Lab 08: Basic Textures

CS423: Computer Graphics

Contents

1 Overview

2 Instructions

Let's start with the basic HTML we need to use:

```
<!DOCTYPE html>

<html>

<head>
    <title>Example 10.01 - Basic textures</title>
    <script type="text/javascript" src="../libs/three.js"></script>

    <script type="text/javascript" src="../libs/stats.min.js"></script>
    <style>
        body {
            /* set margin to 0 and overflow to hidden, to go fullscreen */
            margin: 0;
            overflow: hidden;
        }
    </style>
</head>
<body>

<div id="Stats-output">
</div>
<!-- Div which will hold the Output -->
<div id="WebGL-output">
</div>

<!-- Javascript code that runs our Three.js examples -->
<script type="text/javascript" src="./04-basic-texture.js">
</script>
</body>
</html>
```

Now we do the Javascript we need to put in to make this work. Let's set the scene by putting the following into `04-basic-material.js`:

```
function init() {
    var stats = initStats();
    // create a scene, that will hold all our elements such as
```

```
    // objects, cameras and lights.
    var scene = new THREE.Scene();
    // create a camera, which defines where we're looking at.
    var camera = new THREE.PerspectiveCamera(45,
                    window.innerWidth / window.innerHeight,
                    0.1, 1000);
    // create a render and set the size
    var webGLRenderer = new THREE.WebGLRenderer();
    webGLRenderer.setClearColor(new THREE.Color(0xEEEEEE, 1.0));
    webGLRenderer.setSize(window.innerWidth, window.innerHeight);
    webGLRenderer.shadowMapEnabled = true;
    // position and point the camera to the center of the scene
    camera.position.x = 00;
    camera.position.y = 12;
    camera.position.z = 28;
    camera.lookAt(new THREE.Vector3(0, 0, 0));
    var ambiLight = new THREE.AmbientLight(0x141414);
    scene.add(ambiLight);
    var light = new THREE.DirectionalLight();
    light.position.set(0, 30, 20);
    scene.add(light);
    // Add the objects to scene here
    // add the output of the renderer to the html element
    document.getElementById("WebGL-output")
            .appendChild(webGLRenderer.domElement);
     // call the render function
    var step = 0;
    render();

    // Function for building a mesh will go here

    // Rendering function to go here
    function initStats() {

        var stats = new Stats();
        stats.setMode(0); // 0: fps, 1: ms

        // Align top-left
        stats.domElement.style.position = 'absolute';
        stats.domElement.style.left = '0px';
        stats.domElement.style.top = '0px';

        document.getElementById("Stats-output").appendChild(stats.domElement);

        return stats;
    }
}
window.onload = init;
```

Now to build a textured mesh. Note the pattern: create a loader, load a texture, assign the texture to material map, and build the mesh. Add the following code to your Javascript at the point indicated:

```
function createMesh(geom, imageFile) {
    var texLoader = new THREE.TextureLoader();
    var texture = texLoader.load("../assets/textures/general/" + imageFile);
    var mat = new THREE.MeshPhongMaterial();
    mat.map = texture;

    var mesh = new THREE.Mesh(geom, mat);
    return mesh;
}
```

Now we use this function to add a polyhedron, sphere, and brick wall textured objects. Add the following code where indicated:

```
var polyhedron = createMesh(new THREE.IcosahedronGeometry(5, 0),
                            "metal-rust.jpg");
polyhedron.position.x = 12;
scene.add(polyhedron);
var sphere = createMesh(new THREE.SphereGeometry(5, 20, 20),
                            "floor-wood.jpg");
scene.add(sphere);

var cube = createMesh(new THREE.BoxGeometry(5, 5, 5),
                            "brick-wall.jpg");
cube.position.x = -12;
scene.add(cube);
```

And then add a function to the do the rendering:

```
function render() {
    stats.update();

    polyhedron.rotation.y = step += 0.01;

    polyhedron.rotation.x = step;
    cube.rotation.y = step;
    cube.rotation.x = step;
    sphere.rotation.y = step;
    sphere.rotation.x = step;

    // render using requestAnimationFrame
    requestAnimationFrame(render);
    webGLRenderer.render(scene, camera);
}
```

Save both files and check out the result.

3 Submission instructions

Please create a PDF file with the following:

- A screen-shot of both your webapps displayed in the browser.
- HTML and JS files for each webapp

Attach this PDF file to the submission link in Blackboard.