지형 편집 툴로 만든 JSON파일 사용하기.

계정 비쥬얼 시뮬레이션 연구실

지형 편집 툴을 사용해서 원하는 지형과 마그마를 만들었다면 지형과 마그마를 저장한 json파일을 사용해서 화면에 만들어 보자.  
이번 예제는 다음과 같은 순서로 진행된다.

1. 필요한 js파일 연결하기.
2. Three.js의 기본 요소 만들기.
3. skybox 만들기.
4. 지형 정보를 가져오기.
5. 마그마 정보를 가져오기.

**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>magma Terrain</title>  </head>  <body>  <script src="./lib/three/build/three.min.js"></script>  <script src="./lib/three/examples/js/controls/OrbitControls.js"></script>  <script src="./AdditiveTerrain.js"></script>  <script src="./FlowLines.js"></script>  </body>  </html> |

이번 예제에서 사용하는 js파일은 <code1>의 8, 9, 10, 11번째 줄에 나와있는 4가지 이다.

**Step 2. Three.js의 기본 요소 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>magma Terrain</title>  </head>  <body>  <script src="./lib/three/build/three.min.js"></script>  <script src="./lib/three/examples/js/controls/OrbitControls.js"></script>  <script src="./AdditiveTerrain.js"></script>  <script src="./FlowLines.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, 1, 100000 );          camera.setLens(18);            var camScale = 0.5;          camera.position.y =   800\*camScale;          camera.position.x = -3000\*camScale;          camera.position.z =  2000\*camScale;            scene = new THREE.Scene();            renderer = new THREE.WebGLRenderer( { antialias: true } );          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> |

camera, scene, renderer, controls와 같은 Three.js의 기본적인 요소들을 선언하고 초기화 했다.

**Step 3. skybox 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  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 | <!DOCTYPE html>  <html lang="en">  <head>      <meta charset="UTF-8">      <title>magma Terrain</title>  </head>  <body>  <script src="./lib/three/build/three.min.js"></script>  <script src="./lib/three/examples/js/controls/OrbitControls.js"></script>  <script src="./AdditiveTerrain.js"></script>  <script src="./FlowLines.js"></script>  <script>      var container;      var camera, scene, renderer;      var controls;        init();      loop();        function init() {          container = document.createElement( 'div' );          document.body.appendChild( container );            camera = new THREE.PerspectiveCamera( 60, window.innerWidth / window.innerHeight, 1, 100000 );          camera.setLens(18);            var camScale = 0.5;          camera.position.y =   800\*camScale;          camera.position.x = -3000\*camScale;          camera.position.z =  2000\*camScale;            var path = "./textures/skybox/";          var format = '.jpg';          var urls = [              path + 'px' + format, path + 'nx' + format,              path + 'py' + format, path + 'ny' + format,              path + 'pz' + format, path + 'nz' + format          ];            var textureCube = THREE.ImageUtils.loadTextureCube( urls );          textureCube.mapping = THREE.CubeRefractionMapping;            var shader = THREE.ShaderLib[ "cube" ];          shader.uniforms[ "tCube" ].value = textureCube;            var material = new THREE.ShaderMaterial( {              fragmentShader: shader.fragmentShader,              vertexShader: shader.vertexShader,              uniforms: shader.uniforms,              depthWrite: false,              side: THREE.BackSide          } );            var skyBoxMesh = new THREE.Mesh( new THREE.BoxGeometry( 10000, 10000, 10000 ), material );            scene = new THREE.Scene();          scene.add(skyBoxMesh);            renderer = new THREE.WebGLRenderer( { antialias: true } );          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);      }        function render() {          renderer.render( scene, camera );      }  </script>  </body>  </html> |

이번 단계에서는 skybox를 만든다. Skybox를 만듦으로써 주변 배경을 하늘에 떠 있는 것처럼 보이게 할 수 있다.

