이 게시물은 peter4k님의 블로그에 2016-01-29 오후 2:17:14에 게시되었습니다.

불꽃놀이 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. 불꽃놀이에 사용할 파티클 시스템 설정하기.
4. 무작위 위치에서 불꽃 터트리기.
5. 불꽃의 색깔을 무작위로 바꿔보기.
6. 불꽃놀이의 폭발의 크기를 바꿔보기.
7. 예제 완성하기.

**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 Work</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>  </body>  </html> |

이번 예제에서 필요한 js파일은 <code1>에서 나오는 4개이다.

**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 | <!DOCTYPE html>  <html lang="en">  <head>      <meta charset="UTF-8">      <title>Fire Work</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>      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 = 500;**  **camera.position.z = 1000;**            scene = new THREE.Scene();            renderer = new THREE.WebGLRenderer( { antialias: true } );  **renderer.setClearColor( 0x111111 );**          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> |

이번 예제에서는 camera의 위치를 조금 멀리에 위치시켰다. 그리고 <code2>의 29번째 줄에서 renderer의 기본 색을 검정색에 가깝게 해서 생성되는 파티클들이 조금 더 눈에 띄도록 만들었다.

**Step 3. 불꽃놀이에 사용할 파티클 시스템 설정하기**

|  |  |
| --- | --- |
| <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 | <!DOCTYPE html>  <html lang="en">  <head>      <meta charset="UTF-8">      <title>Fire Work</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>      var container;      var camera, scene, renderer;      var controls;    **particleSystem1 = new ParticleSystem();**  **particleSystem1.initialize(1000);**  **particleSystem1.setParameters(Examples.fireworks);**        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 = 500;          camera.position.z = 1000;            scene = new THREE.Scene();            renderer = new THREE.WebGLRenderer( { antialias: true } );          renderer.setClearColor( 0x111111 );          renderer.setSize( window.innerWidth, window.innerHeight );          container.appendChild(renderer.domElement);            controls = new THREE.OrbitControls(camera, renderer.domElement);            scene.add(particleSystem1.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;        var rad = 5;      var cen = new THREE.Vector3(0,0,0);      var dt;        var intervalTime1 = 700;        function loop() {          setTimeout(loop, 1000/fps);            var now = new Date().getTime();          dt = (now - (time||now))/1000;          time = now;            particleSystem1.updateParticles(dt);            controls.update();            requestAnimationFrame(render);      }        function render() {          renderer.render( scene, camera );      }    **function generate(ps){**  **ps.addParticlesFromSphere(1000, cen, rad);**  **}**    **function doFirework1(){**  **generate(particleSystem1);**  **setTimeout(doFirework1, intervalTime1);**  **}**        init();      loop();  **doFirework1();**    </script>  </body>  </html> |

기본적으로 Example에 있는 설정을 가져다 사용했다.

불꽃놀이의 느낌을 내기 위해서 파티클을 sphere에서 만들도록 했다. 그리고 폭발해서 생기는 불꽃놀이이므로 짧은 시간 동안만 파티클을 생성하게 하고 일정시간이 지난 다음 다시 파티클을 만들도록 했다.. 어느 정도의 빠르기로 파티클이 작은 구에서 생성되어 바깥쪽으로 나가면 폭발하는 듯한 느낌을 줄 수 있다.

잠깐의 시간 동안에만 파티클을 생성하게 하기 위해서 <code3>의 78번째 줄에서처럼 generate함수를 만들어 addParticlesFromSphere을 따로 부르도록 했다. 58번째 줄의 intervalTime이라는 변수는 불꽃이 폭발하는 시간 간격을 나타내는 함수이고 doFirework1함수를 만들어서 앞에서 설정한 작업들을 일괄적으로 처리하도록 하고 intervalTime1을 이용해서 일정시간간격을 두고 다시 doFirework1함수를 부르도록 했다.

