ThreeJS

Object3D

ThreeJS의 보이는 모든 것은 Object3D이다.

Object3D 속성과 메서드

위치

position, scale, matrix

translateX/Y/Z(distance),
translateOnAxis(axis, distance),
updateMatrix()

Object3D 속성과 메서드

회전

`rotation, eulerOrder, quaternion`

`rotateX/Y/Z(angle),`
`rotateOnAxis(axis, angle),`
`lookAt(vector)`

Object3D 속성과 메서드

트리

parent, children

add(object), remove(object),
traverse(callback)

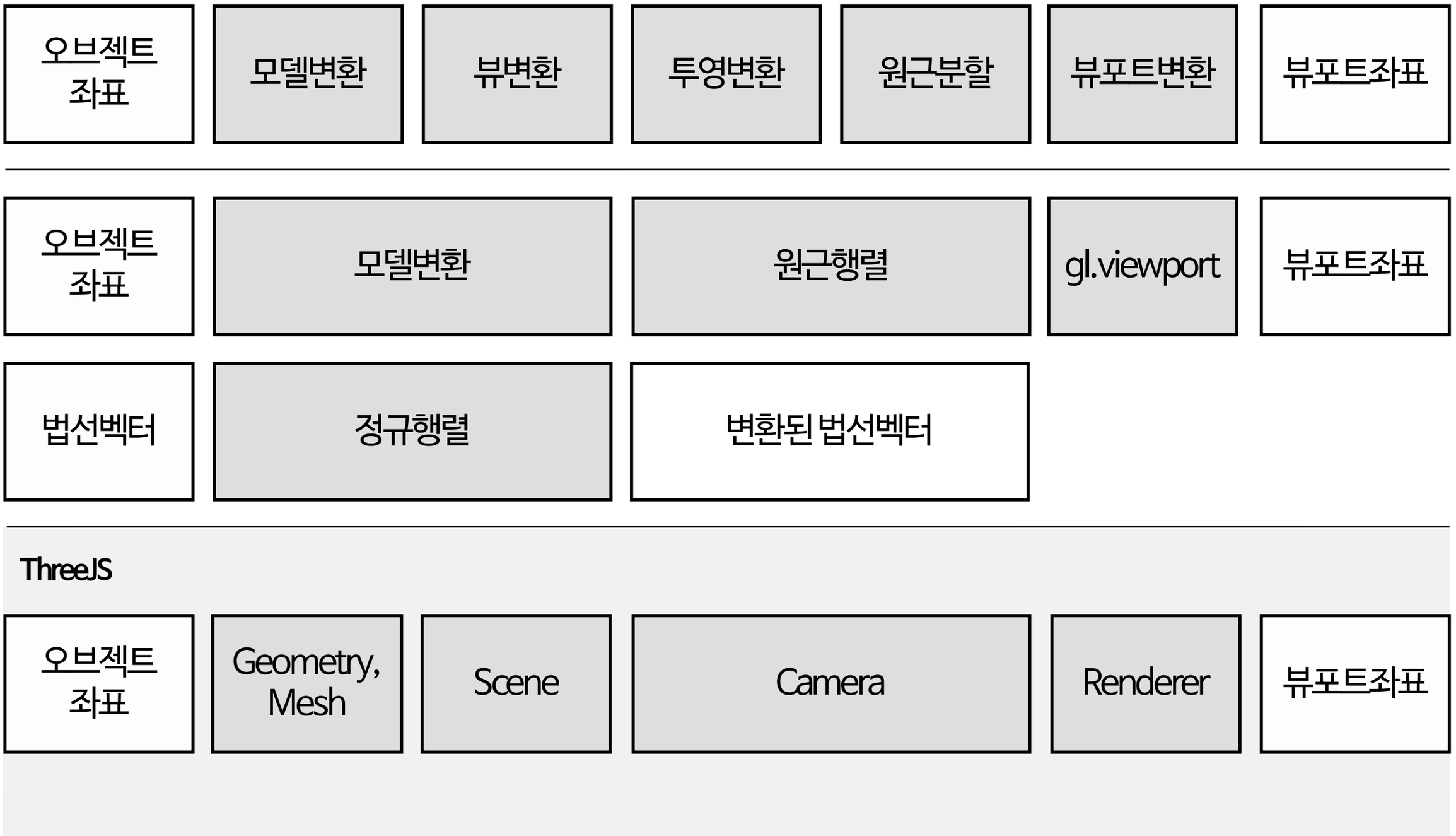
Object3D 상속자들

Scene

Camera

Mesh

Light


오브젝트 좌표
모델변환
뷰변환
투영변환
원근분할
뷰포트변환
뷰포트좌표
오브젝트 좌표
모델변환
원근행렬
gl.viewport
뷰포트좌표
법선벡터
정규행렬
변환된 법선벡터
ThreeJS
오브젝트 좌표
Geometry, Mesh
Scene
Camera
Renderer
뷰포트좌표

그리기 순서

Scene만들기 -> 카메라 놓기 -> Geometry로 도형 모델링
-> Material 결정 -> Mesh 만들기 -> Renderer로 그리기

Scene 만들기

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

카메라 놓기

```
camera = new THREE.PerspectiveCamera(
  25,                                  // fov
  SCREEN_WIDTH / SCREEN_HEIGHT, // aspect
  50,                                  // near
  1e7                                  // far
);
camera.position.z = radius * 8;
scene.add(camera);
```

Geometry로 도형 모델링

```
geometry = new THREE.SphereGeometry(
