ГУАП

КАФЕДРА № 42

ОТЧЕТ   
ЗАЩИЩЕН С ОЦЕНКОЙ

ПРЕПОДАВАТЕЛЬ

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| доцент |  |  |  | А. В. Аграновский |
| должность, уч. степень, звание |  | подпись, дата |  | инициалы, фамилия |

|  |
| --- |
| ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ №8 |
| Вариант №10 |
| по курсу: Компьютерная Графика |
|  |
|  |

РАБОТУ ВЫПОЛНИЛ

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| СТУДЕНТ гр. № | 4321 |  |  |  | К. А. Лебедев |
|  |  |  | подпись, дата |  | инициалы, фамилия |

Санкт-Петербург 2024

**Цель работы**

Изучение открытой графической библиотеки OpenGL; построить динамическую 3D-сцену на языке программирования высокого

уровня, поддерживающего библиотеку OpenGL.

**Задание к лабораторной работе**

Используя язык программирования и библиотеку OpenGL, разработать динамическую 3D-сцену.

**Теоретические положения, используемые при выполнении лабораторной работы.**

OpenGL (Open Graphics Library) — это кроссплатформенная спецификация, предоставляющая инструменты для создания приложений с двух- и трехмерной графикой. Основной принцип работы OpenGL заключается в обработке геометрических примитивов (точек, линий, треугольников и т.д.) и их преобразовании в растровое изображение. OpenGL предоставляет интерфейс для работы с графическим процессором (GPU), что позволяет эффективно выполнять вычисления для построения графики.

WebGL — **это технология, которая позволяет создавать и отображать трёхмерную (3D) графику прямо в веб-браузере.**

Three.js — кроссбраузерная библиотека JavaScript, используемая для создания и отображения анимированной компьютерной 3D графики при разработке веб-приложений.

Three.js позволяет создавать ускоренную на GPU 3D графику, используя язык JavaScript как часть сайта без подключения проприетарных плагинов для браузера. Это возможно благодаря использованию технологии WebGL.

**Основные концепции OpenGL:**

1. **Геометрические примитивы:**точки, линии, треугольники и многоугольники, которые являются основными элементами для построения сцен.
2. **Буферы:** OpenGL использует буферы цвета и глубины для хранения информации об отображении сцены. Буфер глубины помогает правильно отображать объекты, находящиеся на разной удаленности от камеры.
3. **Матрицы:** для управления положением и ориентацией объектов используются матрицы моделирования, проекции и вида.
4. **Освещение:** OpenGL поддерживает различные модели освещения, включая источники света, материалы и методы затенения.
5. **Двойная буферизация:** используется для предотвращения мерцания изображения при его обновлении. Один буфер используется для отображения, другой — для отрисовки текущей сцены.

**Листинг с кодом программы.**

const scene = new THREE.Scene();

const camera = new THREE.PerspectiveCamera(75, window.innerWidth / window.innerHeight, 0.1, 1000);

camera.position.z = 10;

const renderer = new THREE.WebGLRenderer();

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

document.body.appendChild(renderer.domElement);

const sphereGeometry = new THREE.SphereGeometry(1, 32, 32);

const sphereMaterial = new THREE.MeshBasicMaterial({ color: 0x0077ff, wireframe: false });

const sphere = new THREE.Mesh(sphereGeometry, sphereMaterial);

scene.add(sphere);

const triangleGeometry = new THREE.BufferGeometry();

const vertices = new Float32Array([

0, 1, 0,

-1, -1, 0,

1, -1, 0

]);

triangleGeometry.setAttribute('position', new THREE.BufferAttribute(vertices, 3));

const triangleMaterial = new THREE.MeshBasicMaterial({ color: 0xff7700, side: THREE.DoubleSide });

const triangle = new THREE.Mesh(triangleGeometry, triangleMaterial);

scene.add(triangle);

let angle = 0;

const radius = 5;

*// Функция анимации*

function animate() {

requestAnimationFrame(animate);

sphere.rotation.y += 0.01;

angle += 0.01;

sphere.position.x = radius \* Math.cos(angle);

sphere.position.z = radius \* Math.sin(angle);

triangle.rotation.z += 0.01;

renderer.render(scene, camera);

}

animate();

window.addEventListener('resize', () => {

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

camera.updateProjectionMatrix();

