БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

ИИТ БГУИР

Программное обеспечение информационных технологий

Отчет

По лабораторной работе №5

по курсу «Алгоритмы компьютерной графики»

Вариант 12

Выполнил студент группы 481064 Сорока А.А.

Проверил преподаватель: Коренская И.Н.

Минск 2016 г

**Цель работы**:

**Задание:**

При выполнении данной лабораторной работы необходимо построить заданную проекцию заданной фигуры.

В процессе выполнения работы необходимо:

1. задать матрицу преобразования;
2. построить проволочную модель объекта;
3. удалить невидимые ребра;
4. обеспечить заданную динамику отображения объекта.

**Код программы:**

index.html

<html lang="en">

<head>

<title>three.js webgl - geometry - cube</title>

<meta charset="utf-8">

<meta name="viewport" content="width=device-width, user-scalable=no, minimum-scale=1.0, maximum-scale=1.0">

<style>

body {

color: #ffffff;

font-family: Monospace;

font-size: 13px;

text-align: center;

font-weight: bold;

background-color: #000000;

margin: 0px;

overflow: hidden;

}

#info {

color: #fff;

position: absolute;

top: 0px;

width: 100%;

padding: 5px;

z-index: 100;

}

</style>

</head>

<body>

<script src="three.js"></script>

<script>

var arrayLand = Array(6);

var arrayEdges = Array(6);

var arrayBottom = Array(6);

var height = 300;

var rotation = 0;

var size = 10;

var camera, scene, renderer;

var mesh;

var line;

var line2;

var dash = 1;

var geometry;

var geometry2;

var material;

var material2;

var arr = Array(1);

var arrayPoints = [[0, -100, 0], [100, -100, -173], [273, -100, -173], [373, -100, 0], [273, -100, 173], [100, -100, 173], [0, -100, 0]];

var tmp;

init();

animate();

function init() {

camera = new THREE.OrthographicCamera(window.innerWidth / -2, window.innerWidth / 2, window.innerHeight / 2, window.innerHeight / -2, 1, 5000);

// camera = new THREE.PerspectiveCamera(30, window.innerWidth / window.innerHeight, 1, 2000);

camera.position.z = 800;

camera.position.x = 200;

scene = new THREE.Scene();

material2 = new THREE.LineDashedMaterial({

color: 0xffffff,

dashSize: dash,

gapSize: 0.00001

});

tmp = new THREE.Geometry();

tmp.vertices.push(new THREE.Vector3(0, 0, 0));

tmp.vertices.push(new THREE.Vector3(400, 0, 0));

tmp.computeLineDistances();

arr[3] = new THREE.Line(tmp, material2);

scene.add(arr[3]);

for (var i = 0; i < 6; i++) {

arrayLand[i] = {

geometry: new THREE.Geometry(),

material: new THREE.LineDashedMaterial({

color: 0xffaa00,

dashSize: 5,

gapSize: 0.0001,

linewidth: 10

}),

line: null

};

arrayLand[i].geometry.vertices.push(new THREE.Vector3(arrayPoints[i][0], arrayPoints[i][1], arrayPoints[i][2]));

arrayLand[i].geometry.vertices.push(new THREE.Vector3(arrayPoints[i + 1][0], arrayPoints[i + 1][1], arrayPoints[i + 1][2]));

arrayLand[i].geometry.computeLineDistances();

arrayLand[i].line = new THREE.LineSegments(arrayLand[i].geometry, arrayLand[i].material);

scene.add(arrayLand[i].line);

arrayBottom[i] = {

geometry: new THREE.Geometry(),

material: new THREE.LineDashedMaterial({

color: 0xffaa00,

dashSize: 5,

gapSize: 0.0001,

linewidth: 10

}),

line: null

};

arrayBottom[i].geometry.vertices.push(new THREE.Vector3(arrayPoints[i][0], arrayPoints[i][1] + height, arrayPoints[i][2]));

