**Московский авиационный институт**

**(Национальный исследовательский университет)**

Институт: «Информационные технологии и прикладная математика»

Кафедра: 806 «Вычислительная математика и программирование»

Дисциплина: «Компьютерная графика»

**Лабораторная работа № 6**

Тема: Создание шейдерных анимационных эффектов в OpenGL

Студент: Алексеев Владислав Евгеньевич

Группа: М8О-306Б-19

Преподаватель: Чернышов Л.Н.

Дата:

Оценка:

Москва, 2021

1. **Постановка задачи**

Для поверхности, созданной в л.р. №4-5, обеспечить выполнение следующего шейдерного эффекта - Анимация. Цветовые координаты изменяются по синусоидальному закону.

1. **Описание программы**

Программа состоит из окна для взаимодействия с многогранником, из canvas для визуализации, вращения многогранника и изменения некоторых его параметров.

1. **Набор тестов**

1. Изначальный вид.

2. Вращение многогранника.

3. Увеличение точности аппроксимации.

4. Увеличение интенсивности света.

5. Увеличение масштаба многогранника.

6. Уменьшение масштаба многогранника.

1. **Результаты выполнения тестов**

1. Изначальный вид.

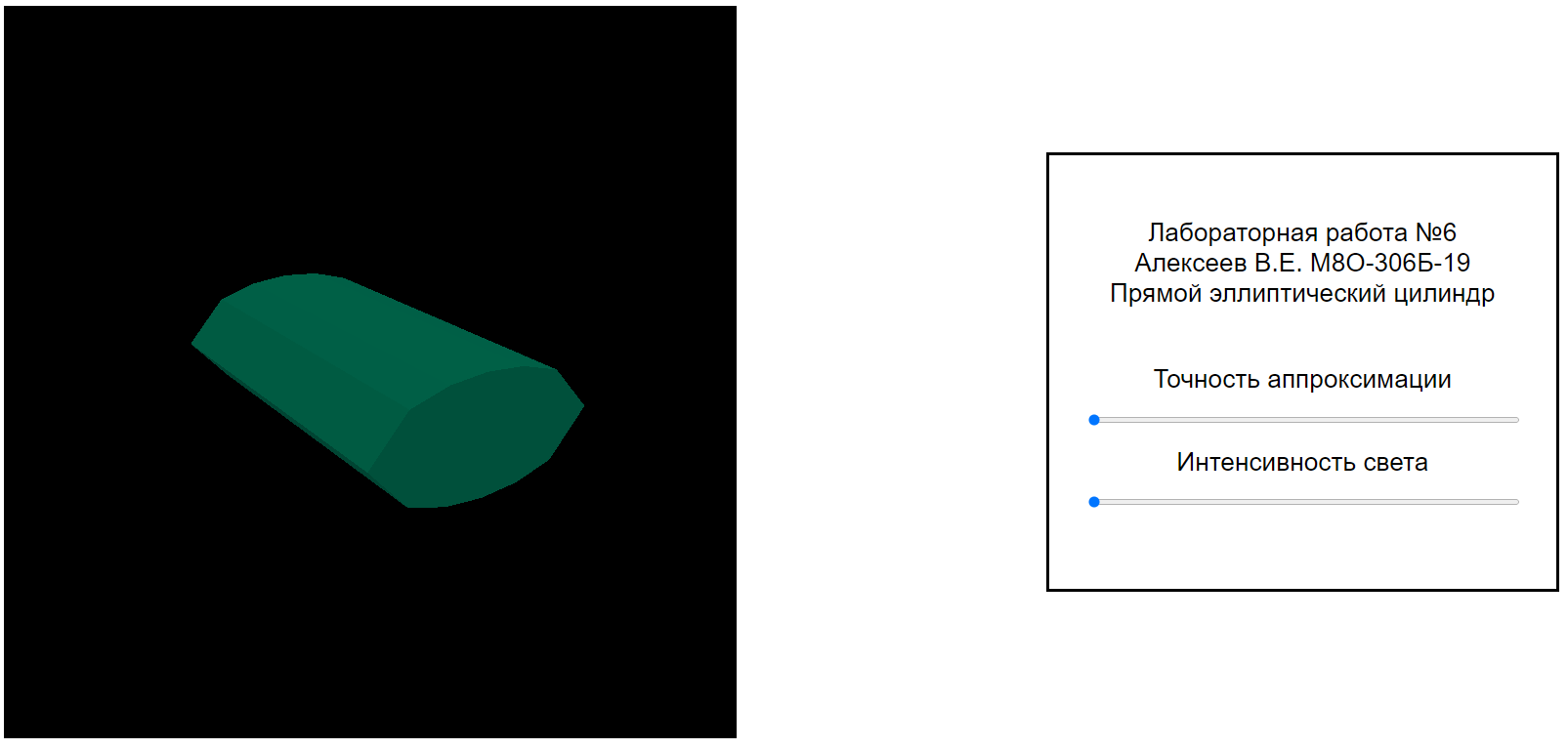


Рис. 4.1 Результат открытия окна

2. Вращение многогранника.

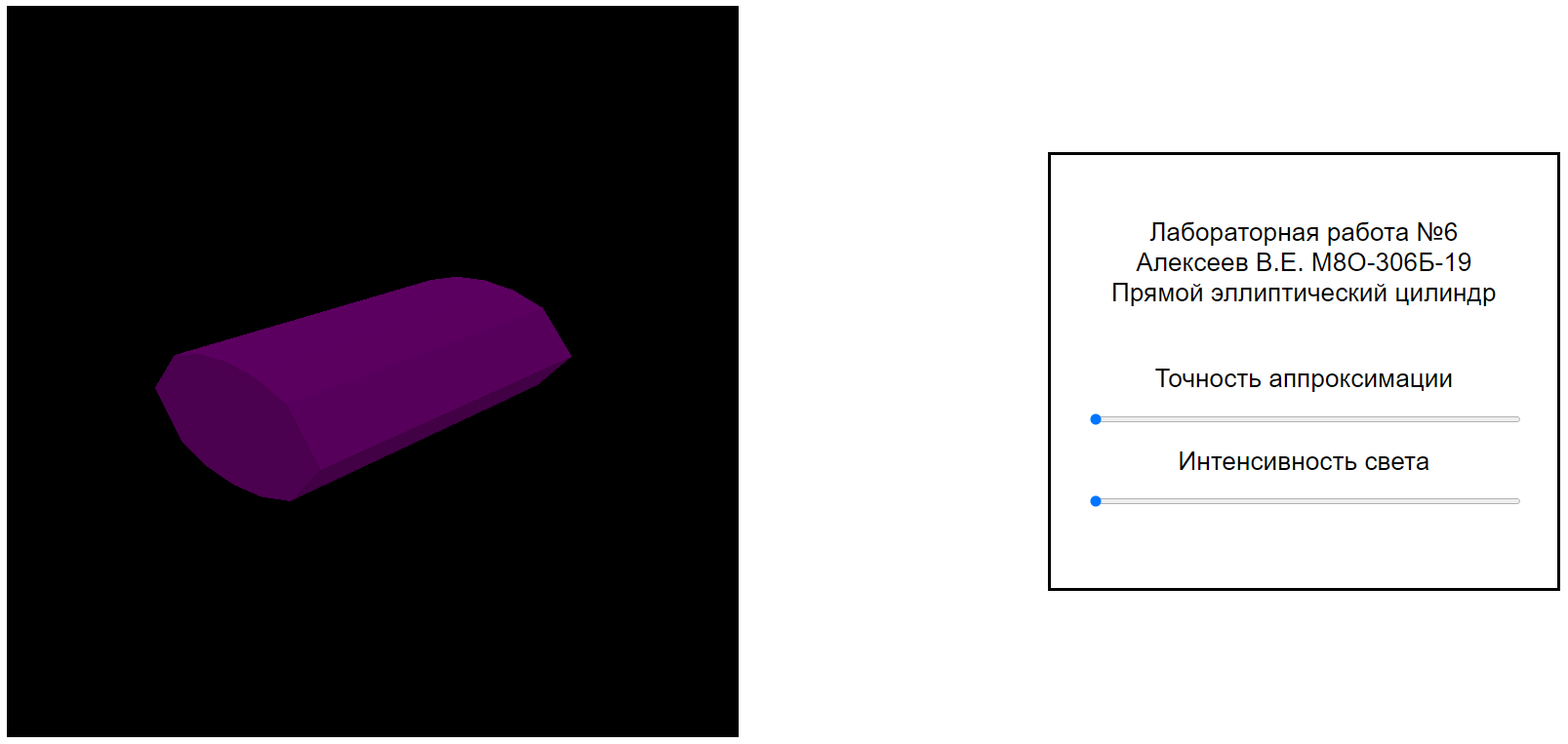


Рис. 4.2 Результат вращения многогранника

3. Увеличение точности аппроксимации.

