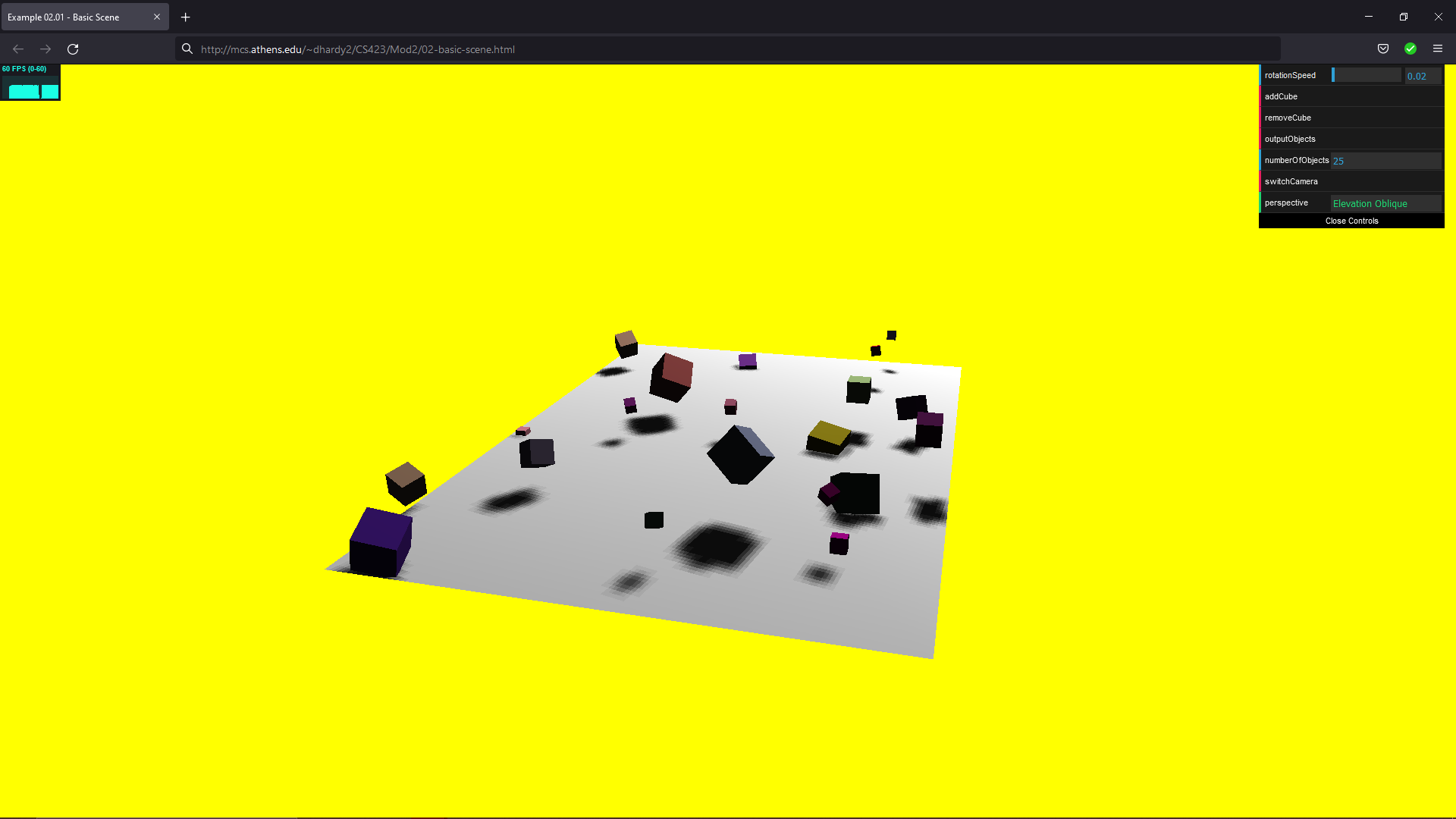
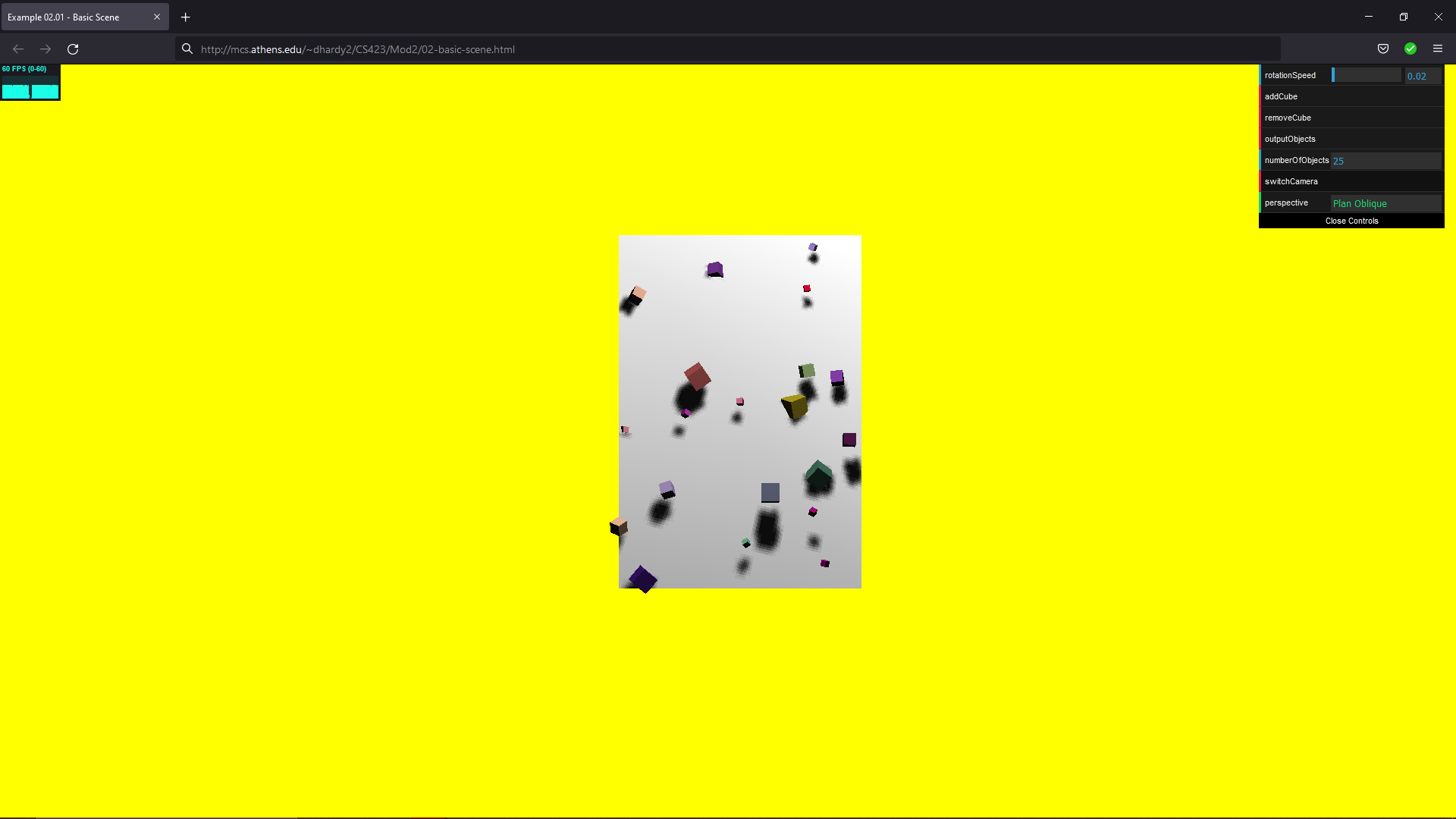
Devin Hardy