   radius, // radius
   100,    // widthSegment
   50      // heightSegment
);
```

Material 결정

```
var material = new THREE.MeshBasicMaterial({
  color: 0xff3333
});
```

Mesh 만들기

```
meshPlanet = new THREE.Mesh(
   geometry,
   material
);
scene.add(meshPlanet);
```

Renderer로 그리기

```
renderer = new THREE.WebGLRenderer();
renderer.setSize(SCREEN_WIDTH, SCREEN_HEIGHT);

container.appendChild(renderer.domElement);

renderer.render(scene, camera);
```

주요 구성요소 훑어보기

Scene

ThreeJS의 모든 오브제들의 공간

월드 좌표계라고 봐도 될듯...

Cameras

`PerspectiveCamera(fov, aspect, near, far)`

`OrthographicCamera(left, right, top, bottom, near, far)`

Objects

```
Mesh(geometry, material)

ParticleSystem(geometry, material)
```

Geometry

.vertices

.colors

.faces

.faceVertexUvs

computeVertexNormals()

computeFaceNormals()

computeTangents()

Geometries

CubeGeometry(width, height, depth, widthSegments, heightSegments, depthSegments)

CylinderGeometry(radiusTop, radiusBottom, height, radiusSegments, heightSegments, openEnded)

SphereGeometry(radius, widthSegments, heightSegments, phiStart, phiLength, thetaStart, thetaLength)

TextGeometry(text, parameters)

