**МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ**

**МИНИСТЕРСТВО ТРАНСПОРТА РОССИЙСКОЙ ФЕДЕРАЦИИ (МИНТРАНС РОССИИ)**

**ФЕДЕРАЛЬНОЕ АГЕНТСТВО ЖЕЛЕЗНОДОРОЖНОГО ТРАНСПОРТА (РОСЖЕЛДОР)**

**ГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ**

**ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ**

**СИБИРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ**

**ПУТЕЙ СООБЩЕНИЯ (СГУПС)**

**Кафедра «Общая информатика»**

**Расчетно-графическая работа**

**Вариант 11**

**по дисциплине**

**«Компьютерная геометрия и графика»**

**Преподаватели: Студент гр. ИСТ-411**

**Машуков В.И. \_\_\_\_\_\_\_\_\_\_\_\_\_Сапронов Е.С.****Роганов С.А.** *(подпись)*

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*(дата сдачи на проверку) (дата проверки)*

**Краткая рецензия:**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**(оценка по результатам защиты) (подпись преподавателя)**

2014 ГОД

**Введение**

**OpenGL — это мощный программный интерфейс**, применяемый для получения высококачественных, программно генерируемых изображений и интерактивных приложений, использующих двух- и трехмерные объекты, а также растровые изображения.

**OpenGL - это графический стандарт в области компьютерной графики**. На данный момент он является одним из самых популярных графических стандартов во всём мире. Ещё в 1982 г. в Стенфордском университете была разработана концепция графической машины, на основе которой фирма Silicon Graphics в своей рабочей станции Silicon IRIS реализовала конвейер рендеринга. Таким образом была разработана графическая библиотека IRIS GL. На основе библиотеки IRIS GL, в 1992 году был разработан и утверждён графический стандарт OpenGL. Разработчики OpenGL - это крупнейшие фирмы разработчики как оборудования так и программного обеспечения: Silicon Graphics, Inc., Microsoft, IBM Corporation, Sun Microsystems, Inc., Digital Equipment Corporation (DEC), Evans & Sutherland, Hewlett-Packard Corporation, Intel Corporation и Intergraph Corporation.

OpenGL переводится как Открытая Графическая Библиотека (Open Graphics Library), это означает, что OpenGL - это открытый и мобильный стандарт. Программы, написанные с помощью OpenGL можно переносить практически на любые платформы, получая при этом одинаковый результат, будь это графическая станция или суперкомпьютер. OpenGL освобождает программиста от написания программ для конкретного оборудования. Если устройство поддерживает какую-то функцию, то эта функция выполняется аппаратно, если нет, то библиотека выполняет её программно.

Инструкция пользователя

1. Запустить в современном браузере, с поддержкой WebGl, файл index.html;
2. Для включения/выключения света в доме используйте клавишу Space;
3. Для перемещения дома по оси X, следует выполнить следющие действия:
   1. Зажать клавишу Shift,
   2. Удерживая левую кнопку мыши перемещать объект(дом).

Для обзора дома используйте мышь с зажатой левой кнопкой.

Инструкция разработчика

Работа выполнена с использованием WebGl. В качестве языка программирования был выбран JavaScript. Прослойкой между программистом и WebGl служит JavaScript библиотека THREE.js.

# Алгоритм

Создать модель дома в трехмерном пространстве. У дома должны быть: 4 окна, дверь. Внутри дома находятся: стол, картина, предмет на столе.

Должен быть реализован механизм отбрасывания тени объектами, например тень от дома падает на поверхность травы.

В качестве элементов управления следует использовать мышь и клавиатуру. Мышью выполняется обзор дома и перемещение объектов. Клавиатура служит для управления режимом освещения и переключением с режима обзора на режим перемещения дома.

