**Московский авиационный институт**

**(Национальный исследовательский университет)**

Институт: «Информационные технологии и прикладная математика»

Кафедра: 806 «Вычислительная математика и программирование»

Дисциплина: «Компьютерная графика»

**Курсовой проект**

**по дисциплине «Компьютерная графика»**

Студент: Алексеев Владислав Евгеньевич

Группа: М8О-306Б-19

Преподаватель: Чернышов Л.Н.

Дата:

Оценка:

Москва, 2021

1. **Постановка задачи**

Составить и отладить программу, обеспечивающую каркасную визуализацию порции поверхности заданного типа. Исходные данные готовятся самостоятельно и вводятся из файла или в панели ввода данных. Должна быть обеспечена возможность тестирования программы на различных наборах исходных данных. Программа должна обеспечивать выполнение аффинных преобразований для заданной порции поверхности, а также возможность управлять количеством изображаемых параметрических линий. Для визуализации параметрических линий поверхности разрешается использовать только функции отрисовки отрезков в экранных координатах.

**Вариант:** Кинематическая поверхность. Образующая – эллипс, направляющая – Cardinal Spline 3D.

1. **Описание программы**

**Cardinal spline 3D**

Для начала поговорим о Cardinal spline 3D. Это один из видов гладких кривых, строится изначально по массиву точек pk, а также mk, что является тангенсом наклона кривой при прохождении данной точки. Для каждых двух точек из заданного массива имеем аппроксимацию:

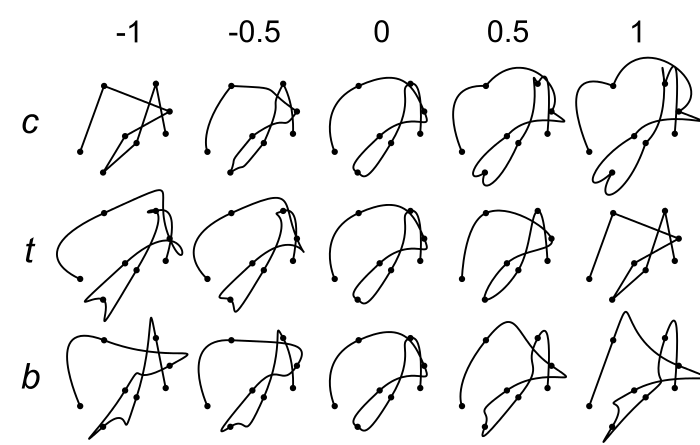
где для k = 1, …, n - интерполяционная сетка, t ∈ [0, 1], а

или

где c ∈ [0, 1], а при c = 0 называется сплайном Катмалла-Рома.

Но в данной работе для вычисления я использовал функцию Kochanek-Bartels Spline, выглядят они примерно так:

где t - регулирует длину вектора тангенса, b - направление вектора тангенса, c - так называемый continuity, задающий остроту угла между векторами. Далее пример:

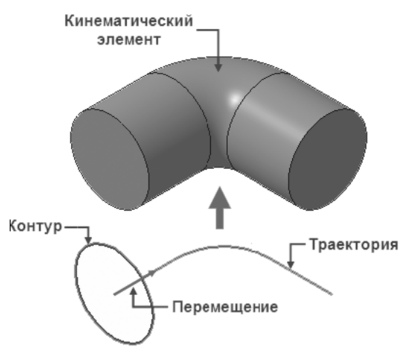


**Кинематическая поверхность**

Сам способ задания такого рода поверхности состоит из двух составляющих: направляющая, а также сама фигура.

С направляющей разобрались, она задается пользователем по точкам, далее интерполируется, достраивается до определенной точности, чтобы казаться достаточно гладкой.

Далее стоит разобраться с видами, есть кинематические поверхности переноса, они перемещают образующую фигуру по направлению направляющей кривой. Также существует вариант также с каждым переносом образующей, поворачивать ее вдоль тангенса наклона направляющей, к примеру поместив круг в основание, перемещая и вращая образующую мы получим фигуру трубы постоянного радиуса.



**3. Набор тестов**

1. Изначальный вид.

2. Уменьшение параметра a.

