МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

Отчет
по лабораторной работе №3, 4
по дисциплине «КОМПЬЮТЕРНАЯ 3D-ГРАФИКА»
Тема: 3D ОСВЕЩЕНИЕ. ТЕКСТУРЫ ИЗОБРАЖЕНИЯ

Студентка гр. 5303 _____ Допира В.Е.

Преподаватель _____ Герасимова Т.В.

Санкт-Петербург
2019

Требования к работе:

- Добавить источник света.
- Добавить второй источник света.
- Отредактируйте свою собственную картинку и сделайте ее текстурной картой.
- Разложите текстуру по изображению.

Узнайте больше о взаимодействии материалов и свойств освещения и примените это на вашей сцене.

Ход работы

Шаг 1. Переписать код программы на TreeJS.

Шаг 2. Освещение.

1. Создание переменной для света, который излучается из одной точки во всех направлениях. Общий случай использования - свет, испускаемый от лампочки. В функции `THREE.PointLight` необязательный передаваемый параметр задает шестнадцатеричный цвет света. Значение по умолчанию-0xffffff (белый). Далее для источник света задается позиция, и он помещается на сцену.

```
var light = new THREE.PointLight(0xffffff);
light.position.set(100,1000,-500);
scene.add(light);
```

2. Добавлен второй источник света, который еще будет отбрасывать тень.

```
var light1 = new THREE.PointLight(0xffffff);
light1.position.set(100,500,300);
light1.castShadow = true;
scene.add(light1);
```

Шаг 3. Наложение текстур.

1. Наложение текстур для шаров.

Сначала необходимо создать переменную для материала и текстуры.

```
var material;
var texture;
```

Далее с помощью метода loadTexture задается изображение текстуры, которое будет накладываться на объект. Примеры текстур на рисунках 1, 2, 3.

```
texture = new THREE.ImageUtils.loadTexture( 'images/whiteball.png' );
```

Рисунок 1 — Текстура для красного шара

Рисунок 2 — Текстура для шара 1

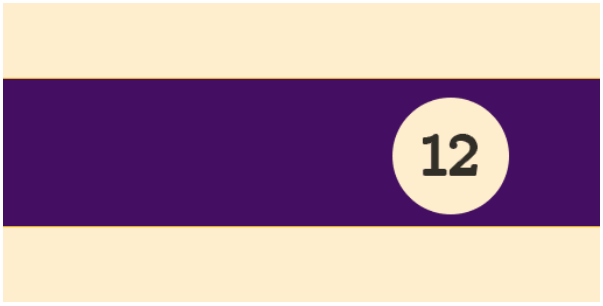

Рисунок 3 — Текстура для шара 12

С помощью метода `MeshLambertMaterial` задается материал для неблестящих поверхностей, без зеркальных бликов. Материал использует нефизическую Ламбертову модель для расчета коэффициента отражения. Она хорошо имитирует такие поверхности, как: необработанную древесину или камень, но не может имитировать блестящие поверхности с зеркальными бликами (например, лакированную древесину).

Затенение рассчитывается с использованием модели затенения Гуро. Оно вычисляется для каждой вершины (т. е. в шейдере вершин) и интерполирует результаты по граням полигона.

```
material = new THREE.MeshLambertMaterial( { map: texture } );
```

Далее материал накладывается на шар и помещается на сцену.

```
var whiteball = new THREE.Mesh( geometry, material );
whiteball.position.set(-300,0,264);
```

```
scene.add(whiteball);
```

Аналогичным образом текстуры накладываются на другие шары, кубик и кий.

2. Наложение текстуры для стен, стола и потолка.

Создается массив для объектов. Далее на каждый объект накладывается текстура, и каждый объект добавляется в массив. Затем они добавляются на сцену. Примеры текстур на рисунке 4.

```
var materialArray = [];
materialArray.push(new THREE.MeshBasicMaterial( { map:
THREE.ImageUtils.loadTexture( 'images/px.png' ) }));
materialArray.push(new THREE.MeshBasicMaterial( { map:
THREE.ImageUtils.loadTexture( 'images/nx.png' ) }));
materialArray.push(new THREE.MeshBasicMaterial( { map:
THREE.ImageUtils.loadTexture( 'images/py.png' ) }));
materialArray.push(new THREE.MeshBasicMaterial( { map:
THREE.ImageUtils.loadTexture( 'images/ny.png' ) }));
materialArray.push(new THREE.MeshBasicMaterial( { map:
THREE.ImageUtils.loadTexture( 'images/pz.png' ) }));
materialArray.push(new THREE.MeshBasicMaterial( { map:
THREE.ImageUtils.loadTexture( 'images/nz.png' ) }));
for (var i = 0; i < 6; i++)
   materialArray[i].side = THREE.BackSide;
```