이제 init과 loop, doFireWork1을 실행시키면 폭죽이 터지는 듯한 모습을 볼 수 있다.

|  |
| --- |
| 시스템 생성 대체 텍스트: Fire Work  localhost:63342/Particles/firework.html |
| <그림1> |

**Step 4. 무작위 위치에서 불꽃 터트리기**

|  |  |
| --- | --- |
| <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 | <!DOCTYPE html>  <html lang="en">  <head>      <meta charset="UTF-8">      <title>Fire Work</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>      var container;      var camera, scene, renderer;      var controls;        particleSystem1 = new ParticleSystem();      particleSystem1.initialize(1000);      particleSystem1.setParameters(Examples.fireworks);        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 = 500;          camera.position.z = 1000;            scene = new THREE.Scene();            renderer = new THREE.WebGLRenderer( { antialias: true } );          renderer.setClearColor( 0x111111 );          renderer.setSize( window.innerWidth, window.innerHeight );          container.appendChild(renderer.domElement);            controls = new THREE.OrbitControls(camera, renderer.domElement);            scene.add(particleSystem1.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;        var rad = 5;      var dt;        var intervalTime1 = 700;        function loop() {          setTimeout(loop, 1000/fps);            var now = new Date().getTime();          dt = (now - (time||now))/1000;          time = now;            particleSystem1.updateParticles(dt);            controls.update();            requestAnimationFrame(render);      }        function render() {          renderer.render( scene, camera );      }        function generate(ps, cen){          ps.addParticlesFromSphere(1000, cen, rad);      }        function getRandomInt( min, max ) {          return Math.floor( Math.random() \* ( max - min + 1 ) ) + min;      }        function randomPos(){          var x = getRandomInt(-400, 400);          var y = getRandomInt(100, 400);          var z = getRandomInt(-300, 300);          var cen = new THREE.Vector3(x,y,z);            return cen;      }        function doFirework1(){          var cen = randomPos();          generate(particleSystem1, cen);          setTimeout(doFirework1, intervalTime1);      }        init();      loop();      doFirework1();    </script>  </body>  </html> |

무작위 위치에서 불꽃을 터트리는 일은 간단하다. 주기적으로 sphere의 중심위치만 바꿔주면 되기 때문이다. 이를 위해서 <code4>의 82번째 줄에 나와있는 getRandomInt라는 주어진 범위에서 난수를 구하는 함수를 만들었다. 그리고 86번째 줄에서부터 93번째 줄에서 보이는 것처럼 randomPos함수를 사용해서 x, y, z좌표에 해당하는 랜덤 넘버를 구하고 이것으로 벡터를 만들어서 addParticlesFromSphere의 매개변수로 넣었다. 그렇게 하면 지정된 범위 안의 무작위 위치에서 불꽃이 생성된다.

렌덤한 좌표를 구해서 중심위치로 설정하는 것은 doFirework1에서 처리하도록 했으며 generate함수에서 생성한 좌표를 매개변수로 받아 그 자리에서 파티클이 생성되도록 했다.

위의 <code4>를 실행시켜보면 아래의 <그림2>와 같이 불꽃의 위치가 계속해서 변하는 것을 볼 수 있다.

|  |
| --- |
| 시스템 생성 대체 텍스트: Fire Work  C  localhost:63342/Particles/firework.html |
| <그림2> |

**Step 5. 불꽃의 색깔을 무작위로 바꿔보기**

|  |  |
| --- | --- |
| <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 | <!DOCTYPE html>  <html lang="en">  <head>      <meta charset="UTF-8">      <title>Fire Work</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>      var container;      var camera, scene, renderer;      var controls;        particleSystem1 = new ParticleSystem();      particleSystem1.initialize(1000);      particleSystem1.setParameters(Examples.fireworks);        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 = 500;          camera.position.z = 1000;            scene = new THREE.Scene();            renderer = new THREE.WebGLRenderer( { antialias: true } );          renderer.setClearColor( 0x111111 );          renderer.setSize( window.innerWidth, window.innerHeight );          container.appendChild(renderer.domElement);            controls = new THREE.OrbitControls(camera, renderer.domElement);            scene.add(particleSystem1.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;        var rad = 5;      var dt;        var intervalTime1 = 1700;        function loop() {          setTimeout(loop, 1000/fps);            var now = new Date().getTime();          dt = (now - (time||now))/1000;          time = now;            particleSystem1.updateParticles(dt);            controls.update();            requestAnimationFrame(render);      }        function render() {          renderer.render( scene, camera );      }        function generate(ps, cen){          ps.addParticlesFromSphere(1000, cen, rad);      }        function getRandomInt( min, max ) {          return Math.floor( Math.random() \* ( max - min + 1 ) ) + min;      }        function randomPos(){          var x = getRandomInt(-400, 400);          var y = getRandomInt(100, 400);          var z = getRandomInt(-300, 300);          var cen = new THREE.Vector3(x,y,z);            return cen;      }        function randomColor(){          var colorCode = Math.round(Math.random() \* 0xffffff);          return colorCode;      }        function doFirework1(){          var cen = randomPos();          particleSystem1.setParameters({particleColor: new THREE.Color(randomColor())});          generate(particleSystem1, cen);          setTimeout(doFirework1, intervalTime1);      }        init();      loop();      doFirework1();    </script>  </body>  </html> |