3. Уменьшение параметра b.

4. Увеличение параметра a.

5. Увеличение параметра b.

6. Изменение масштаба.

7. Вращение поверхности.

**4. Результаты выполнения тестов**

1. Изначальный вид.

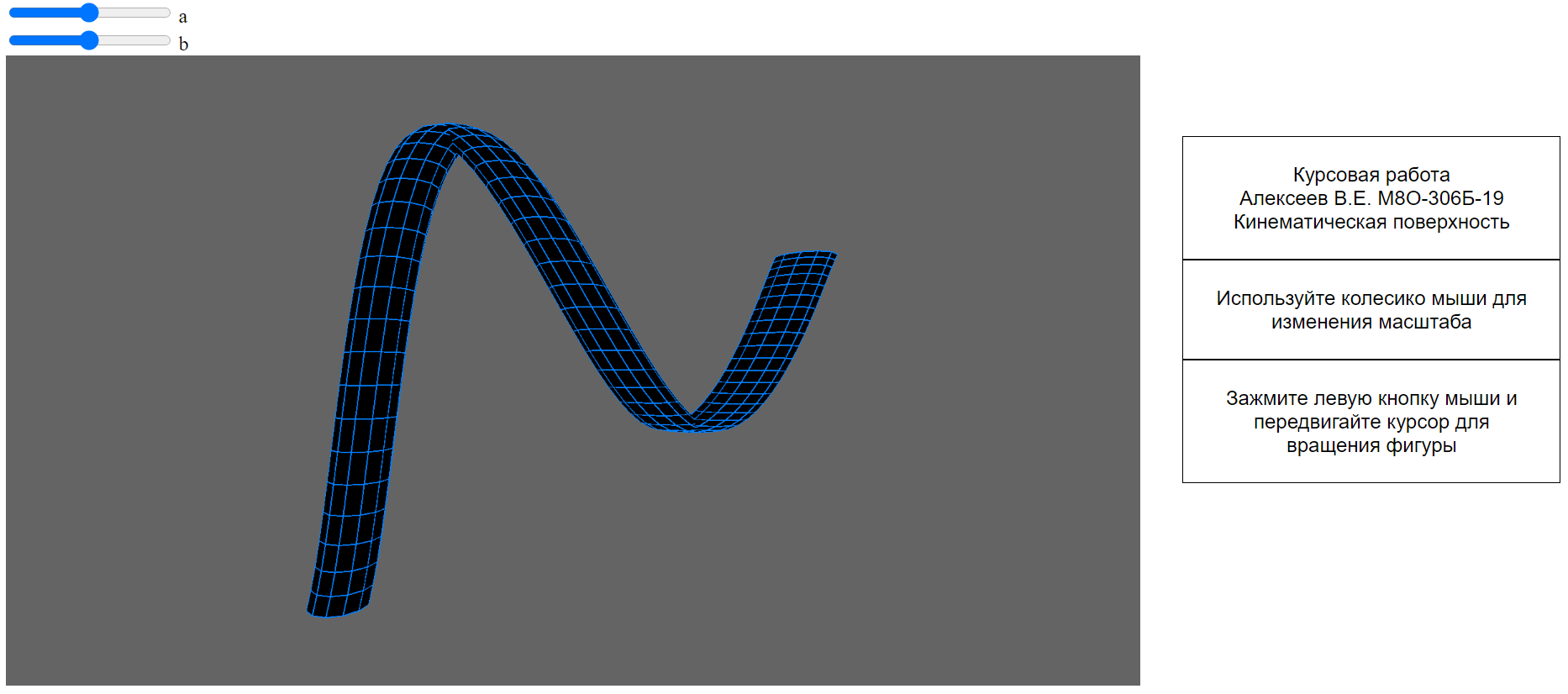


Рис. 4.1 Результат открытия окна

2. Уменьшение параметра a.