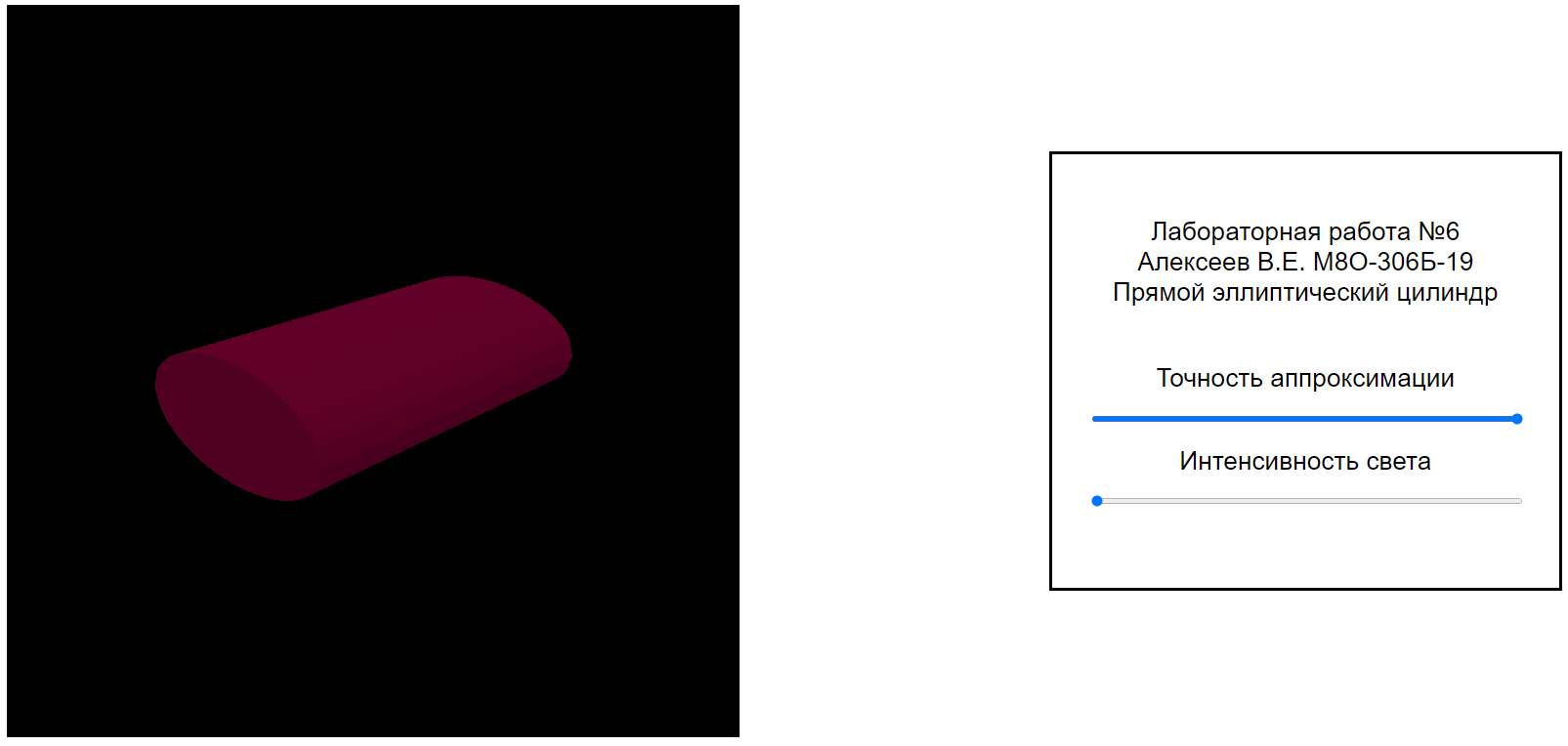


Рис. 4.3 Результат увеличение точности аппроксимации

4. Увеличение интенсивности света.

