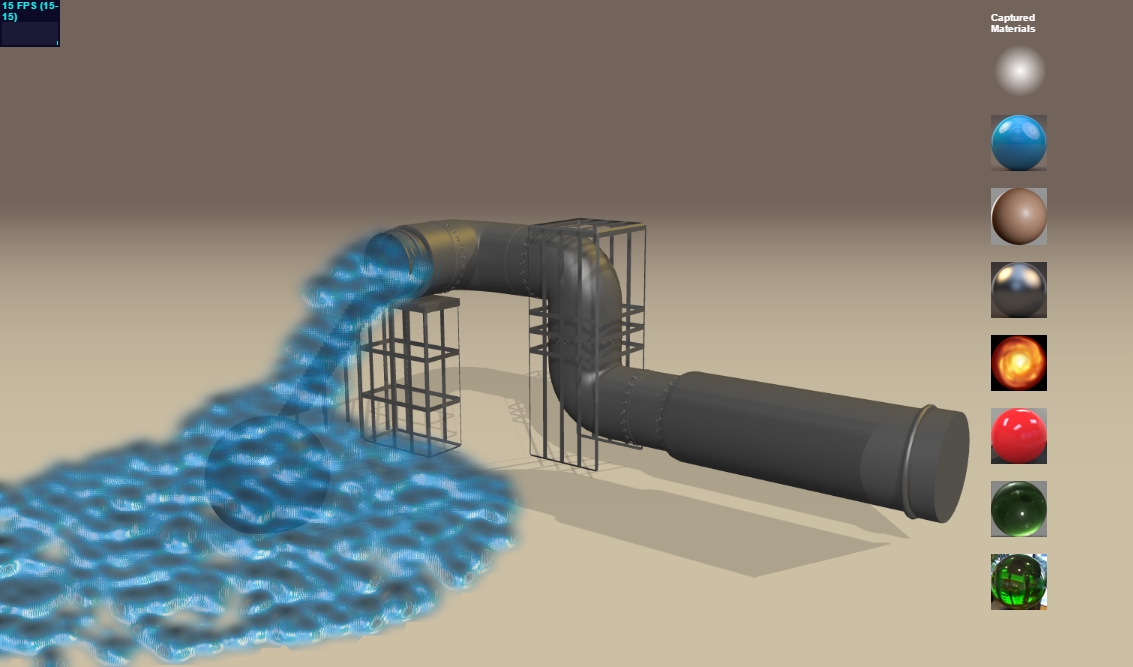
[실전 예제 : Pipe Water]



**[실전 예제 “Pipe Water”의 결과물 : Pipe Line모형의 객체에서 우측 textures의 종류에 따라 Water의 색이 변하는 예제]**

***★앞선 Tutorial들과 실전 예제 Fire Ball을 꼭 참고해주세요! – 설명을 자세히 하지 않는 부분은 이전에 설명을 했던 부분입니다.***

**▶ Pipe객체에서 Water객체가 흐르는 이번 예제를 위해서는 다음과 같은 순서로 진행을 할 것입니다.**

1. 필요한 js라이브러리 연결시키기
2. HTML문서 Style태그 설정
3. HTML문서 Div태그로 우측 Texture선택란 만들기
4. 필요 변수 선언
5. 우측 각 Texture 선택시 Water의 Mesh대체하기
6. Init( )함수 선언 및 정의
7. addShadowedLight( ) 함수 선언 및 정의
8. onWindowResize( ) 함수 선언 및 정의
9. loop( ) 함수 선언 및 정의
10. render( ) 함수 선언 및 정의
11. 전체 코드 공개.
12. 만들어진 최종 결과물 객체 확인하기

