Workshop 04: A look at three.js

Baseline

Based on the first chapter of "Learning Three.js: The JavaScript 3D Library for WebGL" (Jos Dirksen, p19)

```
<!DOCTYPE html>
<html>
<head>
  <title>Three.js baseline</title>
  <style>
    canvas { : 100%; height: 100% }
  </style>
</head>

<body>
  <script src="three.min.js"></script>
  <script>
    var scene = new THREE.Scene();
    var camera = new THREE.PerspectiveCamera(45, window.innerWidth/window.innerHeight, 0.1,
1000);

    var renderer = new THREE.WebGLRenderer();
    renderer.setClearColorHex(0xEEEEEE);
    renderer.setSize(window.innerWidth, window.innerHeight);
    document.body.appendChild(renderer.domElement);

    var axes = new THREE.AxisHelper(20);
    scene.add(axes);
    var planeGeometry = new THREE.PlaneGeometry(60, 20, 1, 1);
    var planeMaterial = new THREE.MeshBasicMaterial({
      color: 0xcccccc
    });

    var plane = new THREE.Mesh(planeGeometry, planeMaterial);
    plane.rotation.x = -0.5 * Math.PI;
    plane.position = new THREE.Vector3(15, 0, 0);
    scene.add(plane);
    var cubeGeometry = new THREE.CubeGeometry(4, 4, 4);
    var cubeMaterial = new THREE.MeshBasicMaterial({
      color: 0xff0000,
      wireframe: true
    });

    var cube = new THREE.Mesh(cubeGeometry, cubeMaterial);
    cube.position.x = -4;
    cube.position.y = 3;
    cube.position.z = 0;
    scene.add(cube);
    var sphereGeometry = new THREE.SphereGeometry(4, 20, 20);
    var sphereMaterial = new THREE.MeshBasicMaterial({
      color: 0x7777ff,
      wireframe: true
    });

    var sphere = new THREE.Mesh(sphereGeometry, sphereMaterial);
    sphere.position.x = 20;
    sphere.position.y = 4;
    sphere.position.z = 2;
    scene.add(sphere);
    camera.position.x = -30;
    camera.position.y = 40;
    camera.position.z = 30;
    camera.lookAt(scene.position);
```

```
    renderer.render(scene, camera);
  </script>
</body>
</html>
```

Note: direct assignment of `THREE.Vector3` to `plane.position` line 30.

Add a bit of interaction

Add an orbit control – the orbit control is not part of three.min.js and can be found in the *threejs folder > examples > js > controls > OrbitControls.js*

```
...
<body>
  <script src="three.min.js"></script>
  <script src="OrbitControls.js"></script>
  <script>
  ...
```

Attach the orbit control

```
    ...
    camera.position.z = 30;
    camera.lookAt(scene.position);

    var controls = new THREE.OrbitControls(camera, renderer.domElement);
    ...
```

Add and enable animation

```
    ...
    var controls = new THREE.OrbitControls(camera, renderer.domElement);

    (function animate() {
      requestAnimationFrame(animate);

      renderer.render(scene, camera);
      controls.update();
    })();
  </script>
  ...
```

Change material

Change `MeshBasicMaterial` to `MeshLambertMaterial` for the plane, sphere and cube, and turn off wireframe (remove the line) where present.

```
    var planeGeometry = new THREE.PlaneGeometry(60, 20, 1, 1);
    var planeMaterial = new THREE.MeshLambertMaterial({
      color: 0xcccccc
    });

    ...

    var cubeGeometry = new THREE.CubeGeometry(4, 4, 4);
    var cubeMaterial = new THREE.MeshLambertMaterial({
      color: 0xff0000,
    });

    ...

    var sphereGeometry = new THREE.SphereGeometry(4, 20, 20);
    var sphereMaterial = new THREE.MeshLambertMaterial({
      color: 0x7777ff,
    });
```

Add a light and position marker

```
...
sphere.position.z = 2;
scene.add(sphere);

var spotLight = new THREE.SpotLight( 0xffffff );
spotLight.position.set( -20, 20, -20 );
scene.add( spotLight );

sphereGeometry = new THREE.SphereGeometry(1, 20, 20);
sphereMaterial = new THREE.MeshBasicMaterial({
  color: 0xffffff
});
var spotLightMarker = new THREE.Mesh(sphereGeometry, sphereMaterial);
spotLightMarker.position = spotLight.position;
scene.add(spotLightMarker);

camera.position.x = -30;
camera.position.y = 40;
...
```

