WebGL / Three.js

1. WebGL이란?

- 웹 브라우저에서 제공하는 3차원 그래픽 자바스크립트 API
- Flash, Unity, Silverlight 등의 Plug-in들이 전혀 필요 없음.
- GPU가속지원
- OpenGL ES 2.0 기반
- 지원 웹브라우저 : Chrome 8.0+, Firefox 4.0+, Safari 5.1+, IE 10+ / 모바일 공식지원 안됨

2. Three.js란?

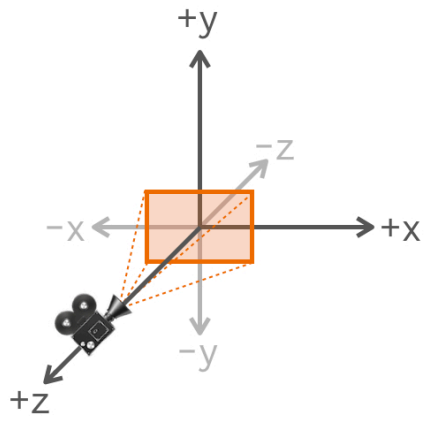
- WebGL의 생산성 향상을 위해 좀 더 단순화 시킨 경량 자바스크립트 라이브러리
- Three.min.js(370kb)
- 대략의 문법

```
var renderer, scene, camera, mesh;

function init() {
  renderer = new THREE.WebGLRenderer();
  renderer.setSize(innerWidth, innerHeight);
  document.body.appendChild(renderer.domElement);

  scene = new THREE.Scene();

  var aspect = innerWidth / innerHeight;
  camera = new THREE.PerspectiveCamera(50, aspect, 1, 10000);
  camera.position.z = 1000;
}
```

2. 요소

1) WebGL 좌표계

- 기본적으로 객체를 생성했을때의 좌표는 (x,y,z)=(0,0,0)이며 이는 컴퓨터 디스플레이에 일직선 상으로 배치 되어 있다는 것을 의미함.



2) Renderer

- 최종 결과물을 그려주는 객체
- WebGlRenderer, CanvasRenderer, SVGRenderer 등 다양한 곳에서 사용.

```
renderer = new THREE.WebGLRenderer();
renderer.setSize(innerWidth, innerHeight);
document.body.appendChild(renderer.domElement);
```

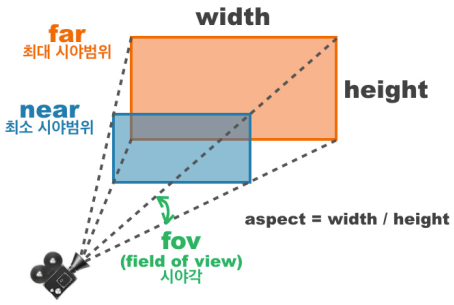
3) Scene

- 화면을 구성하는 기본 객체
- 단일 또는 여러 개의 Object나 Mesh, Light 등으로 구성.

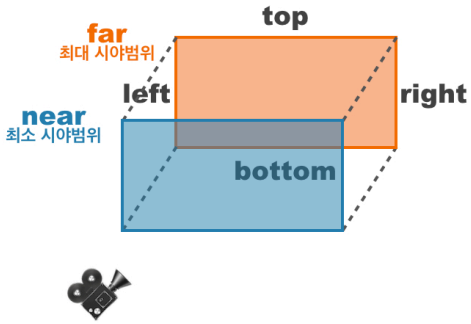
```
scene = new THREE.Scene();

scene.add(object);
scene.add(mesh);
scene.add(light);
```

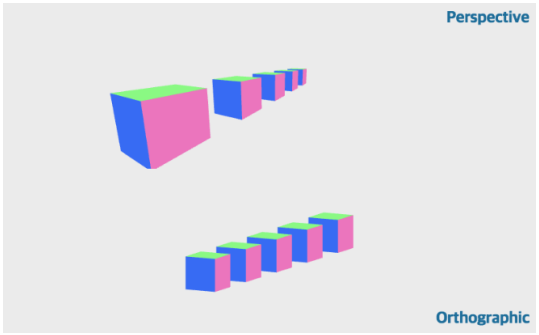
4) Camera

1. PerspectiveCamera
    a. 원근투영 / 투시투영 (Perspective Projection)
    b. 원근법에 따라 가까운 물체는 크게 먼 물체는 작게 보임.
    c. var camera = new THREE.PerspectiveCamera(fov, aspect, near, far);



2. OrthographicCamera
    a. 직교투영(Orthograpgic Projection)
    b. 바라보는 방향은 있으니 특정 지점(투시점)은 없음
    c. 같은 크기의 물체라면 거리에 상관없이 같은 크기로 보임.
    d. 이해가 잘 안되면 클래시 오브 클랜의 건물들 크기를 생각해보면 될 듯
    e. var camera = new THREE.OrthographicCamera(left, right, top, bottom, near, far);