*(부분적인 소스코드를 설명하며 마지막에 전체 코드가 첨부됩니다. 각 Step별로 해당 Line 소스코드에 집중해주세요!)*

**Step1. 필요한 js라이브러리 연결시키기**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 이번 Pipe Water예제를 수행하기 위하여 필요한 라이브러리는 총17가지이다.   1. three.min.js 2. OrbitControls.js 3. Detector.js 4. Stat.min.js 5. tween.min.js 6. VRMLLoader.js 7. OBJLoader.js 8. CopyShader.js 9. BokehShader.js 10. HorizontalBlurShader.js 11. VerticalBlurShader.js 12. EffectComposer.js 13. RenderPass.js 14. ShaderPass.js 15. MaskPass.js 16. BokehPass.js 17. ParticleSystem.js   Three.js라이브러리를 사용하기위하여 three.min.js와 추가적인 16가지가 있다.  카메라를 조절할 OrbitControls.js가 있고, 접속자가 webgl 가능환경인지 아닌지 검사하는데에 필요한 Detector.js가 있다. 이외에 Loader에 필요한 라이브러리와 Shader에 필요한 라이브러리 후 처리에 필요한 라이브러리들이 있다.  또한 이전에도 사용하였던 핵심 라이브러리인 ParticlaSystem.js가 있다.  <코드1>의 5~6줄은 meta태그를 이용하여 HTML문서 설정을 하는 부분이다.  <코드 1>의 8~31번째줄의 부분이 상대경로를 이용한 js라이브러리를 연결시키는 부분이다. | |  |  |  | | --- | --- | --- | | 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 | <!DOCTYPE html>  <html lang="en">  <head>      <title>three.js webgl - PLY</title>      <meta charset="utf-8">      <meta name="viewport" content="width=device-width, user-scalable=no, minimum-scale=1.0, maximum-scale=1.0">        <script src="./lib/three/build/three.min.js"></script>      <script src="./lib/three/examples/js/controls/OrbitControls.js"></script>          <script src="./lib/three/examples/js/Detector.js"></script>      <script src="./lib/three/examples/js/libs/stats.min.js"></script>      <script src="./lib/three/examples/js/libs/tween.min.js"></script>        <script src="./lib/three/examples/js/loaders/VRMLLoader.js"></script>      <script src="./lib/three/examples/js/loaders/OBJLoader.js"></script>        <script src="./lib/three/examples/js/shaders/CopyShader.js"></script>      <script src="./lib/three/examples/js/shaders/BokehShader.js"></script>        <script src="./lib/three/examples/js/shaders/HorizontalBlurShader.js"></script>      <script src="./lib/three/examples/js/shaders/VerticalBlurShader.js"></script>        <script src="./lib/three/examples/js/postprocessing/EffectComposer.js"></script>      <script src="./lib/three/examples/js/postprocessing/RenderPass.js"></script>      <script src="./lib/three/examples/js/postprocessing/ShaderPass.js"></script>      <script src="./lib/three/examples/js/postprocessing/MaskPass.js"></script>      <script src="./lib/three/examples/js/postprocessing/BokehPass.js"></script>        <script src="./ParticleSystem.js"></script> |  | |

**<코드1>**

**Step2. HTML문서 Style태그 설정**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| <코드 2>의 33~39번째줄의 부분은 body부분의 style을 설정해주는 부분이다. 여백을 0으로 주고, 배경색과 글씨와 같은 속성의 표현범위를 설정하는 overflow와 글꼴을 설정한다.  <코드 2>의 41~42번째줄의 부분은 menu태그의 속성을 설정하는 부분이다. 색상, 글씨굵기, 크기등을 설정하고, munu의 img와 canvas의 너비를 75px로 여백을 10px로 설정하였다.  <코드 2>의 44~46번째줄은 이미지와 배경, 라벨에 관한 속성을 설정하는 부분이다.  <코드 2>의 48~50번째줄은 lb에 대한 각종 속성을 설정하는 부분이다.  <코드 2>의 52~59번째줄은 bnt에 대하 각종 속성을 설정하는 부분이다. | |  |  |  | | --- | --- | --- | | 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 | <style>          body {              margin: 0px;              background-color: #111;              overflow: hidden;              font-family: arial;          }          .menu { color: #fff; font-weight: bold; font-size: 12px; z-index: 100; width: 75px; position: absolute; top: 0px; padding: 16px; }          .menu img, .menu canvas { width: 75px; margin: 10px 0 }          #images { background: rgba(0,0,0,0); right: 100px; }          #backgrounds { background: rgba(0,0,0,0.0); left: 107px; }          #labels { background: rgba(0,0,0,0.75); left: 214px; width: 100px }          .lbl { color: #fff; z-index: 150; float:left; padding: 0.25em; width: 75px; display: block  }          #lbl\_dst { background:#800; }          #lbl\_src { background:green; }          .btn { background: darkorange; width: 100px; cursor: pointer }          #btn\_sub { background: transparent }          #btn\_rsub { background: transparent }          #btn\_pre { background: transparent }          #btn\_rsub, #btn\_nopre { margin-bottom: 2em }      </style>    </head> |  | |

**<코드2>**

**Step3. HTML문서 Div태그로 우측 Texture선택란 만들기**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| <코드 3>의 67번째줄을 div태그를 시작하는 부분으로써 id값은 images로 받고 class형식은 menu로 설정하였다.  <코드 3>의 68번째줄은 결과창 우측창의 Texture선택란의 상단 제목으로 출력할 문자열을 설정하였다.  <코드 3>의 69~77번째줄을 각 image의 종류에 따라 그에 맞는 이미지 texture파일을 설정하고 클릭시 onclick이벤트가 일어나게 설정하였다.  (하단에서 클릭 시 Tuxture를 변환하여 Water를 표현하도록 설정하는데 사용된다.) | |  |  |  | | --- | --- | --- | | 64  65  66  67  68  69  70  71  72  73  74  75  76  77 | <body>        <div id="images" class="menu">          Captured Materials          <a id="img\_0" href="#" onclick="onclickImg0()"><img src="textures/gradient.png" /></a>          <a id="img\_1" href="#" onclick="onclickImg1()"><img src="textures/matcaps/blue.jpg" /></a>          <a id="img\_2" href="#" onclick="onclickImg2()"><img src="textures/matcaps/permalink.jpg" /></a>          <a id="img\_3" href="#" onclick="onclickImg3()"><img src="textures/matcaps/rough-aluminium.jpg" /></a>          <a id="img\_4" href="#" onclick="onclickImg4()"><img src="textures/matcaps/fireball.jpg" /></a>          <a id="img\_5" href="#" onclick="onclickImg5()"><img src="textures/matcaps/red.jpg" /></a>          <a id="img\_6" href="#" onclick="onclickImg6()"><img src="textures/matcaps/green.jpg" /></a>          <a id="img\_7" href="#" onclick="onclickImg7()"><img src="textures/matcaps/emerald.jpg" /></a>      </div> |  | |

