FLOOR - 50;          mesh.position.z = 20;          mesh.castShadow = true;          mesh.receiveShadow = true;          scene.add( mesh );            var loader = new THREE.FontLoader();          loader.load( 'lib/three/examples/fonts/helvetiker\_bold.typeface.js', function ( font ) {                var textGeo = new THREE.TextGeometry( "THREE.JS", {                  font: font,                  size: 200,                  height: 50,                  curveSegments: 12,                  bevelThickness: 2,                  bevelSize: 5,                  bevelEnabled: true              });                textGeo.computeBoundingBox();              var centerOffset = -0.5 \* ( textGeo.boundingBox.max.x - textGeo.boundingBox.min.x );                var textMaterial = new THREE.MeshPhongMaterial( { color: 0xff0000, specular: 0xffffff } );                var mesh = new THREE.Mesh( textGeo, textMaterial );              mesh.position.x = centerOffset;              mesh.position.y = FLOOR + 67;                mesh.castShadow = true;              mesh.receiveShadow = true;                scene.add( mesh );          } );      }        function animate() {          requestAnimationFrame( animate );          render();      }        function render() {          var delta = clock.getDelta();            controls.update( delta );          renderer.clear();          renderer.render( scene, camera );      }  </script>  </body>  </html> |

이번 단계에서는 땅과 안개, 글자를 생성할 것이다. 우선 <code3>의 37번째 줄을 보면 지면의 높이를 일치시키기 위한 변수FLOOR을 설정해 놓은 것을 볼 수 있다. 안개를 만들기 위해서 52번째 줄의 코드를 사용했다. 장면에 안개를 만드는 함수 fog를 사용했다. 각각의 매개변수는 안개의 색, 안개 효과가 나타나는 거리, 최대 가시거리를 나타낸다. 아래 <그림1>을 보면 먼 글자일수록 뿌옇게 보이는 것을 확인할 수 있다. 이 안개가 자연스럽게 보이게 하기 위해서 74번째 줄처럼 renderer의 배경색상도 안개의 색과 동일하게 했다.

createScene함수를 만들어서 지면과 글자를 만들었다. createScene함수는 init함수안에서 실행된다. 104번째 줄에 나와있는 createScene함수를 보면 처음으로 하고 있는 일은 지면을 만드는 것이다. 간단하게 PlaneBufferGeometry를 사용해서 ground를 만들었다. ground에 대한 설정은 110번째 줄에서 114번째 줄까지에 나와있다. 그 다음으로 글자를 받히는 box를 만들었다. Box를 겹치게 놓아서 마치 계단이 있는 것 같이 보일 수 있도록 크기와 위치를 설정했다. 글자를 만들기 위해서 FontLoader을 사용했다. 미리 정의되어 있는 js파일을 사용해서 만들고 135번째 줄에 나와있는 textGeo를 이용해서 상세한 설정을 한다. 만드는 글자의 위치를 지정하기 위해서 computeBoundingBox함수를 사용했다. 이 함수로 글자를 감싸는 box를 계산한 후 중심위치를 계산해서 만들어둔 단위에 올릴 수 있도록 위치를 조정한다.

만든 땅과 글자, 단을 확인하기 위해서 animate과 render함수를 만들었다. animate함수에서는 계속해서 animate함수를 부르고 동시에 render함수를 불러서 지속적으로 화면에 장면을 그려주게 한다. 실질적으로 화면에 장면을 그리는 것은 render함수에서 수행한다.

<code3>을 브라우저에서 실행시키면 아래의 <그림1>과 같은 모습을 확인할 수 있다. 안개와 글자, 단이 제대로 만들어진 것을 확인할 수 있다.