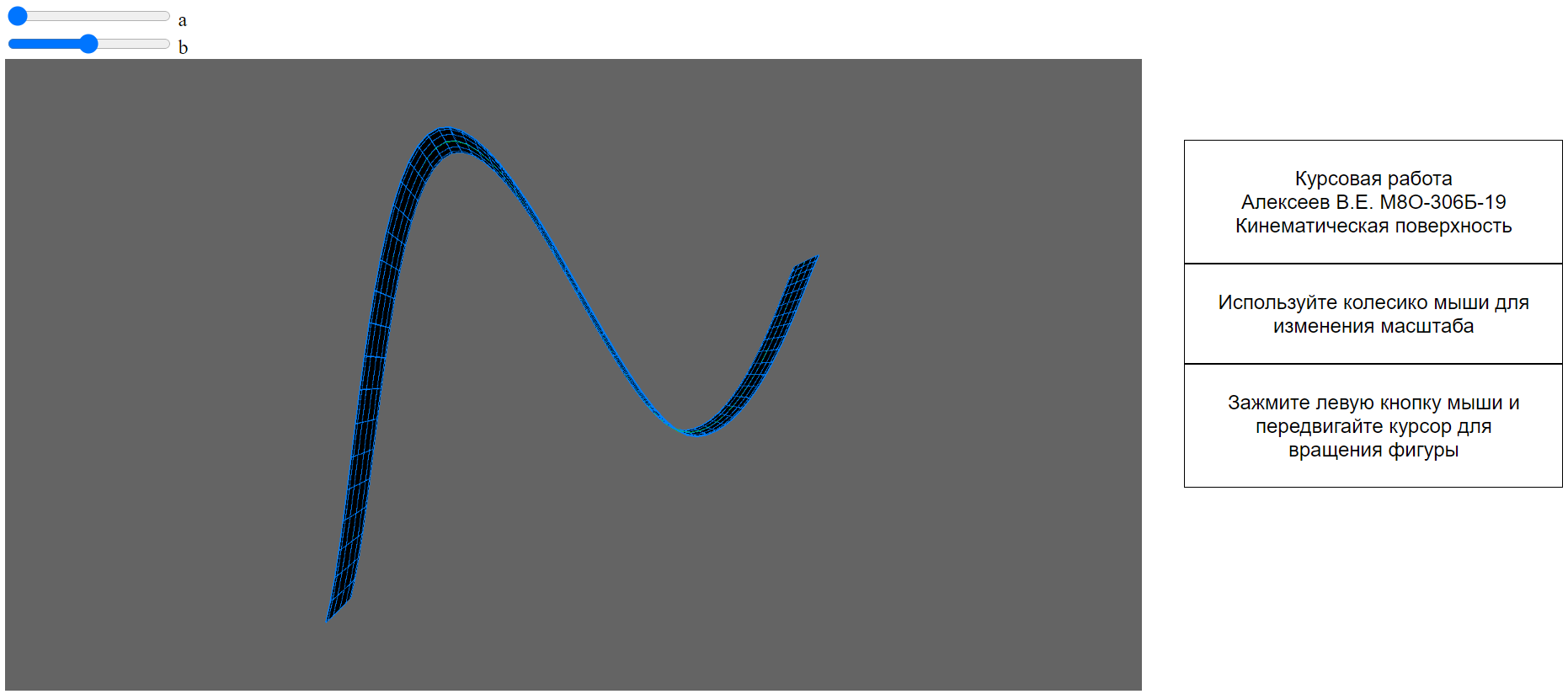


Рис. 4.2 Результат уменьшения параметра a

3. Уменьшение параметра b.