**<코드3>**

**Step4. 필요 변수 선언**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| <코드 4>의 81번째 줄은 Detector라이브러리의 Detector를 호출하여 접속자가 webgl 가능환경인지 아닌지를 검사하는 부분이다.  <코드 4>의 83~89번째 줄은 필요한 변수들인 컨테이너변수 통계치로 사용할 변수인 stats변수 main객체변수들과 핵심 변수들인 ps와 fog변수를 선언한다.  ps변수는 particlesystem을 줄여 표현한 변수명이며, fog변수는 Water를 표현하기위해 필요한 변수이다.  highRendered변수를 선언하여 true로 초기화한다. | |  |  |  | | --- | --- | --- | | 78  79  80  81  82  83  84  85  86  87  88  89 | <script>            if ( ! Detector.webgl ) Detector.addGetWebGLMessage();            var container, stats;          var camera, cameraTarget, scene, renderer;            var ps;          var fog = new THREE.Fog( 0x72645b, 2, 15 );            var highRendered = true; |  | |

**<코드4>**

**Step5. 우측 각 Texture선택시 Water의 Mesh대체하기**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| <코드 5>의 91~94번째줄은 img\_0을 선택했을 시 일어나는 이벤트인 onclicking0( )이벤트 함수를 선언하는 부분이다. Scene에 particlesystem의 Mesh를 가져와 붙인다. (이때는 pointMesh로서 직접 Mesh를 만든다.) 그리고 scene에 기존에 있던(스크린에 있는)Mesh를 가져와서 제거한다.  <코드 5>의 95~99번째줄의 각 5개의 줄은 위의 img\_0과는 약간 다르다. 먼저 Particlesystem으로서 파라미터를 해당jpg이미지파일로 설정한다. 이후 Particlesystem라이브러리에서 직접 만드는 Mesh를 제거하고 해당이미지파일을 screen에 Mesh로서 붙인다.  <코드 5>의 100~129번째 줄의 나머지 6부분도 onclickimg1과 똑 같은 방식으로 설정한다.  <코드 5>의 131~132번째 줄은 init( )과 loop( )함수의 실행 부분이다. 선언 및 정의 부분은 밑에서 다루기로 한다. | |  |  |  | | --- | --- | --- | | 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 | function onclickImg0() {              scene.add(ps.getMesh());              scene.remove(ps.getScreenMesh());          }          function onclickImg1() {              ps.setParameters({ matcapFile: "./textures/matcaps/blue.jpg" });              scene.remove(ps.getMesh());              scene.add(ps.getScreenMesh());          }          function onclickImg2() {              ps.setParameters({ matcapFile: "./textures/matcaps/permalink.jpg" });              scene.remove(ps.getMesh());              scene.add(ps.getScreenMesh());          }          function onclickImg3() {              ps.setParameters({ matcapFile: "./textures/matcaps/rough-aluminium.jpg" });              scene.remove(ps.getMesh());              scene.add(ps.getScreenMesh());          }          function onclickImg4() {              ps.setParameters({ matcapFile: "./textures/matcaps/fireball.jpg" });              scene.remove(ps.getMesh());              scene.add(ps.getScreenMesh());          }          function onclickImg5() {              ps.setParameters({ matcapFile: "./textures/matcaps/red.jpg" });              scene.remove(ps.getMesh());              scene.add(ps.getScreenMesh());          }          function onclickImg6() {              ps.setParameters({ matcapFile: "./textures/matcaps/green.jpg" });              scene.remove(ps.getMesh());              scene.add(ps.getScreenMesh());          }          function onclickImg7() {              ps.setParameters({ matcapFile: "./textures/matcaps/emerald.jpg" });              scene.remove(ps.getMesh());              scene.add(ps.getScreenMesh());          }            init();          loop(); |  | |

