Reickah Collins

Winter 2020

Assignment 1

2/2/2020

Problem #1

/\*\*\*\*\*\*\*\*\*\*\*

\* triangle015.js

\* A square with orbit control

\* M. Laszlo

\* September 2019

\*\*\*\*\*\*\*\*\*\*\*/

let camera, scene, renderer;

let cameraControls;

let clock = new THREE.Clock();

function createScene() {

let n = 8;

let inner = new THREE.Color(0xFFD700);

let outer = new THREE.Color(0x781c2e);

let axes = new THREE.AxesHelper(10);

let polygon\_mesh = regularPolygonGeometry(n, inner, outer);

scene.add(polygon\_mesh);

scene.add(axes);

}

function regularPolygonGeometry(n , innerColor, outerColor)

{

var new\_geo = new THREE.Geometry();

//new\_geo.vertices.push( new THREE.Vector3(0,0,0));

//Creates vertices

for ( var vt = 0 ; vt < n; vt++ )

{

// Rotates counterclockwise around 90 degrees

var angle = (Math.PI/2) + (vt / n) \* 2 \* Math.PI;

var x = 2 \* Math.cos( angle );

var y = 2 \* Math.sin( angle );

//Stores the vertex location

new\_geo.vertices.push( new THREE.Vector3(x,y,0));

}

//Saves colors for each vertex of each face

for ( var i = 0; i< n-2; i++) {

new\_geo.faces.push( new THREE.Face3(0,i+1,i+2));

new\_geo.faces[i].vertexColors[0] = new THREE.Color(innerColor);

new\_geo.faces[i].vertexColors[1] = new THREE.Color(outerColor);

new\_geo.faces[i].vertexColors[2] = new THREE.Color(outerColor);

}

var material = new THREE.MeshBasicMaterial( {vertexColors:THREE.VertexColors, side:THREE.DoubleSide} );

var polygon = new THREE.Mesh(new\_geo, material);

return polygon;

}

function animate() {

window.requestAnimationFrame(animate);

render();

}

function render() {

let delta = clock.getDelta();

cameraControls.update(delta);

renderer.render(scene, camera);

}

function init() {

let canvasWidth = window.innerWidth;

let canvasHeight = window.innerHeight;

let canvasRatio = canvasWidth / canvasHeight;

scene = new THREE.Scene();

renderer = new THREE.WebGLRenderer({antialias : true, preserveDrawingBuffer: true});

renderer.gammaInput = true;

renderer.gammaOutput = true;

renderer.setSize(canvasWidth, canvasHeight);

renderer.setClearColor(0x000000, 1.0);

camera = new THREE.PerspectiveCamera( 40, canvasRatio, 1, 1000);

camera.position.set(0, 0, 30);

camera.lookAt(new THREE.Vector3(0, 0, 0));

cameraControls = new THREE.OrbitControls(camera, renderer.domElement);

}

function addToDOM() {

let container = document.getElementById('container');

let canvas = container.getElementsByTagName('canvas');

if (canvas.length>0) {

container.removeChild(canvas[0]);

}

container.appendChild( renderer.domElement );

}

init();

createScene();

addToDOM();

render();

animate();

Problem #2

/\*\*\*\*\*\*\*\*\*\*\*

\* triangle015.js

\* A square with orbit control

\* M. Laszlo

\* September 2019

\*\*\*\*\*\*\*\*\*\*\*/

let camera, scene, renderer;

let cameraControls;

let clock = new THREE.Clock();

function createScene() {

let n = 10; //number of sides for the cylinder

let rd = 5.0; //radius

let ht = 8.0; //height

let clr = 0x228B22; //cylinder's color

let axes = new THREE.AxesHelper(10);

let cylinder\_mesh = linedCylinder(n, rd, ht, clr);

scene.add(cylinder\_mesh);

//scene.add(axes);

}

function linedCylinder(n, radius, height, color)

{

//Creates a new cylinder object

var new\_geo = new THREE.CylinderGeometry(radius, radius, height,n,1,true);

//Creates material with user generated color and wireframe

var material = new THREE.MeshBasicMaterial( {color: color, wireframe: true} );

//Creates mesh

var cylinder = new THREE.Mesh(new\_geo, material);

//Returns mesh

return cylinder;

}

function animate() {

window.requestAnimationFrame(animate);

render();

}

function render() {

let delta = clock.getDelta();

cameraControls.update(delta);

renderer.render(scene, camera);

