**Московский авиационный институт**

**(Национальный исследовательский университет)**

Институт: «Информационные технологии и прикладная математика»

Кафедра: 806 «Вычислительная математика и программирование»

Дисциплина: «Компьютерная графика»

**Курсовая работа**

**по дисциплине «Компьютерная графика»**

Студент: Лошманов Юрий Андреевич

Группа: 80-306

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:** NURBS поверхность порядка 4x4.

1. **Описание программы**

**NURBS**

Неоднородный рациональный B-сплайн, NURBS (англ. Non-uniform rational B-spline) — математическая форма, применяемая в компьютерной графике для генерации и представления кривых и поверхностей. Как следует из названия, является частным случаем B-сплайнов, причём широко распространённым из-за своей стандартизированности и относительной простоты.

**NURBS моделирование**

В основе NURBS-моделирования лежат математические формулы, намного более сложные, чем для создания полигонов. Так как сцены с NURBS-моделямп требуют дополнительных ресурсов, они используются в ситуациях, когда визуализация производится заблаговременно, например в анимационных фильмах или на телевидении. Так как NURBS-модели отличаются плавными и искривленными формами, моделирование на основе неоднородных рациональных сплайнов Безье чаще всего используется для создания персонажей автомобилей и тому подобных объектов.

Как уже упоминалось, аббревиатура NURBS расшифровывается как «неоднородные рациональные сплайны Безье». NURBS-геометрия основана на кривых, математическая концепция которых была разработана французским инженером Пьером Безье. Кривые Безье располагаются между управляющими вершинами (CV — control vertices) и основаны на уравнениях третьего порядка.

Для создания кривой Безье требуются как минимум четыре управляющих вершины. Установка каждой следующей такой вершины приводит к появлению очередного участка кривой, или сплайна, изогнутого наиболее удобным способом.

Как показано на рис. 2.1, управляющие вершины контролируют кривизну. Они соединяются друг с другом с помощью каркаса (hull), который используется для одновременного выделения наборов вершин. Первая управляющая вершина выглядит как закрытый квадратик, а вторая, задающая направление кривой, — как открытый. Кривой принадлежат только первая и последняя управляющие вершины, которые определяют ее начало и конец.

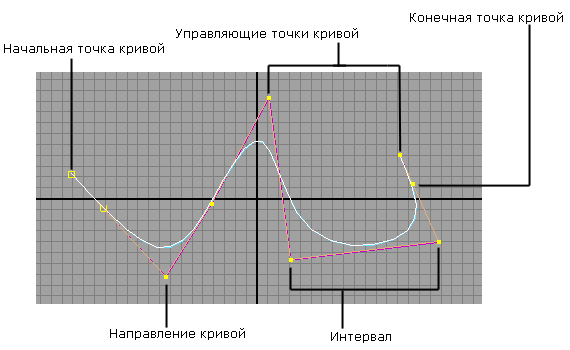


Рис. 2.1. Кривая Безье и ее компоненты

В то время как управляющие точки контролируют кривизну сплайна Безье, сами сплайны, называемые изопараметрическими кривыми (isoparams), определяют Кривизну NURBS-поверхности (рис. 2.2). Расстояние между двумя изопараметрическими кривыми называется интервалом (span). Чем больше количество интервалов, тем более детализированной является поверхность. Однако имейте В Виду, что наличие в сцене поверхностей с высокой детализацией замедляет процесс визуализации.

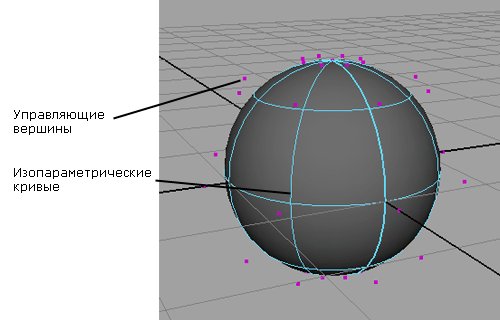


Рис. 2.2. NURBS-поверхность создается между изопараметрическими кривыми; отредактировать ее форму можно перемещением управляющих вершин

Изменение формы NURBS-поверхности основано на интерполяции кривых, в то время как деформация полигональных моделей связана с изменением ориентации наборов граней. Именно по этой причине намного проще получить гладкую деформацию NURBS-поверхности с небольшим набором управляющих вершин. Чтобы достичь аналогичного результата у сетки полигонов, потребуется увеличить детализацию поверхности.