좌표를 난수로 했던 것의 연장선상에 있는 단계이다. <code5>의 94번째 줄에 나온 것처럼randomColor라는 함수를 정의해서 렌덤한 colorCode를 생성하고 particleSystem의 setParameter을 사용해서 파티클 시스템의 색상을 바꿔주면 쉽게 색깔을 바꿀 수 있다.

이 작업을 처리하는 것도 doFirework1에서 수행한다. setParameters함수가 particleSystem1자체를 바꾸는 것이기 때문에 따로 color code를 매개변수로 보낼 필요는 없고 particleSystem1을 매개변수로 보낸 것으로 충분하다.

다만 intervalTIme1을 너무 짧게 잡으면 이미 터지고 있던 불꽃들의 색상도 같이 변하기 때문에 시간간격을 충분히 길게 둬서 이전의 불꽃이 모두 사라진 다음 색상을 변하게 하는 것이 중요하다. 그 시간간격은 particleSystem의 seedLife값에 따라 다르지만 여기에서는 최소 700이상이면 충분하다.

|  |
| --- |
| 시스템 생성 대체 텍스트: Fire Work  C  localhost:63342/Particles/firework.html  Fire Work  C  localhost:63342/Particles/firework.html |
| <그림3> |

**Step 6. 불꽃놀이의 폭발의 크기를 바꿔보기**

|  |  |
| --- | --- |
| <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  98  99  100  101  102  103  104  105  106  107  108  109  110  111  112  113  114  115  116  117  118 | <!DOCTYPE html>  <html lang="en">  <head>      <meta charset="UTF-8">      <title>Fire Work</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>      var container;      var camera, scene, renderer;      var controls;        particleSystem1 = new ParticleSystem();      particleSystem1.initialize(1000);      particleSystem1.setParameters(Examples.fireworks);        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 = 500;          camera.position.z = 1000;            scene = new THREE.Scene();            renderer = new THREE.WebGLRenderer( { antialias: true } );          renderer.setClearColor( 0x111111 );          renderer.setSize( window.innerWidth, window.innerHeight );          container.appendChild(renderer.domElement);            controls = new THREE.OrbitControls(camera, renderer.domElement);            scene.add(particleSystem1.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;        var rad = 5;      var dt;        var intervalTime1 = 1700;        function loop() {          setTimeout(loop, 1000/fps);            var now = new Date().getTime();          dt = (now - (time||now))/1000;          time = now;            particleSystem1.updateParticles(dt);            controls.update();            requestAnimationFrame(render);      }        function render() {          renderer.render( scene, camera );      }        function generate(ps, cen){          ps.addParticlesFromSphere(1000, cen, rad);      }        function getRandomInt( min, max ) {          return Math.floor( Math.random() \* ( max - min + 1 ) ) + min;      }        function randomPos(){          var x = getRandomInt(-400, 400);          var y = getRandomInt(100, 400);          var z = getRandomInt(-300, 300);          var cen = new THREE.Vector3(x,y,z);            return cen;      }        function randomColor(){          var colorCode = Math.round(Math.random() \* 0xffffff);          return colorCode;      }        function randomMag(){          var mag = getRandomInt(70,250);          return mag;      }        function doFirework1(){          var cen = randomPos();          particleSystem1.setParameters({particleColor: new THREE.Color(randomColor())});          particleSystem1.setParameters({seedVelMag: randomMag()});          generate(particleSystem1, cen);          setTimeout(doFirework1, intervalTime1);      }        init();      loop();      doFirework1();    </script>  </body>  </html> |

폭발의 크기를 바꾸는 것은 particleSystem의 속성 중에서 seedVelMag값을 바꾸는 것으로 가능해 진다. 이 속성은 입자의 속도에 관련된 값으로 폭발이 크다는 것이 짧은 시간에 입자가 멀리 간다는 것과 유사하다는 것에서 연관이 있다.

