|  |
| --- |
| Университет итмо |
| Лабораторная работа №1 по дисциплине «Компьютерная графика» |
| WebGL |
| Группа Р3402 |
| **Выполнила: Орлова Кристина Александровна** |
| **Преподаватель: Королёва Юлия Александровна** |

|  |
| --- |
| *13.12.19* |

Задание

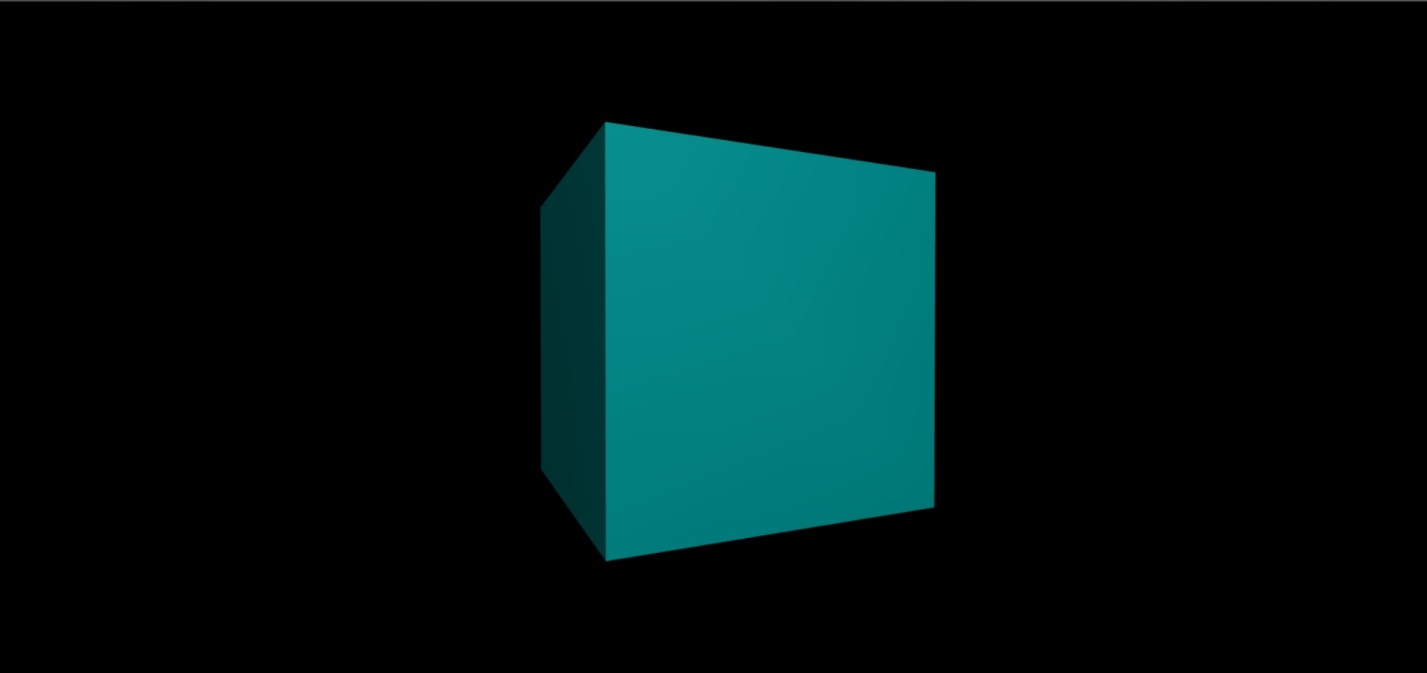
В рамках лабораторной работы было принято решение реализовать простейший вывод на экран куба с источником света, который бы его освещал, и материалами, которые бы отражали этот свет.

Данная проблема была выполнена на WebGL и Three.js. Так же на Three.js в качестве отдельного задания была выгружена 3d-модель в формате, поддерживаемым Three.js, gltf.

