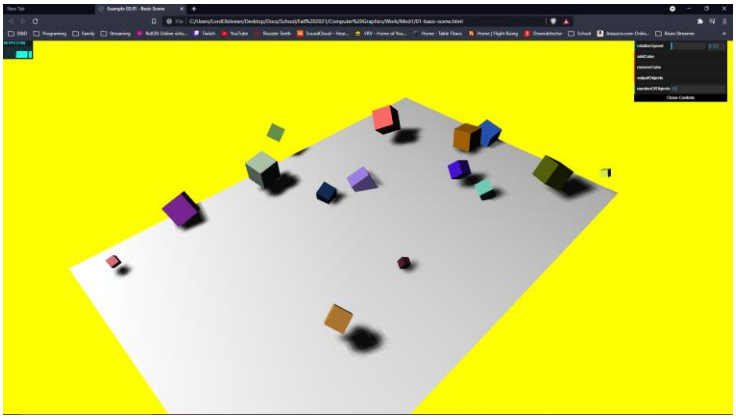
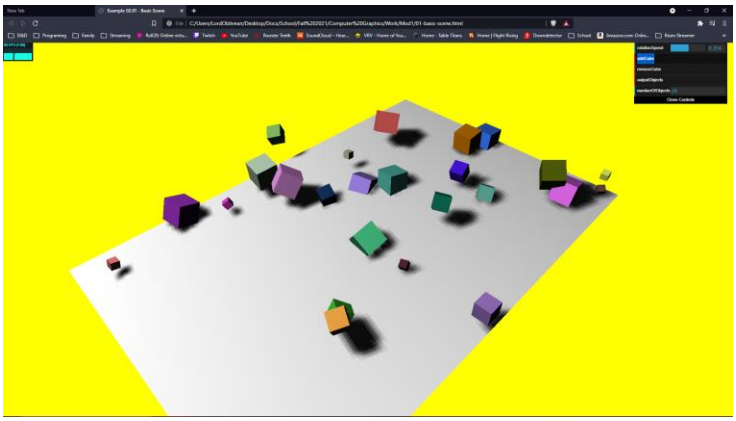
Devin Hardy

00076619

CS423

Basic Scene





```
<!DOCTYPE html>
<HTML>
<HEAD>
	<TITLE>Example 02.01 - Basic Scene</TITLE>
	<SCRIPT TYPE="text/javascript" SRC="../libs/three.js"></SCRIPT>
	<SCRIPT TYPE="text/javascript" SRC="../libs/stats.min.js"></SCRIPT>
	<SCRIPT type="text/javascript" src="../libs/dat.gui.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="01-basic-scene.js">
</script>
</body>
</HTML>

//
// File: 01-basic-scene.js
// Demo some of the basics of working with the scenegraph
// This is an extension of code from he Learning THREE.js textbook
// once everything is loaded. we run our Three.js stuff
function init() {

    var stats = initStats();

    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;
        }

        // 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);
    scene.add(camera);

    // create a render and set the size
    var renderer = new THREE.WebGLRenderer();

    renderer.setClearColor(new THREE.Color(0xEEEEEE, 1.0));
    renderer.setSize(window.innerWidth, window.innerHeight);
    renderer.shadowMapEnabled = true;

    // create the ground plane
    var planeGeometry = new THREE.PlaneGeometry(60, 40, 1, 1);
    var planeMaterial = new THREE.MeshLambertMaterial({color: 0xffffff});
    var plane = new THREE.Mesh(planeGeometry, planeMaterial);
```

```
plane.receiveShadow = true;

// rotate and position the plane
plane.rotation.x = -0.5 * Math.PI;
plane.position.x = 0;
plane.position.y = 0;
plane.position.z = 0;

// add the plane to the scene
scene.add(plane);

// position and point the camera to the center of the scene
camera.position.x = -30;
camera.position.y = 40;
camera.position.z = 30;
camera.lookAt(scene.position);

// add subtle ambient lighting
var ambientLight = new THREE.AmbientLight(0x0c0c0c);
scene.add(ambientLight);

// add spotlight for the shadows
var spotLight = new THREE.SpotLight(0xffffff);
spotLight.position.set(-40, 60, -10);
spotLight.castShadow = true;
scene.add(spotLight);

// add the output of the renderer to the html element
document.getElementById("WebGL-output").appendChild(renderer.domElement);
```

```
// call the render function
var step = 0;

var controls = new function () {
   this.rotationSpeed = 0.02;
   this.numberOfObjects = scene.children.length;

   this.removeCube = function () {
      var allChildren = scene.children;
      var lastObject = allChildren[allChildren.length - 1];
      if (lastObject instanceof THREE.Mesh) {
         scene.remove(lastObject);
         this.numberOfObjects = scene.children.length;
      }
   };

   this.addCube = function () {

      var cubeSize = Math.ceil((Math.random() * 3));
      var cubeGeometry = new THREE.BoxGeometry(cubeSize, cubeSize, cubeSize);
      var cubeMaterial = new THREE.MeshLambertMaterial({color: Math.random() * 0xffffff});
      var cube = new THREE.Mesh(cubeGeometry, cubeMaterial);
      cube.castShadow = true;
      cube.name = "cube-" + scene.children.length;


      // position the cube randomly in the scene
```

```
        cube.position.x = -30 + Math.round((Math.random() * planeGeometry.parameters.width));
        cube.position.y = Math.round((Math.random() * 5));
        cube.position.z = -20 + Math.round((Math.random() * planeGeometry.parameters.height));

        // add the cube to the scene
        scene.add(cube);
        this.numberOfObjects = scene.children.length;
    };

    this.outputObjects = function () {
        console.log(scene.children);
    }
};

var gui = new dat.GUI();
gui.add(controls, 'rotationSpeed', 0, 0.5);
gui.add(controls, 'addCube');
gui.add(controls, 'removeCube');
gui.add(controls, 'outputObjects');
gui.add(controls, 'numberOfObjects').listen();

render();

function render() {
    stats.update();

    // rotate the cubes around its axes
    scene.traverse(function (e) {
        if (e instanceof THREE.Mesh && e != plane) {
```

```
            e.rotation.x += controls.rotationSpeed;
            e.rotation.y += controls.rotationSpeed;
            e.rotation.z += controls.rotationSpeed;
          }
        });

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

}
window.onload = init;
```

Not included three.js, stats.min.js, dat.gui.min.js