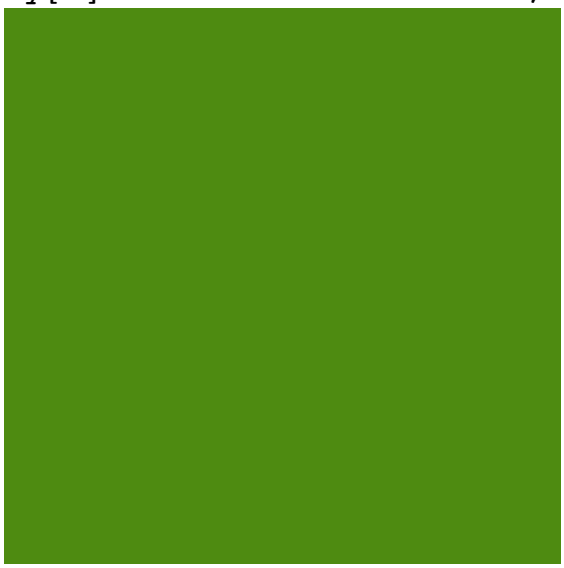

Рисунок 4 — Текстура для стола

Результат выполнения программы показан на рисунке 5.

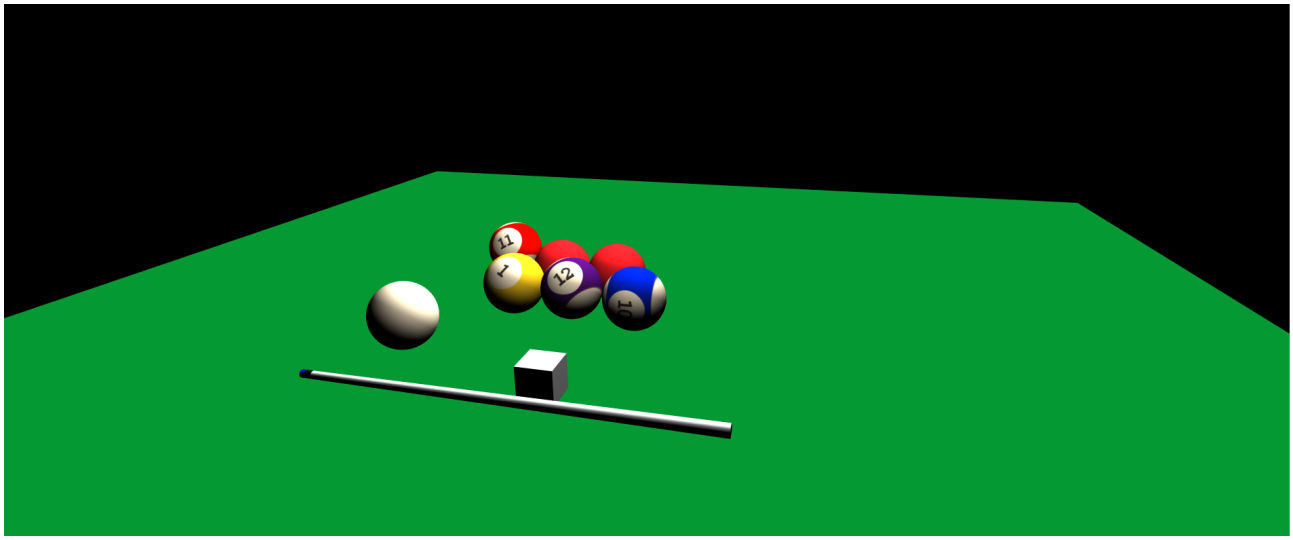

Рисунок 5 — Результат работы программы

Вывод

В ходе лабораторной работы было добавлено освещение объектов и на них наложены текстуры.

