이 게시물은 peter4k님의 블로그에 2016-01-29 오후 2:02:12에 게시되었습니다.

눈 내리는 초원 예제 따라 하기

계정 비쥬얼 시뮬레이션 연구실

초원에 눈이 내리는 환경을 만드는 것이 이번 예제의 목표이다.

이번 예제는 다음과 같은 순서로 진행할 것이다.

1. 필요한 js파일 연결하기.
2. Three.js의 기본적인 요소들 만들기.
3. 초원 지형 만들기.
4. 만들어진 초원 지형을 화면에서 확인하기.
5. 눈을 표현할 파티클 시스템 설정하기.
6. 눈을 초원 지형에 추가하기.
7. 눈이 쌓이는 효과주기.

**Step 1. 필요한 js파일 연결하기**

|  |  |
| --- | --- |
| <code1> | |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15 | <!DOCTYPE html>  <html lang="en">  <head>      <meta charset="UTF-8">      <title>falling snow</title>  </head>  <body>    <script src="./lib/three/build/three.min.js"></script>  <script src="./lib/three/examples/js/controls/OrbitControls.js"></script>  <script src="./ParticleSystem.js"></script>  <script src="./paticleParameter.js"></script>  <script src="./AdditiveTerrain.js"></script>  </body>  </html> |

초원에 눈을 내리게 하기 위해서 필요한 js파일은 <코드 1>의 9째 줄에서 13번째 줄에 나타나 있는 5가지 이다.

이번에 사용할 다양한 함수들은 다섯 개의 js파일에서 정의하고 있기 때문에 스크립트 태그로 연결을 해 주는 것 만으로 더 쉽게 코드를 작성할 수 있게 한다.

**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 | <!DOCTYPE html>  <html lang="en">  <head>      <meta charset="UTF-8">      <title>falling snow</title>  </head>  <body>    <script src="./lib/three/build/three.min.js"></script>  <script src="./lib/three/examples/js/controls/OrbitControls.js"></script>  <script src="./ParticleSystem.js"></script>  <script src="./paticleParameter.js"></script>  <script src="./AdditiveTerrain.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( 60, window.innerWidth / window.innerHeight, 0.01, 10000 );          camera.setLens(18);          camera.position.y = 200;          camera.position.z = 400;            scene = new THREE.Scene();            renderer = new THREE.WebGLRenderer( { antialias: true } );          renderer.setClearColor( 0xaaaaaa );          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> |

<코드 2>의 15번째 줄에서 17번째 줄을 보면 스크립트 태그 안에서 전역 변수를 선언하고 있는 것을 확인할 수 있다. 이 변수들은 다양한 함수 안에서 접근할 것이기 때문에 전역변수로 선언했다.

선언한 전역변수들은 <코드 2>의 19번째 줄에서 38번째 줄에 걸쳐서 나타나는 init함수로 초기화를 한다. 새로운 객체를 만들고 설정하는 일련의 작업이 init함수에서 하는 일이다.

init함수에서 브라우저 창의 크기변화에 대응할 수 있는 이벤트 리스너를 설정한다. <코드 2>의 37번째 줄을 보면 브라우저 창의 크기변화가 일어날 때 마다 onWindowResize함수를 호출하도록 되어있는 것을 확인할 수 있다. 이에 따른 onWindowResize함수는 <코드 2>의 40번째 줄에서부터 45번째 줄에 나타나 있다. 이 함수에서는 camera의 aspect를 변경하고 renderer의 size를 변경하는 일을 하게 된다.

15 ~ 17 : container, camera, scene, renderer, controls를 전역변수로 선언했다.

20 ~ 35 : 전역변수로 선언한 요소들을 init함수에서 초기화한다.

37 : 브라우저 창의 크기변화에 대응할 함수, onWindowResize를 이벤트에 연결시킨다.

40 ~ 45 : 브라우저 창의 크기변화에 따라 camera의 aspect와 renderer의 size를 변화시킨다.