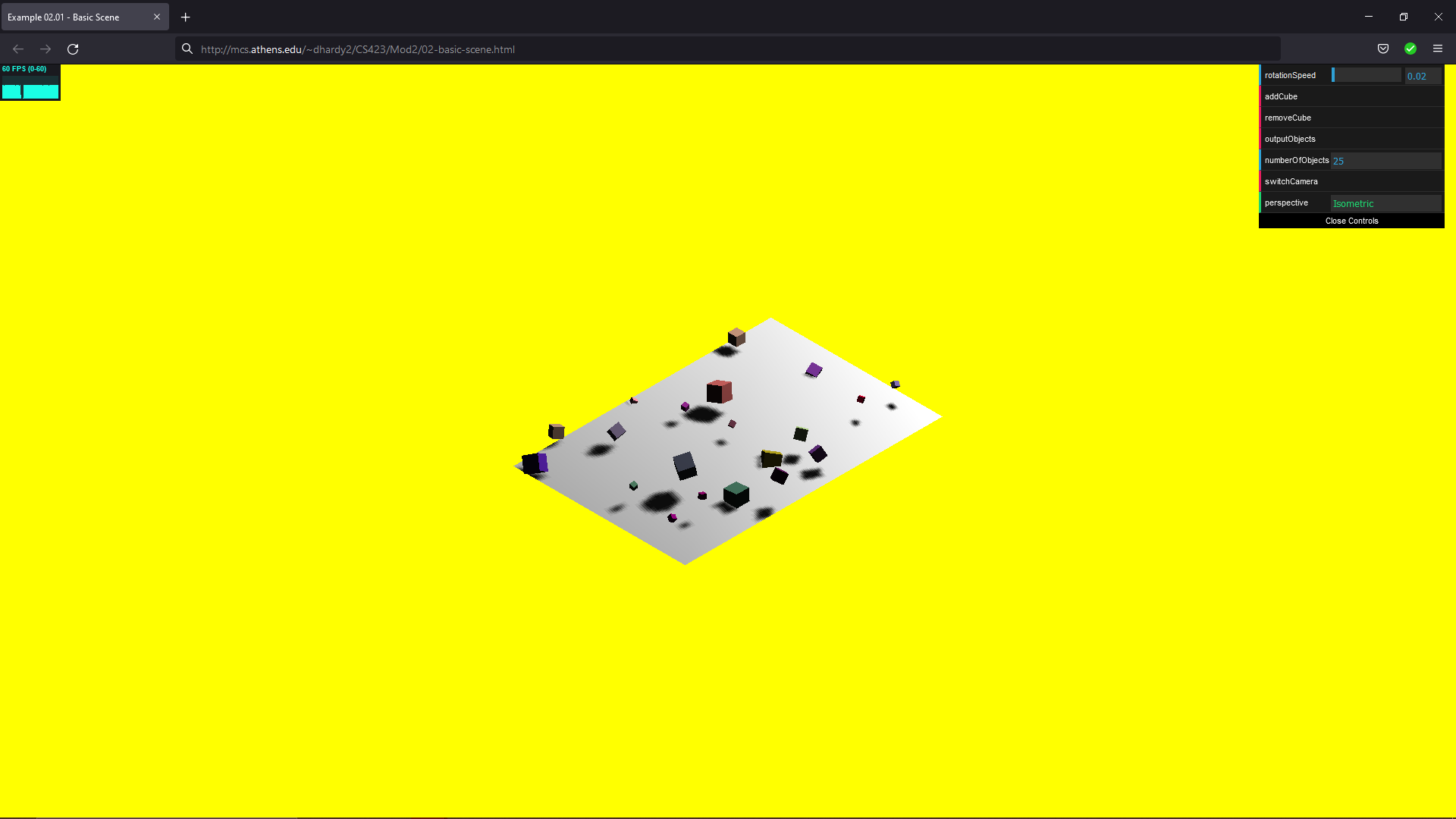
00076619

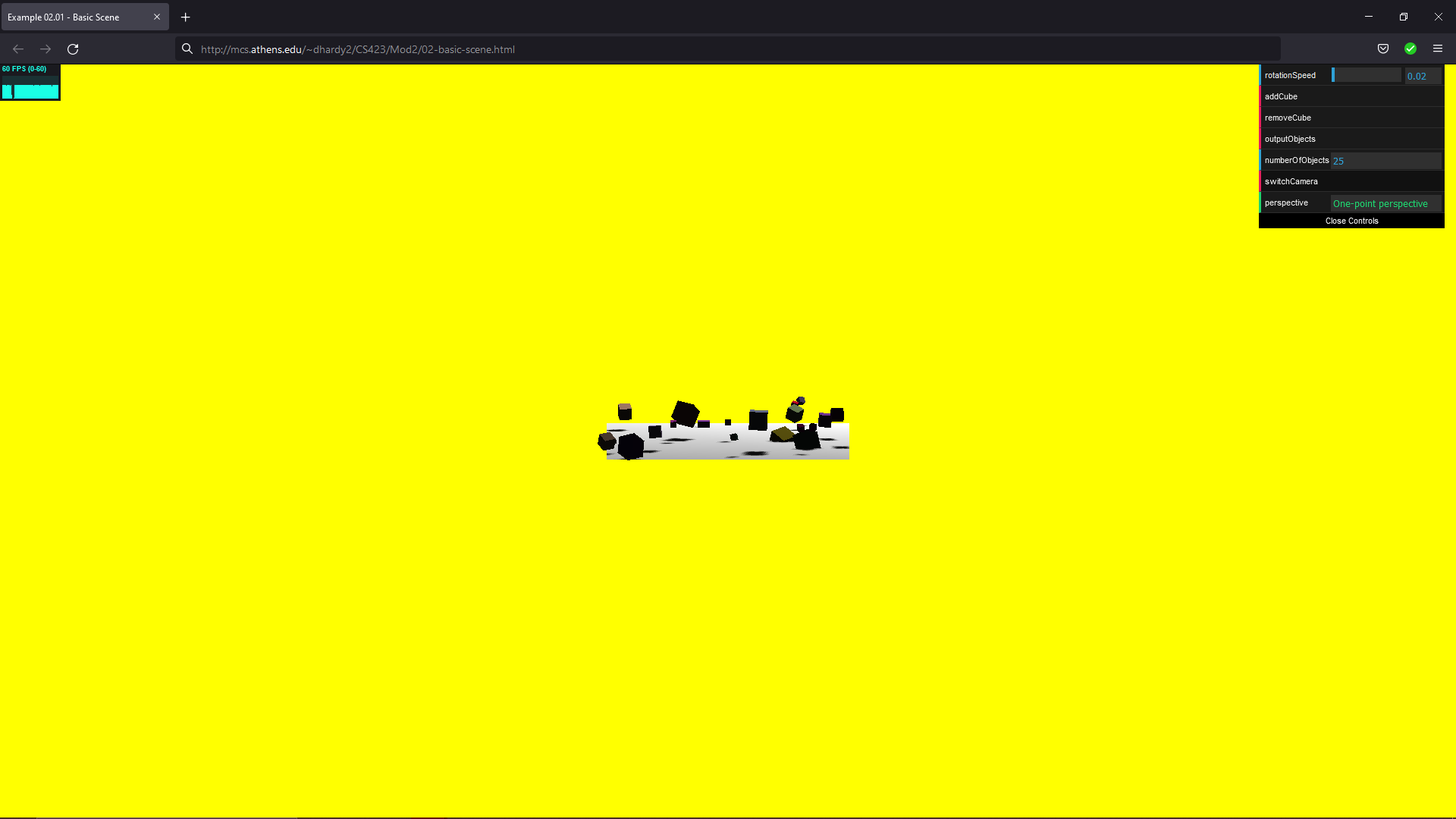
CS423

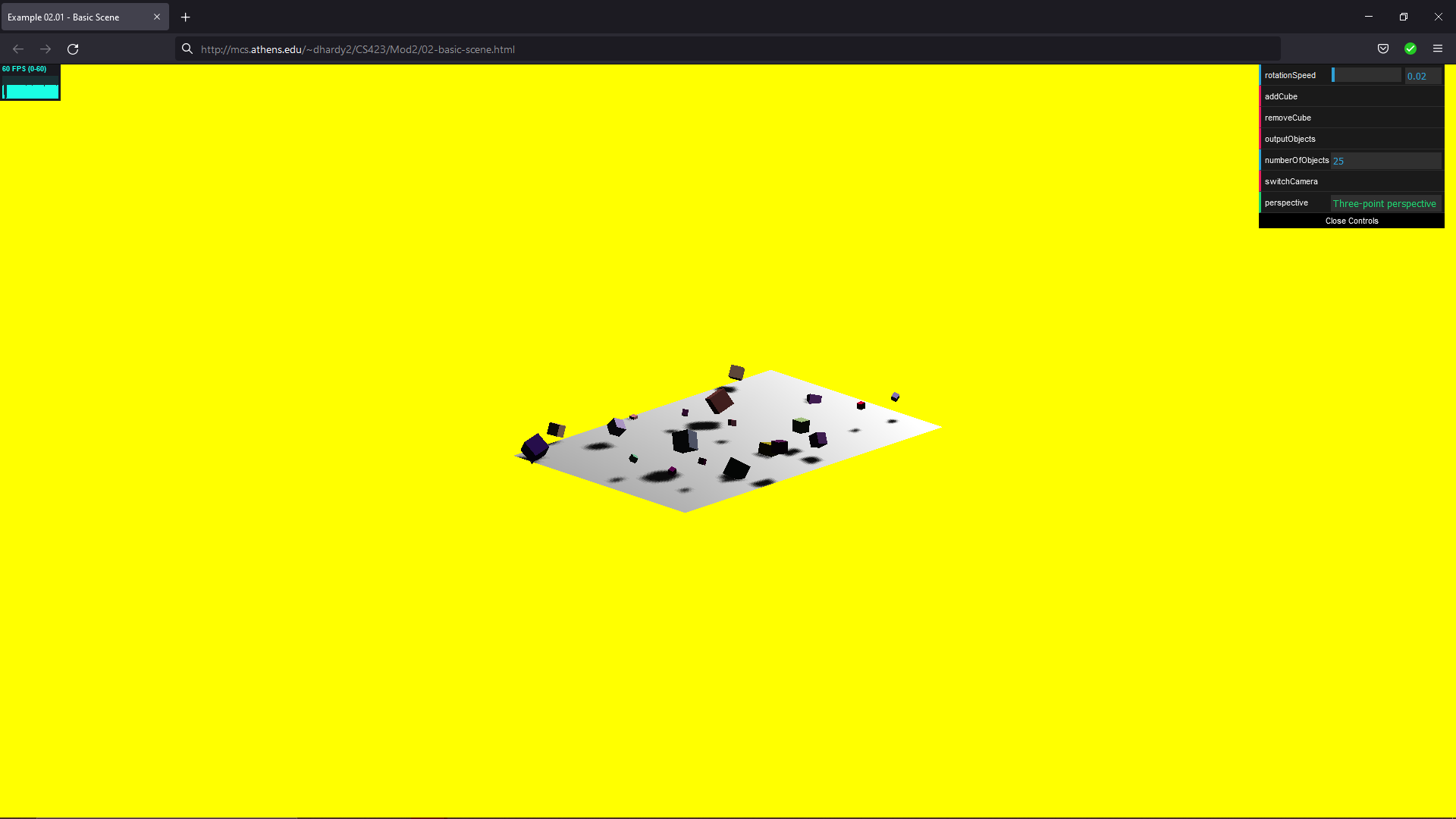












Basic Scene

<!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="02-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);

// 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 camAngle = 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);

}

// Camera Change

this.perspective = "Perspective";

this.switchCamera = function () {

if (camAngle == 0) {

camAngle = 1;

camera = new THREE.OrthographicCamera(window.innerWidth / -16, window.innerWidth / 16, window.innerHeight / 16, window.innerHeight / -16, -200, 500);

camera.position.x = 20;

camera.position.y = 0;

camera.position.z = 0;

camera.lookAt(scene.position);

this.perspective = "Front Elevation";

} else if (camAngle == 1) {

camAngle = 2;

camera = new THREE.PerspectiveCamera(45, window.innerWidth / window.innerHeight, 0.1, 1000);

camera.position.x = 75;

camera.position.y = 30;

camera.position.z = -20;

camera.lookAt(scene.position);

this.perspective = "Elevation Oblique";

} else if (camAngle == 2) {

camAngle = 3;

camera = new THREE.OrthographicCamera(window.innerWidth / -16, window.innerWidth / 16, window.innerHeight / 16, window.innerHeight / -16, -200, 500);

camera.position.x = 5;

camera.position.y = 20;

camera.position.z = 0;

camera.lookAt(scene.position);

camera.position.x = 10;

camera.position.z = 2;

this.perspective = "Plan Oblique";