|  |
| --- |
|  |
| <그림1> |

**Step 4. 말이 달리게 하기**

|  |  |
| --- | --- |
| <code4> | |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99  100  101  102  103  104  105  106  107  108  109  110  111  112  113  114  115  116  117  118  119  120  121  122  123  124  125  126  127  128  129  130  131  132  133  134  135  136  137  138  139  140  141  142  143  144  145  146  147  148  149  150  151  152  153  154  155  156  157  158  159  160  161  162  163  164  165  166  167  168  169  170  171  172  173  174  175  176  177  178  179  180  181  182  183  184  185  186  187  188  189  190  191  192  193  194  195  196  197  198  199  200  201  202  203  204  205  206  207  208  209  210  211  212  213  214  215  216  217  218  219  220  221  222  223  224  225  226  227  228  229  230  231  232  233  234  235  236  237  238  239 | <!DOCTYPE html>  <html lang="en">  <head>      <title>three.js webgl - shadow map</title>      <meta charset="utf-8">      <meta name="viewport" content="width=device-width, user-scalable=no, minimum-scale=1.0, maximum-scale=1.0">      <style>          #info {              position: absolute;              top: 20px;              width: 100%;              text-align: center;              z-index: 100;              display:block;              font-size: 30px;          }      </style>  </head>  <body>  <div id="info">      Press KEY<br>      earthquake : e  </div>  <script src="./lib/three/build/three1.min.js"></script>  <script src="./lib/three/examples/js/controls/OrbitControls.js"></script>  <script src="./lib/three/examples/js/shaders/UnpackDepthRGBAShader.js"></script>  <script src="./PostProcessor.js"></script>  <script>      var camera, controls, scene, renderer;      var container;        var clock = new THREE.Clock();        var quake = false;      var morph, morphs = [], mixer, animGroups = [];        var SHADOW\_MAP\_WIDTH = 2048, SHADOW\_MAP\_HEIGHT = 1024;      var FLOOR = -250;      var ANIMATION\_GROUPS = 25;        init();      animate();        function init() {          container = document.createElement( 'div' );          document.body.appendChild( container );            camera = new THREE.PerspectiveCamera( 23, window.innerWidth / window.innerHeight, 5, 3000 );          camera.position.set( 600, -100, 1900 );            controls = new THREE.OrbitControls( camera );            scene = new THREE.Scene();          scene.fog = new THREE.Fog( 0x59472b, 1000, 3000 );            var ambient = new THREE.AmbientLight( 0x444444 );          scene.add( ambient );            var light = new THREE.SpotLight( 0xffffff, 1, 0, Math.PI/2, 1 );          light.position.set( 0, 1500, 1000 );          light.target.position.set( 0, 0, 0 );          light.castShadow = true;          light.shadow.camera.near = 700;          light.shadow.camera.far = camera.far;          light.shadow.camera.fov = 50;          light.shadow.bias = 0.0001;          light.shadow.mapSize.width = SHADOW\_MAP\_WIDTH;          light.shadow.mapSize.height = SHADOW\_MAP\_HEIGHT;            scene.add(light);            renderer = new THREE.WebGLRenderer( { antialias: false } );          renderer.setPixelRatio(window.devicePixelRatio);          renderer.setSize( window.innerWidth, window.innerHeight );          renderer.autoClear = false;          renderer.setClearColor( scene.fog.color );          renderer.shadowMap.enabled = true;          renderer.shadowMap.type = THREE.PCFSoftShadowMap;            container.appendChild( renderer.domElement );            createScene();            window.addEventListener( 'resize', onWindowResize, false );          window.addEventListener( 'keydown', onKeyDown, false );      }        function onKeyDown(e) {          if(e.keyCode == "69"){              if(quake){                  quake = false;              }              else {                  quake = true;              }          }      }        function onWindowResize() {          camera.aspect = window.innerWidth / window.innerHeight;          camera.updateProjectionMatrix();            renderer.setSize( window.innerWidth, window.innerHeight );      }        function createScene() {          var geometry = new THREE.PlaneBufferGeometry( 100, 100 );          var planeMaterial = new THREE.MeshPhongMaterial( { color: 0xffdd99 } );            var ground = new