**Step 3. 초원 지형 만들기**

|  |  |
| --- | --- |
| <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 | <!DOCTYPE html>  <html lang="en">  <head>      <meta charset="UTF-8">      <title>falling snow</title>  </head>  <body>    <script src="./lib/three/build/three.min.js"></script>  <script src="./lib/three/examples/js/controls/OrbitControls.js"></script>  <script src="./ParticleSystem.js"></script>  <script src="./paticleParameter.js"></script>  <script src="./AdditiveTerrain.js"></script>  <script>      var container;      var camera, scene, renderer;      var controls;        var min = new THREE.Vector3(-300, 0, -300);      var max = new THREE.Vector3( 300, 0,  300);      var iSize = 500;      var jSize = 500;      var totalNum = 100000;      var ptSize = 0.8;      var texPath = "./textures/grass.jpg";      var dtexPath = "./textures/snowflake.png";        var terrain = new AdditiveTerrain();      terrain.initialize(min, max, iSize, jSize, totalNum, ptSize, texPath, dtexPath);        function init() {          container = document.createElement( 'div' );          document.body.appendChild( container );            camera = new THREE.PerspectiveCamera( 60, window.innerWidth / window.innerHeight, 0.01, 10000 );          camera.setLens(18);          camera.position.y = 200;          camera.position.z = 400;            scene = new THREE.Scene();            renderer = new THREE.WebGLRenderer( { antialias: true } );          renderer.setClearColor( 0xaaaaaa );          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> |

눈이 내릴 지형을 만드는 것이 다음 순서이다. Step1에서 연결할 js파일 중에서 'AdditiveTerrain.js'가 이번에 사용된다.

'AdditiveTerrain.js'에서 정의되어 있는 AdditiveTerrain클래스를 사용하기 위해서 terrain이라는 변수 이름으로 AdditivTerrain객체를 생성했다 (<코드 3>의 28번째 줄). 생성한 객체의 맴버 변수를 초기화 하기 위해서 <코드 3>의 29번째 줄에서처럼 initialize함수를 사용한다. Initialize함수에 직접 벡터나 값들을 넣어도 되지만 여기에서는 <코드 3>의 19번째 줄에서 26번째 줄까지 나온 것처럼 각각을 변수로 선언해서 사용했다.

각각의 변수의 의미를 설명하면 다음의 <표 1>과 같다.

19 ~ 26 : 초원 지형을 만들기 위한 값들을 저장하는 변수

28 : terrain이라는 이름으로 AdditiveTerrain의 객체를 만들었다.

29 : 위에서 선언한 변수들을 이용해서 terrain의 맴버 변수를 설정했다.

|  |  |
| --- | --- |
| min | 지면을 만들기 위한 최소값을 나타내는 벡터 |
| max | 지면을 만들기 위한 최대값을 나타내는 벡터 |
| iSize | 지면의 텍스처를 입힐 때 가로의 비율을 나타내는 정수 |
| jSize | 지면의 텍스처를 입힐 때 세로의 비율을 나타내는 정수 |
| totalNum | 지면에 눈을 쌓이게 할 때 표시될 눈의 최대 개수 |
| ptSize | 지면에 눈이 쌓일 때 눈 입자의 크기 |
| texPath | 지면의 텍스처를 입힐 때 사용할 파일의 경로 |
| dtexPath | 지면에 쌓일 눈을 표현하기 위해 사용할 파일의 경로 |
| <표 1> | |