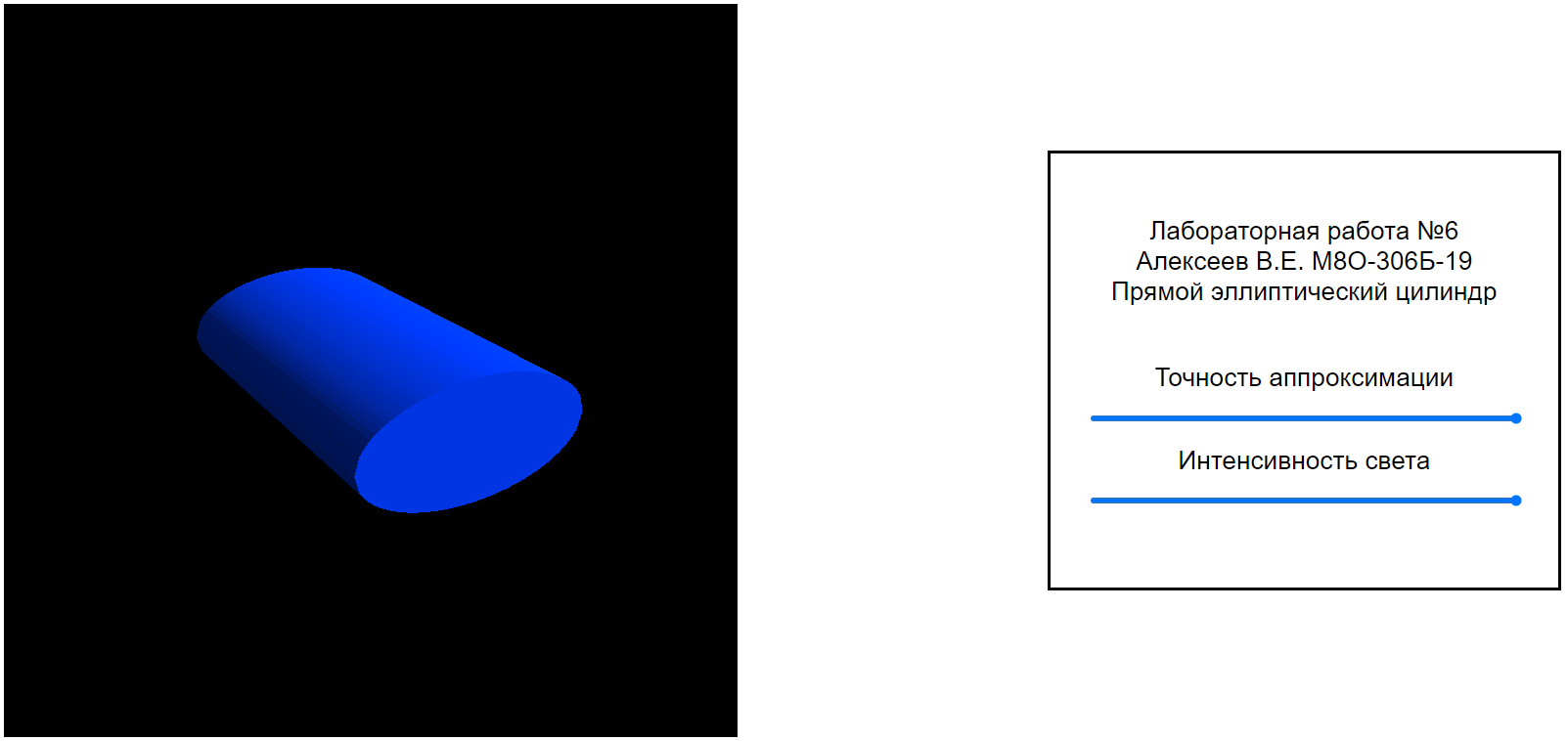


Рис. 4.4 Результат увеличения интенсивности света

5. Увеличение масштаба многогранника.