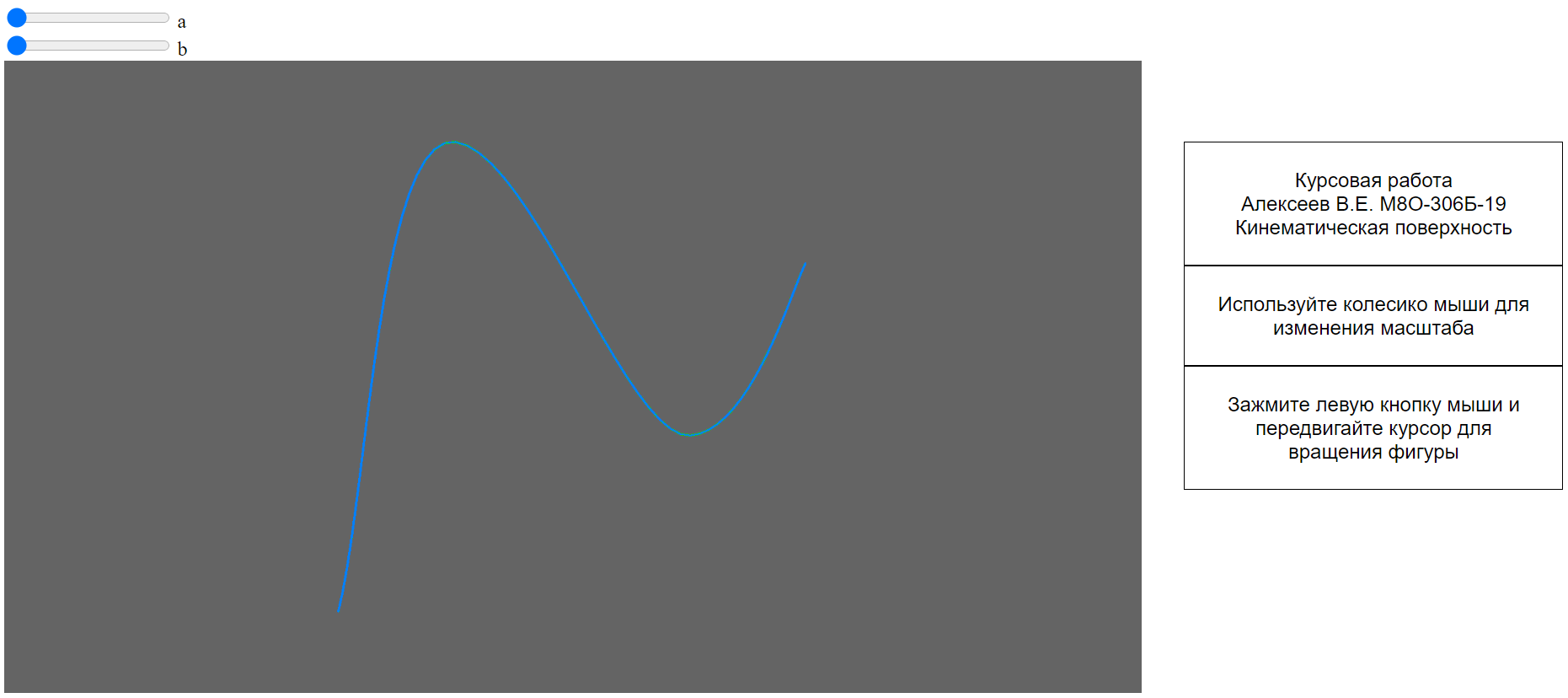


Рис. 4.3 Результат уменьшения параметра b

4. Увеличение параметра a.