**Step 4. 만들어진 초원 지형을 화면에서 확인하기**

|  |  |
| --- | --- |
| <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 | <!DOCTYPE html>  <html lang="en">  <head>      <meta charset="UTF-8">      <title>falling snow</title>  </head>  <body>    <script src="./lib/three/build/three.min.js"></script>  <script src="./lib/three/examples/js/controls/OrbitControls.js"></script>  <script src="./ParticleSystem.js"></script>  <script src="./paticleParameter.js"></script>  <script src="./AdditiveTerrain.js"></script>  <script>      var container;      var camera, scene, renderer;      var controls;        var min = new THREE.Vector3(-300, 0, -300);      var max = new THREE.Vector3( 300, 0,  300);      var iSize = 500;      var jSize = 500;      var totalNum = 100000;      var ptSize = 0.8;      var texPath = "./textures/grass.jpg";      var dtexPath = "./textures/snowflake.png";        var terrain = new AdditiveTerrain();      terrain.initialize(min, max, iSize, jSize, totalNum, ptSize, texPath, dtexPath);        init();      loop();        function init() {          container = document.createElement( 'div' );          document.body.appendChild( container );            camera = new THREE.PerspectiveCamera( 60, window.innerWidth / window.innerHeight, 0.01, 10000 );          camera.setLens(18);          camera.position.y = 200;          camera.position.z = 400;            scene = new THREE.Scene();            renderer = new THREE.WebGLRenderer( { antialias: true } );          renderer.setClearColor( 0xaaaaaa );          renderer.setSize( window.innerWidth, window.innerHeight );          container.appendChild(renderer.domElement);            controls = new THREE.OrbitControls(camera, renderer.domElement);            scene.add(terrain.getMesh());            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);      }        function render() {          renderer.render( scene, camera );      }  </script>  </body>  </html> |

만들어진 초원을 화면에서 확인하기 위해서는 render을 해야 한다. 그런데 한 번만 render을 하면 한 번만 화면에 그려지기 때문에 조작에 따라서 달라지는 것도 없고, 상태변화에 따라 변하는 모습도 확인할 수 없다. 그렇기 때문에 loop함수를 사용해서 주기적으로 계속 render가 일어나게 만들어 줘야 한다.

<코드 4>의 64번째 줄부터 76번째 줄까지 주기적으로 계속 render가 일어나도록 하는 코드가 작성 되어 있다. 우선 <코드 4>의 64번째 줄에 fps라는 변수가 선언 되어 있는 것을 볼 수 있다. 이 변수는 어떤 시간 간격으로 render가 일어나게 할 것인지를 결정하는 변수이다. 이 변수는 <코드 4>의 67번째 줄의 setTimeout함수에서 사용 되고 있다. setTimeout함수는 자바 스크립트에서 제공하는 함수로 일정 시간이 지나면 특정한 함수를 호출해준다. 여기에서는 (1000/fps) msec마다 loop함수가 호출된다.

<코드 4>의 66번째 줄에서 72번째 줄까지 나와 있는 loop함수에서는 다음의 기능을 수행한다. 우선 특정시간을 주기로 loop함수를 다시 호출한다. 그리고 마우스를 사용해서 화면을 조작한 상태를 업데이트 한다. 마지막으로 render함수를 호출한다. 여기서 requestAnimationFrame는 브라우저의 다른 탭을 사용하고 있는 경우에 다시 render을 하지 않겠다는 의미이다.

<코드 4>의 74번째 줄에서 76번째 줄에 나타나있는 render함수에서 실질적으로 화면에 지형을 그려주는 일을 하게 된다. Three.js에서 제공하는 render함수를 사용해서 지형을 나타내 준다.

render함수에서는 장면에 있는 요소들을 그리기 때문에 장면에 추가하지 않은 요소들은 화면에 나타나지 않는다. 따라서 <코드 4>의 52번째 나온 것처럼 지면을 장면에 추가해 줘야 한다.

이 과정을 모두 마쳤다면 아래의 <그림 1>과 같은 초원이 브라우저 화면에 표시되는 것을 확인할 수 있을 것이다.

31 ~ 32 : init함수와 loop함수를 호출한다.

52 : terrain을 장면에 추가했다.

34 : 화면을 얼마나 자주 보여줄 것인지를 설정하는 fps를 선언했다.

66 ~ 72 : 1000/fps msec마다 loop함수를 호출하고 loop함수에서는 render함수를 호출한다.