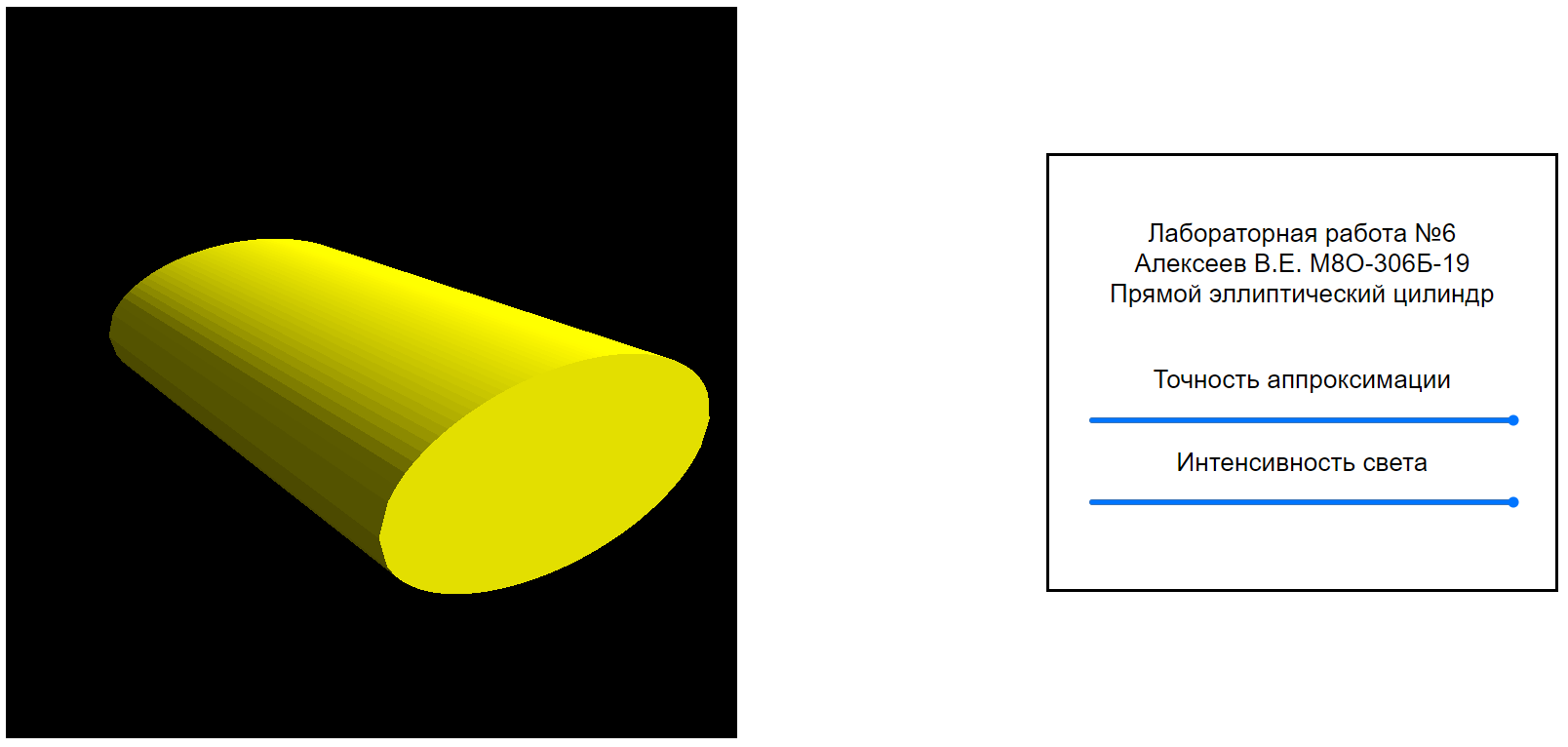


Рис. 4.5 Результат увеличения масштаба многогранника

6. Уменьшение масштаба многогранника.