arrayBottom[i].geometry.vertices.push(new THREE.Vector3(arrayPoints[i + 1][0], arrayPoints[i + 1][1] + height, arrayPoints[i + 1][2]));

arrayBottom[i].geometry.computeLineDistances();

arrayBottom[i].line = new THREE.LineSegments(arrayBottom[i].geometry, arrayBottom[i].material);

scene.add(arrayBottom[i].line);

arrayEdges[i] = {

geometry: new THREE.Geometry(),

material: new THREE.LineDashedMaterial({

color: 0xffaa00,

dashSize: 5,

gapSize: 0.0001,

linewidth: 10

}),

line: null

};

arrayEdges[i].geometry.vertices.push(new THREE.Vector3(arrayPoints[i][0], arrayPoints[i][1], arrayPoints[i][2]));

arrayEdges[i].geometry.vertices.push(new THREE.Vector3(arrayPoints[i][0], arrayPoints[i][1] + height, arrayPoints[i][2]));

arrayEdges[i].geometry.computeLineDistances();

arrayEdges[i].line = new THREE.LineSegments(arrayEdges[i].geometry, arrayEdges[i].material);

scene.add(arrayEdges[i].line);

}

for (var i = 0; i < 6; i++) {

arrayLand[i].line.rotation.y = 0.2;

arrayEdges[i].line.rotation.y = 0.2;

arrayBottom[i].line.rotation.y = 0.2;

}

renderer = new THREE.WebGLRenderer({

antialias: true

});

renderer.setClearColor(0x111111, 1);

renderer.setPixelRatio(window.devicePixelRatio);

renderer.setSize(window.innerWidth, window.innerHeight);

document.body.appendChild(renderer.domElement);

window.addEventListener('resize', onWindowResize, false);

}

function onWindowResize() {

camera.aspect = window.innerWidth / window.innerHeight;

camera.updateProjectionMatrix();

renderer.setSize(window.innerWidth, window.innerHeight);

}

function animate() {

requestAnimationFrame(animate);

for (var i = 0; i < 6; i++) {

arrayLand[i].line.rotation.x = rotation;

arrayEdges[i].line.rotation.x = rotation;

arrayBottom[i].line.rotation.x = rotation;

}

rotation = (rotation >= 6.28 ? 0 : rotation + 0.005)

if (rotation < 3.14) {

mid = rotation < 1.57;

} else {

mid = rotation < 4.71;

}

for (var i = 0; i < 6; i++) {

arrayLand[i].material.gapSize = 0.0001;

arrayEdges[i].material.gapSize = 0.0001;

arrayBottom[i].material.gapSize = 0.0001;

}

if (rotation < 3.14) {

if (mid) {

arrayLand[0].material.gapSize = size;

arrayLand[1].material.gapSize = size;

arrayLand[2].material.gapSize = size;

arrayEdges[1].material.gapSize = size;

arrayEdges[2].material.gapSize = size;

}

if (!mid) {

arrayLand[3].material.gapSize = size;

arrayLand[4].material.gapSize = size;

arrayLand[5].material.gapSize = size;

arrayEdges[4].material.gapSize = size;

arrayEdges[5].material.gapSize = size;

}

} else {

if (mid) {

arrayBottom[3].material.gapSize = size;

arrayBottom[4].material.gapSize = size;

arrayBottom[5].material.gapSize = size;

arrayEdges[4].material.gapSize = size;

arrayEdges[5].material.gapSize = size;

}

if (!mid) {

arrayBottom[0].material.gapSize = size;

arrayBottom[1].material.gapSize = size;

arrayBottom[2].material.gapSize = size;

arrayEdges[1].material.gapSize = size;

arrayEdges[2].material.gapSize = size;

}

}

renderer.render(scene, camera);