Приложение 1

Листинг программы

```
<!doctype html>
<html lang="en">
<head>
	<title>WebGL</title>
	<meta charset="utf-8">
	<meta name="viewport" content="width=device-width, user-scalable=no, minimum-scale=1.0, maximum-scale=1.0">
</head>

<body>
<script src="js/Three.js"></script>
<script src="js/Detector.js"></script>
<script src="js/OrbitControls.js"></script>
<script src="js/THREEx.KeyboardState.js"></script>
<script src="js/THREEx.WindowResize.js"></script>
<div id="ThreeJS" style="position: absolute; left:0px; top:0px"></div>

<script>

// variables
var container, scene, camera, renderer, controls;
var keyboard = new THREEx.KeyboardState();
var sphereCamera;

init();
animate();

// FUNCTIONS
function init()
{
	// SCENE
	scene = new THREE.Scene();

	// CAMERA
	var SCREEN_WIDTH = window.innerWidth, SCREEN_HEIGHT = window.innerHeight;
	var VIEW_ANGLE = 45, ASPECT = SCREEN_WIDTH / SCREEN_HEIGHT, NEAR = 0.1, FAR = 20000;
	camera = new THREE.PerspectiveCamera( VIEW_ANGLE, ASPECT, NEAR, FAR);
	scene.add(camera);
	camera.position.set(200, 400, 800);
	camera.rotation.x = Math.PI / 3;
	camera.lookAt(scene.position);
```

```
	// RENDERER
	if ( Detector.webgl )
		renderer = new THREE.WebGLRenderer( {antialias:true} );
	else
		renderer = new THREE.CanvasRenderer();
	renderer.setSize(SCREEN_WIDTH, SCREEN_HEIGHT);
	container = document.getElementById( 'ThreeJS' );
	container.appendChild( renderer.domElement );

	// EVENTS
	THREEx.WindowResize(renderer, camera);

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

	// LIGHT
	var light = new THREE.PointLight(0xffffff);
	light.position.set(100,1000,-500);
	scene.add(light);
	var light1 = new THREE.PointLight(0xffffff);
	light1.position.set(100,500,300);
	light1.castShadow = true;
	scene.add(light1);

	// FLOOR
	var floorGeometry = new THREE.PlaneGeometry(2000,2000);
	var floorMaterial = new THREE.MeshBasicMaterial({color:
0x009933, side: THREE.DoubleSide});
	var floor = new THREE.Mesh(floorGeometry, floorMaterial);
	//floor.position.y = -0.5;
	floor.rotation.x = Math.PI / 2;
	floor.position.set(0,-50,0);
	floor.receiveShadow = true;
	scene.add(floor);

	// SKY/FOG
	var materialArray = [];
	materialArray.push(new THREE.MeshBasicMaterial( { map:
THREE.ImageUtils.loadTexture( 'images/px.png' ) }));
	materialArray.push(new THREE.MeshBasicMaterial( { map:
THREE.ImageUtils.loadTexture( 'images/nx.png' ) }));
	materialArray.push(new THREE.MeshBasicMaterial( { map:
THREE.ImageUtils.loadTexture( 'images/py.png' ) }));
	materialArray.push(new THREE.MeshBasicMaterial( { map:
THREE.ImageUtils.loadTexture( 'images/ny.png' ) }));
	materialArray.push(new THREE.MeshBasicMaterial( { map:
THREE.ImageUtils.loadTexture( 'images/pz.png' ) }));
	materialArray.push(new THREE.MeshBasicMaterial( { map:
THREE.ImageUtils.loadTexture( 'images/nz.png' ) }));
```

```
    for (var i = 0; i < 6; i++)
       materialArray[i].side = THREE.BackSide;
    var skyboxMaterial = new
THREE.MeshFaceMaterial( materialArray );
    var skyboxGeom = new THREE.CubeGeometry( 2000, 2000, 2000 );
    var skybox = new THREE.Mesh( skyboxGeom, skyboxMaterial );
    scene.add( skybox );

    var material;
    var sphereGeom =  new THREE.SphereGeometry( 50, 32, 16 ); //
radius, segmentsWidth, segmentsHeight

    sphereCamera = new THREE.CubeCamera( 0.1, 5000, 512 );
    sphereCamera.renderTarget.minFilter =
THREE.LinearMipMapLinearFilter;
    scene.add( sphereCamera );

    geometry = new THREE.SphereGeometry( 50, 64, 30  );
    texture = new
THREE.ImageUtils.loadTexture( 'images/whiteball.png' );
    material = new THREE.MeshLambertMaterial( { map: texture } );
    var whiteball = new THREE.Mesh( geometry, material );
    whiteball.metallic = true;
    whiteball.position.set(-300,0,264);