NURBS-поверхность позволяет получить более гладкую деформацию (рис. 2.3), в то время как полигональная сетка разбивается на набор граней в местах расположения ребер. Для достижения такого же результата у сеток полигонов требуется дополнительная детализация.

Таким образом, если объект имеет плавные очертания, для его моделирования лучше использовать NURBS-поверхность. Если вы сомневаетесь в правильности выбора формата, все равно лучше начать моделирование на основе неоднородных рациональных В-сплайнов. Ведь потом готовую модель можно легко преобразовать в сетку полигонов, в то время как обратное преобразование зачастую довольно сложно реализуемо.

Рассмотрим процесс создания NURBS-кривой. Выберите в меню File (Файл) команду New (Создать) и разверните на весь экран окно проекции Perspective (Перспектива). Выберите в меню Create (Создать) команду CV Curve Tool (Построение CV-кривой) и обратите внимание на то, как при этом изменится форма указателя мыши. Несколько раз щелкните в разных точках координатной плоскости, формируя кривую. Как легко заметить, кривая Безье создается между управляющими вершинами. Нажмите клавишу Enter, чтобы закончить кривую.

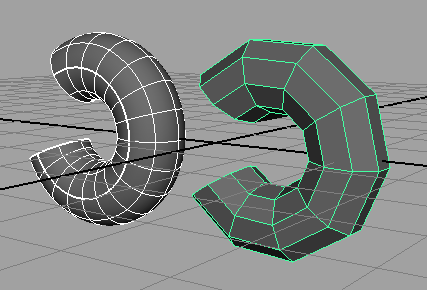


Рис. 2.3. Результат сгиба NURBS-цилиндра (слева) и полигонального цилиндра (справа). NURBS-цилиндр остался гладким, в то время как полигональная модель приобрела фасеточный вид

**3. Набор тестов**

1. Вид test1.txt.

2. Увеличение количества параметрических линий.

3. Вращение плоскости.

4. Вид test2.txt.

5. Увеличение количества параметрических линий.

6. Вращение плоскости.

**4. Результаты выполнения тестов**

1.Вид test1.txt.