} else if (camAngle == 3) {

camAngle = 4;

camera = new THREE.OrthographicCamera(window.innerWidth / -16, window.innerWidth / 16, window.innerHeight / 16, window.innerHeight / -16, -200, 500);

camera.position.x = 10;

camera.position.y = 10;

camera.position.z = -10;

camera.lookAt(scene.position);

this.perspective = "Isometric";

} else if (camAngle == 4) {

camAngle = 5;

camera = new THREE.OrthographicCamera(window.innerWidth / -16, window.innerWidth / 16, window.innerHeight / 16, window.innerHeight / -16, -200, 500);

camera.position.x = 20;

camera.position.y = 2;

camera.position.z = 0;

camera.lookAt(scene.position);

this.perspective = "One-point perspective";

} else {

camAngle = 0;

camera = new THREE.OrthographicCamera(window.innerWidth / -16, window.innerWidth / 16, window.innerHeight / 16, window.innerHeight / -16, -200, 500);

camera.position.x = 20;

camera.position.y = 10;

camera.position.z = -20;

camera.lookAt(scene.position);

this.perspective = "Three-point perspective";

}

};

};

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();

gui.add(controls, 'switchCamera');

gui.add(controls, 'perspective').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

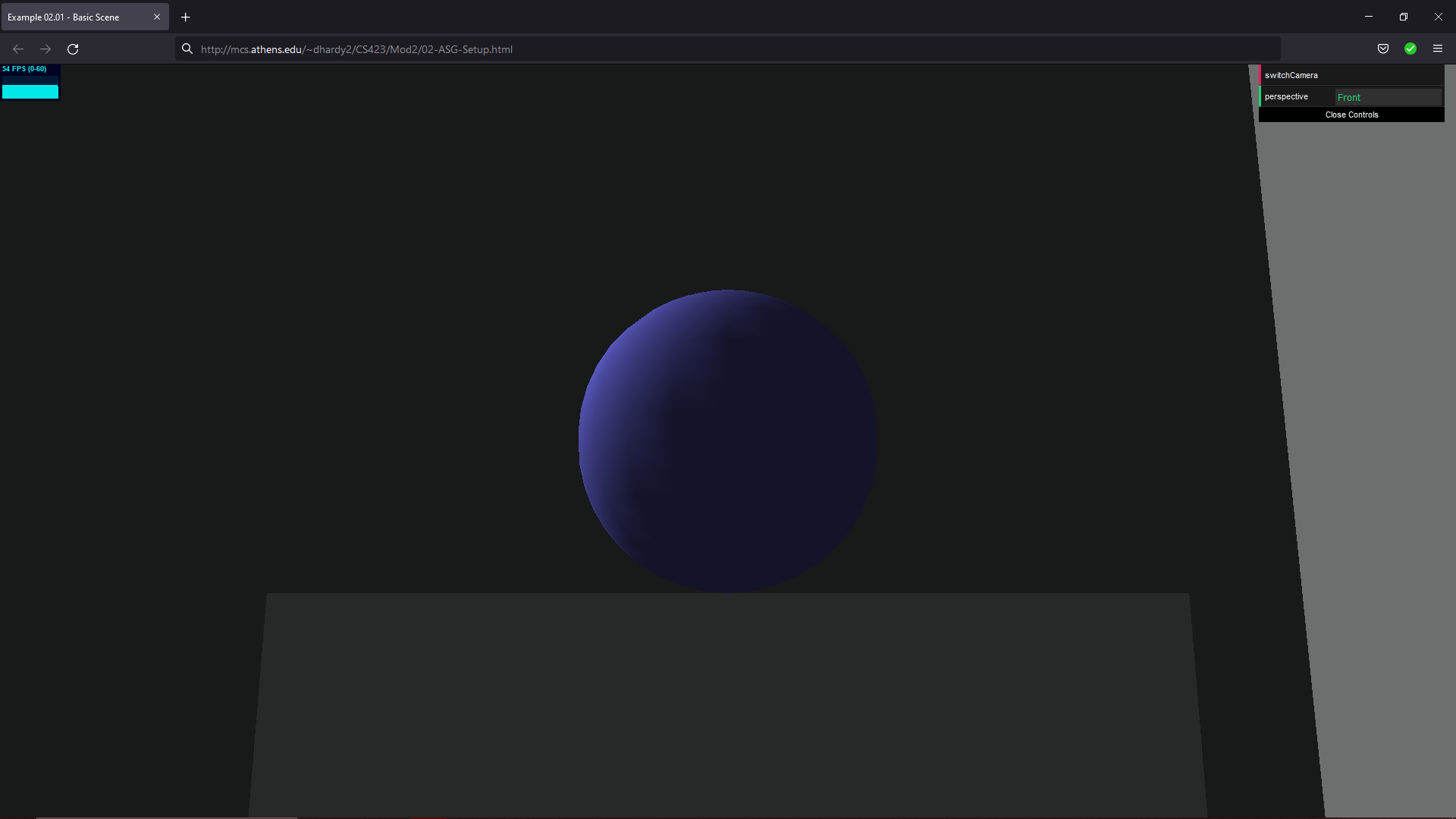
requestAnimationFrame(render);

