이 게시물은 peter4k님의 블로그에 2016-01-29 오후 3:05:34에 게시되었습니다.

불타는 말 예제 따라 하기

계정 비쥬얼 시뮬레이션 연구실

밤하늘에 불꽃놀이를 만드는 것이 이번 예제의 목표이다.

이번 예제는 다음과 같은 순서로 진행할 것이다.

1. 필요한 js파일 연결하기.
2. Three.js의 기본적인 요소들 만들기.
3. 달리는 말을 불러오기.
4. 파티클 시스템을 설정하고 추가하기.

**Step 1. 필요한 js파일 연결하기**

|  |  |
| --- | --- |
| <code1> | |
| 1  2  3  4  5  6  7  8  9  10  11  12  13 | <!DOCTYPE html>  <html lang="en">  <head>      <meta charset="UTF-8">      <title>fire horse</title>  </head>  <body>  <script src="./lib/three/build/three.min.js"></script>  <script src="./ParticleSystem.js"></script>  <script src="./lib/three/examples/js/controls/OrbitControls.js"></script>    </body>  </html> |

<code1>의 8번째 줄에서 10번째 줄에 나오는 3개의 js파일이 이번 예제에서 사용된다.

**Step 2. Three.js의 기본적인 요소들 만들기**

|  |  |
| --- | --- |
| <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 | <!DOCTYPE html>  <html lang="en">  <head>      <meta charset="UTF-8">      <title>fire horse</title>  </head>  <body>  <script src="./lib/three/build/three.min.js"></script>  <script src="./ParticleSystem.js"></script>  <script src="./lib/three/examples/js/controls/OrbitControls.js"></script>  <script>      var container;      var camera, scene, renderer;      var controls;        function init() {          container = document.createElement( 'div' );          document.body.appendChild( container );            camera = new THREE.PerspectiveCamera( 50, window.innerWidth / window.innerHeight, 1, 10000 );          camera.setLens(18);          camera.position.y = 300;          camera.position.x = 350;          camera.position.z = 350;            scene = new THREE.Scene();            var light = new THREE.DirectionalLight( 0xefefff, 2 );          light.position.set( 1, 1, 1 ).normalize();          scene.add( light );            var light = new THREE.DirectionalLight( 0xffefef, 2 );          light.position.set( -1, -1, -1 ).normalize();          scene.add( light );            renderer = new THREE.WebGLRenderer();          renderer.setClearColor( 0x000000 );          renderer.setSize( window.innerWidth, window.innerHeight );          container.appendChild(renderer.domElement);            controls = new THREE.OrbitControls(camera, renderer.domElement);            window.addEventListener( 'resize', onWindowResize, false );      }        function onWindowResize() {          camera.aspect = window.innerWidth / window.innerHeight;          camera.updateProjectionMatrix();            renderer.setSize( window.innerWidth, window.innerHeight );      }  </script>  </body>  </html> |

이전 예제와 동일하게 Three.js의 기본적인 요소들을 만든다. 여기서 추가되는 것은 light이다. 이전에는 사용하지 않았던 light를 쓰는 이유는 달리는 말을 불러왔을 때 빛이 없으면 말이 보이지 않기 때문이다.

<code2>의 28번째줄에서 34번째 줄에 나온 것처럼 빛을 만들고 위치를 조정한 다음 장면에 추가한다.