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

});

**Экранные формы с результатами работы программы.**

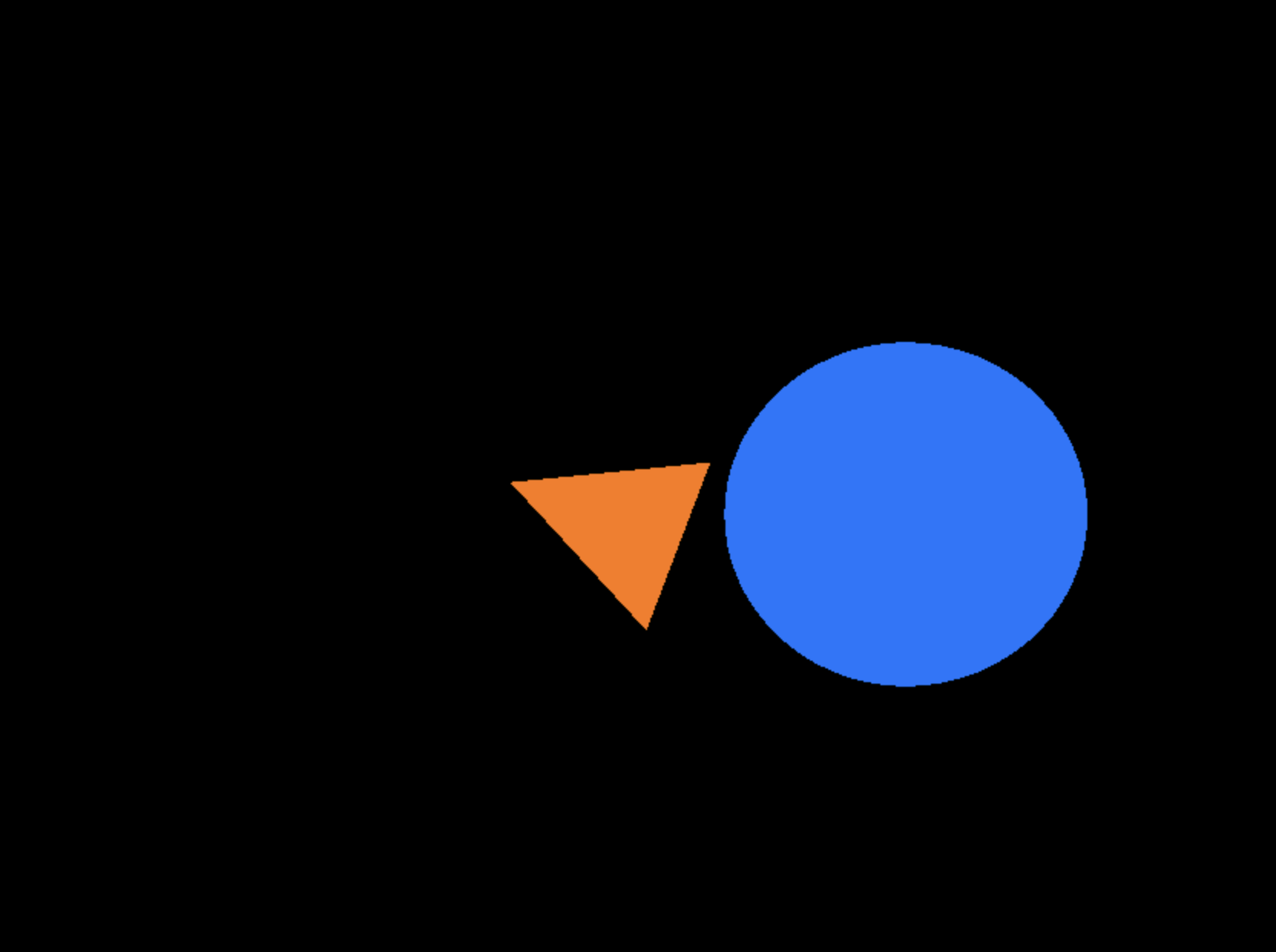


Рисунок 1 – Результат программы

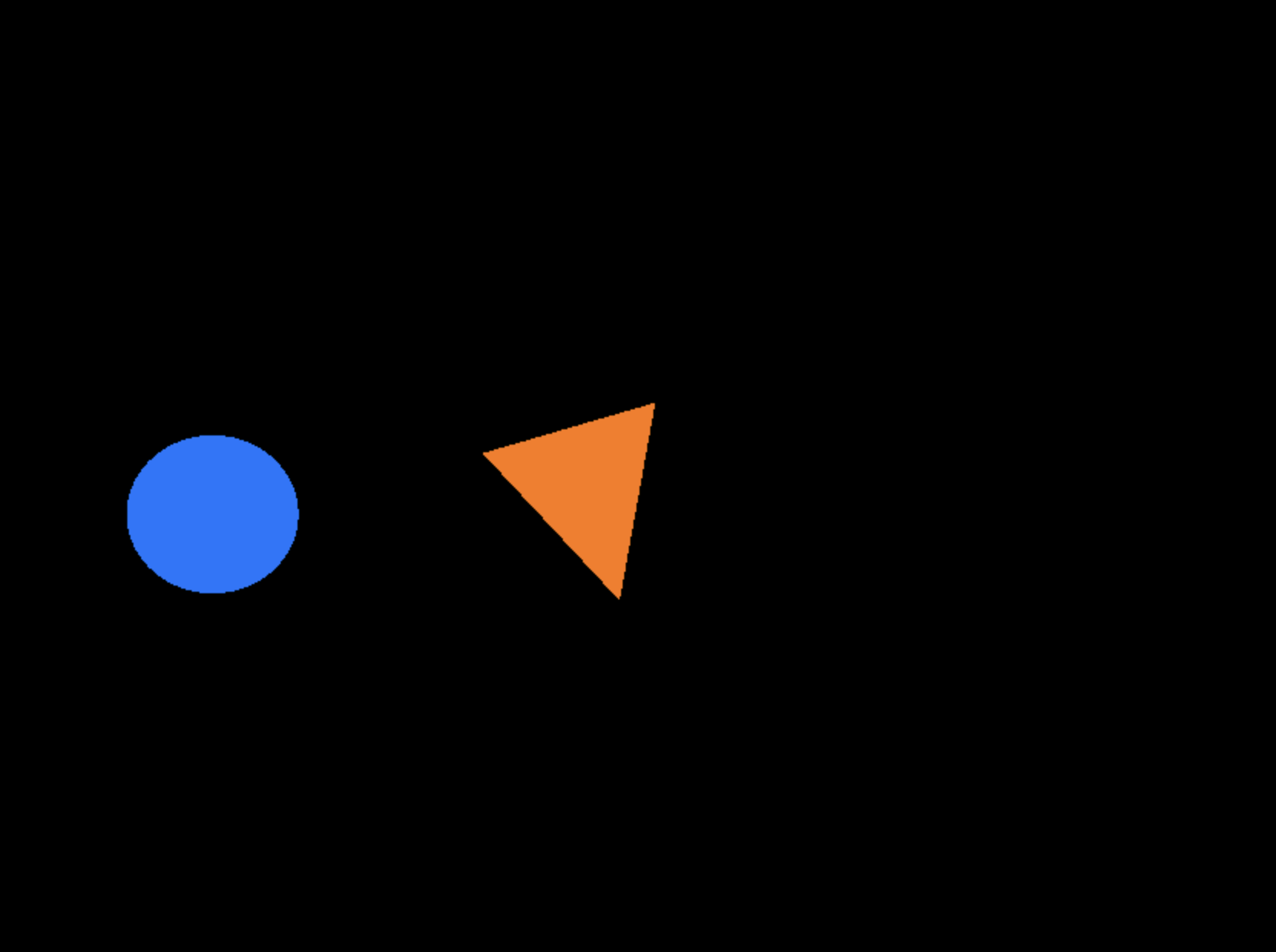


Рисунок 2 – Результат программы

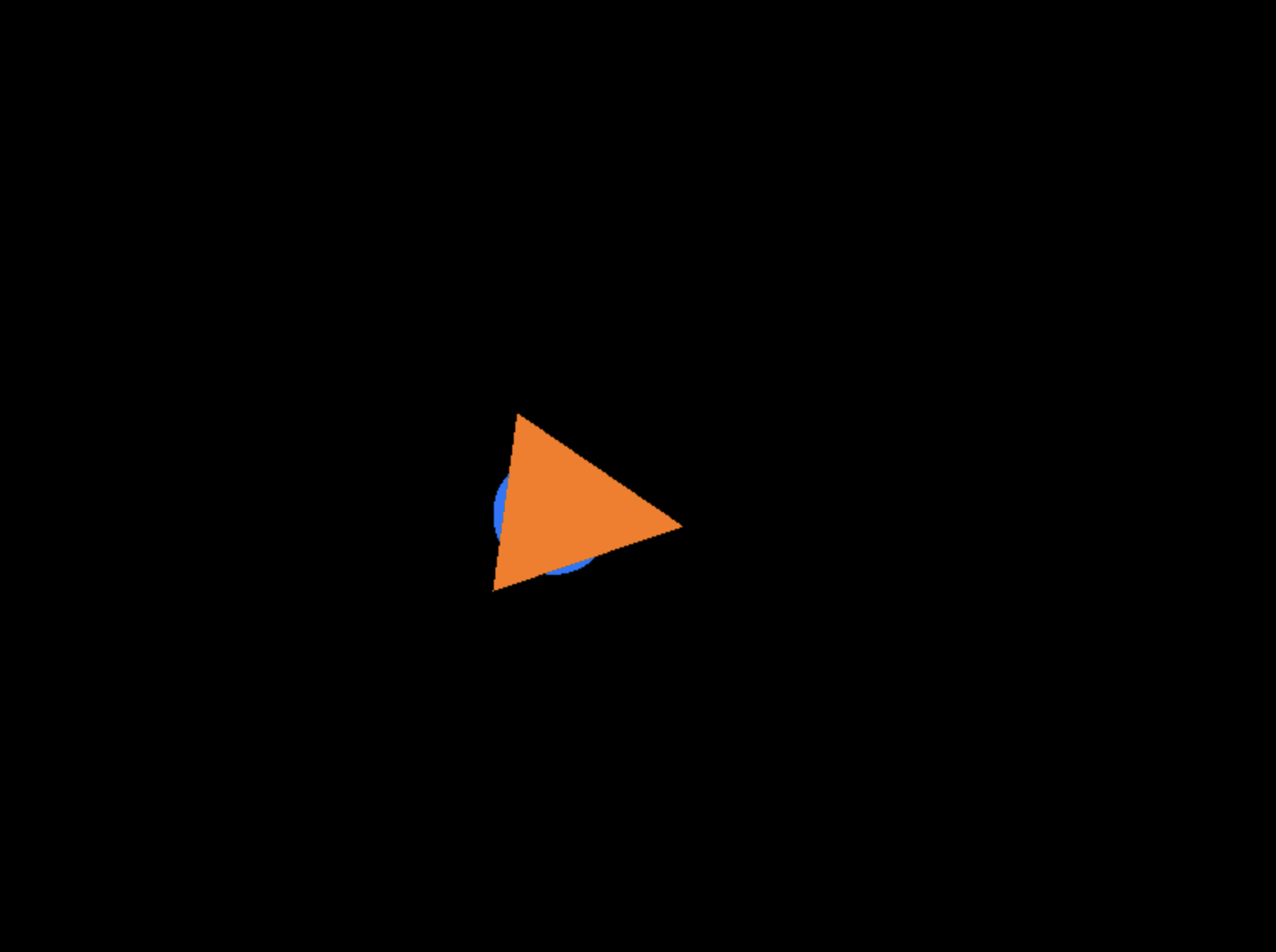


Рисунок 3 – Результат программы

**Выводы по лабораторной работе**

В ходе выполнения лабораторной работы была создана динамическая 3D-сцена с использованием языка JavaScript и библиотеки Three.js. В сцене были реализованы геометрические примитивы, такие как сфера и треугольник, которые анимируются и взаимодействуют друг с другом.

Благодаря использованию Three.js удалось добиться плавной визуализации и анимации объектов, что позволяет наблюдать за движением сферы по круговой траектории и вращением треугольника. Сцена отображается корректно и без мерцания, что обеспечивается использованием встроенных механизмов рендеринга библиотеки.

Также была реализована адаптивность сцены к изменениям размеров окна браузера, что позволяет пользователю комфортно взаимодействовать с 3D-графикой на различных устройствах. В результате лабораторной работы были получены практические навыки работы с 3D-графикой в веб-приложениях, а также понимание основных принципов работы с библиотекой Three.js.