74 : renderer에서 제공하는 render함수를 이용해서 장면을 화면에 출력한다.

|  |
| --- |
| <그림 1> |
| 시스템 생성 대체 텍스트: falling snow  G- C localhost:63342/Particles/fallingsnow.htmI |

**Step 5. 눈을 표현할 파티클 시스템 설정하기**

|  |  |
| --- | --- |
| <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 | <!DOCTYPE html>  <html lang="en">  <head>      <meta charset="UTF-8">      <title>falling snow</title>  </head>  <body>    <script src="./lib/three/build/three.min.js"></script>  <script src="./lib/three/examples/js/controls/OrbitControls.js"></script>  <script src="./ParticleSystem.js"></script>  <script src="./paticleParameter.js"></script>  <script src="./AdditiveTerrain.js"></script>  <script>      var container;      var camera, scene, renderer;      var controls;        var min = new THREE.Vector3(-300, 0, -300);      var max = new THREE.Vector3( 300, 0,  300);      var iSize = 500;      var jSize = 500;      var totalNum = 100000;      var ptSize = 0.8;      var texPath = "./textures/grass.jpg";      var dtexPath = "./textures/snowflake.png";        var terrain = new AdditiveTerrain();      terrain.initialize(min, max, iSize, jSize, totalNum, ptSize, texPath, dtexPath);        particleSystem = new ParticleSystem();      particleSystem.initialize(1000);        particleSystem.setParameters(Examples.snowStorm);        init();      loop();        function init() {          container = document.createElement( 'div' );          document.body.appendChild( container );            camera = new THREE.PerspectiveCamera( 60, window.innerWidth / window.innerHeight, 0.01, 10000 );          camera.setLens(18);          camera.position.y = 200;          camera.position.z = 400;            scene = new THREE.Scene();            renderer = new THREE.WebGLRenderer( { antialias: true } );          renderer.setClearColor( 0xaaaaaa );          renderer.setSize( window.innerWidth, window.innerHeight );          container.appendChild(renderer.domElement);            controls = new THREE.OrbitControls(camera, renderer.domElement);            scene.add(terrain.getMesh());            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);      }        function render() {          renderer.render( scene, camera );      }  </script>  </body>  </html> |

초원 지형을 만들었으니 눈을 내리게 하기 위한 파티클 시스템을 만들고 설정해 보자. 이번에는 연결했던 js파일 중 'ParticleSystem.js'와 'paticleParameter.js'파일을 사용할 것이다. ParticleSystem 클래스가 정의 되어 있는 파일은 'ParticleSystem.js'로 이것만 있으면 눈을 만드는데 문제가 없다. 그렇지만 이번 예제에서는 'paticleParameter.js'에 미리 정의해 놓은 파티클 시스템의 속성을 사용하기 위해서 추가적인 js파일을 사용했다.

<코드 5>의 31번째 줄과 32번째 줄을 보면 ParticleSystem 클래스의 객체를 만들고 초기화하는 것을 볼 수 있다. 여기서 initialize(1000)을 사용했는데 initialize함수의 파라미터로 들어간 1000은 눈 입자의 최대 개수를 의미한다. <코드 5>의 34번째 줄에서는 setParameters함수를 사용해서 ParticleSystem의 맴버 변수를 설정하고 있다. 여기에서는 Examples.snowStorm을 사용해서 변수를 설정했는데 'paticleParameter.js'에서 snowStorm이라는 이름으로 각 변수에 들어갈 값들을 미리 입력해 놨기 때문에 가능한 코드이다. 만약 미리 정의된 속성들이 마음에 들지 않는다면 setParameters함수를 사용해서 각각의 변수에 원하는 값을 넣을 수도 있다. (각각의 속성을 따로 설정하는 것은 다음 예제를 통해서 살펴볼 예정이다.)

31 : particleSystem이라는 이름으로 ParticleSystem객체를 만들었다.

32 : particleSystem이 가질 수 있는 입자의 최대 개수는 1000이다.