THREE.Mesh( geometry, planeMaterial );            ground.position.set( 0, FLOOR, 0 );          ground.rotation.x = - Math.PI / 2;          ground.scale.set( 100, 100, 100 );          ground.castShadow = false;          ground.receiveShadow = true;            scene.add( ground );            var mesh = new THREE.Mesh( new THREE.BoxGeometry( 1500, 220, 150 ), planeMaterial );          mesh.position.y = FLOOR - 50;          mesh.position.z = 20;          mesh.castShadow = true;          mesh.receiveShadow = true;          scene.add( mesh );            var mesh = new THREE.Mesh( new THREE.BoxGeometry( 1600, 170, 250 ), planeMaterial );          mesh.position.y = FLOOR - 50;          mesh.position.z = 20;          mesh.castShadow = true;          mesh.receiveShadow = true;          scene.add( mesh );            var loader = new THREE.FontLoader();          loader.load( 'lib/three/examples/fonts/helvetiker\_bold.typeface.js', function ( font ) {                var textGeo = new THREE.TextGeometry( "THREE.JS", {                  font: font,                  size: 200,                  height: 50,                  curveSegments: 12,                  bevelThickness: 2,                  bevelSize: 5,                  bevelEnabled: true              });                textGeo.computeBoundingBox();              var centerOffset = -0.5 \* ( textGeo.boundingBox.max.x - textGeo.boundingBox.min.x );                var textMaterial = new THREE.MeshPhongMaterial( { color: 0xff0000, specular: 0xffffff } );                var mesh = new THREE.Mesh( textGeo, textMaterial );              mesh.position.x = centerOffset;              mesh.position.y = FLOOR + 67;                mesh.castShadow = true;              mesh.receiveShadow = true;                scene.add( mesh );          } );            mixer = new THREE.AnimationMixer( scene );            for ( var i = 0; i !== ANIMATION\_GROUPS; ++ i ) {              animGroups.push( new THREE.AnimationObjectGroup() );          }            var loader = new THREE.JSONLoader();            loader.load( "lib/three/examples/models/animated/horse.js", function( geometry ) {              for ( var i = - 600; i < 601; i += 2 ) {                  addMorph( geometry, 550, 1, 100 - Math.random() \* 3000, FLOOR, i, true, true );              }          } );      }        function addMorph( geometry, speed, duration, x, y, z, fudgeColor, massOptimization ) {            var material = new THREE.MeshLambertMaterial( { color: 0xffaa55, morphTargets: true, vertexColors: THREE.FaceColors } );            if ( fudgeColor ) {              material.color.offsetHSL( 0, Math.random() \* 0.5 - 0.25, Math.random() \* 0.5 - 0.25 );          }            var mesh = new THREE.Mesh(geometry, material);          mesh.speed = speed;            var clip = geometry.animations[0];            if ( massOptimization ) {              var index = Math.floor( Math.random() \* ANIMATION\_GROUPS);                var animGroup = animGroups[ index ];              animGroup.add( mesh );                if ( ! mixer.existingAction( clip, animGroup ) ) {                  var randomness = 0.6 \* Math.random() - 0.3;                  var phase = ( index + randomness ) / ANIMATION\_GROUPS;                    mixer.clipAction(clip, animGroup).setDuration(duration).startAt(-duration \* phase).play();              }          }          else {              mixer.clipAction(clip, mesh).setDuration(duration).startAt(-duration \* Math.random()).play();          }            mesh.position.set( x, y, z );          mesh.rotation.y = Math.PI/2;          mesh.castShadow = true;          mesh.receiveShadow = true;            scene.add( mesh );          morphs.push( mesh );      }        function animate() {          requestAnimationFrame( animate );          render();      }        function render() {          var delta = clock.getDelta();          if ( mixer ) mixer.update( delta );            for ( var i = 0; i < morphs.length; i ++ ) {              morph = morphs[ i ];              morph.position.x += morph.speed \* delta;              if ( morph.position.x  > 2000 )  {                  morph.position.x = -1000 - Math.random() \* 500;              }          }            controls.update( delta );          renderer.clear();          renderer.render( scene, camera );      }  </script>  </body>  </html> |