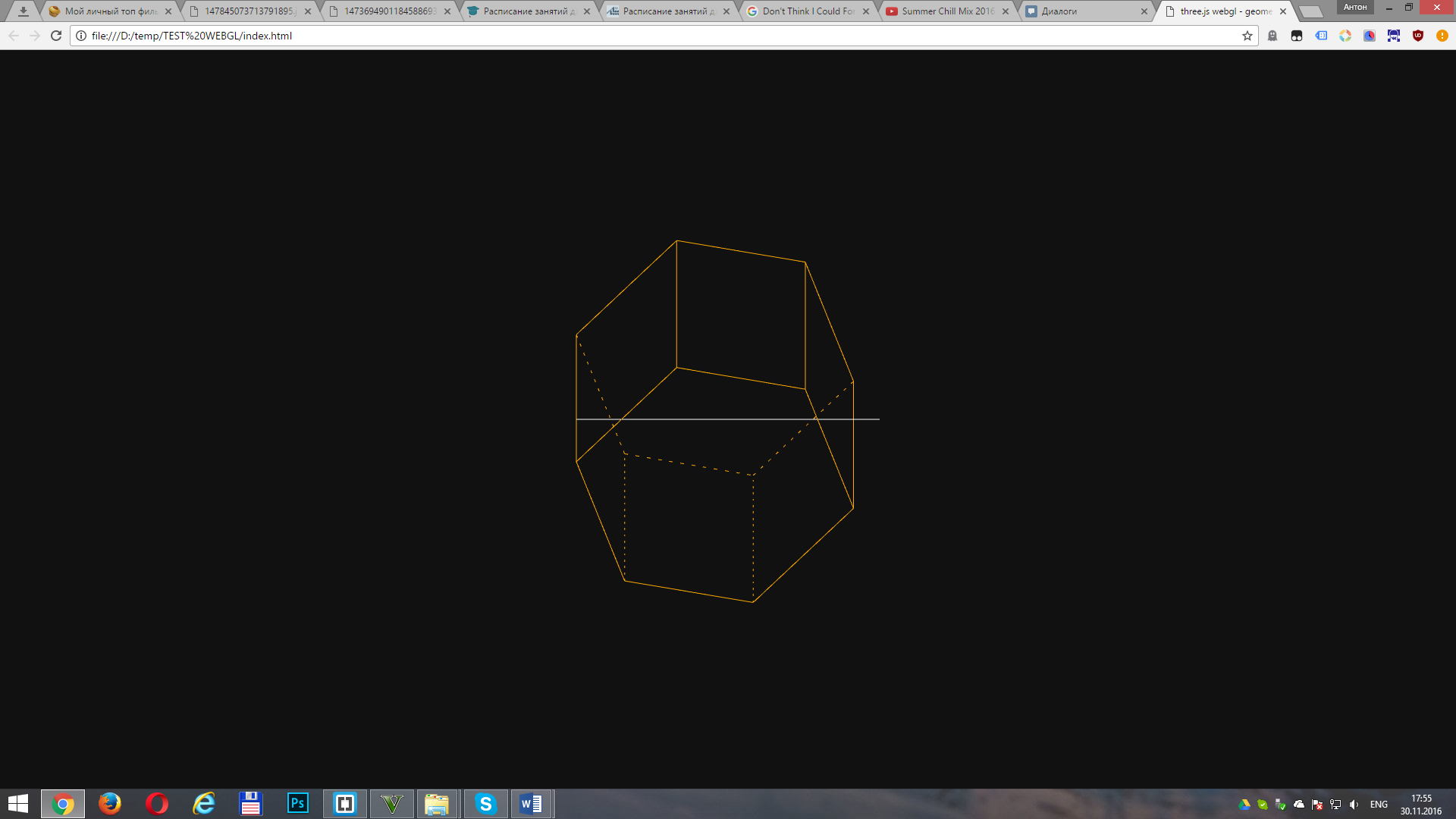
}

</script>

</body>

</html>

**Скриншот выполнения программы:**



**Вывод**: в ходе выполнения лабораторной работы была разработано программное средство на языке JavaScript, служащее для создания анимации движущейся шестиугольной призмы.