**Step 3. 달리는 말을 불러오기**

|  |  |
| --- | --- |
| <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 | <!DOCTYPE html>  <html lang="en">  <head>      <meta charset="UTF-8">      <title>fire horse</title>  </head>  <body>  <script src="./lib/three/build/three.min.js"></script>  <script src="./ParticleSystem.js"></script>  <script src="./lib/three/examples/js/controls/OrbitControls.js"></script>  <script>      var container;      var camera, scene, renderer;      var controls;        var mesh, animation;          init();      loop();        function init() {          container = document.createElement( 'div' );          document.body.appendChild( container );            camera = new THREE.PerspectiveCamera( 50, window.innerWidth / window.innerHeight, 1, 10000 );          camera.setLens(18);          camera.position.y = 300;          camera.position.x = 350;          camera.position.z = 350;            scene = new THREE.Scene();            var light = new THREE.DirectionalLight( 0xefefff, 2 );          light.position.set( 1, 1, 1 ).normalize();          scene.add( light );            var light = new THREE.DirectionalLight( 0xffefef, 2 );          light.position.set( -1, -1, -1 ).normalize();          scene.add( light );            var loader = new THREE.JSONLoader( true );          loader.load( "./models/horse.json", function( geometry ) {                mesh = new THREE.Mesh( geometry, new THREE.MeshLambertMaterial( { color: 0x606060, morphTargets: true } ) );              mesh.scale.set( 1.5, 1.5, 1.5 );              mesh.position.set(0,-100,0);                scene.add( mesh );                animation = new THREE.MorphAnimation( mesh );              animation.play();                mesh.geometry.computeMorphNormals();          } );            renderer = new THREE.WebGLRenderer();          renderer.setClearColor( 0x000000 );          renderer.setSize( window.innerWidth, window.innerHeight );          container.appendChild(renderer.domElement);            controls = new THREE.OrbitControls(camera, renderer.domElement);            window.addEventListener( 'resize', onWindowResize, false );      }        function onWindowResize() {          camera.aspect = window.innerWidth / window.innerHeight;          camera.updateProjectionMatrix();            renderer.setSize( window.innerWidth, window.innerHeight );      }        var fps = 40;      function loop() {          setTimeout(loop, 1000/fps);          controls.update();          requestAnimationFrame(render);      }        var prevTime = Date.now();        function render() {          if ( animation ) {              var time = Date.now();                animation.update( time - prevTime );              prevTime = time;          }            renderer.render( scene, camera );          controls.update();      }  </script>  </body>  </html> |

<code3>의 42번째줄에서 55번째 줄에 나온 것처럼 미리 구현되어 있는 달리는 말의 모델을 JSONLoader을 통해서 가져온다. 일반적인 Mesh와 동일하게 Material을 통해서 색을 지정할 수 있다. 그리고 mesh.scale.set을 통해서 1.5배 확대한 모습으로 말을 볼 것이다. 또 mesh의 위치를 mesh.position.set을 사용해서 (0,-100, 0)으로 결정했다.

말이 달리는 모습을 보기 위해서 <code3>의 51번째 줄에 나온 것처럼 animation을 만들었다. 이 animation은 미리 정의되어있는 horse.json에서 가져온 것이다. <code3>의 52번째 줄에 나온 것처럼 play를 시키고 <code3>의 84번째 줄에서 89번째 줄에서 보이는 것처럼 render함수에서 시간의 변화에 따라 animation.update를 시키면 말이 달리는 애니메이션을 볼 수 있다.

<code3>을 브라우저에서 실행 시키면 아래의 <그림1>과 같은 달리는 말을 볼 수 있을 것이다.

|  |
| --- |
| 시스템 생성 대체 텍스트: fire horse  C localhost:63342/Particles/fire_horse2.html  NAVE-R n Daum RI. |
| <그림1> |