Materials

`MeshBasicMaterial(parameters)`

`MeshLambertMaterial(parameters)`

`MeshPhongMaterial(parameters)`

`ShaderMaterial(parameters)`

```
THREE.MeshBasicMaterial = function ( parameters ) {

    THREE.Material.call( this );

    this.color = new THREE.Color( 0xffffff ); // emissive

    this.map = null;

    this.lightMap = null;

    this.specularMap = null;

    this.envMap = null;
    this.combine = THREE.MultiplyOperation;
    this.reflectivity = 1;
    this.refractionRatio = 0.98;

    this.fog = true;

    this.shading = THREE.SmoothShading;

    this.wireframe = false;
    this.wireframeLinewidth = 1;
    this.wireframeLinecap = 'round';
    this.wireframeLinejoin = 'round';

    this.vertexColors = THREE.NoColors;

    this.skinning = false;
    this.morphTargets = false;

    this.setValues( parameters );

};
```

```
THREE.MeshLambertMaterial = function ( parameters ) {

    THREE.Material.call( this );

    this.color = new THREE.Color( 0xffffff ); // diffuse
    this.ambient = new THREE.Color( 0xffffff );
    this.emissive = new THREE.Color( 0x000000 );

    this.wrapAround = false;
    this.wrapRGB = new THREE.Vector3( 1, 1, 1 );

    this.map = null;

    this.lightMap = null;

    this.specularMap = null;

    this.envMap = null;
    this.combine = THREE.MultiplyOperation;
    this.reflectivity = 1;
    this.refractionRatio = 0.98;

    this.fog = true;

    this.shading = THREE.SmoothShading;

    this.wireframe = false;
    this.wireframeLinewidth = 1;
    this.wireframeLinecap = 'round';
    this.wireframeLinejoin = 'round';

    this.vertexColors = THREE.NoColors;

    this.skinning = false;
    this.morphTargets = false;
    this.morphNormals = false;

    this.setValues( parameters );

};
```

```
THREE.MeshPhongMaterial = function ( parameters ) {

    THREE.Material.call( this );

    this.color = new THREE.Color( 0xffffff ); // diffuse
    this.ambient = new THREE.Color( 0xffffff );
    this.emissive = new THREE.Color( 0x000000 );
    this.specular = new THREE.Color( 0x111111 );
    this.shininess = 30;

    this.metal = false;
    this.perPixel = true;

    this.wrapAround = false;
    this.wrapRGB = new THREE.Vector3( 1, 1, 1 );

    this.map = null;

    this.lightMap = null;

    this.bumpMap = null;
    this.bumpScale = 1;

    this.normalMap = null;
    this.normalScale = new THREE.Vector2( 1, 1 );

    this.specularMap = null;

    this.envMap = null;
    this.combine = THREE.MultiplyOperation;
    this.reflectivity = 1;
    this.refractionRatio = 0.98;

    this.fog = true;

    this.shading = THREE.SmoothShading;

    this.wireframe = false;
    this.wireframeLinewidth = 1;
    this.wireframeLinecap = 'round';
    this.wireframeLinejoin = 'round';

    this.vertexColors = THREE.NoColors;

    this.skinning = false;
    this.morphTargets = false;
    this.morphNormals = false;

    this.setValues( parameters );

};
```

```
THREE.ShaderMaterial = function ( parameters ) {

    THREE.Material.call( this );

    this.fragmentShader = "void main() {}";
    this.vertexShader = "void main() {}";
    this.uniforms = {};
    this.defines = {};
    this.attributes = null;

    this.shading = THREE.SmoothShading;

    this.linewidth = 1;

    this.wireframe = false;
    this.wireframeLinewidth = 1;

    this.fog = false; // set to use scene fog

    this.lights = false; // set to use scene lights

    this.vertexColors = THREE.NoColors; // set to use "color" attribute stream

    this.skinning = false; // set to use skinning attribute streams

    this.morphTargets = false; // set to use morph targets
    this.morphNormals = false; // set to use morph normals

    // When rendered geometry doesn't include these attributes but the material does,
    // use these default values in WebGL. This avoids errors when buffer data is missing.
    this.defaultAttributeValues = {
        "color" : [ 1, 1, 1],
        "uv" : [ 0, 0 ],
        "uv2" : [ 0, 0 ]
    };

    // By default, bind position to attribute index 0. In WebGL, attribute 0
    // should always be used to avoid potentially expensive emulation.
    this.index0AttributeName = "position";

    this.setValues( parameters );

};
```

Lights

AmbientLight(hex)

DirectionalLight(hex, intensity)

PointLight(hex, intensity, distance)

SpotLight(hex, intensity, distance, angle, exponent)

Links

http://threejs.org/
http://threejs.org/examples/
http://threejs.org/docs/

http://gitlab.map.naver.com/fe/earth