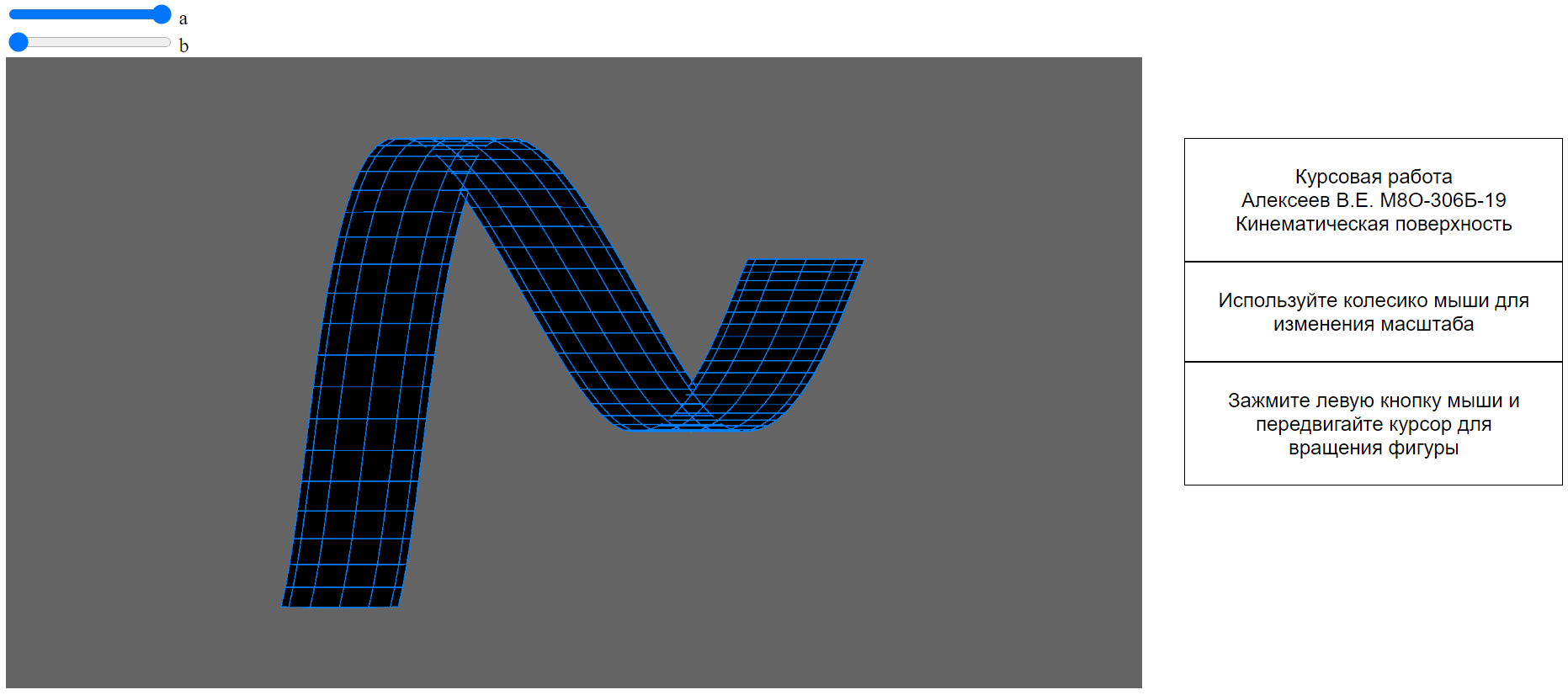


Рис. 4.4 Результат увеличения параметра a

5. Увеличение параметра b.