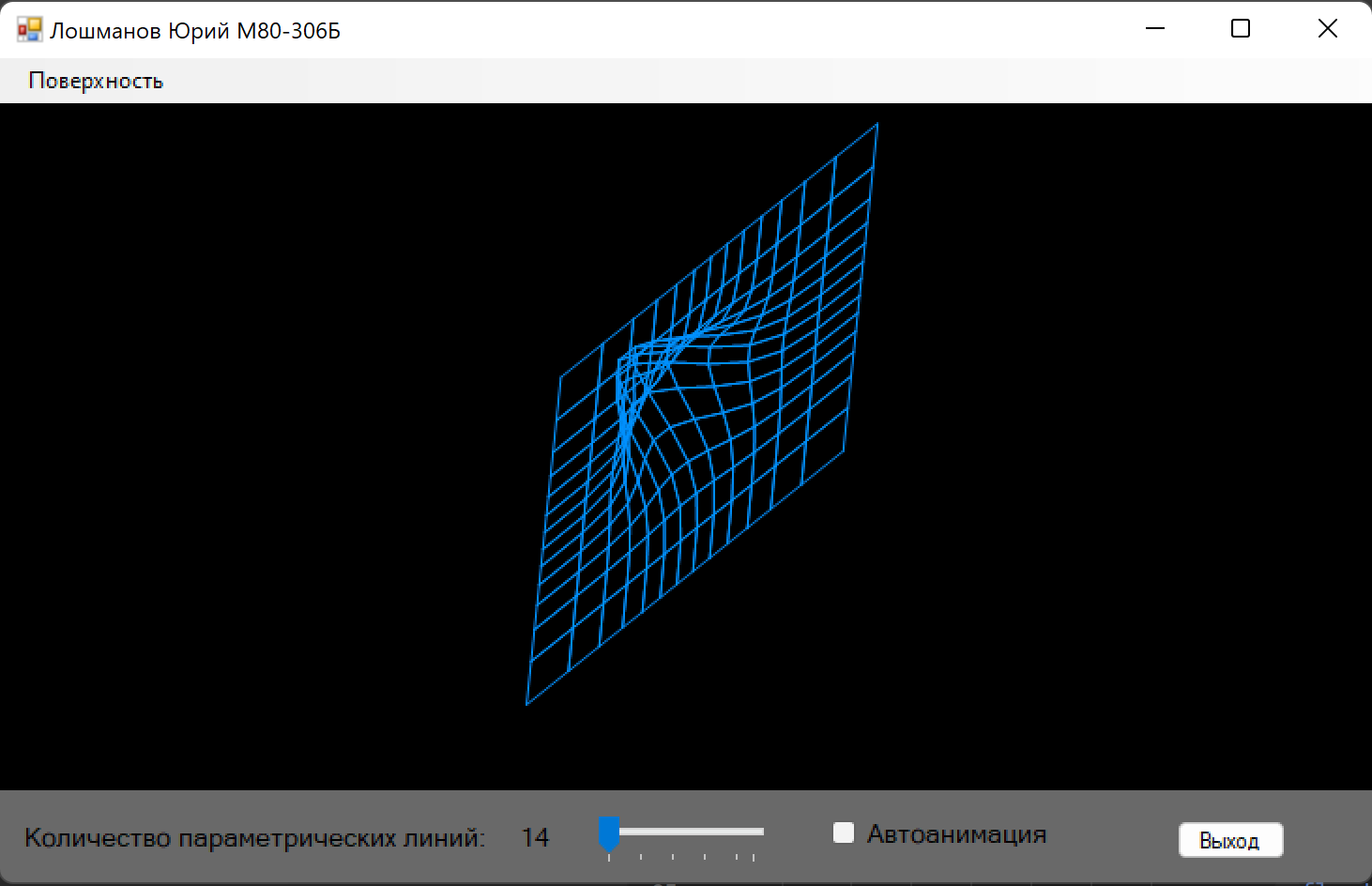


Рис. 4.1 Результат вида test1.txt

2.Увеличение количества параметрических линий.