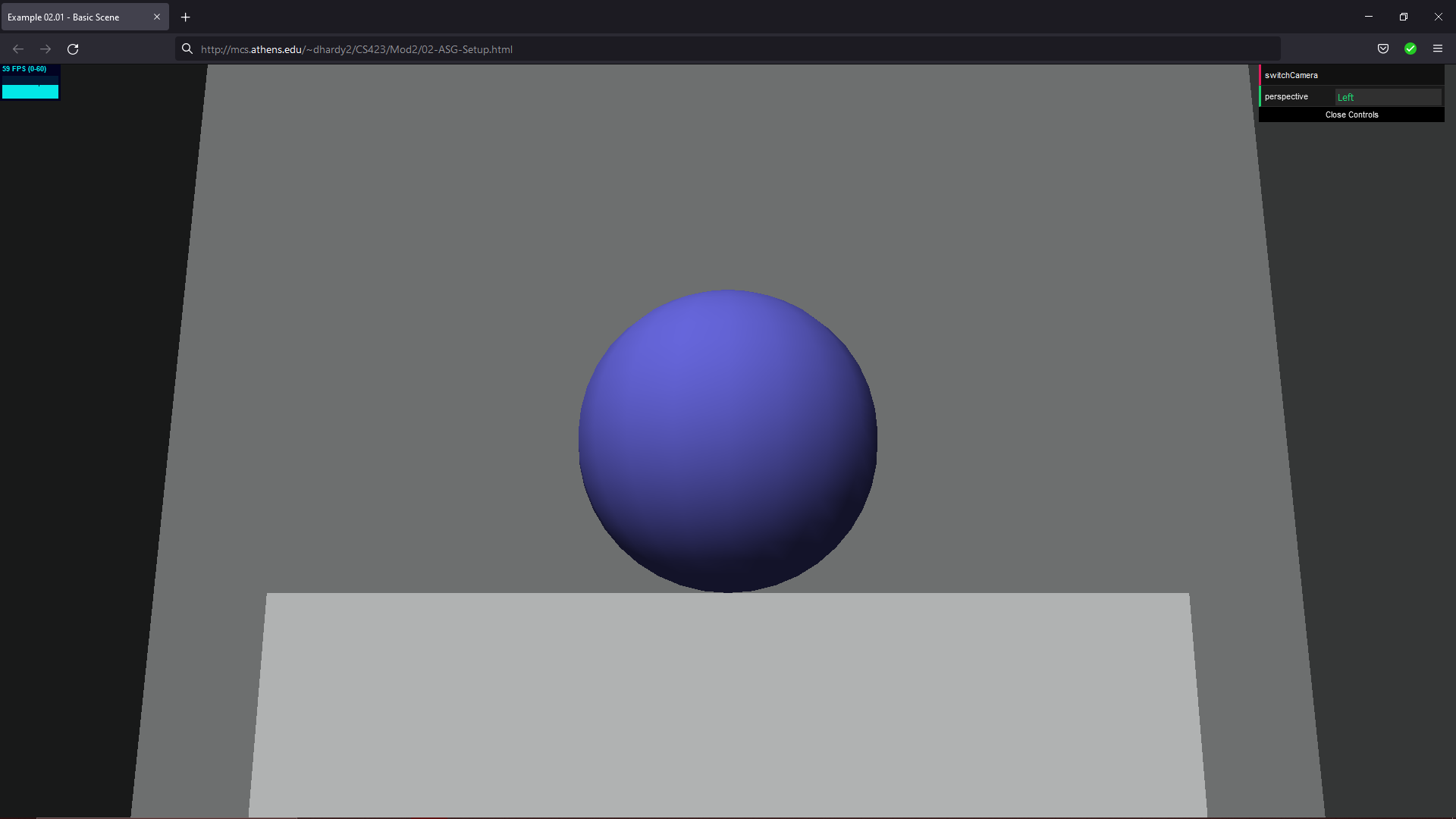
renderer.render(scene, camera);