createScene함수에 코드를 추가해서 말을 생성할 것이다. <code4>의 162번째 줄에서부터 174번째 줄까지가 createScene함수에서 추가된 코드이다. 그런데 172번째 줄에 addMorph함수를 사용하기 위해서 함수를 정의해야 한다. createScene함수 바로 아래에 addMorph함수가 정의되어 있다. 이 함수를 통해서 말을 추가하고, 말의 위치를 결정하고, 말이 달리는 애니메이션을 실행시킨다. 말을 가져오기 위해서 미리 정의되어 있는 horse.js를 사용했다.

말을 추가했다면 시간의 변화에 따라서 말의 상태를 업데이트 해줘야 한다. 그 부분은 render함수에 추가되어 있다. 223번째 줄에서 각각의 말에 대한 애니메이션을 업데이트하고 225번째 줄에서 각각의 말에 대한 위치정보를 업데이트한다. 위치를 업데이트함으로써 말이 달려서 앞으로 가고 있는 모습을 연출할 수 있다.

<code4>를 브라우저에서 실행시켜보면 <그림2>와 같이 많은 말들이 달려가는 모습을 확인할 수 있을 것이다.

|  |
| --- |
|  |
| <그림 2> |

**Step 5. 지진효과 부여하기**

|  |  |
| --- | --- |
| <code5> | |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99  100  101  102  103  104  105  106  107  108  109  110  111  112  113  114  115  116  117  118  119  120  121  122  123  124  125  126  127  128  129  130  131  132  133  134  135  136  137  138  139  140  141  142  143  144  145  146  147  148  149  150  151  152  153  154  155  156  157  158  159  160  161  162  163  164  165  166  167  168  169  170  171  172  173  174  175  176  177  178  179  180  181  182  183  184  185  186  187  188  189  190  191  192  193  194  195  196  197  198  199  200  201  202  203  204  205  206  207  208  209  210  211  212  213  214  215  216  217  218  219  220  221  222  223  224  225  226  227  228  229  230  231  232  233  234  235  236  237  238  239  240  241  242  243  244  245 | <!DOCTYPE html>  <html lang="en">  <head>      <title>three.js webgl - shadow map</title>      <meta charset="utf-8">      <meta name="viewport" content="width=device-width, user-scalable=no, minimum-scale=1.0, maximum-scale=1.0">      <style>          #info {              position: absolute;              top: 20px;              width: 100%;              text-align: center;              z-index: 100;              display:block;              font-size: 30px;          }      </style>  </head>  <body>  <div id="info">      Press KEY<br>      earthquake : e  </div>  <script src="./lib/three/build/three1.min.js"></script>  <script src="./lib/three/examples/js/controls/OrbitControls.js"></script>  <script src="./lib/three/examples/js/shaders/UnpackDepthRGBAShader.js"></script>  <script src="./PostProcessor.js"></script>  <script>      var camera, controls, scene, renderer;      var container;        var clock = new THREE.Clock();        var quake = false, postProcessorSystem;      var morph, morphs = [], mixer, animGroups = [];        var SHADOW\_MAP\_WIDTH = 2048, SHADOW\_MAP\_HEIGHT = 1024;      var FLOOR = -250;      var ANIMATION\_GROUPS = 25;        init();      animate();        function init() {          container = document.createElement( 'div' );          document.body.appendChild( container );            camera = new THREE.PerspectiveCamera( 23, window.innerWidth / window.innerHeight, 5, 3000 );          camera.position.set( 600, -100, 1900 );            controls = new THREE.OrbitControls( camera );            scene = new THREE.Scene();          scene.fog = new THREE.Fog( 0x59472b, 1000, 3000 );            var ambient = new THREE.AmbientLight( 0x444444 );          scene.add( ambient );            var light = new THREE.SpotLight( 0xffffff, 1, 0, Math.PI/2, 1 );          light.position.set( 0, 1500, 1000 );          light.target.position.set( 0, 0, 0 );          light.castShadow = true;          light.shadow.camera.near = 700;          light.shadow.camera.far = camera.far;          light.shadow.camera.fov = 50;          light.shadow.bias = 0.0001;          light.shadow.mapSize.width = SHADOW\_MAP\_WIDTH;          light.shadow.mapSize.height = SHADOW\_MAP\_HEIGHT;            scene.add(light);            renderer = new THREE.WebGLRenderer( { antialias: false } );          renderer.setPixelRatio(window.devicePixelRatio);          renderer.setSize( window.innerWidth, window.innerHeight );          renderer.autoClear = false;          renderer.setClearColor( scene.fog.color );          renderer.shadowMap.enabled = true;          renderer.shadowMap.type = THREE.PCFSoftShadowMap;            container.appendChild( renderer.domElement );            createScene();            postProcessorSystem = new PostProcessor();            window.addEventListener( 'resize', onWindowResize, false );          window.addEventListener( 'keydown', onKeyDown, false );      }        function onKeyDown(e) {          if(e.keyCode == "69"){              if(quake){                  quake = false;              }              else {                  quake = true;              }          }      }        function onWindowResize() {          camera.aspect = window.innerWidth / window.innerHeight;          camera.updateProjectionMatrix();            renderer.setSize( window.innerWidth, window.innerHeight );      }        function createScene() {          var geometry = new THREE.PlaneBufferGeometry( 100, 100 );          var planeMaterial = new