WEBGL #2

Three.js

04 06, 2015 - 이준호

READY

- 예제 & 실습 - https://github.com/projectBS/webGL
- Three.js - https://github.com/mrdoob/three.js
- Editor - https://www.jetbrains.com/idea
- GitHub Tool (GUI) - http://www.sourcetreeapp.com
- BSWebGL - https://github.com/projectBS/bsWebGL
- BSWebGL Blog - http://www.bswebgl.com

Three.js Core

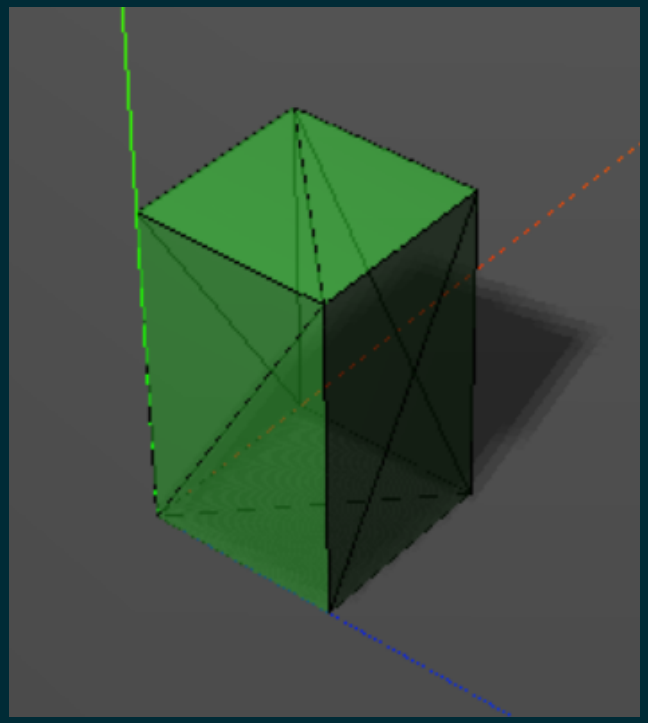
- Scene : 랜더링할 모든 오브젝트를 저장하는 컨테이너.
- Camera : Scene을 랜더링한 결과를 정의. 시점 기반.
- Renderer : Scene이 Camera 앵글에 기반하여 보여지도록 랜더링 연산.
- Object3D : 그릴 영역을 정의하는 Geometry, 영역의 Color를 정의하는 Material.
- Light : 3D 표현의 필수 요소.

DEMO

Mesh

```
var sphereGeometry = new THREE.SphereGeometry( 4, 20, 20 );
var sphereMaterial = new THREE.MeshBasicMaterial({color:0x7777FF});
var sphere = new THREE.Mesh( sphereGeometry, sphereMaterial );
```

- Geometry에서 오브젝트가 어떤 형태로 보여질지 정의. (그려질 영역)
- Material에서 영역이 어떻게 칠해질지 정의.
- Mesh에서 Geometry와 Material을 결합.



Geometry

- 기본은 3D 공간에서 점의 집합.
- 이 점들을 연결하여 생성된 face 들의 집합.
- 8개의 꼭지점 (Vertex)
- 6개의 면 (Face)

Three.js에 이미 준비되어 있는 Geometry들.