**Step 4. 파티클 시스템을 설정하고 추가하기**

|  |  |
| --- | --- |
| <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 | <!DOCTYPE html>  <html lang="en">  <head>      <meta charset="UTF-8">      <title>fire horse</title>  </head>  <body>  <script src="./lib/three/build/three.min.js"></script>  <script src="./ParticleSystem.js"></script>  <script src="./lib/three/examples/js/controls/OrbitControls.js"></script>  <script>      var container;      var camera, scene, renderer;      var controls;        var mesh, animation;        var particleSystem = new ParticleSystem();      particleSystem.initialize(5000);        particleSystem.setParameters({          seedVelDir: new THREE.Vector3(0,1,-1),          seedVelMag: 100.0,          globalForce: new THREE.Vector3(0, 10, -1000),          windStrength: 10,          seedSize: 300,          seedLife: 2.0,          alpha: 0.7,          texFile : "./textures/flame.png"      });        init();      loop();        function init() {          container = document.createElement( 'div' );          document.body.appendChild( container );            camera = new THREE.PerspectiveCamera( 50, window.innerWidth / window.innerHeight, 1, 10000 );          camera.setLens(18);          camera.position.y = 300;          camera.position.x = 350;          camera.position.z = 350;            scene = new THREE.Scene();            var light = new THREE.DirectionalLight( 0xefefff, 2 );          light.position.set( 1, 1, 1 ).normalize();          scene.add( light );            var light = new THREE.DirectionalLight( 0xffefef, 2 );          light.position.set( -1, -1, -1 ).normalize();          scene.add( light );            var loader = new THREE.JSONLoader( true );          loader.load( "./models/horse.json", function( geometry ) {                mesh = new THREE.Mesh( geometry, new THREE.MeshLambertMaterial( { color: 0x606060, morphTargets: true } ) );              mesh.scale.set( 1.5, 1.5, 1.5 );              mesh.position.set(0,-100,0);                scene.add( mesh );                animation = new THREE.MorphAnimation( mesh );              animation.play();                mesh.geometry.computeMorphNormals();          } );            renderer = new THREE.WebGLRenderer();          renderer.setClearColor( 0x000000 );          renderer.setSize( window.innerWidth, window.innerHeight );          container.appendChild(renderer.domElement);            scene.add(particleSystem.getMesh());            controls = new THREE.OrbitControls(camera, renderer.domElement);            window.addEventListener( 'resize', onWindowResize, false );      }        function onWindowResize() {          camera.aspect = window.innerWidth / window.innerHeight;          camera.updateProjectionMatrix();            renderer.setSize( window.innerWidth, window.innerHeight );      }        var fps = 40;      var time;      function loop() {          setTimeout(loop, 1000/fps);            var now = new Date().getTime();          var dt = (now - (time||now))/1000;          time = now;            if(animation != undefined ) {              var frame = animation.currentFrame;              var tar = mesh.geometry.morphTargets[frame];                for(var i=0; i<tar.vertices.length; i++) {                    if(Math.random() < 0.1) {                        var v = new THREE.Vector3();                      v = tar.vertices[i].clone();                      v = v.multiplyScalar(1.5);                      v.y -= 100;                        if(v.y < 20) continue;                        particleSystem.addParticlesFromSphere(1, v, 0);                  }              }          }          particleSystem.updateParticles(dt);            controls.update();          requestAnimationFrame(render);      }        var prevTime = Date.now();        function render() {          if ( animation ) {              var time = Date.now();                animation.update( time - prevTime );              prevTime = time;          }            renderer.render( scene, camera );          controls.update();      }  </script>  </body>  </html> |

이전 예제에서 파티클 시스템을 사용했던 것처럼 <code4>의 18번째 줄에서부터 30번째 줄까지 파티클 시스템을 생성하고 속성을 정의했다. Init함수 안에서 장면에 만든 파티클 시스템을 추가했다.

이전 예제들과는 다르게 파티클 시스템을 생성하는 addParticlesFromSphere함수를 사용할 때 일정한 좌표를 사용하지 않았다. 대신 <code4>의 98번째 줄에서부터 116번째 줄에 나온 것처럼 mesh의 vertices값을 사용해서 말의 표면에서 불을 나타내는 파티클을 생성했다. 특히 104번째 줄의 random함수를 사용해서 불규칙적으로 불이 생성되도록 했고, 111번째 줄의 (v.y < 20) 이라는 조건을 사용해서 말의 다리부분에서는 불이 생성되지 않고 말의 몸통과 머리부분에서만 불이 생성되도록 했다.

말을 불러올 때 1.5배 크게 하고 말의 위치의 y좌표를 -100으로 설정했기 때문에 108번째 줄에서 v.multiplyScalar(1.5)를 했고, 109번째 줄처럼 v.y -=100을 했다.

<code4>와 같이 작성한 html파일을 브라우저에서 실행하면 아래의 <그림2>와 같은 불타는 말이 달리는 모습을 볼 수 있다.

|  |
| --- |
| 시스템 생성 대체 텍스트: fire horse  C localhost:63342/Particles/fire_horse2.html  NAVE-R n Daum RI. |
| <그림2> |