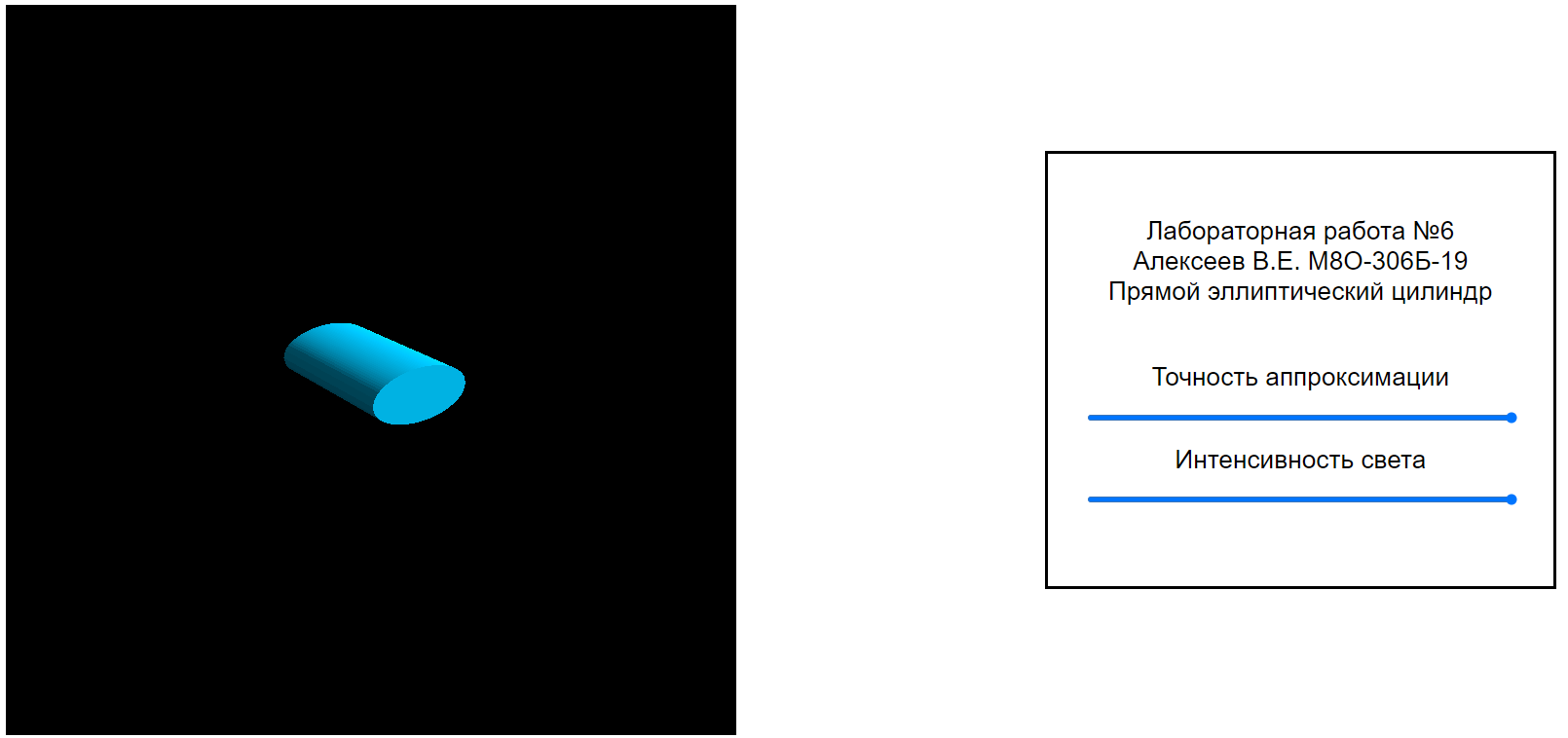


Рис. 4.6 Результат уменьшения масштаба многогранника

1. **Листинг программы**

В программе использовались готовые библиотеки **Three.js** и **OrbitControls.js**

**main.html**

<!-- Алексеев В.Е. М8О-306Б-19

Создание шейдерных анимационных эффектов в OpenGL 2.1

Вариант многогранника: Прямой эллиптический цилиндр. -->

<!DOCTYPE html>

<html>

<head>

<meta charset = "utf-8">

<link rel = 'stylesheet' href = 'style.css'>

<title>Прямой эллиптический цилиндр</title>

</head>

<body onLoad = "onLoad();">

<div id = "container" style = "width:1000px; height:1000px"></div>

<div class = 'controls'>

<div class = 'note'>

<p> Лабораторная работа №6 <br>

Алексеев В.Е. М8О-306Б-19 <br>

Прямой эллиптический цилиндр <br> <br>

</p>

</div>

<label for = "approx">Точность аппроксимации</label>

<div class = 'range'>

<input type = "range" id = "approx" name = "volume"

min = "6" max = "50" step = "1" value = "6">

</div>

<label for = "intens">Интенсивность света</label>

<div class = 'range'>

<input type = "range" id = "intens" name = "volume"

min = "0" max = "1" step = "0.01" value = "0">

</div>

</div>

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

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

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

<script type = "main.js"></script>

</body>

</html>

**style.css**

body {

background-color: white;

padding-top: 50px;

display: flex;

align-items: center;

justify-content: space-around;

}

.controls {

color: black;

font-family: sans-serif;

font-size: 35px;

border: 5px solid black;

width: 700px;

height: 600px;

padding: 50px;

box-sizing: border-box;

text-align: center;

align-items: center;