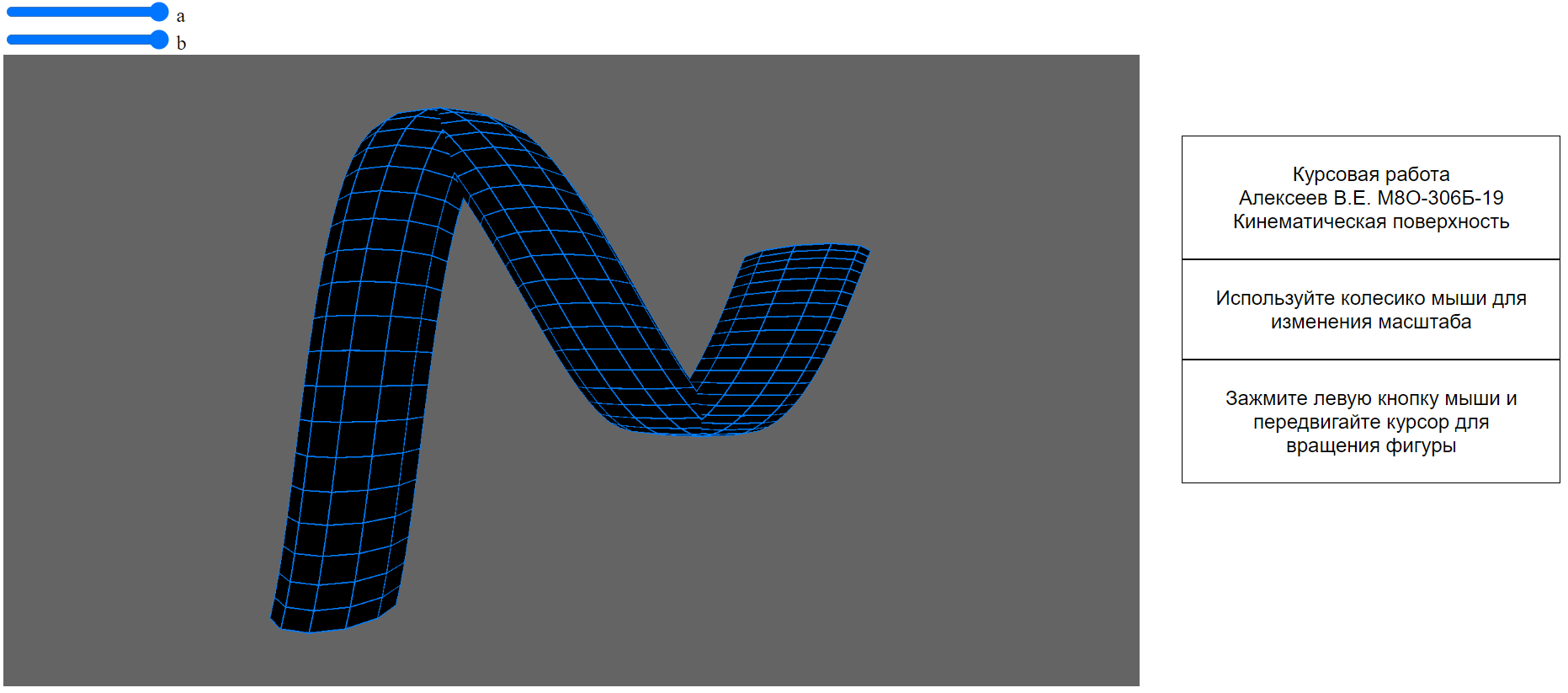


Рис. 4.5 Результат увеличения параметра b

6. Изменение масштаба.



Рис. 4.6 Результат изменения масштаба

7. Вращение поверхности.



Рис. 4.7 Результат вращения поверхности

**5. Листинг программы**

**main.html**

<!-- Алексеев В.Е. М8О-306Б-19

Каркасная визуализация порции поверхности заданного типа.

Вариант: Кинематическая поверхность. Образующая – эллипс, направляющая – Cardinal Spline 3D. -->

<html>

<head>

<meta charset = 'utf-8'>

<title>Курсовая работа по комьютерной графике</title>

<link rel = 'stylesheet' href = './styles.css'>

<script src = "https://cdnjs.cloudflare.com/ajax/libs/p5.js/0.7.2/p5.js"></script>

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

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

</head>

<body>

<div>

<input id = "a" type = "range" min = "1" max = "100" step = "1" value = "50">

a <br>

</div>

<div>

<input id = "b" type = "range" min = "1" max = "100" step = "1" value = "50">

b <br>

</div>

<div class = 'info'>

<div>

Курсовая работа <br>

Алексеев В.Е. М8О-306Б-19 <br>

Кинематическая поверхность

</div>

<div>

Используйте колесико мыши для изменения масштаба

</div>

<div>

Зажмите левую кнопку мыши и передвигайте курсор для вращения фигуры

</div>

</div>

</body>

</html>

**style.css**

.info {

position: fixed;

font-family: sans-serif;

top: 20%;

right: 3%;

width: 300px;

height: 700px;

text-align: center;

}

.info div {

padding: 20px;

border: 1px solid black;

}

**drow.js**

function draw\_based\_points(point) {

var curve\_x = point.x - (mouseX - point.x);

var curve\_y = point.y - (mouseY - point.y);

point.curve\_point\_front = {x: mouseX, y: mouseY};

point.curve\_point\_back = {x: curve\_x, y: curve\_y};

strokeWeight(1);

stroke(0, 0, 250);

line(point.x, point.y, mouseX, mouseY);

line(point.x, point.y, curve\_x, curve\_y);

strokeWeight(1);

stroke(10);

fill(0, 0, 250);

ellipse(mouseX, mouseY, 6);

ellipse(curve\_x, curve\_y, 6);

strokeWeight(2);

stroke(0, 0, 250);

noFill();

rect(mouseX - 6, mouseY - 6, 12, 12);

rect(curve\_x - 6, curve\_y - 6, 12, 12);