34 : particleSystem의 맴버 변수는 Examples.snowStorm을 이용해서 설정한다.

**Step 6. 눈을 초원 지형에 추가하기**

|  |  |
| --- | --- |
| <code6> | |
| 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 | <!DOCTYPE html>  <html lang="en">  <head>      <meta charset="UTF-8">      <title>falling snow</title>  </head>  <body>    <script src="./lib/three/build/three.min.js"></script>  <script src="./lib/three/examples/js/controls/OrbitControls.js"></script>  <script src="./ParticleSystem.js"></script>  <script src="./paticleParameter.js"></script>  <script src="./AdditiveTerrain.js"></script>  <script>      var container;      var camera, scene, renderer;      var controls;        var min = new THREE.Vector3(-300, 0, -300);      var max = new THREE.Vector3( 300, 0,  300);      var iSize = 500;      var jSize = 500;      var totalNum = 100000;      var ptSize = 0.8;      var texPath = "./textures/grass.jpg";      var dtexPath = "./textures/snowflake.png";        var terrain = new AdditiveTerrain();      terrain.initialize(min, max, iSize, jSize, totalNum, ptSize, texPath, dtexPath);        particleSystem = new ParticleSystem();      particleSystem.initialize(1000);        particleSystem.setParameters(Examples.snowStorm);        init();      loop();        function init() {          container = document.createElement( 'div' );          document.body.appendChild( container );            camera = new THREE.PerspectiveCamera( 60, window.innerWidth / window.innerHeight, 0.01, 10000 );          camera.setLens(18);          camera.position.y = 200;          camera.position.z = 400;            scene = new THREE.Scene();            renderer = new THREE.WebGLRenderer( { antialias: true } );          renderer.setClearColor( 0xaaaaaa );          renderer.setSize( window.innerWidth, window.innerHeight );          container.appendChild(renderer.domElement);            controls = new THREE.OrbitControls(camera, renderer.domElement);            scene.add(terrain.getMesh());          scene.add(particleSystem.getMesh());            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;            var cen = new THREE.Vector3(0,200,0);          var nor = new THREE.Vector3(0,-1,0);          var rad = 200;            particleSystem.addParticlesFromDisk(10, cen, nor, rad);          particleSystem.updateParticles(dt, terrain);            controls.update();            requestAnimationFrame(render);      }        function render() {          renderer.render( scene, camera );      }  </script>  </body>  </html> |

설정한 파티클 시스템을 장면에 추가하고 눈을 내리게 해보자.

일단 눈을 표현하는 particleSystem을 장면에 추가한다(<코드 6>의 58번쨰 줄).

<코드 6>의 71번째 줄과 76,77,78번째 줄을 보면 시간에 관련된 여러 가지 값을 사용하고 있다. 이 값들은 시간에 변화에 맞춰서 상태를 업데이트하기 위해서 사용된다.

조금 더 신경 써서 살펴볼 부분은 그 다음에 나온다. <코드 6>의 80번째에서 85번째 줄이 그 부분이다. 눈을 내리게 하기 위해서는 눈이 만들어지는 장소를 설정해야 한다. 여기에서는 disk모양의 눈이 만들어지는 장소를 사용했다. ParticleSystem 클래스의 함수 중에서 addParticlesFromDisk를 사용하면 disk에서 눈을 만들수 있다. 이런 disk의 모양과 크기, 위치를 설정하는 변수가 <코드 6>의 80, 81, 82번째 줄에 나와 있다. cen은 disk의 중심 위치를 나타내고, nor은 disk의 방향을 나타내고, rad는 disk의 지름을 나타낸다. 또 addParticlesFromDisk의 첫 번째 파라미터는 disk에서 만들어지는 입자의 개수를 나타낸다.