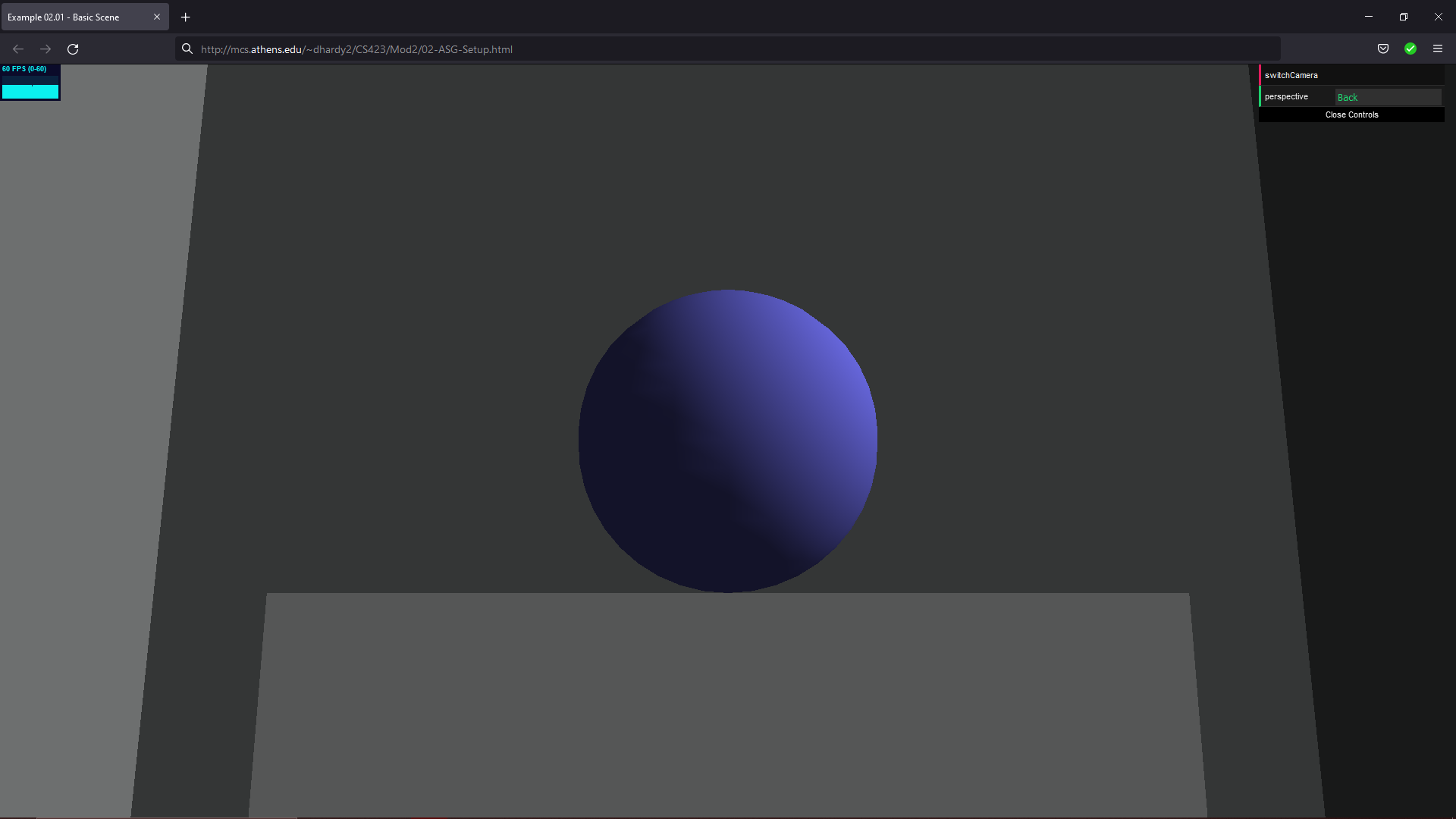
}

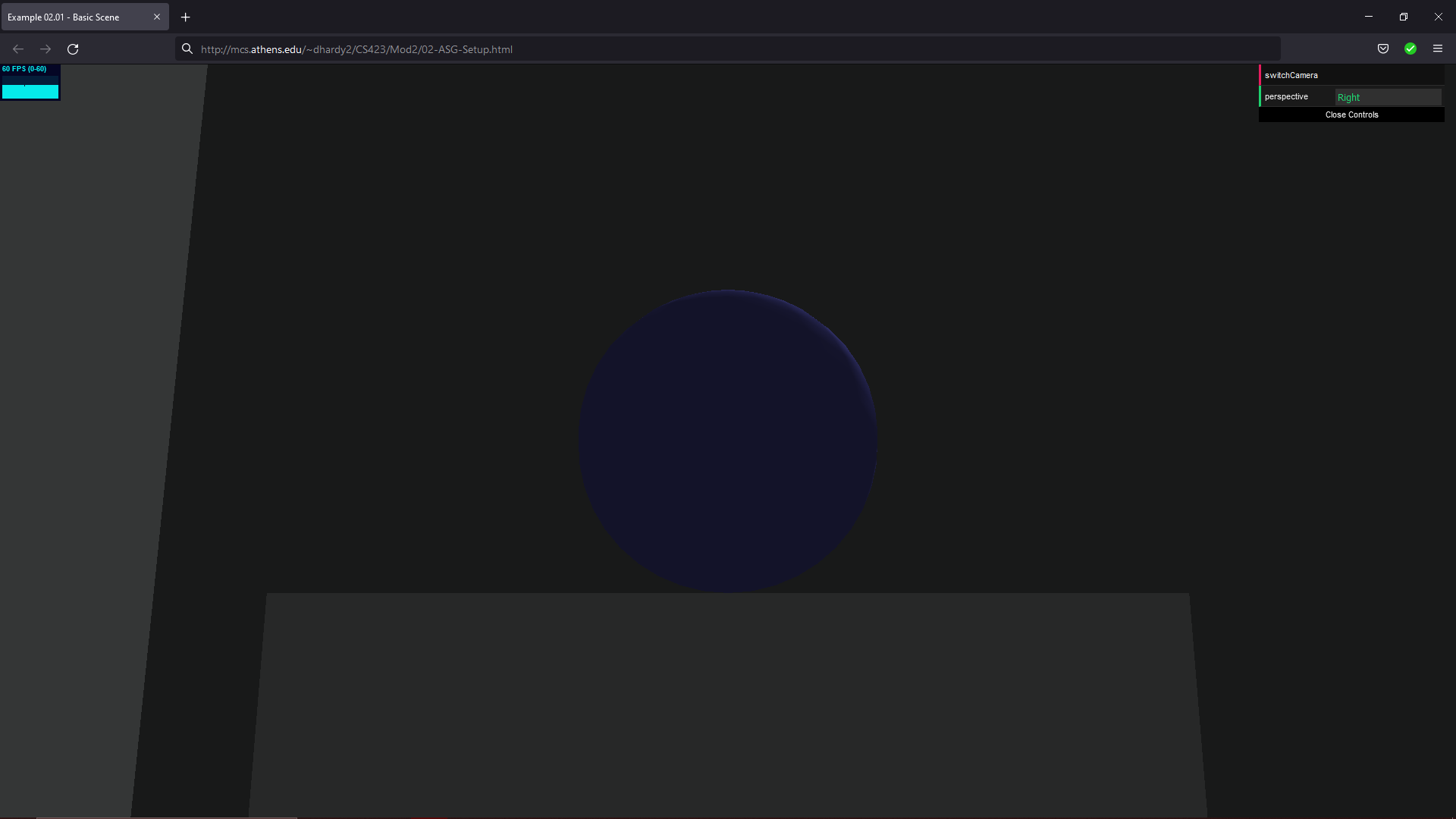
}

window.onload = init;









ASG

<!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-ASG-Setup.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 = 30;

camera.position.y = 20;

camera.position.z = 0;

// 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);

// Room

var cubeGeometry = new THREE.BoxGeometry(1,100,100);

var cubeMaterial = new THREE.MeshLambertMaterial({color: 0x979A9A});

var cube = new THREE.Mesh(cubeGeometry, cubeMaterial);

cube.position.x = 50;

cube.position.y = 25;

cube.position.z = 0;

scene.add(cube);

var cubeGeometry = new THREE.BoxGeometry(100,100,1);

var cubeMaterial = new THREE.MeshLambertMaterial({color: 0x979A9A});

var cube = new THREE.Mesh(cubeGeometry, cubeMaterial);

cube.position.x = 0;

cube.position.y = 25;

cube.position.z = 50;

scene.add(cube);

var cubeGeometry = new THREE.BoxGeometry(1,100,100);

var cubeMaterial = new THREE.MeshLambertMaterial({color: 0x979A9A});

var cube = new THREE.Mesh(cubeGeometry, cubeMaterial);

cube.position.x = -50;

cube.position.y = 25;

cube.position.z = 0;

scene.add(cube);

var cubeGeometry = new THREE.BoxGeometry(100,100,1);

var cubeMaterial = new THREE.MeshLambertMaterial({color: 0x979A9A});

var cube = new THREE.Mesh(cubeGeometry, cubeMaterial);

cube.position.x = 0;

cube.position.y = 25;

cube.position.z = -50;

scene.add(cube);

// Pedestal cube

var cubeGeometry = new THREE.BoxGeometry(20,20,20);

var cubeMaterial = new THREE.MeshLambertMaterial({color: 0xF4F6F7});

var cube = new THREE.Mesh(cubeGeometry, cubeMaterial);

cube.position.x = 0;

cube.position.y = 10;