}

function draw\_control\_points() {

strokeWeight(1);

stroke(10);

if (control\_points.length == 1) {

return;

}

for (var i = 0; i < control\_points.length; i++) {

noFill();

ellipse(control\_points[i].x, control\_points[i].y, 12);

fill(0, 0, 250);

ellipse(control\_points[i].x, control\_points[i].y, 4);

if (control\_points[i].curve\_point\_back && control\_points[i].curve\_point\_front) {

stroke(0, 0, 250);

line(

control\_points[i].curve\_point\_back.x,

control\_points[i].curve\_point\_back.y,

control\_points[i].x,

control\_points[i].y

);

line(

control\_points[i].curve\_point\_front.x,

control\_points[i].curve\_point\_front.y,

control\_points[i].x,

control\_points[i].y

);

stroke(10);

fill(0, 0, 250);

ellipse(control\_points[i].curve\_point\_front.x, control\_points[i].curve\_point\_front.y, 6);

ellipse(control\_points[i].curve\_point\_back.x, control\_points[i].curve\_point\_back.y, 6);

}

}

}

function draw\_cardinal\_spline() {

strokeWeight(4);

stroke(0, 230, 10);

for (var i = 0; i < res\_dots.length - 1; i++) {

line(res\_dots[i].x, res\_dots[i].y, res\_dots[i].z, res\_dots[i + 1].x, res\_dots[i + 1].y, res\_dots[i + 1].z);

}

}

function drawOval(x, y, z) {

stroke(0, 128, 255);

for (var i = 0; i < 360; i++) {

var x\_1 = x + cos(i / 360 \* 2 \* PI) \* document.getElementById('a').value;

var y\_1 = y + sin(i / 360 \* 2 \* PI) \* document.getElementById('b').value;

var x\_2 = x + cos((i + 1) / 360 \* 2 \* PI) \* document.getElementById('a').value;

var y\_2 = y + sin((i + 1) / 360 \* 2 \* PI) \* document.getElementById('b').value;

line(x\_1, y\_1, z, x\_2, y\_2, z);

}

}

function draw\_kinematic\_plane(percent) {

for (var i = 0; i < res\_dots.length - 1; i++) {

push();

stroke(0, 128, 255);

strokeWeight(2);

var piv = createVector(res\_dots[i + 1].x - res\_dots[i].x, res\_dots[i + 1].y - res\_dots[i].y, res\_dots[i + 1].z - res\_dots[i].z);

var a = res\_dots[i];

var b = res\_dots[i+1];

var step = 30;

for (var fi = 0; fi <= 360; fi += step) {

var x\_1 = cos(fi / 360 \* 2 \* PI) \* document.getElementById('a').value;

var z\_1 = sin(fi / 360 \* 2 \* PI) \* document.getElementById('b').value;

var x\_2 = cos((fi + step) / 360 \* 2 \* PI) \* document.getElementById('a').value;

var z\_2 = sin((fi + step) / 360 \* 2 \* PI) \* document.getElementById('b').value;

beginShape();

vertex(x\_1 + a.x, a.y, z\_1 + a.z);

vertex(x\_2 + a.x, a.y, z\_2 + a.z);

vertex(x\_2 + a.x + piv.x, a.y + piv.y, z\_2 + a.z + piv.z);

vertex(x\_1 + a.x + piv.x, a.y + piv.y, z\_1 + a.z + piv.z);

endShape(CLOSE);

}

beginShape();

for (var fi = 0; fi <= 360; fi += step) {

var x\_1 = cos(fi / 360 \* 2 \* PI) \* document.getElementById('a').value;

var z\_1 = sin(fi / 360 \* 2 \* PI) \* document.getElementById('b').value;

vertex(x\_1 + a.x, a.y, z\_1 + a.z);

}

endShape(CLOSE);

beginShape();

for (var fi = 0; fi <= 360; fi += step) {

var x\_1 = cos(fi / 360 \* 2 \* PI) \* document.getElementById('a').value;

var z\_1 = sin(fi / 360 \* 2 \* PI) \* document.getElementById('b').value;

vertex(x\_1 + a.x + piv.x, a.y + piv.y, z\_1 + a.z + piv.z);