# Документация

Основные функции:

* *function init()* — инициализирует основные объекты: сцену, камеру, контейнер для отображения сцены, отрисовщик. Вызывает функции для создания 3D объектов.
* *function createMaterials()* — создает материалы, которые используются в качестве поверхности для объектов.
* *function createLights() —* создает объекты освещения.
* *function createGround()* — создает землю под домом.
* *function createWalls()* — создает стены дома.
* *function createRoof()* — создает крышу дома.
* *function createTable()* — создает стол.
* *function createFloor()* — создает пол в доме.
* *function createPicture()* — создает картину на стене.
* *function createModel()* — создает модель компьютера на столе.
* *function createWorld()* — создает мир вокруг дома.
* *function animate()* — перезапускает отрисовку.
* *function Render()* — метод в котором происходит отрисовка кадра.

Основные глобальные переменные:

* *group* — объект, содержащий в себе все элементы дома.
* *container* — DOM элемент в котором происходит отрисовка.
* *camera* — камера для обзора сцены.
* *scene* — объект сцены.

# Линстинг

var container;

var camera, scene, renderer, controls, light;

var group;

var houseLight, pointLight, lamp;

var brickMaterial, wallMaterial, roofMaterial, woodMaterial, floorMaterial, grassMaterial, pictureMaterial, worldMaterial;

var house, mouse\_x;

createMaterials();

init();

animate();

function init() {

scene = new THREE.Scene();

container = document.createElement('div');

document.body.appendChild(container);

var SCREEN\_WIDTH = window.innerWidth, SCREEN\_HEIGHT = window.innerHeight;

var VIEW\_ANGLE = 45, ASPECT = SCREEN\_WIDTH / SCREEN\_HEIGHT, NEAR = 0.1, FAR = 10000;

camera = new THREE.PerspectiveCamera(VIEW\_ANGLE, ASPECT, NEAR, FAR);

camera.position.set(0, 50, 200);

camera.lookAt(scene.position);

renderer = new THREE.WebGLRenderer();

renderer.setSize(SCREEN\_WIDTH, SCREEN\_HEIGHT);

renderer.shadowMapEnabled = true;

renderer.shadowMapSoft = true;

container.appendChild(renderer.domElement);

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

group = new THREE.Object3D();

group.position.z = -5;

scene.add(group);

createLights();

createGround();

createWalls();

createRoof();

createTable();

createFloor();

createPicture();

createModel();

createWorld();

document.addEventListener('keydown', onDocumentKeyDown, false);

document.addEventListener('keyup', onDocumentKeyUp, false);

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

}

function onMouseDown(event) {

mouse\_x = event.clientX;

house = group;

}

function onMouseMove(event) {

if (!house) return;

var x\_diff = event.clientX - mouse\_x;

house.\_\_dirtyPosition = true;

var newPosition = house.position.x + x\_diff;

if (newPosition < 480 && newPosition > -480) {

house.position.x = newPosition;

mouse\_x = event.clientX;

}

}

function onMouseUp(event) {

house = false;

}

function createModel() {

var loader = new THREE.STLLoader();

loader.addEventListener('load', function (event) {

var geometry = event.content;

var material = new THREE.MeshPhongMaterial({ ambient: 0xff5533, color: 0xff5533, specular: 0x111111, shininess: 200 });

var mesh = new THREE.Mesh(geometry, material);

mesh.scale.set(0.05, 0.05, 0.05);

mesh.position.y = 5.85;

mesh.position.z = 1;

mesh.position.x = 1;

mesh.rotation.y = -Math.PI / 2;

mesh.castShadow = true;

mesh.receiveShadow = true;

group.add(mesh);

});

loader.load('model/model.stl');

}

function createTable() {

var geometry = new THREE.BoxGeometry(7, 0.2, 5);

var mesh = new THREE.Mesh(geometry, woodMaterial);

mesh.position.y = 3;

mesh.position.z = 2.5;

mesh.castShadow = true;

mesh.receiveShadow = true;

group.add(mesh);

var geometry = new THREE.BoxGeometry(0.2, 3, 0.2);

var mesh = new THREE.Mesh(geometry, woodMaterial);

mesh.position.y = 1.5;

mesh.position.z = 1;

mesh.position.x = 2;

mesh.castShadow = true;

mesh.receiveShadow = true;

group.add(mesh);

var geometry = new THREE.BoxGeometry(0.2, 3, 0.2);

var mesh = new THREE.Mesh(geometry, woodMaterial);

mesh.position.y = 1.5;

mesh.position.z = 1;