cube.position.z = 0;

scene.add(cube);

// Item Sphere

var sphereGeometry = new THREE.SphereGeometry(5, 20, 20);

var sphereMaterial = new THREE.MeshLambertMaterial({color: 0x7777ff});

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

sphere.position.x = 0;

sphere.position.y = 25;

sphere.position.z = 0;

scene.add(sphere);

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 camSpot = 1;

var controls = new function () {

this.perspective = "Front";

this.switchCamera = function () {

if (camSpot == 0) {

camSpot = 1;

camera = new THREE.PerspectiveCamera(45, window.innerWidth / window.innerHeight, 0.1, 1000);

camera.position.x = 30;

camera.position.y = 20;

camera.position.z = 0;

camera.lookAt(0, 25, 0);

this.perspective = "Front";

} else if (camSpot == 1) {

camSpot = 2;

camera = new THREE.PerspectiveCamera(45, window.innerWidth / window.innerHeight, 0.1, 1000);

camera.position.x = 0;

camera.position.y = 20;

camera.position.z = 30;

camera.lookAt(0, 25, 0);

this.perspective = "Left";

} else if (camSpot == 2) {

camSpot = 3;

camera = new THREE.PerspectiveCamera(45, window.innerWidth / window.innerHeight, 0.1, 1000);

camera.position.x = -30;

camera.position.y = 20;

camera.position.z = 0;

camera.lookAt(0, 25, 0);

this.perspective = "Back";

} else {

camSpot = 0;

camera = new THREE.PerspectiveCamera(45, window.innerWidth / window.innerHeight, 0.1, 1000);

camera.position.x = 0;

camera.position.y = 20;

camera.position.z = -30;

camera.lookAt(0, 25, 0);

this.perspective = "Right";

}

};

};

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(0, 25, 0);

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