**<코드5>**

**Step6. Init( )함수 선언 및 정의**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| <코드 6>의 136~137번째 줄은 문서의 div태그부분을 컨테이너 변수에 저장하고 child로서 body부분에 붙이는 부분이다.  <코드 6>의 139~159번째줄은 카메라를 설정하는 부분이다.  (앞선 Tutorial 예제들을 참고 바란다.)  <코드 6>의 161~162번째 줄은 Scene변수를 선언하고 scene의 fog에 이전에 선언한 fog변수를 저장하는 부분이다.  <코드 6>의 165~166번째 줄은 ps변수를 선언하고 초기화 하는 부분이다.  (여기서의 ps변수는 전역변수의 ps와는 다른 변수이다.)  <코드 6>의 168~178번째 줄은 ps변수의 파라미터들을 설정하는 부분이다. 이전의 설명했던 부분을 제외하고 추가적으로 설명할 부분만 설명하자면,  globalForce는 힘이 가해지는 방향을 설정하는데 여기서 y축으로 -10만큼이므로 중력과 같은 힘으로 아래로 떨어지게 하였다. 바람의 세기인 windStrength는 0.05로 설정하였고, texFile과 matcapFile을 적당한 이미지파일로 설정한다. alpha값은 투명도를 나타내는 수치값인데 이는 Water의 사실성을 표현하는데 중요하다.  (따로 설명하지 않은 부분은 앞선 Tutorial 예제들을 참고 바란다.)  <코드 6>의 181~201번째 줄은 plane과 sphere Mesh를 만드는 부분이다. 각 Mesh의 특성들을 설정하고, scene에 붙인다.  (자세히 설명하지 않은 부분은 앞선 예제를 꼭 참고 해보길 바란다.)  <코드 6>의 205~226번째 줄은 pipe객체를 설정해주는 부분이다. main객체와 같이 자주 사용되는 loader객체를 이용하여 객체로더로 pipe객체를 로드한다. 로드할때, meterial을 만들고 child로서 그 값들을 설정하면서 로드하였다. 만들어진 object객체의 position y값, scale, rotation x값들을 설정하고 scene에 만들어진 pipe object객체를 붙인다.  <코드 6>의 229번째 줄은 위에서 만들어 놓은 ps변수를 이용하여 초기로 선택될 mesh를 화면에 보여주는 부분이다.  <코드 6>의 232~250번째 줄은 먼저 objectField를 만들고 초기화 한다 이후 각각 planeObject와 sphereObject각각 만들고 초기화 하며, objectField에 관하여 장애물로서 위의 2가지를 add한다. 마지막으로 ps의 장애물 Field에 정리가 되어있는 objectField를 add한다.  <코드 6>의 254~257번째 줄은 main객체인 Light객체에 관한 내용이다.  <코드 6>의 261~270번째 줄은 main 객체인 renderer객체에 관한 내용이다. 이전과 조금 다른 점이 있다면, 그림자를 만들어주기 위하여 코드 4줄정도가 추가되었다. 이후 renderer.domElement를 컨테이너의 child로 붙인다.  <코드 6>의 274~277번째 줄은 Stat( )함수를 사용하여, 통계에 필요한 수치들을 설정하고 renderer와 똑같이 컨테이너의 child로 붙인다.  <코드 6>의 281번째 줄은 윈도우 창resize에 관한 소스이다.  (따로 설명하지 않은 부분은 앞선 Tutorial 예제들을 참고 바란다.) | |  |  |  | | --- | --- | --- | | 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  246  247  248  249  250  251  252  253  254  255  256  257  258  259  260  261  262  263  264  265  266  267  268  269  270  271  272  273  274  275  276  277  278  279  280  281  282 | function init() {                container = document.createElement( 'div' );              document.body.appendChild( container );                camera = new THREE.PerspectiveCamera( 35, window.innerWidth / window.innerHeight, 1, 100 );                camera.position.y = 0.8;              camera.position.x = 2.2;              camera.position.z = 3;                controls = new THREE.OrbitControls(camera);                controls.rotateSpeed = 0.3;                controls.noZoom = false;              controls.noPan = true;                controls.staticMoving = true;              controls.dynamicDampingFactor = 0.3;                controls.minPolarAngle = 0;              controls.maxPolarAngle = Math.PI;                controls.minAzimuthAngle = -Math.PI\*0.4;              controls.maxAzimuthAngle =  Math.PI\*0.4;                scene = new THREE.Scene();              scene.fog = fog;                // initialize particle system;              ps = new ParticleSystem();              ps.initialize(10000);                ps.setParameters({seedVelDir: new THREE.Vector3(0,0,1),                                seedVelMag: 2.0,                                globalForce: new THREE.Vector3(0, -10,0),                                windStrength: 0.05,                                seedSize: 0.6,                                seedLife: 2.0,                                texFile : "./textures/gradient.png",                                matcapFile: "./textures/matcaps/blue.jpg",                                particleColor: new THREE.Color(0.1, 0.1, 1.0),                                alpha : 0.7              });                // Ground              var plane = new THREE.Mesh(                      new THREE.PlaneBufferGeometry( 40, 40 ),                      new THREE.MeshPhongMaterial( { color: 0x555555, specular: 0x000000 } )              );              plane.rotation.x = -Math.PI/2;              plane.position.y = -0.55;              plane.receiveShadow = true;                scene.add( plane );                  var sphereMesh = new THREE.Mesh(                      new THREE.SphereGeometry(0.2,100,100),                      new THREE.MeshPhongMaterial( { color: 0x111111, specular: 0x111111, shininess: 100 } )              );                sphereMesh.receiveShadow = true;              sphereMesh.castShadow = true;              sphereMesh.position.set(-0.57,-0.4,0.9);                scene.add( sphereMesh );                  // Loader              var loader = new THREE.OBJLoader();              loader.load('./models/pipe.obj', function(object){                    var material = new THREE.MeshPhongMaterial( { color: 0x111111, specular: 0x111111, shininess: 30 });                  material.side = THREE.DoubleSide;                    object.traverse( function ( child ) {                        if ( child instanceof THREE.Mesh ) {                            child.material = material;                          child.castShadow = true;                          child.receiveShadow = true;                      }                  });                    object.position.y = -0.5;                  object.scale.set(0.02, 0.02, 0.02);                  object.rotation.x = -Math.PI\*0.5;                    scene.add(object);              });                //scene.add(ps.getMesh());              scene.add(ps.getScreenMesh());                //              var objectField = new ObjectField();              objectField.initialize();                var planeObject = new PlaneObject();              planeObject.initialize(new THREE.Vector3(0,-0.5,0), new THREE.Vector3(0,1,0));              planeObject.setParameters(0.3, 0.8);                objectField.addObstacle(planeObject);                  var sphereObject = new SphereObject();              sphereObject.initialize(new THREE.Vector3(-0.57,-0.4,0.9), 0.2);              sphereObject.setParameters(0.3, 0.8);                  objectField.addObstacle(sphereObject);                  ps.setObstacleField(objectField);                // Lights                scene.add( new THREE.AmbientLight( 0x777777 ) );                addShadowedLight( 1, 1, 1, 0xffffff, 1.35 );              addShadowedLight( 0.5, 1, -1, 0xffaa00, 1 );                // renderer                renderer = new THREE.WebGLRenderer( { antialias: true, alpha:true } );              renderer.setClearColor( fog.color, 0.0 );              renderer.setSize( window.innerWidth, window.innerHeight );                renderer.gammaInput = true;              renderer.gammaOutput = true;              renderer.shadowMapEnabled = true;              renderer.shadowMapCullFace = THREE.CullFaceBack;                container.appendChild( renderer.domElement );                // stats                stats = new Stats();              stats.domElement.style.position = 'absolute';              stats.domElement.style.top = '0px';              container.appendChild( stats.domElement );                // resize                window.addEventListener( 'resize', onWindowResize, false );          } |  | |