}

endShape(CLOSE);

pop();

}

}

**sketch.js**

var control\_points = [];

var res\_dots = [];

function setup\_curve() {

control\_points = [

{x:-400, y:400, z:0},

{x:-200, y:-400, z:0},

{x:200, y:100, z:0},

{x:400, y:-200, z:0}

];

}

function update\_curve() {

if (control\_points.length == 1) {

return;

}

res\_dots = [];

var t = 0.3;

var b = -0.3;

var c = 0.7;

for (var i = 0; i < control\_points.length; i++) {

var curr = control\_points[i];

if (i == 0) {

var next = control\_points[i + 1];

var p = createVector(curr.x, curr.y, curr.z);

var pn = createVector(next.x, next.y, next.z);

pn.sub(p);

control\_points[i].d = p5.Vector.mult(pn, (1-t)\*(1-b)\*(1+c)/2);

} else if (i == control\_points.length - 1) {

var prev = control\_points[i - 1];

var p = createVector(curr.x, curr.y, curr.z);

var pp = createVector(prev.x, prev.y, prev.z);

p.sub(pp);

control\_points[i].d = p5.Vector.mult(pn, (1-t)\*(1+b)\*(1+c)/2);

} else {

var next = control\_points[i + 1];

var prev = control\_points[i - 1];

var p = createVector(curr.x, curr.y, curr.z);

var pn = createVector(next.x, next.y, next.z);

var pp = createVector(prev.x, prev.y, prev.z);

pn.sub(p);

p.sub(pp);

pn.mult((1 - t) \* (1 - b) \* (1 + c) / 2);

p.mult((1 - t) \* (1 + b) \* (1 + c) / 2);

control\_points[i].d = p5.Vector.add(p,pn);

}

}

for (var i = 0; i < control\_points.length - 1; i++) {

var a = control\_points[i];

var b = control\_points[i + 1];

for (var t = 0; t <= 1; t += 0.05) {

var p = createVector(a.x, a.y, a.z);

var pn = createVector(b.x, b.y, b.z);

var m = createVector(a.d.x, a.d.y, a.d.z);

var mn = createVector(b.d.x, b.d.y, b.d.z);

p.mult(2 \* pow(t, 3) - 3 \* pow(t, 2) + 1);

m.mult(pow(t, 3) - 2 \* pow(t, 2) + t);

pn.mult(-2 \* pow(t, 3) + 3 \* pow(t, 2));

mn.mult(pow(t, 3) - pow(t, 2));

res\_dots.push(p5.Vector.add(p5.Vector.add(p, m), p5.Vector.add(pn, mn)));

}

}

}

function setup() {

createCanvas(900, 500, WEBGL);

background(100);

setup\_curve();

update\_curve();

ambientLight(100);

ambientMaterial(0, 0, 0);

}

function draw() {

background(100);

orbitControl();

draw\_cardinal\_spline();

draw\_kinematic\_plane(100);

}

**6. Выводы**

Выполнив данный курсовой проект я закрепил и усовершенствовал знания OpenGl. Смог применить знания всех предыдущих ЛР по курсе, научился строить сложные параметрические поверхности и изменять их. Использование графической библиотеки p5.js для визуализации алгоритмических процессов, таких как представление кривых, объектов и структур очень удобно. А веб модель реализует кроссплатформенность.

**ЛИТЕРАТУРА**

1. Кинематические поверхности [Электронный ресурс]URL: <https://studme.org/288224/tehnika/kinematicheskie_poverhnosti> (Дата обращения: 09.12.2021).

2. Геометрическое моделирование [Электронный ресурс]URL: <http://www.vixri.ru/d3/Golovanov%20N.N._Geometricheskoe%20modelirovanie_2002.pdf> (Дата обращения: 09.12.2021).

3. Кривые и поверхности на экране компьютера. Руководство по сплайнам для пользователя [Электронный ресурс]URL: <https://www.studmed.ru/shikin-ev-plis-li-krivye-i-poverhnosti-na-ekrane-kompyutera_c34ae521ee6.html> (Дата обращения: 09.12.2021).