このノートでは、earth exampleを例に、Three.jsの基本中の基本について解説する。

# 表示領域

まずは、アプリケーションを表示する領域を<div>タグで確保する。

<div id=”container”></div>

このタグは、Javascriptからは*document.getElementById()*関数で取得できる。

container = document.getElementById(‘container’);

そして、領域に、例えば3Dを表示する領域を追加する。3Dを表示する領域は、3Dを表示するレンダラ（*THREE.CanvasRenderer*）を構築した後、*domElement*プロパティで取得できる。

renderer = new THREE.CanvasRenderer();

renderer.setClearColor(0xffffff);

renderer.setPixelRatio(window.devicePixelRatio);

renderer.setSize(window.innerWidth, window.innerHeight);

container.appendChild(renderer.domElement);

# シーン

3Dのシーン全体を扱うオブジェクト。*Three.Scene()*コンストラクタで構築するだけ。

scene = new THREE.Scene();

全てのオブジェクト、ライト、カメラなどは、このシーンオブジェクトに*add*することで、表示される。fogとかも設定できるみたい。実際にシーンを表示するには、3Dレンダラの*render*関数に、シーンオブジェクトを渡すだけ。

renderer.render(scene, camera);

# カメラ

標準的なカメラは、*THREE.PerspectiveCamera()*コンストラクタで簡単に構築できる。引数には、fov、縦横比、near、farを指定する。

camera = new THREE.PerspectiveCamera(60, window.innerWidth/window.innerHeight, 1, 2000);

また、カメラの座標は、*position*プロパティを通して変更できる。

camera.position.z = 500;

さらに、カメラの向きは、*lookAt*プロパティを通して変更できる。

camera.lookAt(scene.position);

なお、カメラの内部パラメータである縦横比などを変更した場合は、*updateProjectionMatrix()*を実行して内部パラメータ行列を更新する必要がある。

camera.aspect = window.innerWidth / window.innerHeight;

camera.updateProjectionMatrix();

# ジオミトリ

簡単なジオミトリは、組み込み関数で簡単に作成できる。例えば、球は、*THREE.SphereGeometry()*で作成できる。

var geometry = new THREE.SphereGeometry(半径, 水平方向の分割数, 垂直方向の分割数);

平面なら、*THREE.PlaneGeometry()*だ。

var geometry = new THREE.PlaneGeometry(幅, 縦, 幅方向の分割数, 縦方向の分割数);

ただし、ジオミトリは、直接シーンに追加できない。かならずマテリアルをセットしてメッシュを作成してから、シーンに追加する必要がある。

# マテリアル

マテリアルは、オブジェクトの色などを指定するものだ。基本的なマテリアルなら、*THREE.MeshBasicMaterial()*で作成できる。

var material = new THREE.MeshBasicMaterial( {color: 0xff0000} );

マテリアルを作成したら、*THREE.Mesh()*コンストラクタを使って、対象となるジオミトリにセットする。

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

これを見ると分かる通り、メッシュは、ジオミトリとマテリアルから構成される。そして、シーンには、このメッシュを追加するのだ。

scene.add(mesh);

こうして、シーンにオブジェクトが追加されるのだ。

# テクスチャ

まず、テクスチャローダ（*THREE.TextureLoader*）を使って、画像を読み込む必要がある。

var loader = new THREE.TextureLoader();

loader.load(‘画像ファイル名’,

function(texture) {

// 画像がロードされた後に、この関数が実行される。

}

);

で、画像がロードされた後に何をやれば良いのか？

テクスチャを直接オブジェクトに貼ることは出来ない。まず、このテクスチャを使ってマテリアルを作成する必要がある。基本的なマテリアルなら、*THREE.MeshBasicMaterial()*コンストラクタで簡単に作成できる。*map*プロパティでテクスチャを指定する。*overdraw*プロパティは、各三角形を少しだけ拡大して表示する。三角形の間にギャップが見える時に、これによりきれいに見えるようになる。※ショボイけど、よく使われるテクニックらしい。

var material = new THREE.MeshBasicMaterial( {map: texture, overdarw: 0.5} );

# ウィンドウのリサイズ

表示領域のサイズが固定の場合は、ウィンドウをリサイズしても心配要らないよね。

でも、表示領域がウィンドウ全体の場合は、ウィンドウをリサイズしたら、それに合わせて１）カメラの縦横比の更新、２）レンダラに表示領域のサイズを通知、する必要がある。

そのために、まず、Javascriptにおいて、ウィンドウのリサイズイベントのハンドラを追加する必要がある。

window.addEventListener(‘resize’, onWindowResize, false);

次に、イベントハンドラを定義する。

function onWindowResize() {

camera.aspect = window.innerWidth / window.innerHeight;

camera.updateProjectionMatrix();

renderer.setSize(window.innerWidth, window.innerHeight);

}

# マウス操作

まず、Javascriptにおいて、マウスイベントハンドラを追加する必要がある。

document.addEventListener(‘mousemove’, onDocumentMouseMove, false);

次に、この関数を定義する。

Function onDocumentMouseMove(event) {

// マウスX座標は、event.clientX

// マウスY座標は、event.clientY

}

もし表示を更新したい場合は、この関数の中で、*renderer.render()*を実行すれば良い。

# もっと複雑なジオミトリ

自分でジオミトリを作成することもできる。まず、*THREE.Geometry()*で空のジオミトリを構築する。

var geometry = new THREE.Geometry();

次に、*vertices*プロパティに頂点座標を追加していく。

geometry.vertices.push(new THREE.Vector3(X座標, Y座標, Z座標));

さらに、*faces*プロパティに面情報を追加していく。

geometry.faces.push(

new THREE.Face3(1つ目の頂点ID, 2つ目の頂点ID, 3つ目の頂点ID, 法線ベクトル, 色, マテリアルID)

);

法線ベクトルは、後で自動計算できるので、nullを指定しておけば良い。

テクスチャを使用する場合は、*faceVertexUvs*に、テクスチャ座標を追加していく。

geometry.faceVertexUvs[0].push(

[1つ目の頂点のテクスチャ座標, 2つ目の頂点のテクスチャ座標, 3つ目の頂点のテクスチャ座標]

);

さらに、*materials*プロパティを通して、マテリアルをセットする。このマテリアルは、*faces*プロパティにセットしたマテリアルIDで参照される。

geometry.materials = [

new THREE.MeshBasicMaterial( {color: 0xff0000} ),

new THREE.MeshBasicMaterial( {color: 0x0000ff} )

];

法線ベクトルは、最後に自動で計算できる。

geometry.computeFaceNormals();

geometry.computeVertexNormals();

最後に、メッシュを作成する。なお、この例のように、１つのジオミトリで複数のマテリアルを使用している場合は、マテリアルとして*THREE.MeshFaceMaterial()*を使用する。

var mesh = new THREE.Mesh(geometry, new THREE.MeshFaceMaterial());

# ピッキング

マウスクリックした座標にあるオブジェクトを調べるためには、THREE.Raycasterを使用する。

var raycaster = new THREE.Raycaster();

raycaster.setFromCamera(マウスのXY座標, camera);

var intersects = raycaster.intersectObjects(オブジェクトのリスト);

ただし、マウスのXY座標は、-1～1にnormalizeされた座標にしておくこと。

もしオブジェクトをピック出来れば、戻り値のリストに、オブジェクトがカメラから近い順に格納される。