**<코드6>**

**Step7. addShadowedLight( )함수 선언 및 정의**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| <코드 7>의 284~308번째 줄은 init( )함수에서 사용한 addShadowedLight( )함수에 대한 정의 부분이다.  Line 284 : addShadowedLight함수는 매개변수로 5가지를 받는다.   * X : 절대위치의 x축 값을 의미한다. * Y : 절대위치의 y축 값을 의미한다. * Z : 절대위치의 z축 값을 의미한다. * Color : 빛의 색상을 의미하며 RGB값으로 표현한다. * Intensity : 빛의 강도를 의미한다.   Line 286~288 : directionLight라는 빛의 방향성을 결정할 객체를 선언한다. 객체의 위치를 x, y, z로 설정을 한 뒤 scene에 붙인다.  Line 290 : 앞서 만든 directionLight라는 객체에 그림자효과를 true로 설정한다.  Line 293 : 1과 -1로 사용할 변수 d선언  Line 294~297 : 각 좌, 우, 위, 아래의 카메라가 그림자를 표현하는 수치를 1과 -1로 조절한다.  Line 299~300 : 카메라가 그림자를 식별하는 거리감을 near와 far로 조절한다.  Line 302~303 : 픽셀적 수치에 있어 그림자 맵의 텍스쳐 너비와 높이를 설정한다.  Line 305~306 : 그림자의 표면을 결정할 때, 표준화된 깊이로부터 얼마의 수치만큼 빼고 더할 것인지를 정하는 Bias와 그림자의 어두운 정도를 표현하는 Darkness를 설정한다.  (꼭 앞선 예제들을 공부하고 보길 바란다.) | |  |  |  | | --- | --- | --- | | 283  284  285  286  287  288  289  290  291  292  293  294  295  296  297  298  299  300  301  302  303  304  305  306  307  308 | function addShadowedLight( x, y, z, color, intensity ) {                var directionalLight = new THREE.DirectionalLight( color, intensity );              directionalLight.position.set( x, y, z )              scene.add( directionalLight );                directionalLight.castShadow = true;              // directionalLight.shadowCameraVisible = true;                var d = 1;              directionalLight.shadowCameraLeft = -d;              directionalLight.shadowCameraRight = d;              directionalLight.shadowCameraTop = d;              directionalLight.shadowCameraBottom = -d;                directionalLight.shadowCameraNear = 1;              directionalLight.shadowCameraFar = 4;                directionalLight.shadowMapWidth = 1024;              directionalLight.shadowMapHeight = 1024;                directionalLight.shadowBias = -0.005;              directionalLight.shadowDarkness = 0.15;            } |  | |