Skybox에서 사용할 텍스처를 불러오기 위해서 이미지 파일의 경로와 이름을 설정한다. <code3>의 32번째 줄에서부터 38번째 줄에 파일이름을 만들고 urls에 저장하는 과정이 나온다. 상자모양으로 만들 것이기 때문에 필요한 텍스처 파일은 6개이다. 이렇게 만든 urls를 이용해서 textureCube를 만든다. 다음으로 shader을 만든다. 43번째 줄과 44번째 줄을 보면 ShaderLib을 사용해서 shader을 만드는 것을 볼 수 있다. 이렇게 만든 shader을 이용해서 material을 설정했다. 이제 54번째 줄처럼 skyBoxMesh를 만들고 scene에 추가하면 skybox를 만드는 과정은 끝이 난다.

만든 skybox를 화면에서 확인해 보기 위해서 loop함수와 render함수를 추가한다. loop함수에서는 1000/fps의 시간간격으로 loop함수를 호출하고 리퀘스트가 있을 때마다 render함수를 호출한다. 그리고 controls의 상태를 업데이트 한다. render함수는 실질적으로 scene의 내용을 화면에 그려주는 역할을 하는 함수이다. 이렇게 loop함수와 render함두를 만든 다음 <code3>을 브라우저에서 실행시켜 보면 아래의 <그림1>과 같은 모습을 볼 수 있다.

|  |
| --- |
|  |
| <그림1> |

**Step 4. 지형 정보를 가져오기**

|  |  |
| --- | --- |
| <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 | <!DOCTYPE html>  <html lang="en">  <head>      <meta charset="UTF-8">      <title>magma Terrain</title>  </head>  <body>  <script src="./lib/three/build/three.min.js"></script>  <script src="./lib/three/examples/js/controls/OrbitControls.js"></script>  <script src="./AdditiveTerrain.js"></script>  <script src="./FlowLines.js"></script>  <script>      var container;      var camera, scene, renderer;      var controls;        var terrain = new AdditiveTerrain();      var min = new THREE.Vector3(-2000.0, 0.0, -2000.0);      var max = new THREE.Vector3( 2000.0, 0.0,  2000.0);      var terrainParams = {          terrainImage:"./textures/rock.jpg",          terrainImage1:"./textures/grass.jpg",          decalColor:0xffaa00,          load:"./models/magma\_mountain.json"      };      terrain.initialize(min, max, 200, 200, 0, 0, terrainParams);        init();      loop();        function init() {          container = document.createElement( 'div' );          document.body.appendChild( container );            camera = new THREE.PerspectiveCamera( 60, window.innerWidth / window.innerHeight, 1, 100000 );          camera.setLens(18);            var camScale = 0.5;          camera.position.y =   800\*camScale;          camera.position.x = -3000\*camScale;          camera.position.z =  2000\*camScale;            var path = "./textures/skybox/";          var format = '.jpg';          var urls = [              path + 'px' + format, path + 'nx' + format,              path + 'py' + format, path + 'ny' + format,              path + 'pz' + format, path + 'nz' + format          ];            var textureCube = THREE.ImageUtils.loadTextureCube( urls );          textureCube.mapping = THREE.CubeRefractionMapping;            var shader = THREE.ShaderLib[ "cube" ];          shader.uniforms[ "tCube" ].value = textureCube;            var material = new THREE.ShaderMaterial( {              fragmentShader: shader.fragmentShader,              vertexShader: shader.vertexShader,              uniforms: shader.uniforms,              depthWrite: false,              side: THREE.BackSide          } );            var skyBoxMesh = new THREE.Mesh( new THREE.BoxGeometry( 10000, 10000, 10000 ), material );            scene = new THREE.Scene();          scene.add(skyBoxMesh);          scene.add(terrain.getMesh());            renderer = new THREE.WebGLRenderer( { antialias: true } );          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);      }        function render() {          renderer.render( scene, camera );      }  </script>  </body>  </html> |

지형 편집 툴을 사용해서 만들었던 지형정보를 가져오기 위해서 terrain을 만들었다. <code4>의 26번째 줄에서부터 36번째 줄에 terrain을 만드는 과정이 나타나 있다. Terrain을 만들기 위해서 사용한 변수는 총 7가지로 initialize함수의 매개변수로 사용되었다. 각각 terrain의 최소 위치 벡터, terrain의 최대 위치 벡터, terrain의 텍스처를 입히는 x축 비율, terrain의 텍스처를 입히는 y축 비율, decal의 개수, decal의 크기, 다른 terrain을 설정하는 정보를 의미한다. decal관련된 값은 여기에서 사용하지 않기 때문에 0으로 뒀다. 특히, 24번째 줄에서 ‘./models/magma\_mountain.json’파일을 가져오는 부분이 있는데 이것이 바로 지형 편집 툴에서 만들었던 지형 정보를 가지고 있는 것이다. 이렇게 만든 terrain을 init함수에서 scene에 추가시키면 아래의 <그림2>와 같은 모습을 볼 수 있다.