}

function init() {

let canvasWidth = window.innerWidth;

let canvasHeight = window.innerHeight;

let canvasRatio = canvasWidth / canvasHeight;

scene = new THREE.Scene();

renderer = new THREE.WebGLRenderer({antialias : true, preserveDrawingBuffer: true});

renderer.gammaInput = true;

renderer.gammaOutput = true;

renderer.setSize(canvasWidth, canvasHeight);

renderer.setClearColor(0x000000, 1.0);

camera = new THREE.PerspectiveCamera( 40, canvasRatio, 1, 1000);

camera.position.set(0, 0, 30);

camera.lookAt(new THREE.Vector3(0, 0, 0));

cameraControls = new THREE.OrbitControls(camera, renderer.domElement);

}

function addToDOM() {

let container = document.getElementById('container');

let canvas = container.getElementsByTagName('canvas');

if (canvas.length>0) {

container.removeChild(canvas[0]);

}

container.appendChild( renderer.domElement );

}

init();

createScene();

addToDOM();

render();

animate();

Problem #3

/\*\*\*\*\*\*\*\*\*\*\*

\* assignment1-3.js

\* Create a cylinder function

\* Reickah Collins

\* Winter 2020

\*\*\*\*\*\*\*\*\*\*\*/

let camera, scene, renderer;

let cameraControls;

let clock = new THREE.Clock();

function createScene() {

let cylinder = cylinderGeometry(10, 4, 4);

let color = new THREE.Color(0, 1, 0);

let mat = new THREE.MeshLambertMaterial({color: color, side: THREE.FrontSide});

mat.polygonOffset = true;

mat.polygonOffsetUnits = 1;

mat.polygonOffsetFactor = 1;

// try side: THREE.FrontSide and THREE.Backside

let pyramid = new THREE.Mesh(cylinder, mat);

let basicMat = new THREE.MeshBasicMaterial({color: 'red', wireframe: true, wireframeLinewidth: 2});

let pyramidWiremesh = new THREE.Mesh(cylinder, basicMat);

let light = new THREE.PointLight(0xFFFFFF, 1, 1000);

light.position.set(0, 0, 10);

let light2 = new THREE.PointLight(0xFFFFFF, 1, 1000);

light2.position.set(0, -10, -10);

let ambientLight = new THREE.AmbientLight(0x222222);

scene.add(light);

scene.add(light2);

scene.add(ambientLight);

scene.add(pyramid);

scene.add(pyramidWiremesh);

}

function cylinderGeometry(n, rad, len) {

let len2 = len / 2;

let geom = new THREE.Geometry();

// push n + 1 vertices

// first the apex...

geom.vertices.push(new THREE.Vector3(rad \* Math.cos(0), -len2, 0));

// and then the vertices of the base

let inc = 2 \* Math.PI / n;

//top verticies of cylinder

for (let i = 0, a = 0; i < n; i++, a += inc) {

let cos = Math.cos(a);

let sin = Math.sin(a);

geom.vertices.push(new THREE.Vector3(rad \* cos, len2, rad \* sin));

}

//bottom verticies of cylinder

for (let i = 0, a = 0; i < n; i++, a += inc) {

let cos = Math.cos(a);

let sin = Math.sin(a);

geom.vertices.push(new THREE.Vector3(rad \* cos, -len2, rad \* sin));

}

for (let i = 0; i < n; i++) {

let face = new THREE.Face3(i, i+1, i+n);

geom.faces.push(face);

let face2 = new THREE.Face3(i+n, i+1+n, i+1);

geom.faces.push(face2);

}

let face = new THREE.Face3(1, n, 0);

geom.faces.push(face);

// and then push the n-2 faces of the base

for (let i = 1; i < n; i++) {

let face = new THREE.Face3(i, i+1, 1);

geom.faces.push(face);

}

// set face normals and return the geometry

geom.computeFaceNormals();

//geom.computeCentroids();

//geom.computeVertexNormals();

return geom;

}

function animate() {

window.requestAnimationFrame(animate);

render();

}

function render() {

let delta = clock.getDelta();

cameraControls.update(delta);

renderer.render(scene, camera);

}

function init() {

let canvasWidth = window.innerWidth;

let canvasHeight = window.innerHeight;

let canvasRatio = canvasWidth / canvasHeight;

scene = new THREE.Scene();

renderer = new THREE.WebGLRenderer({antialias : true});

renderer.gammaInput = true;

renderer.gammaOutput = true;

renderer.setSize(canvasWidth, canvasHeight);

renderer.setClearColor(0x000000, 1.0);

camera = new THREE.PerspectiveCamera(40, canvasRatio, 1, 1000);

camera.position.set(0, 0, 12);

camera.lookAt(new THREE.Vector3(0, 0, 0));

cameraControls = new THREE.OrbitControls(camera, renderer.domElement);

}

function addToDOM() {

let container = document.getElementById('container');

let canvas = container.getElementsByTagName('canvas');

if (canvas.length>0) {

container.removeChild(canvas[0]);

}

container.appendChild( renderer.domElement );

}

init();

createScene();

addToDOM();

render();

animate();