**<코드7>**

**Step8. onWindowResize( )함수 선언 및 정의**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| <코드 8>의 310~317번째 줄은 window창을 변경했을 때 다시 렌더링을 해주는 onWindowResize( )함수에 대한 부분이다.  (설명하지 않은 부분은 Tutorial2와 Tutorial2\_1을 꼭 참고해주세요!) | |  |  |  | | --- | --- | --- | | 309  310  311  312  313  314  315  316  317 | function onWindowResize() {                camera.aspect = window.innerWidth / window.innerHeight;              camera.updateProjectionMatrix();                renderer.setSize( window.innerWidth, window.innerHeight );            } |  | |

**<코드8>**

**Step9. loop( )함수 선언 및 정의**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| <코드 9>의 319~320번째 줄은 fps변수의 선언과 초기화를 하고 time변수도 하나 만든다.  <코드 9>의 321~342번재 줄은 loop함수에 관한 선언 및 정의를 하는 부분이다.   * setTimeout은 1초/fps초마다 loop가 실행되게 하였다. * 현재시간(now)과 시간차(dt)변수를 만들어 설정하고 계산하여 현시간은 time변수에 저장한다. * Stats을 이용하여 통계적 계산을 시작하게하였다. * Cen과 rad을 설정하고 ps의 각종 particle관련하여 설정한뒤 controls를 없데이트 한다. * 이후 requestAnimationFrame을 사용하여 render를 랜더링 해준다. * Stats을 이용하여 통계적 계산을 init( )함수 내에서 종료하게 한다. | |  |  |  | | --- | --- | --- | | 318  319  320  321  322  323  324  325  326  327  328  329  330  331  332  333  334  335  336  337  338  339  340  341  342 | var fps = 40;          var time;          function loop() {                setTimeout(loop, 1000/fps);                var now = new Date().getTime();              var dt = (now - (time||now))/1000;              time = now;                  stats.begin();                var cen = new THREE.Vector3(-0.57,0.24,0.24);              var rad = 0.11;                ps.addParticlesFromSphere(50, cen, rad);              ps.updateParticles(dt);              controls.update();                requestAnimationFrame(render);                stats.end();          } | [cs](http://colorscripter.com/info#e) | |

**<코드9>**

**Step10. render( )함수 선언 및 정의**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| <코드 10>의 344~352번째 줄은 render( )함수에 관한 내용이다.  <코드 10>의 346~347번째줄은 renderer가 그리는 색상을 fog.color로 클리어 하고 ps를 update하는데 renderer, scene, camera를 모두 적용하여 update한다.  <코드 10>의 349~350번째줄은 다시 한번 renderer가 그리는 색상을 fog,color로 클리어 하고 renderer.render를 사용하여 scene과 camera객체를 다시 랜더링 해준다. | |  |  |  | | --- | --- | --- | | 343  344  345  346  347  348  349  350  351  352  353  354  355  356 | function render() {                renderer.setClearColor( fog.color, 0.0 );              ps.updateTexture(renderer, scene, camera);                renderer.setClearColor( fog.color, 1.0 );              renderer.render(scene, camera);            }      </script>    </body> |  | |