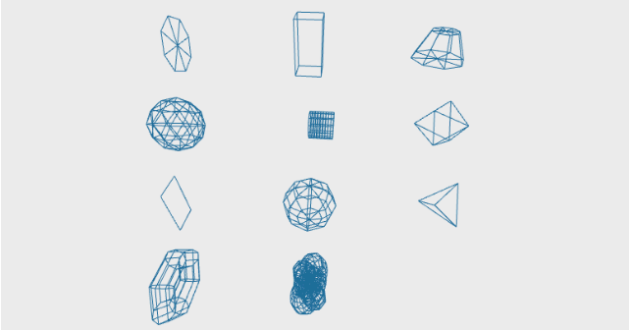
3. CubeCamera
4. CombinedCamera

5) Mesh

1. Mesh = Geometry + Material
2. 즉, 도형과 질감을 적용한 물체를 뜻함

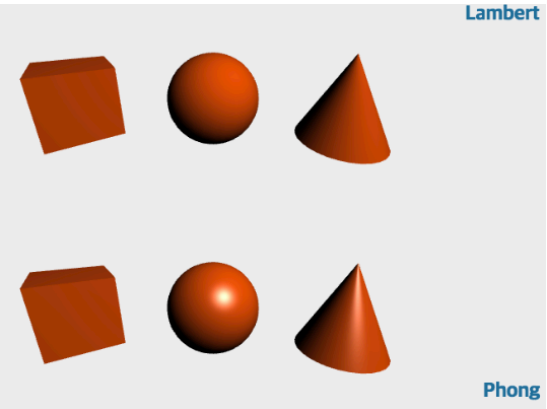
6) Geometry

1. 도형을 의미한다.
2. CircleGeometry, PlaneGeometry, CubeGeometry 등 도형이름 뒤에 Geometry가 붙는다.



7) Material

1. 재질을 의미한다.
2. MeshBasicMaterial, MeshLambertMaterial, MeshPhongMaterial 등



8) Mesh를 사용하기 위한 간단한 스크립트

1. Geomery 생성 = SphereGeometry(100) = 직경이 100인 구를 생성
2. Material 생성 = 색이 #f15023이고 와이어프레임으로 이루어진 BasicMaterial을 생성
3. Mesh 생성 = 위에서 생성한 Geometry와 material을 하나의 Mesh로 생성
4. 이렇게 생성된 Mesh를 Scene에 추가 / 안하면 안보인다.(ActionScript의 AddChild나 Andriod의 show()같은 개념)

```
var geometry = new THREE.SphereGeometry(100);

var material = new THREE.MeshBasicMaterial({
    color: 0xf15023,
    wireframe: true
});

mesh = new THREE.Mesh(geometry, material);

scene.add(mesh);
```

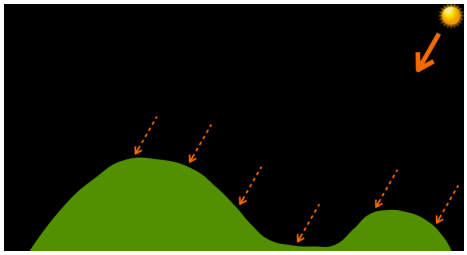
8) Light(http://fallroot.github.io/getting-started-webgl-with-threejs/#/54)

1. AmbientLight(주변광, 간접광)

- var light = new THREE.AmbientLight(hex);

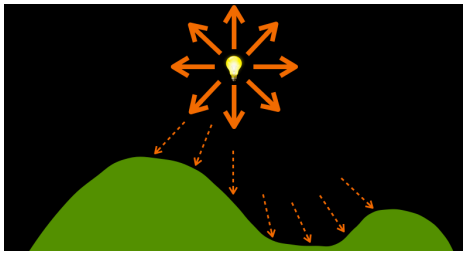
2. DirectionalLight(방향광)

- var light = new THREE.DirectionalLight(hex, intensity, distance);



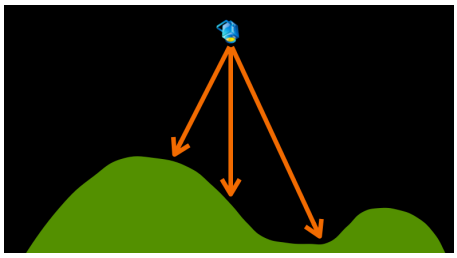
3. PointLight(점광)

- var light = new THREE.PointLight(color, intensity, distance);



4. SpotLight(집중광)

a. var light = new THREE.SpotLight(color, intensity, distance, castShadow);



8) Shadow

1. 조명의 세기와 방향을 반영.

```
renderer.shadowMapEnabled = true;

light.castShadow = true;

object.castShadow = true;
object.receiveShadow = true;
```

8) Texture

1. 물체의 표면을 이미지파일로 처리

```
var texture = THREE.ImageUtils.loadTexture('any.jpg');

new THREE.MeshBasicMaterial({
    map: texture
});
```

8) Model

1. 다른 어플리케이션에서 제작된 3차원 모델을 사용
2. Exporter : blender, ctm, fbx, max, obj, uft-8
3. Loader : ColladaLoader, OBJLoader, MTLLoader, CTMLoader, OBJMTLLoader, STLLoader, UTF8Loader, VTKLoader
4. 대략적인 Loader의 사용법

```
var loader = new THREE.OBJLoader();

loader.addEventListener('load', function(event) {
  var object = event.content;
  scene.add(object);
});

loader.load('model.obj');
```

참고 : http://fallroot.github.io/getting-started-webgl-with-threejs/#/