위의 좌표나 색깔을 바꾸는 것과 유사하게 <code4>의 99번째 줄에서처럼 randomMag함수를 만들어서 doFirework1에서 사용하도록 하면 폭발의 크기를 렌덤하게 바꿀 수 있다.

|  |
| --- |
| 시스템 생성 대체 텍스트: Fire Work  localhost:63342/Particles/firework.html  Fire Work  C  localhost:63342/Particles/firework.html |
| <그림4> |

**Step 7. 예제 완성하기**

|  |  |
| --- | --- |
| <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  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 | <!DOCTYPE html>  <html lang="en">  <head>      <meta charset="UTF-8">      <title>Fire Work</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>      var container;      var camera, scene, renderer;      var controls;        particleSystem1 = new ParticleSystem();      particleSystem1.initialize(1000);      particleSystem1.setParameters(Examples.fireworks);        particleSystem2 = new ParticleSystem();      particleSystem2.initialize(1000);      particleSystem2.setParameters(Examples.fireworks);        particleSystem3 = new ParticleSystem();      particleSystem3.initialize(1000);      particleSystem3.setParameters(Examples.fireworks);        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 = 500;          camera.position.z = 1000;            scene = new THREE.Scene();            renderer = new THREE.WebGLRenderer( { antialias: true } );          renderer.setClearColor( 0x111111 );          renderer.setSize( window.innerWidth, window.innerHeight );          container.appendChild(renderer.domElement);            controls = new THREE.OrbitControls(camera, renderer.domElement);            scene.add(particleSystem1.getMesh());          scene.add(particleSystem2.getMesh());          scene.add(particleSystem3.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;        var rad = 5;      var dt;        var intervalTime1 = 1700;      var intervalTime2 = 1100;      var intervalTime3 = 1300;        function loop() {          setTimeout(loop, 1000/fps);            var now = new Date().getTime();          dt = (now - (time||now))/1000;          time = now;            particleSystem1.updateParticles(dt);          particleSystem2.updateParticles(dt);          particleSystem3.updateParticles(dt);            controls.update();            requestAnimationFrame(render);      }        function render() {          renderer.render( scene, camera );      }        function generate(ps, cen){          ps.addParticlesFromSphere(1000, cen, rad);      }        function getRandomInt( min, max ) {          return Math.floor( Math.random() \* ( max - min + 1 ) ) + min;      }        function randomPos(){          var x = getRandomInt(-400, 400);          var y = getRandomInt(100, 400);          var z = getRandomInt(-300, 300);          var cen = new THREE.Vector3(x,y,z);            return cen;      }        function randomColor(){          var colorCode = Math.round(Math.random() \* 0xffffff);          return colorCode;      }        function randomMag(){          var mag = getRandomInt(70,250);          return mag;      }        function doFirework1(){          var cen = randomPos();          particleSystem1.setParameters({particleColor: new THREE.Color(randomColor())});          particleSystem1.setParameters({seedVelMag: randomMag()});          generate(particleSystem1, cen);          setTimeout(doFirework1, intervalTime1);      }        function doFirework2(){          var cen = randomPos();          particleSystem2.setParameters({particleColor: new THREE.Color(randomColor())});          particleSystem2.setParameters({seedVelMag: randomMag()});          generate(particleSystem2,cen);          setTimeout(doFirework2, intervalTime2);      }        function doFirework3(){          var cen = randomPos();          particleSystem3.setParameters({particleColor: new THREE.Color(randomColor())});          particleSystem3.setParameters({seedVelMag: randomMag()});          generate(particleSystem3,cen);          setTimeout(doFirework3, intervalTime3);      }        init();      loop();      doFirework1();      doFirework2();      doFirework3();    </script>  </body>  </html> |

하나의 폭발만 사용하는 것은 화려함이 덜하기 때문에 particleSystem을 3개 사용해서 폭발이 계속 일어나게 만들었다. 다만 intervalTime의 값을 최대한 겹치지 않게 해서 폭발이 일어나는 주기를 랜덤하계 보이도록 했다.

|  |
| --- |
| 시스템 생성 대체 텍스트: Fire Work  C  localhost:63342/Particles/firework.html |
| <그림5> |