**<코드10>**

**Step11. 전체 코드 공개**

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | | 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  246  247  248  249  250  251  252  253  254  255  256  257  258  259  260  261  262  263  264  265  266  267  268  269  270  271  272  273  274  275  276  277  278  279  280  281  282  283  284  285  286  287  288  289  290  291  292  293  294  295  296  297  298  299  300  301  302  303  304  305  306  307  308  309  310  311  312  313  314  315  316  317  318  319  320  321  322  323  324  325  326  327  328  329  330  331  332  333  334  335  336  337  338  339  340  341  342  343  344  345  346  347  348  349  350  351  352  353  354  355  356 | <!DOCTYPE html>  <html lang="en">  <head>      <title>three.js webgl - PLY</title>      <meta charset="utf-8">      <meta name="viewport" content="width=device-width, user-scalable=no, minimum-scale=1.0, maximum-scale=1.0">        <script src="./lib/three/build/three.min.js"></script>      <script src="./lib/three/examples/js/controls/OrbitControls.js"></script>          <script src="./lib/three/examples/js/Detector.js"></script>      <script src="./lib/three/examples/js/libs/stats.min.js"></script>      <script src="./lib/three/examples/js/libs/tween.min.js"></script>        <script src="./lib/three/examples/js/loaders/VRMLLoader.js"></script>      <script src="./lib/three/examples/js/loaders/OBJLoader.js"></script>        <script src="./lib/three/examples/js/shaders/CopyShader.js"></script>      <script src="./lib/three/examples/js/shaders/BokehShader.js"></script>        <script src="./lib/three/examples/js/shaders/HorizontalBlurShader.js"></script>      <script src="./lib/three/examples/js/shaders/VerticalBlurShader.js"></script>        <script src="./lib/three/examples/js/postprocessing/EffectComposer.js"></script>      <script src="./lib/three/examples/js/postprocessing/RenderPass.js"></script>      <script src="./lib/three/examples/js/postprocessing/ShaderPass.js"></script>      <script src="./lib/three/examples/js/postprocessing/MaskPass.js"></script>      <script src="./lib/three/examples/js/postprocessing/BokehPass.js"></script>        <script src="./ParticleSystem.js"></script>        <style>          body {              margin: 0px;              background-color: #111;              overflow: hidden;              font-family: arial;          }          .menu { color: #fff; font-weight: bold; font-size: 12px; z-index: 100; width: 75px; position: absolute; top: 0px; padding: 16px; }          .menu img, .menu canvas { width: 75px; margin: 10px 0 }          #images { background: rgba(0,0,0,0); right: 100px; }          #backgrounds { background: rgba(0,0,0,0.0); left: 107px; }          #labels { background: rgba(0,0,0,0.75); left: 214px; width: 100px }          .lbl { color: #fff; z-index: 150; float:left; padding: 0.25em; width: 75px; display: block  }          #lbl\_dst { background:#800; }          #lbl\_src { background:green; }          .btn { background: darkorange; width: 100px; cursor: pointer }          #btn\_sub { background: transparent }          #btn\_rsub { background: transparent }          #btn\_pre { background: transparent }          #btn\_rsub, #btn\_nopre { margin-bottom: 2em }      </style>    </head>    <body>    <div id="images" class="menu">      Captured Materials      <a id="img\_0" href="#" onclick="onclickImg0()"><img src="textures/gradient.png" /></a>      <a id="img\_1" href="#" onclick="onclickImg1()"><img src="textures/matcaps/blue.jpg" /></a>      <a id="img\_2" href="#" onclick="onclickImg2()"><img src="textures/matcaps/permalink.jpg" /></a>      <a id="img\_3" href="#" onclick="onclickImg3()"><img src="textures/matcaps/rough-aluminium.jpg" /></a>      <a id="img\_4" href="#" onclick="onclickImg4()"><img src="textures/matcaps/fireball.jpg" /></a>      <a id="img\_5" href="#" onclick="onclickImg5()"><img src="textures/matcaps/red.jpg" /></a>      <a id="img\_6" href="#" onclick="onclickImg6()"><img src="textures/matcaps/green.jpg" /></a>      <a id="img\_7" href="#" onclick="onclickImg7()"><img src="textures/matcaps/emerald.jpg" /></a>  </div>    <script>        if ( ! Detector.webgl ) Detector.addGetWebGLMessage();        var container, stats;      var camera, cameraTarget, scene, renderer;        var ps;      var fog = new THREE.Fog( 0x72645b, 2, 15 );        var highRendered = true;        function onclickImg0() {          scene.add(ps.getMesh());          scene.remove(ps.getScreenMesh());      }      function onclickImg1() {          ps.setParameters({ matcapFile: "./textures/matcaps/blue.jpg" });          scene.remove(ps.getMesh());          scene.add(ps.getScreenMesh());      }      function onclickImg2() {          ps.setParameters({ matcapFile: "./textures/matcaps/permalink.jpg" });          scene.remove(ps.getMesh());          scene.add(ps.getScreenMesh());      }      function onclickImg3() {          ps.setParameters({ matcapFile: "./textures/matcaps/rough-aluminium.jpg" });          scene.remove(ps.getMesh());          scene.add(ps.getScreenMesh());      }      function onclickImg4() {          ps.setParameters({ matcapFile: "./textures/matcaps/fireball.jpg" });          scene.remove(ps.getMesh());          scene.add(ps.getScreenMesh());      }      function onclickImg5() {          ps.setParameters({ matcapFile: "./textures/matcaps/red.jpg" });          scene.remove(ps.getMesh());          scene.add(ps.getScreenMesh());      }      function onclickImg6() {          ps.setParameters({ matcapFile: "./textures/matcaps/green.jpg" });          scene.remove(ps.getMesh());          scene.add(ps.getScreenMesh());      }      function onclickImg7() {          ps.setParameters({ matcapFile: "./textures/matcaps/emerald.jpg" });          scene.remove(ps.getMesh());          scene.add(ps.getScreenMesh());      }        init();      loop();        function init() {            container = document.createElement( 'div' );          document.body.appendChild( container );            camera = new THREE.PerspectiveCamera( 35, window.innerWidth / window.innerHeight, 