mesh.position.x = -2;

mesh.castShadow = true;

mesh.receiveShadow = true;

group.add(mesh);

var geometry = new THREE.BoxGeometry(0.2, 3, 0.2);

var mesh = new THREE.Mesh(geometry, woodMaterial);

mesh.position.y = 1.5;

mesh.position.z = 4;

mesh.position.x = 2;

mesh.castShadow = true;

mesh.receiveShadow = true;

group.add(mesh);

var geometry = new THREE.BoxGeometry(0.2, 3, 0.2);

var mesh = new THREE.Mesh(geometry, woodMaterial);

mesh.position.y = 1.5;

mesh.position.z = 4;

mesh.position.x = -2;

mesh.castShadow = true;

mesh.receiveShadow = true;

group.add(mesh);

}

function createWalls() {

var material = new THREE.MeshPhongMaterial({ side: THREE.DoubleSide, opacity: 0.3, transparent: true });

var material2 = new THREE.MeshPhongMaterial({ side: THREE.DoubleSide, opacity: 0.0, transparent: true});

var materials = [

material,

brickMaterial,

wallMaterial,

material2

];

// Back wall with 2 windows

var geometry = new THREE.PlaneGeometry(40, 10, 10, 10);

var l = geometry.faces.length;

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

geometry.faces[i].materialIndex = 1;

}

for (var i = 44; i <= 134; i += 10) {

geometry.faces[i].materialIndex = 0;

geometry.faces[i + 1].materialIndex = 0;

}

var mesh = new THREE.Mesh(geometry, new THREE.MeshFaceMaterial(materials));

mesh.position.y = 5;

mesh.position.z = 10;

mesh.castShadow = true;

mesh.receiveShadow = true;

group.add(mesh);

// Front wall with door

var geometry = new THREE.PlaneGeometry(10, 10, 10, 10);

geometry.applyMatrix(new THREE.Matrix4().makeRotationY(-Math.PI / 2));

var l = geometry.faces.length;

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

geometry.faces[i].materialIndex = 1;

}

for (var i = 82; i <= 182; i += 20) {

geometry.faces[i].materialIndex = 3;

geometry.faces[i + 1].materialIndex = 3;

geometry.faces[i + 2].materialIndex = 3;

geometry.faces[i + 3].materialIndex = 3;

}

var mesh = new THREE.Mesh(geometry, new THREE.MeshFaceMaterial(materials));

mesh.position.y = 5;

mesh.position.z = 5;

mesh.position.x = 20;

mesh.castShadow = true;

mesh.receiveShadow = true;

group.add(mesh);

// Wall

var geometry = new THREE.PlaneGeometry(40, 10, 10, 10);

var l = geometry.faces.length;

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

geometry.faces[i].materialIndex = 1;

}

var mesh = new THREE.Mesh(geometry, new THREE.MeshFaceMaterial(materials));

mesh.position.y = 5;

// mesh.position.z = 10;

mesh.castShadow = true;

mesh.receiveShadow = true;

group.add(mesh);

// Side wall with two windows

var geometry = new THREE.PlaneGeometry(10, 10, 10, 10);

geometry.applyMatrix(new THREE.Matrix4().makeRotationY(-Math.PI / 2));

var l = geometry.faces.length;

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

geometry.faces[i].materialIndex = 1;

}

for (var i = 42; i <= 132; i += 10) {

geometry.faces[i].materialIndex = 0;

geometry.faces[i + 1].materialIndex = 0;

geometry.faces[i + 2].materialIndex = 0;

geometry.faces[i + 3].materialIndex = 0;

}

var mesh = new THREE.Mesh(geometry, new THREE.MeshFaceMaterial(materials));

mesh.position.y = 5;

mesh.position.z = 5;

mesh.position.x = -20;

mesh.castShadow = true;

mesh.receiveShadow = true;

group.add(mesh);