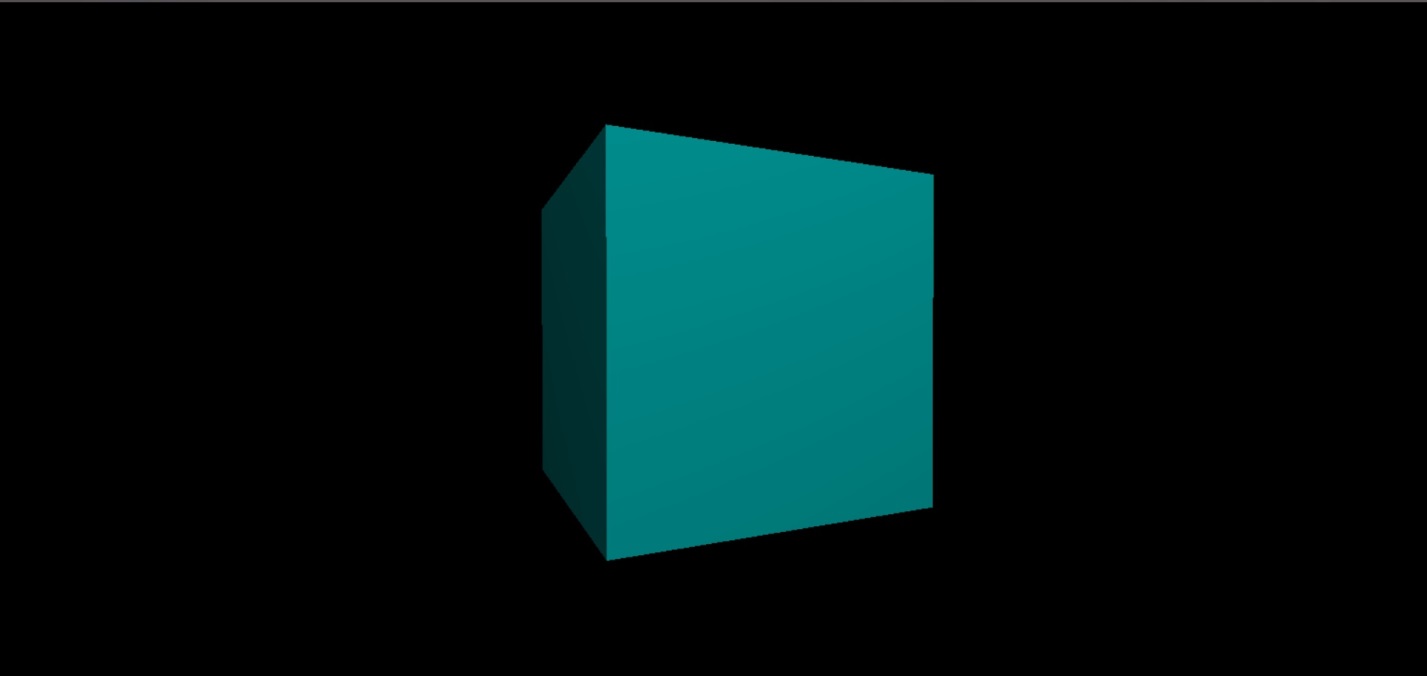
Three.js



WebGL vs Three.js



**WebGL**



**Three.js**

Опытным путем было установлено, что 1 строчка на Three.js заменяет 6 строчек кода на WebGL.

Рисовка куба

**Three.js**

var geometry = new THREE.CubeGeometry(200, 200, 200);

**WebGL**

const points = [

|  |
| --- |
| // передняя сторона |
| -0.5, -0.5, 0.5, // 0 |
| -0.5, 0.5, 0.5, // 1 |
| 0.5, 0.5, 0.5, // 2 |
| 0.5, -0.5, 0.5, // 3 |
| // задняя сторона |
| -0.5, -0.5, -0.5, // 4 |
| -0.5, 0.5, -0.5, // 5 |
| 0.5, 0.5, -0.5, // 6 |
| 0.5, -0.5, -0.5, // 7 |
| // левый бок |
| -0.5, -0.5, 0.5, // 0 |
| -0.5, 0.5, 0.5, // 1 |
| -0.5, 0.5, -0.5, // 5 |
| -0.5, -0.5, -0.5, // 4 |
| // правый бок |
| 0.5, -0.5, 0.5, // 3 |
| 0.5, 0.5, 0.5, // 2 |
| 0.5, 0.5, -0.5, // 6 |
| 0.5, -0.5, -0.5, // 7 |
| // верх |
| 0.5, 0.5, 0.5, // 2 |
| -0.5, 0.5, 0.5, // 1 |
| -0.5, 0.5, -0.5, // 5 |
| 0.5, 0.5, -0.5, // 6 |
| // низ |
| 0.5, -0.5, 0.5, // 3 |
| -0.5, -0.5, 0.5, // 0 |
| -0.5, -0.5, -0.5, // 4 |
| 0.5, -0.5, -0.5 // 7 |
| ]; |

|  |
| --- |
| const indexes = [ |
| // передняя сторона |
| 0, 1, 2, |
| 2, 3, 0, |
| // задняя сторона |
| 4, 5, 6, |
| 6, 7, 4, |
| // левый бок |
| 8, 9, 10, |
| 10, 11, 8, |
| // правый бок |
| 12, 13, 14, |
| 14, 15, 12, |
| // верх |
| 16, 17, 18, |
| 18, 19, 16, |
| // низ |
| 20, 21, 22, |
| 22, 23, 20 |
| ]; |