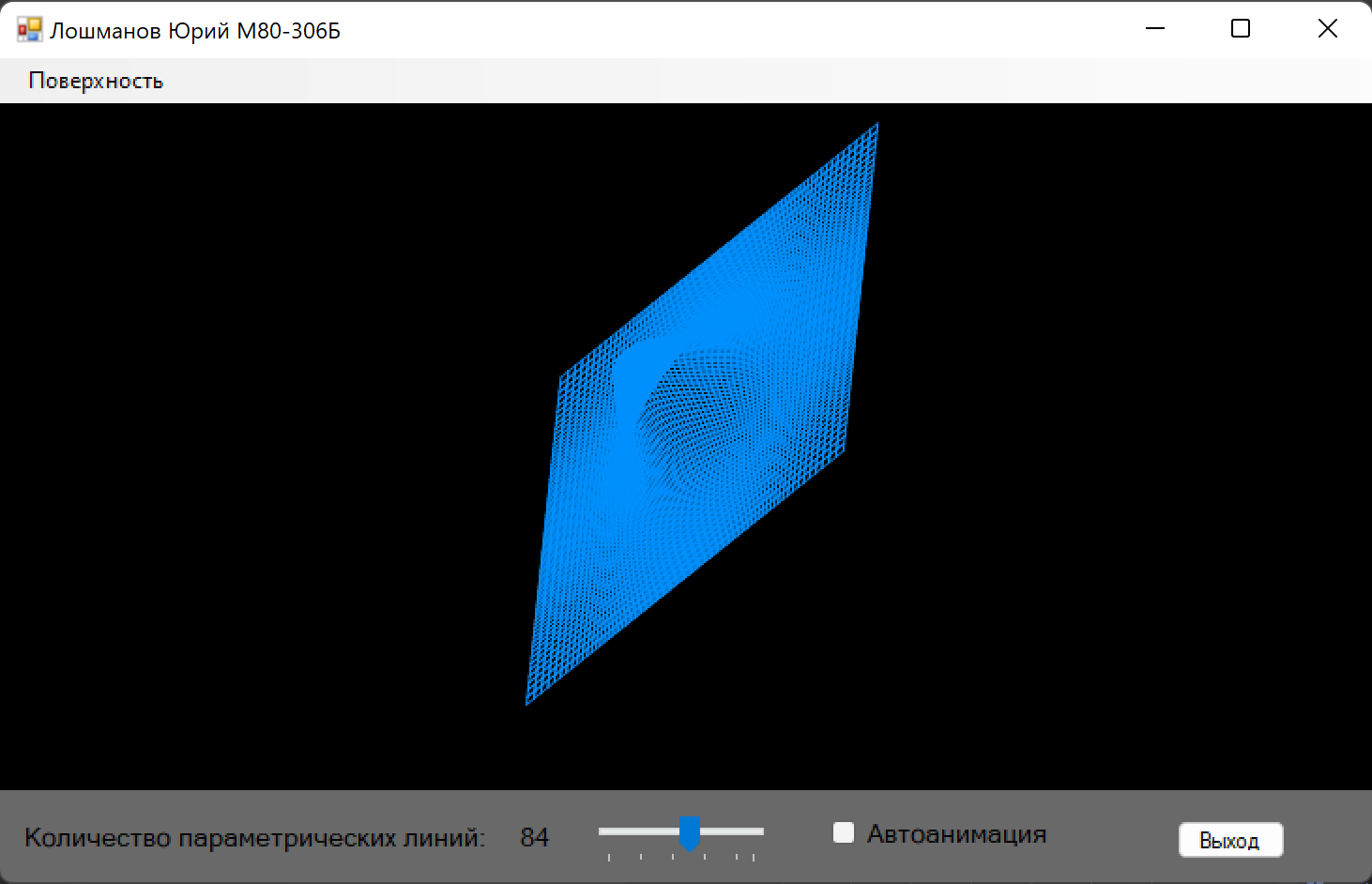


Рис. 4.2 Результат увеличения количества параметрических линий

3. Вращение поверхности.

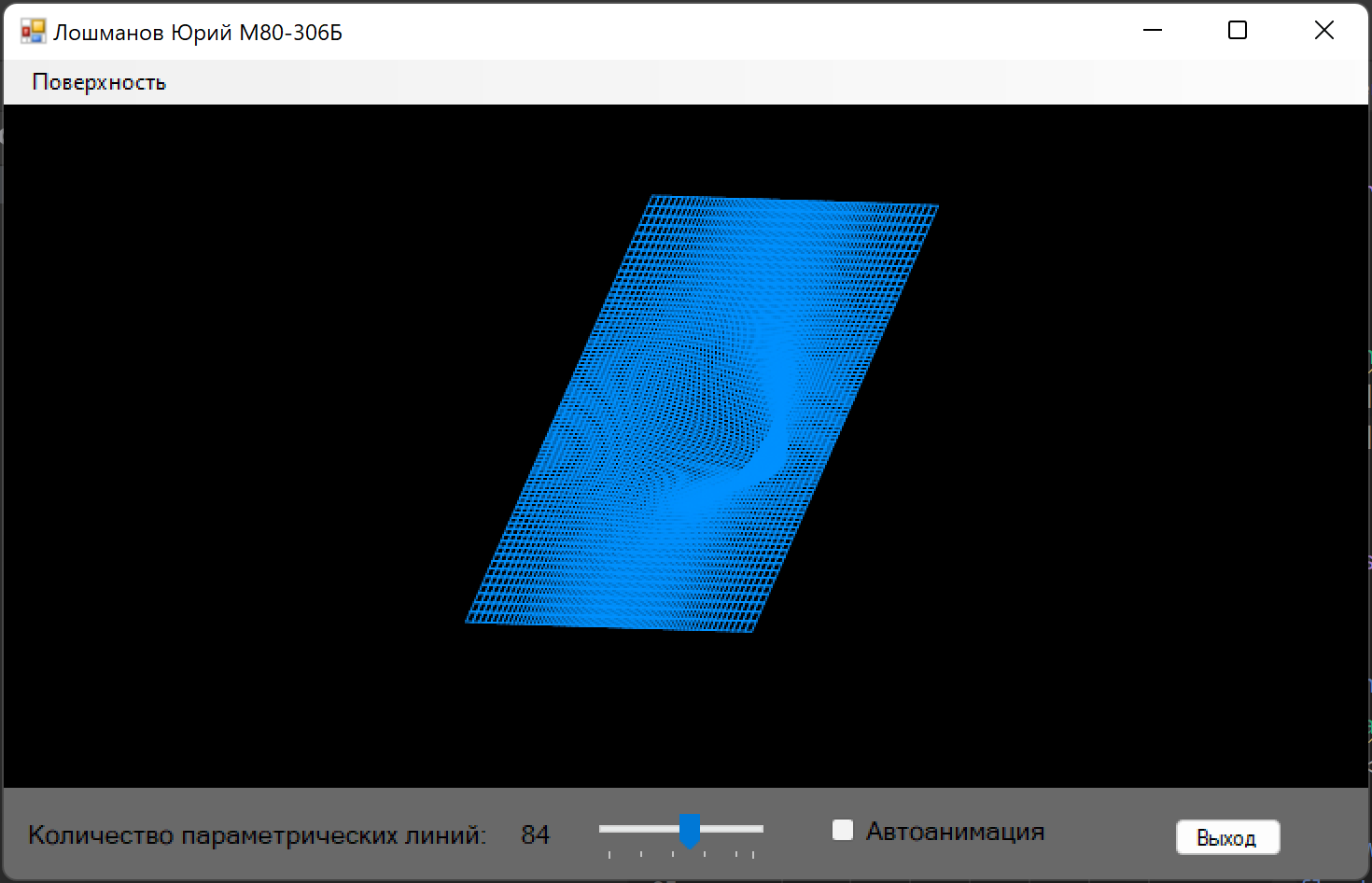


Рис. 4.3 Результат вращения плоскости

4. Вид test2.txt.