Increase plane poly count

The plane looks almost entirely black; this is due to low poly count and interpolation. Increase the poly count.

```
...
scene.add(axes);
var planeGeometry = new THREE.PlaneGeometry(60, 20, 50, 50);
var planeMaterial = new THREE.MeshBasicMaterial({
...
```

Change the direction of the spotlight

Point the spotlight at the sphere

```
...
var spotLight = new THREE.SpotLight( 0xffffff );
spotLight.position.set( -20, 20, -20 );
spotlight.target.position = sphere.position;
scene.add( spotLight );
...
```

Add shadows

Set the light to cast shadows,

```
...
spotLight.target.position = sphere.position;
spotLight.castShadow = true;
scene.add( spotLight );
...
```

Set the plane to receive shadows

```
...
renderer.setSize(window.innerWidth, window.innerHeight);
renderer.shadowMapEnabled = true;
document.body.appendChild(renderer.domElement);
...
plane.position = new THREE.Vector3(15, 0, 0);
plane.receiveShadow = true;
scene.add(plane);
...
cube.position.z = 0;
cube.castShadow = true;
scene.add(cube);
```

```
...
sphere.position.z = 2;
sphere.castShadow = true;
scene.add(sphere);
...
```

Fine-tune shadow

The cube isn't casting a shadow, the spotlight's shadow camera's near clipping plane needs to be reduced.

```
spotLight.target.position = sphere.position;
spotLight.shadowCameraNear = spotLight.shadowCameraNear/2;
spotLight.castShadow = true;
```

Shaders (effects)

Add an ASCII render effect.

```
<script src="OrbitControls.js"></script>
<script src="AsciiEffect.js"></script>
<script>
  ...
  document.body.appendChild(renderer.domElement);
  var effect = new THREE.AsciiEffect( renderer );
  effect.setSize( window.innerWidth, window.innerHeight );

  document.body.appendChild(effect.domElement);

  camera.lookAt(scene.position);
  ...
  var controls = new THREE.OrbitControls(camera, effect.domElement);

  (function animate() {
    requestAnimationFrame(animate);

    effect.render(scene, camera);
    controls.update();
```

Import a model

This won't work on a local drive, needs to be run on a server 😐

```
...
sphereMaterial.color = spotLight.color;

var jsonLoader = new THREE.JSONLoader();
jsonLoader.load( "android.js", function(geometry, materials){
  var material = new THREE.MeshFaceMaterial( materials );
  android = new THREE.Mesh( geometry, material );
  android.position.x = 10;
  android.castShadow = true;
  scene.add( android );
});

var spotLightMarker = new THREE.Mesh(sphereGeometry, sphereMaterial);
...
```

Babylon

```
<!DOCTYPE html>
<html>
<head>
  <title>Babylon.js baseline</title>
  <style>
    canvas { width: 100%; height: 75% }
  </style>
</head>

<body>
  <div id="rootDiv">
    <canvas id="renderCanvas"></canvas>
  </div>
  <script src="babylon.min.js"></script>
  <script src="hand.js"></script>
  <script>
  // Get the Canvas element from our HTML below
  var canvas = document.getElementById("renderCanvas");
  canvas.width  = window.innerWidth * 0.75;
  canvas.height  = window.innerHeight * 0.75;

  // Load BABYLON 3D engine
  var engine = new BABYLON.Engine(canvas, true);
  var scene = new BABYLON.Scene(engine);

  // Creating a camera looking to the zero point (0,0,0), a light, and a sphere of size 1
  var camera = new BABYLON.ArcRotateCamera("Camera", 1, 0.8, 10, new BABYLON.Vector3(0, 0, 0),
scene);
  var light0 = new BABYLON.PointLight("Omni", new BABYLON.Vector3(0, 0, 10), scene);
  var origin = BABYLON.Mesh.CreateSphere("origin", 10, 1.0, scene);

  // Attach the camera to the scene
  scene.activeCamera.attachControl(canvas);

  // Once the scene is loaded, just register a render loop to render it
  engine.runRenderLoop(function () {
    scene.render();
  });
  </script>
</body>
</html>
```