    scene.add(whiteball);

    var redball1;
    texture = new
THREE.ImageUtils.loadTexture( 'images/redball.png' );
    material = new THREE.MeshLambertMaterial( { map: texture } );
    redball1 = new THREE.Mesh( geometry, material );
    redball1.position.set(-150,0,10);
    scene.add(redball1);

    var redball2;
    redball2 = new THREE.Mesh( geometry, material );
    redball2.position.set(-50,0,10);
    scene.add(redball2);

    texture = new
THREE.ImageUtils.loadTexture( 'images/1ball.png' );
    material = new THREE.MeshLambertMaterial( { map: texture } );
    var ball1 = new THREE.Mesh( geometry, material );
    ball1.position.set(-200,0,100);
    ball1.rotation.x = -Math.PI/4;
    ball1.rotation.y = Math.PI;
    ball1.rotation.z = Math.PI/6;
    ball1.castShadow = true;
    ball1.receiveShadow = true;
    scene.add(ball1);
```

```
    texture = new
THREE.ImageUtils.loadTexture( 'images/12ball.png' );
    material = new THREE.MeshLambertMaterial( { map: texture } );
    var ball12 = new THREE.Mesh( geometry, material );
    ball12.position.set(-100,0,100);
    ball12.rotation.x = -Math.PI/4;
    ball12.rotation.y = Math.PI;
    ball12.rotation.z = -Math.PI/4;
    ball12.castShadow = true;
    ball12.receiveShadow = true;
    scene.add(ball12);

    texture = new
THREE.ImageUtils.loadTexture( 'images/10ball.png' );
    material = new THREE.MeshLambertMaterial( { map: texture } );
    var ball10 = new THREE.Mesh( geometry, material );
    ball10.position.set(10,0,120);
    ball10.rotation.y = Math.PI;
    ball10.rotation.x = 0.0;
    ball10.rotation.z = Math.PI/2;
    ball10.castShadow = true;
    ball10.receiveShadow = true;
    scene.add(ball10);

    texture = new
THREE.ImageUtils.loadTexture( 'images/11ball.png' );
    material = new THREE.MeshLambertMaterial( { map: texture } );
    ;
    var ball11 = new THREE.Mesh( geometry, material );
    ball11.position.set(-280,0,-80);
    ball11.rotation.x = -Math.PI/4+Math.PI/16;
    ball11.rotation.y = Math.PI;
    ball11.rotation.z = -Math.PI/6;
    ball11.castShadow = true;
    ball11.receiveShadow = true;
    scene.add(ball11);

    var cubeGeometry = new THREE.CubeGeometry(50,50,50,50);
                                texture = new
THREE.TextureLoader().load( 'images/grayball.png' );
    material = new THREE.MeshLambertMaterial( { map: texture } );
    var cube = new THREE.Mesh( cubeGeometry, material );
    cube.position.set(-50,-24,350)
    scene.add(cube);

    var cylindergeometry = new THREE.CylinderGeometry(5, 8, 500,
512, false);
    var cylindermaterial = new THREE.MeshLambertMaterial({color:
0xffffff});
```

```
	cue = new THREE.Mesh(cylindergeometry, cylindermaterial);
	cue.rotation.x = Math.PI / 2;
	cue.rotation.z = Math.PI / 2;
	cue.position.set(-50,0,450);
	scene.add(cue);

	cylindergeometry = new THREE.CylinderGeometry(5, 5, 10, 512,
false);
	cylindermaterial = new THREE.MeshLambertMaterial({color:
0x000000});
	cue_top1 = new THREE.Mesh(cylindergeometry,
cylindermaterial);
	cue_top1.rotation.x = Math.PI / 2;
	cue_top1.rotation.z = Math.PI / 2;
	cue_top1.position.set(-305,0,450);
	scene.add(cue_top1);

	cylindergeometry = new THREE.CylinderGeometry(5, 5, 6, 512,
false);
	cylindermaterial = new THREE.MeshLambertMaterial({color:
0x0000ff});
	cue_top2 = new THREE.Mesh(cylindergeometry,
cylindermaterial);
	cue_top2.rotation.x = Math.PI / 2;
	cue_top2.rotation.z = Math.PI / 2;
	cue_top2.position.set(-312,0,450);
	scene.add(cue_top2);
}

function animate()
{
	requestAnimationFrame( animate );
	render();
	update();
}

function update()
{
	controls.update();
}

function render()
{
	sphereCamera.updateCubeMap(renderer, scene);
	renderer.shadowMapEnabled = true;
	renderer.render(scene, camera);
}
</script>
</body>
</html>
```