|  |
| --- |
| cubeVertexBuffer = context.createBuffer(); |
| cubeIndexBuffer = context.createBuffer(); |
|  |
|  |
| cubeVertexBuffer.itemSize = 3; // число координат на вершину (x, y, z) |
|  |
| cubeIndexBuffer.numberOfItems = indexes.length; |
|  |
|  |
| context.bindBuffer(context.ARRAY\_BUFFER, cubeVertexBuffer); |
| context.bufferData(context.ARRAY\_BUFFER, new Float32Array(points), context.STATIC\_DRAW); |
|  |
| context.bindBuffer(context.ELEMENT\_ARRAY\_BUFFER, cubeIndexBuffer); |
| context.bufferData(context.ELEMENT\_ARRAY\_BUFFER, new Uint16Array(indexes), context.STATIC\_DRAW); |
|  |

|  |
| --- |
| context.bindBuffer(context.ARRAY\_BUFFER, cubeVertexBuffer); |
| context.vertexAttribPointer(shaderProgram.vertexPositionAttribute, cubeVertexBuffer.itemSize, context.FLOAT, false, 0, 0); |
|  |
|  |
|  |
|  |
| context.drawElements(context.TRIANGLES, cubeIndexBuffer.numberOfItems, context.UNSIGNED\_SHORT, 0); // отрисовка примитивов |

Назначение материала кубу

**Three.js**

|  |
| --- |
| var material = new THREE.MeshPhongMaterial({ |
| color: 0x00ffff, |
| specular: 0x00ffff, |
| shininess: 0 |
| }); |
| var cube = new THREE.Mesh(geometry, material); |

**WebGL**

|  |
| --- |
| context.uniform3fv(shaderProgram.uniformAmbientMaterialColor, [1.0, 1.0, 1.0]); |
| context.uniform3fv(shaderProgram.uniformDiffuseMaterialColor, [0.0, 1.0, 1.0]); |
| context.uniform3fv(shaderProgram.uniformSpecularMaterialColor, [1.0, 1.0, 1.0]); |

|  |
| --- |
| const normals = [ |
| // передняя сторона |
| 0.0, 0.0, 1.0, |
| 0.0, 0.0, 1.0, |
| 0.0, 0.0, 1.0, |
| 0.0, 0.0, 1.0, |
|  |
| // задняя сторона |
| 0.0, 0.0, -1.0, |
| 0.0, 0.0, -1.0, |
| 0.0, 0.0, -1.0, |
| 0.0, 0.0, -1.0, |
|  |
| // левый бок |
| -1.0, 0.0, 0.0, |
| -1.0, 0.0, 0.0, |
| -1.0, 0.0, 0.0, |
| -1.0, 0.0, 0.0, |
|  |
| // правый бок |
| 1.0, 0.0, 0.0, |
| 1.0, 0.0, 0.0, |
| 1.0, 0.0, 0.0, |
| 1.0, 0.0, 0.0, |
|  |
| // верх |
| 0.0, 1.0, 0.0, |
| 0.0, 1.0, 0.0, |
| 0.0, 1.0, 0.0, |
| 0.0, 1.0, 0.0, |
|  |
| // низ |
| 0.0, -1.0, 0.0, |
| 0.0, -1.0, 0.0, |
| 0.0, -1.0, 0.0, |
| 0.0, -1.0, 0.0 |
| ]; |
|  |
| cubeVertexNormalBuffer = context.createBuffer(); |
|  |
| cubeVertexNormalBuffer.itemSize = 3; |
|  |
|  |
| context.bindBuffer(context.ARRAY\_BUFFER, cubeVertexNormalBuffer); |
| context.bufferData(context.ARRAY\_BUFFER, new Float32Array(normals), context.STATIC\_DRAW); |

|  |
| --- |
|  |
| context.bindBuffer(context.ARRAY\_BUFFER, cubeVertexNormalBuffer); |
| context.vertexAttribPointer(shaderProgram.vertexNormalAttribute, cubeVertexNormalBuffer.itemSize, context.FLOAT, false, 0, 0); |
|  |
| context.enable(context.DEPTH\_TEST); |
|  |

Установка точечного источника света

**Three.js**

var pointLight = new THREE.PointLight(0xffffff, 0.8);

**WebGL**

|  |
| --- |
| context.uniform3fv(shaderProgram.uniformLightPosition, [0.0, 5.0, 5.0]); |
| context.uniform3fv(shaderProgram.uniformAmbientLightColor, [0.0, 0.0, 0.0]); |
| context.uniform3fv(shaderProgram.uniformDiffuseLightColor, [0.7, 0.7, 0.7]); |
| context.uniform3fv(shaderProgram.uniformSpecularLightColor, [1.0, 1.0, 1.0]); |