}

.note {

margin-bottom: 30px;

}

.range {

margin: 10px 0 20px;

}

.range input {

width: 100%;

}

**Cylinder.js**

var camera, scene, renderer, controls, geom, mesh, light;

var delta = 0, colorFlag = true;

function init() {

// Возьмем наш контейнер

var container = document.getElementById("container");

const approx = document.getElementById("approx");

const intens = document.getElementById("intens");

// Создадим рендерер Three.js и добавим его в наш контейнер

renderer = new THREE.WebGLRenderer();

renderer.setPixelRatio(window.devicePixelRatio);

renderer.setSize(container.offsetWidth, container.offsetHeight);

container.appendChild(renderer.domElement);

// Создадим сцену Three.js

scene = new THREE.Scene();

// Создадим камеру и добавим ее в сцену

camera = new THREE.PerspectiveCamera(45, container.offsetWidth / container.offsetHeight, 1, 500);

camera.position.set(0, 0, 150);

scene.add(camera);

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

// При включении перемещения или автоповорота требуется цикл анимации

controls.enableDamping = true;

controls.dampingFactor = 0.25;

controls.screenSpacePanning = false;

controls.minDistance = 100;

controls.maxDistance = 400;

scene.add(new THREE.HemisphereLight(0x606060, 0x404040));

light = new THREE.DirectionalLight(0xffffff);

light.intensity = intens.value;

light.reflectivity = intens.value;

light.position.set(600, 600, 600).normalize();

scene.add(light);

geom = initGeom();

var color = new THREE.Color("rgb(0, 255, 255)");

// hsl - Hue (оттенок), Saturate (насыщенность) и Lightness (светлота или яркость)

// var color = new THREE.Color("hsl(0, 100%, 50%)");

var material = new THREE.MeshLambertMaterial({ color: color });

mesh = new THREE.Mesh(geom, material);

scene.add(mesh);

}

const A = 20, B = 10, H = 60;

let num = approx.value;

approx.onchange = function (e) {

num = approx.value;

scene.remove(mesh);

geom = initGeom();

var cyan = new THREE.Color("rgb(0, 255, 255)");

var material = new THREE.MeshLambertMaterial({ color: cyan });

mesh = new THREE.Mesh(geom, material);

scene.add(mesh);

render();

}

intens.onchange = function(e) {

light.intensity = intens.value;

}

function fplus(x) {

return Math.sqrt((1 - (x \* x) / (A \* A)) \* B \* B);

}

function fminus(x) {

return -Math.sqrt((1 - (x \* x) / (A \* A)) \* B \* B);

}

function initGeom() {

// Аппроксимируем выпуклое тело треугольниками

var geometry = new THREE.BufferGeometry();

var vertices = [];

var step = 2 \* A / num;

// Лицевые треугольники

var z = H / 2;

var prevX = -A, prevY = fplus(prevX);

for (var x = -A + step; Math.floor(x) < A; x += step) {

var y = fminus(x);

vertices.push(0, 0, z);

vertices.push(prevX, prevY, z);

vertices.push(x, y, z);

geometry.addAttribute('position', new THREE.Float32BufferAttribute(vertices, 3));

vertices.push(x, y, -z);

vertices.push(x, y, z);

vertices.push(prevX, prevY, z);

geometry.addAttribute('position', new THREE.Float32BufferAttribute(vertices, 3));

prevX = x;

prevY = y;

}

var x = A, y = fminus(x);

vertices.push(0, 0, z);

vertices.push(prevX, prevY, z);

vertices.push(x, y, z);

geometry.addAttribute('position', new THREE.Float32BufferAttribute(vertices, 3));

vertices.push(x, y, -z);

vertices.push(x, y, z);

vertices.push(prevX, prevY, z);

prevX = -A, prevY = fminus(prevX);

for (var x = -A + step; x < A; x += step) {

var y = fplus(x);

vertices.push(x, y, z);

vertices.push(prevX, prevY, z);

vertices.push(0, 0, z);

geometry.addAttribute('position', new THREE.Float32BufferAttribute(vertices, 3));

vertices.push(prevX, prevY, -z);

vertices.push(prevX, prevY, z);

vertices.push(x, y, z);

geometry.addAttribute('position', new THREE.Float32BufferAttribute(vertices, 3));

prevX = x;

prevY = y;