ParticleSystem 클래스의 updateParticles를 사용하면 시간의 변화에 따라서 파티클 시스템의 상태를 갱신한다. 이 때 사용하는 dt를 얻기 위해서 time이나 now같은 변수들을 사용한 것이다.

particleSystem을 장면에 추가하고 loop함수에 파티클을 만들고 갱신하는 함수를 추가했다면 아래의 <그림 2>와 같이 초원에 눈이 내리는 모습을 확인할 수 있다. 그런데 눈은 내리는데 초원에 눈이 쌓이지는 않는다.

71 : 시간의 변화를 계산하기 위한 변수를 선언

76 ~ 78 : 시간의 변화를 계산

80 ~ 82 : particleSystem의 입자를 만들 디스크를 설정하기 위한 변수

84 : 위에서 설정한 값을 가지는 디스크에서 입자를 만들어낸다.

85 : 시간에 따른 particleSystem의 상태를 갱신한다.

|  |
| --- |
| <그림2> |
| 시스템 생성 대체 텍스트: BOOS |

**Step 7. 눈이 쌓이는 효과주기**

|  |  |
| --- | --- |
| <code7> | |
| 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 | <!DOCTYPE html>  <html lang="en">  <head>      <meta charset="UTF-8">      <title>falling snow</title>  </head>  <body>    <script src="./lib/three/build/three.min.js"></script>  <script src="./lib/three/examples/js/controls/OrbitControls.js"></script>  <script src="./ParticleSystem.js"></script>  <script src="./paticleParameter.js"></script>  <script src="./AdditiveTerrain.js"></script>  <script>      var container;      var camera, scene, renderer;      var controls;        var min = new THREE.Vector3(-300, 0, -300);      var max = new THREE.Vector3( 300, 0,  300);      var iSize = 500;      var jSize = 500;      var totalNum = 100000;      var ptSize = 0.8;      var texPath = "./textures/grass.jpg";      var dtexPath = "./textures/snowflake.png";        var terrain = new AdditiveTerrain();      terrain.initialize(min, max, iSize, jSize, totalNum, ptSize, texPath, dtexPath);        particleSystem = new ParticleSystem();      particleSystem.initialize(1000);        particleSystem.setParameters(Examples.snowStorm);        init();      loop();        function init() {          container = document.createElement( 'div' );          document.body.appendChild( container );            camera = new THREE.PerspectiveCamera( 60, window.innerWidth / window.innerHeight, 0.01, 10000 );          camera.setLens(18);          camera.position.y = 200;          camera.position.z = 400;            scene = new THREE.Scene();            renderer = new THREE.WebGLRenderer( { antialias: true } );          renderer.setClearColor( 0xaaaaaa );          renderer.setSize( window.innerWidth, window.innerHeight );          container.appendChild(renderer.domElement);            controls = new THREE.OrbitControls(camera, renderer.domElement);            scene.add(terrain.getMesh());          scene.add(terrain.getDecalMesh());          scene.add(particleSystem.getMesh());            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;            var cen = new THREE.Vector3(0,200,0);          var nor = new THREE.Vector3(0,-1,0);          var rad = 200;            particleSystem.addParticlesFromDisk(10, cen, nor, rad);          particleSystem.updateParticles(dt, terrain);            controls.update();            requestAnimationFrame(render);      }        function render() {          renderer.render( scene, camera );      }  </script>  </body>  </html> |

초원에 눈이 쌓이는 효과를 주기 위해서 한 줄의 코드를 더 넣어야 한다. <코드 7>의 58번째 줄에 나오는 코드가 바로 그것이다. 이 코드는 지면에 눈이 쌓이는 것을 표현한 decal이라는 요소를 장면에 추가해 준 것이다. 이제 다시 브라우저에서 실행시키면 <그림 3>과 같이 시간이 지남에 따라 초원에 눈이 쌓이는 것을 확인할 수 있다.

58 : terrain의 decal을 장면에 넣는다. Decal에서 눈이 쌓이는 효과를 준다

|  |
| --- |
| <그림 3> |
| 시스템 생성 대체 텍스트: falling snow  -5 C localhost:63342/Panicles/fallingsnow.html |