1, 100 );            camera.position.y = 0.8;          camera.position.x = 2.2;          camera.position.z = 3;            controls = new THREE.OrbitControls(camera);            controls.rotateSpeed = 0.3;            controls.noZoom = false;          controls.noPan = true;            controls.staticMoving = true;          controls.dynamicDampingFactor = 0.3;            controls.minPolarAngle = 0;          controls.maxPolarAngle = Math.PI;            controls.minAzimuthAngle = -Math.PI\*0.4;          controls.maxAzimuthAngle =  Math.PI\*0.4;            scene = new THREE.Scene();          scene.fog = fog;            // initialize particle system;          ps = new ParticleSystem();          ps.initialize(10000);            ps.setParameters({seedVelDir: new THREE.Vector3(0,0,1),              seedVelMag: 2.0,              globalForce: new THREE.Vector3(0, -10,0),              windStrength: 0.05,              seedSize: 0.6,              seedLife: 2.0,              texFile : "./textures/gradient.png",              matcapFile: "./textures/matcaps/blue.jpg",              particleColor: new THREE.Color(0.1, 0.1, 1.0),              alpha : 0.7          });            // Ground          var plane = new THREE.Mesh(                  new THREE.PlaneBufferGeometry( 40, 40 ),                  new THREE.MeshPhongMaterial( { color: 0x555555, specular: 0x000000 } )          );          plane.rotation.x = -Math.PI/2;          plane.position.y = -0.55;          plane.receiveShadow = true;            scene.add( plane );              var sphereMesh = new THREE.Mesh(                  new THREE.SphereGeometry(0.2,100,100),                  new THREE.MeshPhongMaterial( { color: 0x111111, specular: 0x111111, shininess: 100 } )          );            sphereMesh.receiveShadow = true;          sphereMesh.castShadow = true;          sphereMesh.position.set(-0.57,-0.4,0.9);            scene.add( sphereMesh );              // Loader          var loader = new THREE.OBJLoader();          loader.load('./models/pipe.obj', function(object){                var material = new THREE.MeshPhongMaterial( { color: 0x111111, specular: 0x111111, shininess: 30 });              material.side = THREE.DoubleSide;                object.traverse( function ( child ) {                    if ( child instanceof THREE.Mesh ) {                        child.material = material;                      child.castShadow = true;                      child.receiveShadow = true;                  }              });                object.position.y = -0.5;              object.scale.set(0.02, 0.02, 0.02);              object.rotation.x = -Math.PI\*0.5;                scene.add(object);          });            //scene.add(ps.getMesh());          scene.add(ps.getScreenMesh());            //          var objectField = new ObjectField();          objectField.initialize();            var planeObject = new PlaneObject();          planeObject.initialize(new THREE.Vector3(0,-0.5,0), new THREE.Vector3(0,1,0));          planeObject.setParameters(0.3, 0.8);            objectField.addObstacle(planeObject);              var sphereObject = new SphereObject();          sphereObject.initialize(new THREE.Vector3(-0.57,-0.4,0.9), 0.2);          sphereObject.setParameters(0.3, 0.8);              objectField.addObstacle(sphereObject);              ps.setObstacleField(objectField);            // Lights            scene.add( new THREE.AmbientLight( 0x777777 ) );            addShadowedLight( 1, 1, 1, 0xffffff, 1.35 );          addShadowedLight( 0.5, 1, -1, 0xffaa00, 1 );            // renderer            renderer = new THREE.WebGLRenderer( { antialias: true, alpha:true } );          renderer.setClearColor( fog.color, 0.0 );          renderer.setSize( window.innerWidth, window.innerHeight );            renderer.gammaInput = true;          renderer.gammaOutput = true;          renderer.shadowMapEnabled = true;          renderer.shadowMapCullFace = THREE.CullFaceBack;            container.appendChild( renderer.domElement );            // stats            stats = new Stats();          stats.domElement.style.position = 'absolute';          stats.domElement.style.top = '0px';          container.appendChild( stats.domElement );            // resize            window.addEventListener( 'resize', onWindowResize, false );      }        function addShadowedLight( x, y, z, color, intensity ) {            var directionalLight = new THREE.DirectionalLight( color, intensity );          directionalLight.position.set( x, y, z )          scene.add( directionalLight );            directionalLight.castShadow = true;          // directionalLight.shadowCameraVisible = true;            var d = 1;          directionalLight.shadowCameraLeft = -d;          directionalLight.shadowCameraRight = d;          directionalLight.shadowCameraTop = d;          directionalLight.shadowCameraBottom = -d;            directionalLight.shadowCameraNear = 1;          directionalLight.shadowCameraFar = 4;            directionalLight.shadowMapWidth = 1024;          directionalLight.shadowMapHeight = 1024;            directionalLight.shadowBias = -0.005;          directionalLight.shadowDarkness = 0.15;        }        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;              stats.begin();            var cen = new THREE.Vector3(-0.57,0.24,0.24);          var rad = 0.11;            ps.addParticlesFromSphere(50, cen, rad);          ps.updateParticles(dt);          controls.update();            requestAnimationFrame(render);            stats.end();      }        function render() {            renderer.setClearColor( fog.color, 0.0 );          ps.updateTexture(renderer, scene, camera);            renderer.setClearColor( fog.color, 1.0 );          renderer.render(scene, camera);        }    </script>    </body> |  | |

**<코드11>**

**Step12. 만들어진 최종 결과물 객체 확인하기**

(자~ 오늘은 실전 예제 “Pipe Water”를 배워 보았습니다! 역시 결과물을 확인해 봐야겠죠?)

▶ 실전 예제 Pipe Water입니다. 자세히 설명되지 않은 부분은 꼭 앞부분 Tutorial들을 공부하고 난 뒤 이해하고 넘어 갑시다!

* “Pipe Water” 결과물 링크 : <https://vimeo.com/155429624>

***지금까지 기본적인 내용들을 많이 배웠으니 다른 실전예제들도 쉽게 공부 하실 수 있습니다. ~ !***