}

var x = A, y = fplus(x);

vertices.push(x, y, z);

vertices.push(prevX, prevY, z);

vertices.push(0, 0, z);

geometry.addAttribute('position', new THREE.Float32BufferAttribute(vertices, 3));

vertices.push(prevX, prevY, -z);

vertices.push(prevX, prevY, z);

vertices.push(x, y, z);

geometry.addAttribute('position', new THREE.Float32BufferAttribute(vertices, 3));

// Задаем треуголники

z = -H / 2;

var prevX = -A, prevY = fplus(prevX);

for (var x = -A + step; x < A; x += step) {

var y = fplus(x);

vertices.push(x, y, z);

vertices.push(0, 0, z);

vertices.push(prevX, prevY, z);

geometry.addAttribute('position', new THREE.Float32BufferAttribute(vertices, 3));

vertices.push(x, y, z);

vertices.push(prevX, prevY, z);

vertices.push(x, y, -z);

geometry.addAttribute('position', new THREE.Float32BufferAttribute(vertices, 3));

prevX = x;

prevY = y;

}

var x = A, y = fplus(x);

vertices.push(x, y, z);

vertices.push(0, 0, z);

vertices.push(prevX, prevY, z);

geometry.addAttribute('position', new THREE.Float32BufferAttribute(vertices, 3));

vertices.push(x, y, z);

vertices.push(prevX, prevY, z);

vertices.push(x, y, -z);

geometry.addAttribute('position', new THREE.Float32BufferAttribute(vertices, 3));

prevX = -A, prevY = fminus(prevX);

for (var x = -A + step; x < A; x += step) {

var y = fminus(x);

vertices.push(prevX, prevY, z);

vertices.push(0, 0, z);

vertices.push(x, y, z);

geometry.addAttribute('position', new THREE.Float32BufferAttribute(vertices, 3));

vertices.push(prevX, prevY, -z);

vertices.push(prevX, prevY, z);

vertices.push(x, y, z);

geometry.addAttribute('position', new THREE.Float32BufferAttribute(vertices, 3));

prevX = x;

prevY = y;

}

var x = A, y = fminus(x);

vertices.push(prevX, prevY, z);

vertices.push(0, 0, z);

vertices.push(x, y, z);

geometry.addAttribute('position', new THREE.Float32BufferAttribute(vertices, 3));

vertices.push(prevX, prevY, -z);

vertices.push(prevX, prevY, z);

vertices.push(x, y, z);

geometry.addAttribute('position', new THREE.Float32BufferAttribute(vertices, 3));

geometry.computeVertexNormals();

return geometry;

}

function changeColor(x) {

// scene.children[3].material.color.setHSL(0.7, 1, 0,5);

scene.children[3].material.color = new THREE.Color("hsl(" + x + ", 100%, 50%)");

}

function animate() {

requestAnimationFrame(animate);

scene.children[3].material.color = new THREE.Color("hsl(" + delta % 360 + ", 100%, 50%)");

if (delta >= 360) {

colorFlag = false;

} else if (delta <= 0) {

colorFlag = true;

}

if (colorFlag) {

delta += 1;

} else {

delta -= 1;

}

controls.update();

render();

}

function render() {

renderer.render(scene, camera);

}

function onLoad() {

init();

render();

animate();

}

1. **Выводы**

Выполнив данную лабораторную работу, я закрепил знания по использованию HTML и запуску графический интерфейс, перерисовыванию его в зависимости от изменения окна и отрисовки многогранника. Также познакомился с отображения интерактивной 3D графики. Моделирование графики с использованием Three.js можно сравнить со съемочной площадкой, так как у нас есть возможность оперировать такими понятиями как сцена, свет, камера, объекты и их материалы. Также научился делать шейдерную анимацию эффектов.

**ЛИТЕРАТУРА**

1. Введение в 3D: основы Three.js [Электронный ресурс]URL: <https://habr.com/ru/post/494810/> (Дата обращения: 10.11.2021).

2. Three.js - импорт орбитальных контроллеров [Электронный ресурс]URL: <https://themightyprogrammer.dev/snippet/orbit-controls-webpack> (Дата обращения: 10.11.2021).

3. Орбитальные контроллеры - Three.js [Электронный ресурс]URL: <https://threejs.org/docs/#examples/en/controls/OrbitControls> (Дата обращения: 10.11.2021).