}

function createRoof() {

var geometry = new THREE.PlaneGeometry(40, 8.5, 10, 10);

geometry.applyMatrix(new THREE.Matrix4().makeRotationX(Math.PI / 4));

var mesh = new THREE.Mesh(geometry, roofMaterial);

mesh.position.z = 2;

mesh.position.y = 12;

mesh.castShadow = true;

mesh.receiveShadow = true;

group.add(mesh);

var geometry = new THREE.PlaneGeometry(40, 8.5, 10, 10);

geometry.applyMatrix(new THREE.Matrix4().makeRotationX(-Math.PI / 4));

var mesh = new THREE.Mesh(geometry, roofMaterial);

mesh.position.z = 8;

mesh.position.y = 12;

mesh.castShadow = true;

mesh.receiveShadow = true;

group.add(mesh);

var geometry = new THREE.Geometry();

v1 = new THREE.Vector3(0, 0, 0);

v2 = new THREE.Vector3(10, 0, 0);

v3 = new THREE.Vector3(5, 5, 0);

geometry.vertices.push(v1);

geometry.vertices.push(v2);

geometry.vertices.push(v3);

geometry.faces.push(new THREE.Face3(0, 1, 2));

geometry.applyMatrix(new THREE.Matrix4().makeRotationY(-Math.PI / 2));

var triangleMaterial = new THREE.MeshPhongMaterial({

color: 0xFFFFFF,

side: THREE.DoubleSide

});

var mesh = new THREE.Mesh(geometry, triangleMaterial);

mesh.position.y = 10;

mesh.position.z = 0;

mesh.position.x = 20;

mesh.castShadow = true;

mesh.receiveShadow = true;

group.add(mesh);

var mesh = new THREE.Mesh(geometry, triangleMaterial);

mesh.position.y = 10;

mesh.position.z = 0;

mesh.position.x = -20;

mesh.castShadow = true;

mesh.receiveShadow = true;

group.add(mesh);

}

function createFloor() {

var geometry = new THREE.PlaneGeometry(40, 10, 15, 15);

geometry.applyMatrix(new THREE.Matrix4().makeRotationX(Math.PI / 2));

var mesh = new THREE.Mesh(geometry, floorMaterial);

mesh.position.y = 0.2;

mesh.position.z = 5;

mesh.receiveShadow = true;

group.add(mesh);

}

function createGround() {

var geometry = new THREE.PlaneGeometry(1000, 1000, 10, 14);

geometry.applyMatrix(new THREE.Matrix4().makeRotationX(Math.PI / 2));

var mesh = new THREE.Mesh(geometry, grassMaterial);

mesh.position.y = 0;

mesh.receiveShadow = true;

scene.add(mesh);

}

function createPicture() {

var geometry = new THREE.PlaneGeometry(3, 4);

var mesh = new THREE.Mesh(geometry, pictureMaterial);

mesh.position.z = 0.1;

mesh.position.y = 6;

mesh.position.x = 6;

group.add(mesh);

}

function createWorld() {

var geomemtry = new THREE.BoxGeometry(1000, 1000, 1000)

var mesh = new THREE.Mesh(geomemtry, worldMaterial);

mesh.position.y = 400;

scene.add(mesh);

}

function createLights() {

var sunlight = new THREE.DirectionalLight(0xffffff, 0.6);

sunlight.position.set(140, 100, 100);

sunlight.castShadow = true;

sunlight.shadowDarkness = 0.6;

sunlight.shadowCameraNear = 100;

sunlight.shadowCameraFar = 400;

sunlight.shadowCameraLeft = -100;

sunlight.shadowCameraRight = 100;

sunlight.shadowCameraTop = 100;

sunlight.shadowCameraBottom = -100;

scene.add(sunlight);

var sunlight2 = new THREE.DirectionalLight(0xffffff, 0.6);

sunlight2.position.set(100, 200, -70);

scene.add(sunlight2);

var sunlight3 = new THREE.DirectionalLight(0xffffff, 0.6);

sunlight3.position.set(-100, 200, 70);

scene.add(sunlight3);

houseLight = new THREE.SpotLight(0xffffff, 0.5);

houseLight.position.set(0, 11, 3);

houseLight.castShadow = true;

houseLight.shadowDarkness = 0.5;

houseLight.shadowCameraNear = 3;

houseLight.shadowCameraFar = 12;

houseLight.shadowCameraLeft = -3;

houseLight.shadowCameraRight = 3;

houseLight.shadowCameraTop = 3;

houseLight.shadowCameraBottom = -3;

group.add(houseLight);

pointLight = new THREE.PointLight(0xffffff, 3, 20);

pointLight.position = houseLight.position;

group.add(pointLight);

var sphere = new THREE.SphereGeometry(0.5, 16, 8);

lamp = new THREE.Mesh(sphere, new THREE.MeshBasicMaterial({ color: 0xffffff }));

lamp.position = houseLight.position;

group.add(lamp);

