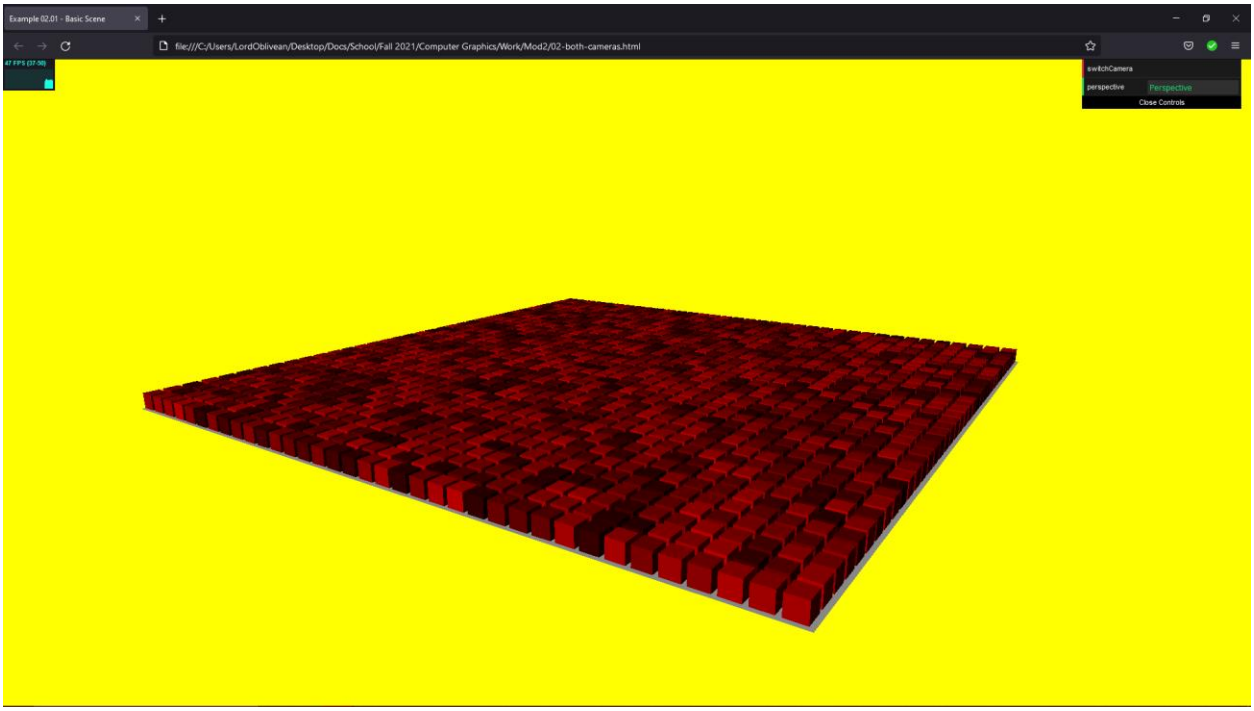
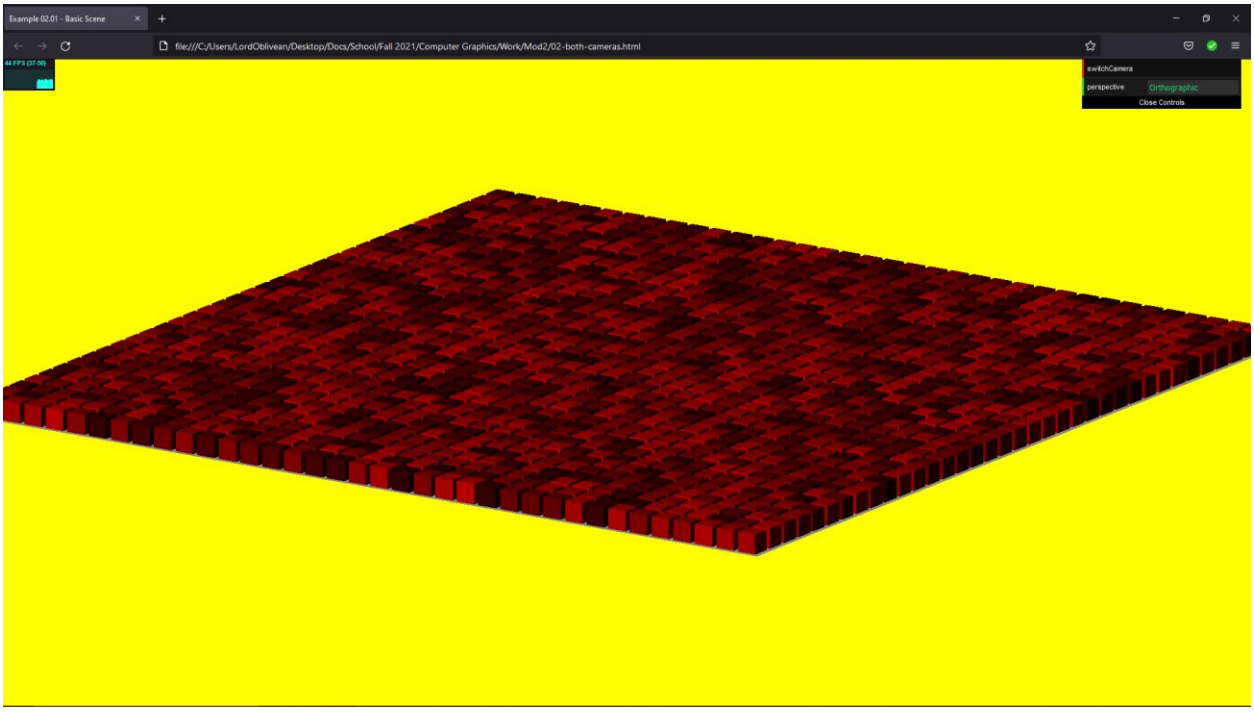
Devin Hardy