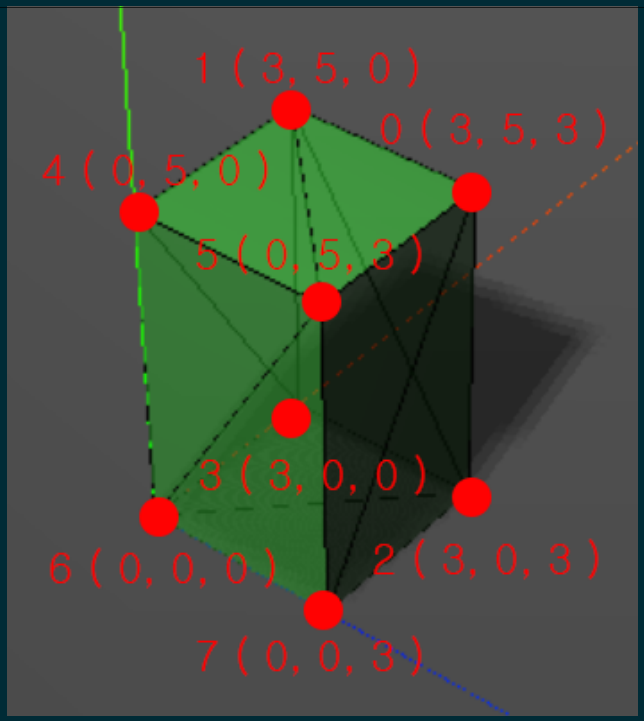
```
geoms = [
    new THREE.CylinderGeometry( 1, 4, 4 ),
    new THREE.BoxGeometry( 2, 2, 2),
    new THREE.SphereGeometry(2),
    new THREE.IcosahedronGeometry(4),
    new THREE.OctahedronGeometry(3),
    new THREE.TetrahedronGeometry(3),
    new THREE.TorusGeometry( 3, 1, 10, 10 ),
    new THREE.TorusKnotGeometry( 3, 0.5, 50, 20 )
];
```

- Material만 결합하면 Mesh를 생성하고 3D Scene에서 사용 가능
- Vertex나 Face를 직접 정의할 필요는 없음.
- Box 지오메트리를 만들기 위해서는 width, height, depth 만 지정하면 됨.
- Vertex와 Face를 Three.js 라이브러리에서 생성해 줌.

DEMO

CUSTOM GEOMETRY

Vertex와 Face를 직접 지정하는 것.




```
var vertices = [
    new THREE.Vector3( 3, 5, 3 ),    //0
    new THREE.Vector3( 3, 5, 0 ),    //1
    new THREE.Vector3( 3, 0, 3 ),    //2
    new THREE.Vector3( 3, 0, 0 ),    //3
    new THREE.Vector3( 0, 5, 0 ),    //4
    new THREE.Vector3( 0, 5, 3 ),    //5
    new THREE.Vector3( 0, 0, 0 ),    //6
    new THREE.Vector3( 0, 0, 3 )     //7
];

var faces = [
    new THREE.Face3( 0, 2, 1 ),
    new THREE.Face3( 2, 3, 1 ),     // left
    new THREE.Face3( 4, 6, 5 ),
    new THREE.Face3( 6, 7, 5 )      // right
```

- vertices : 각 점들의 위치를 정의.
- faces : 점들을 연결하여 삼각형 face를 정의.
- Material과 결합하여 Mesh를 생성할 수 있음.

DEMO

vertices update

```
/* 208 render loop */
mesh.geometry.vertices = vertices;      // face를 다시 정의할 필요는 없음. face는 여전히 연결되어 있음.
mesh.geometry.verticesNeedUpdate = true;   // 성능을 위해 업데이트 필요시 지정.
mesh.geometry.computeFaceNormals();   // face를 다시 계산.
```

clone

```
/* 100 gui */
var geometry = mesh.children[0].geometry.clone(), // Mesh 그룹의 첫번째 Mesh의 Geometry를 복사.
    materials = [
        new THREE.MeshLambertMaterial({opacity:0.6, color:0xFF44FF, transparent:true}),
        new THREE.MeshBasicMaterial({color:0x000000, wireframe:true})
    ],
    mesh2 = THREE.SceneUtils.createMultiMaterialObject( geometry, materials ); // 다른 Material과 결합.
```

multi material

```
/* 164 add mesh */
materials = [
    new THREE.MeshLambertMaterial({opacity:0.6, color:0x44FF44, transparent:true}),
    new THREE.MeshBasicMaterial({color:0x000000, wireframe:true})
];

geom.vertices = vertices;
geom.faces = faces;
geom.computeFaceNormals();

mesh = THREE.SceneUtils.createMultiMaterialObject( geom, materials);
mesh.children.forEach( function(e){e.castShadow = true; } ); // Mesh 그룹을 loop
scene.add(mesh);
```

- THREE.SceneUtils.createMultiMaterialObject(geometry, [material, material, ...])
- 각각의 Mesh를 생성하여 그룹을 지어주는 것

THANK YOU