THREE.MeshPhongMaterial( { color: 0xffdd99 } );            var ground = new THREE.Mesh( geometry, planeMaterial );            ground.position.set( 0, FLOOR, 0 );          ground.rotation.x = - Math.PI / 2;          ground.scale.set( 100, 100, 100 );          ground.castShadow = false;          ground.receiveShadow = true;            scene.add( ground );            var mesh = new THREE.Mesh( new THREE.BoxGeometry( 1500, 220, 150 ), planeMaterial );          mesh.position.y = FLOOR - 50;          mesh.position.z = 20;          mesh.castShadow = true;          mesh.receiveShadow = true;          scene.add( mesh );            var mesh = new THREE.Mesh( new THREE.BoxGeometry( 1600, 170, 250 ), planeMaterial );          mesh.position.y = FLOOR - 50;          mesh.position.z = 20;          mesh.castShadow = true;          mesh.receiveShadow = true;          scene.add( mesh );            var loader = new THREE.FontLoader();          loader.load( 'lib/three/examples/fonts/helvetiker\_bold.typeface.js', function ( font ) {                var textGeo = new THREE.TextGeometry( "THREE.JS", {                  font: font,                  size: 200,                  height: 50,                  curveSegments: 12,                  bevelThickness: 2,                  bevelSize: 5,                  bevelEnabled: true              });                textGeo.computeBoundingBox();              var centerOffset = -0.5 \* ( textGeo.boundingBox.max.x - textGeo.boundingBox.min.x );                var textMaterial = new THREE.MeshPhongMaterial( { color: 0xff0000, specular: 0xffffff } );                var mesh = new THREE.Mesh( textGeo, textMaterial );              mesh.position.x = centerOffset;              mesh.position.y = FLOOR + 67;                mesh.castShadow = true;              mesh.receiveShadow = true;                scene.add( mesh );          } );            mixer = new THREE.AnimationMixer( scene );            for ( var i = 0; i !== ANIMATION\_GROUPS; ++ i ) {              animGroups.push( new THREE.AnimationObjectGroup() );          }            var loader = new THREE.JSONLoader();            loader.load( "lib/three/examples/models/animated/horse.js", function( geometry ) {              for ( var i = - 600; i < 601; i += 2 ) {                  addMorph( geometry, 550, 1, 100 - Math.random() \* 3000, FLOOR, i, true, true );              }          } );      }        function addMorph( geometry, speed, duration, x, y, z, fudgeColor, massOptimization ) {            var material = new THREE.MeshLambertMaterial( { color: 0xffaa55, morphTargets: true, vertexColors: THREE.FaceColors } );            if ( fudgeColor ) {              material.color.offsetHSL( 0, Math.random() \* 0.5 - 0.25, Math.random() \* 0.5 - 0.25 );          }            var mesh = new THREE.Mesh(geometry, material);          mesh.speed = speed;            var clip = geometry.animations[0];            if ( massOptimization ) {              var index = Math.floor( Math.random() \* ANIMATION\_GROUPS);                var animGroup = animGroups[ index ];              animGroup.add( mesh );                if ( ! mixer.existingAction( clip, animGroup ) ) {                  var randomness = 0.6 \* Math.random() - 0.3;                  var phase = ( index + randomness ) / ANIMATION\_GROUPS;                    mixer.clipAction(clip, animGroup).setDuration(duration).startAt(-duration \* phase).play();              }          }          else {              mixer.clipAction(clip, mesh).setDuration(duration).startAt(-duration \* Math.random()).play();          }            mesh.position.set( x, y, z );          mesh.rotation.y = Math.PI/2;          mesh.castShadow = true;          mesh.receiveShadow = true;            scene.add( mesh );          morphs.push( mesh );      }        function animate() {          requestAnimationFrame( animate );          render();      }        function render() {          var delta = clock.getDelta();          if ( mixer ) mixer.update( delta );            for ( var i = 0; i < morphs.length; i ++ ) {              morph = morphs[ i ];              morph.position.x += morph.speed \* delta;              if ( morph.position.x  > 2000 )  {                  morph.position.x = -1000 - Math.random() \* 500;              }          }            if(quake){              camera = postProcessorSystem.earthquake(camera, 50, 0, 10);          }            controls.update( delta );          renderer.clear();          renderer.render( scene, camera );      }  </script>  </body>  </html> |

간단하게 지진효과를 부여할 수 있다. 지진효과를 부여하기 위해서 postProcessorSystem을 선언했다. 그리고 PostProcessor클래스의 객체를 만들었다. 이 객체를 이용해서 지진효과를 부여할 것이다. <code5>의 render함수를 보면 235번째 줄에 quake변수가 ture일 때 postProcssorSystem을 이용해서 지진 효과를 주고 있다. earthquake라는 함수를 사용하는데 이 함수의 매개변수는 camera, x축의 흔들림 정도, y축의 흔들림 정도, z축의 흔들림 정도를 의미한다. 이 값들을 더 크게 하면 더 격렬한 지진이 일어나는 것을 볼 수 있다.