00076619

CS423





The camera appears to be bending and stretching the cubes.

```
<!DOCYTPE 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="02-both-cameras.js">
</SCRIPT>
</BODY>
</HTML>

//
// File:
```

```
// Author:
// Purpose:
//
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);
    camera.position.x = 120;
    camera.position.y = 60;
    camera.position.z = 180;

    // 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);

    // create the ground plane
    var planeGeometry = new THREE.PlaneGeometry(180, 180);
    var planeMaterial = new THREE.MeshLambertMaterial({color: 0xffffff});
    var plane = new THREE.Mesh(planeGeometry, planeMaterial);
```

```
// 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);

var cubeGeometry = new THREE.BoxGeometry(4, 4, 4);

for (var j = 0; j < (planeGeometry.parameters.height / 5); j++) {
   for (var i = 0; i < planeGeometry.parameters.width / 5; i++) {
      var rnd = Math.random() * 0.75 + 0.25;
      var cubeMaterial = new THREE.MeshLambertMaterial();
      cubeMaterial.color = new THREE.Color(rnd, 0, 0);
      var cube = new THREE.Mesh(cubeGeometry, cubeMaterial);

      cube.position.z = -((planeGeometry.parameters.height) / 2) + 2 + (j * 5);
      cube.position.x = -((planeGeometry.parameters.width) / 2) + 2 + (i * 5);
      cube.position.y = 2;

      scene.add(cube);
   }
}

var directionalLight = new THREE.DirectionalLight(0xffffff, 0.7);
directionalLight.position.set(-20, 40, 60);
```

```
    scene.add(directionalLight);



    // add subtle ambient lighting
    var ambientLight = new THREE.AmbientLight(0x292929);
    scene.add(ambientLight);

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

    // call the render function
    var step = 0;

    // Insert Lab03 code here.
        // New Controls
        var controls = new function () {
            this.perspective = "Perspective";
            this.switchCamera = function () {
                if (camera instanceof THREE.PerspectiveCamera) {
                    camera = new THREE.OrthographicCamera(window.innerWidth / -16,
window.innerWidth / 16, window.innerHeight / 16, window.innerHeight / -16, -200, 500);
                    camera.position.x = 120;
                    camera.position.y = 60;
                    camera.position.z = 180;
                    camera.lookAt(scene.position);
                    this.perspective = "Orthographic";
                } else {
                    camera = new THREE.PerspectiveCamera(45, window.innerWidth /
window.innerHeight, 0.1, 1000);
```

```
                                camera.position.x = 120;
                                camera.position.y = 60;
                                camera.position.z = 180;
                                camera.lookAt(scene.position);
                                this.perspective = "Perspective";
                        }
                };
        };

        var gui = new dat.GUI();
        gui.add(controls, 'switchCamera');
        gui.add(controls, 'perspective').listen();

// make sure that for the first time, the
// camera is looking at the scene
camera.lookAt(scene.position);
render();

function render() {

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

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
```