}

function createMaterials() {

var brickTexture = new THREE.ImageUtils.loadTexture('bricks.jpg');

brickTexture.wrapS = brickTexture.wrapT = THREE.RepeatWrapping;

brickTexture.repeat.set(2, 2);

brickMaterial = new THREE.MeshPhongMaterial({map: brickTexture, side: THREE.DoubleSide});

var wallTexture = new THREE.ImageUtils.loadTexture('wall.jpg');

wallTexture.wrapS = wallTexture.wrapT = THREE.RepeatWrapping;

wallTexture.repeat.set(2, 2);

wallMaterial = new THREE.MeshPhongMaterial({map: wallTexture, side: THREE.BackSide});

var roofTexture = new THREE.ImageUtils.loadTexture('roof.jpg');

roofTexture.wrapS = roofTexture.wrapT = THREE.RepeatWrapping;

roofTexture.repeat.set(2, 2);

roofMaterial = new THREE.MeshPhongMaterial({map: roofTexture, side: THREE.DoubleSide});

var woodTexture = new THREE.ImageUtils.loadTexture('wood.jpg');

woodMaterial = new THREE.MeshPhongMaterial({map: woodTexture, side: THREE.DoubleSide});

var floorTexture = new THREE.ImageUtils.loadTexture('checkerboard.jpg');

floorTexture.wrapS = floorTexture.wrapT = THREE.RepeatWrapping;

floorTexture.repeat.set(10, 10);

floorMaterial = new THREE.MeshPhongMaterial({ map: floorTexture, side: THREE.DoubleSide, overdraw: 1 });

var grassTexture = new THREE.ImageUtils.loadTexture('grass.jpg');

grassMaterial = new THREE.MeshPhongMaterial({map: grassTexture, side: THREE.DoubleSide, overdraw: 0});

var pictureTexture = new THREE.ImageUtils.loadTexture('picture.jpg');

pictureMaterial = new THREE.MeshPhongMaterial({map: pictureTexture, side: THREE.FrontSide});

var worldTexture = new THREE.ImageUtils.loadTexture('world.jpg');

worldMaterial = new THREE.MeshPhongMaterial({map: worldTexture, side: THREE.DoubleSide});

return {brickMaterial: brickMaterial, wallMaterial: wallMaterial, roofMaterial: roofMaterial, woodMaterial: woodMaterial, floorMaterial: floorMaterial, grassMaterial: grassMaterial, pictureMaterial: pictureMaterial, worldMaterial: worldMaterial};

}

function onWindowResize() {

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

camera.updateProjectionMatrix();

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

}

function onDocumentKeyDown(event) {

if (event.keyCode == 16) {

controls.enabled = false;

document.addEventListener('mousedown', onMouseDown, false);

document.addEventListener('mouseup', onMouseUp, false);

document.addEventListener('mousemove', onMouseMove, false);

}

if (event.keyCode == 32) {

if (houseLight.intensity == 0) {

houseLight.intensity = 0.5;

pointLight.intensity = 3;

lamp.visible = true;

} else {

houseLight.intensity = 0;

pointLight.intensity = 0;

lamp.visible = false;

}

}

}

function onDocumentKeyUp(event) {

if (event.keyCode == 16) {

controls.enabled = true;

document.removeEventListener('mousedown', onMouseDown, false);

document.removeEventListener('mousemove', onMouseMove, false);

document.removeEventListener('mouseup', onMouseUp, false);

}

}

function animate() {

requestAnimationFrame(animate);

controls.update();

render();

}

function render() {

renderer.render(scene, camera);

}