|  |
| --- |
|  |
| <그림2> |

**Step 5. 마그마 정보를 가져오기**

|  |  |
| --- | --- |
| <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 | <!DOCTYPE html>  <html lang="en">  <head>      <meta charset="UTF-8">      <title>magma Terrain</title>  </head>  <body>  <script src="./lib/three/build/three.min.js"></script>  <script src="./lib/three/examples/js/controls/OrbitControls.js"></script>  <script src="./AdditiveTerrain.js"></script>  <script src="./FlowLines.js"></script>  <script>      var container;      var camera, scene, renderer;      var controls;        var terrain = new AdditiveTerrain();      var min = new THREE.Vector3(-2000.0, 0.0, -2000.0);      var max = new THREE.Vector3( 2000.0, 0.0,  2000.0);      var terrainParams = {          terrainImage:"./textures/rock.jpg",          terrainImage1:"./textures/grass.jpg",          decalColor:0xffaa00,          load:"./models/magma\_mountain.json"      };      terrain.initialize(min, max, 200, 200, 0, 0, terrainParams);        var flowLines = new FlowLines();      flowLines.addLinesPointsFromJSON('./models/magma\_line.json');      flowLines.generate();      flowLines.generateGeometry();      flowLines.updateTerrain(terrain);        init();      loop();        function init() {          container = document.createElement( 'div' );          document.body.appendChild( container );            camera = new THREE.PerspectiveCamera( 60, window.innerWidth / window.innerHeight, 1, 100000 );          camera.setLens(18);            var camScale = 0.5;          camera.position.y =   800\*camScale;          camera.position.x = -3000\*camScale;          camera.position.z =  2000\*camScale;            var path = "./textures/skybox/";          var format = '.jpg';          var urls = [              path + 'px' + format, path + 'nx' + format,              path + 'py' + format, path + 'ny' + format,              path + 'pz' + format, path + 'nz' + format          ];            var textureCube = THREE.ImageUtils.loadTextureCube( urls );          textureCube.mapping = THREE.CubeRefractionMapping;            var shader = THREE.ShaderLib[ "cube" ];          shader.uniforms[ "tCube" ].value = textureCube;            var material = new THREE.ShaderMaterial( {              fragmentShader: shader.fragmentShader,              vertexShader: shader.vertexShader,              uniforms: shader.uniforms,              depthWrite: false,              side: THREE.BackSide          } );            var skyBoxMesh = new THREE.Mesh( new THREE.BoxGeometry( 10000, 10000, 10000 ), material );            scene = new THREE.Scene();          scene.add(skyBoxMesh);          scene.add(terrain.getMesh());          scene.add(flowLines.getMesh());            renderer = new THREE.WebGLRenderer( { antialias: true } );          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;      var time;        function loop() {          setTimeout(loop, 1000/fps);            var now = new Date().getTime();          var dt = (now - (time||now))/1000;          time = now;            flowLines.update(dt);          controls.update();            requestAnimationFrame(render);      }        function render() {          renderer.render( scene, camera );      }  </script>  </body>  </html> |

지형 편집 툴을 사용해서 만든 마그마에 대한 정보를 가져오기 위해서 flowLines를 만들었다. 특히, <code5>의 29번째 줄을 보면 “./models/magma\_line.json”을 사용해서 마그마를 생성하는 것을 볼 수 있다. 이렇게 만든 flowLines를 init함수에서 scene에 추가하고, 시간이 지남에 따라 흐르는 모습을 보여주기 위해서 loop함수에 flowloines.update함수를 추가하면 아래의 <그림3>과 같은 결과를 확인할 수 있다.

|  |
| --- |
|  |
| <그림3> |