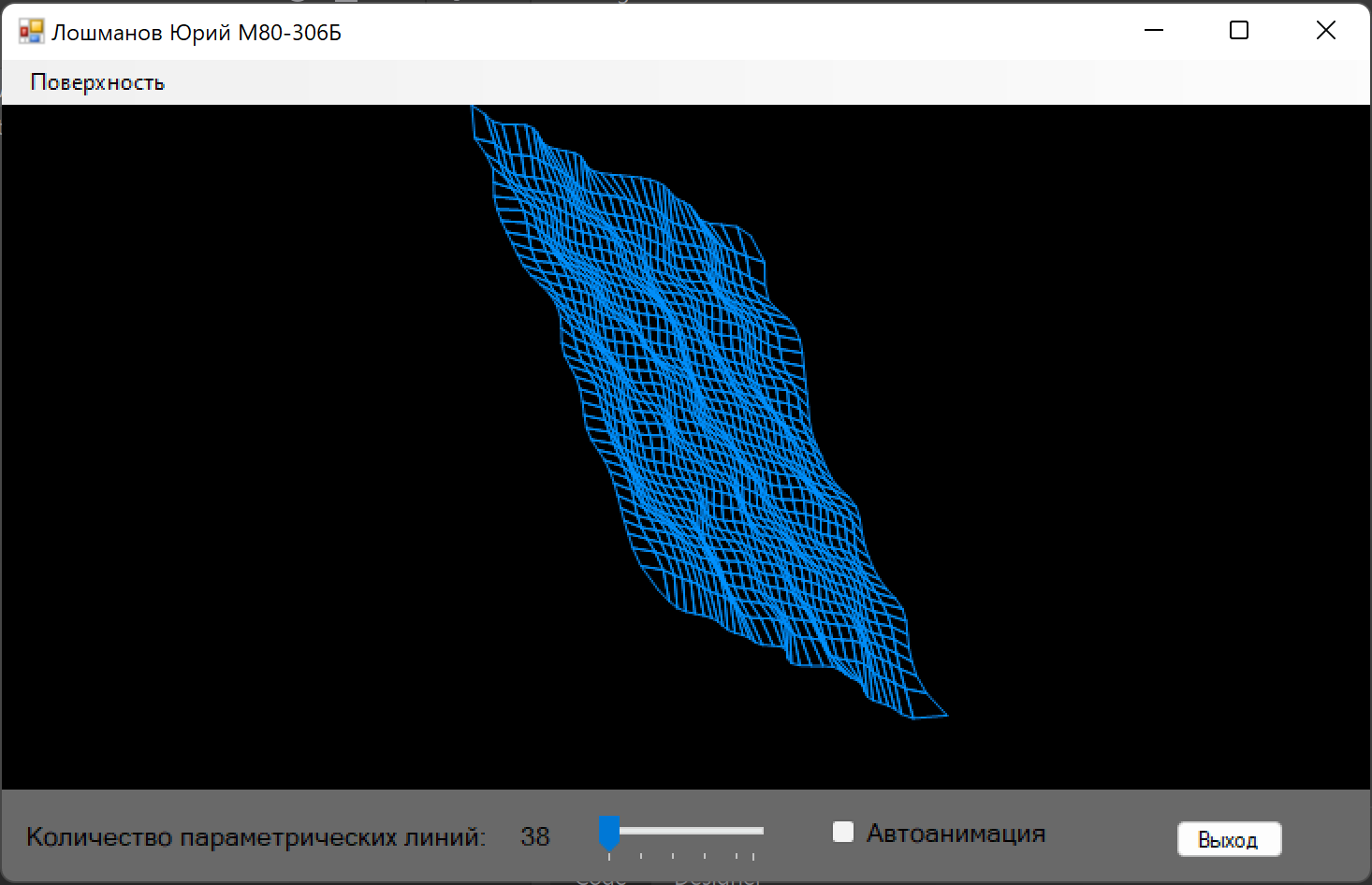


Рис. 4.4 Результат вида test2.txt

5. Увеличение количества параметрических линий.

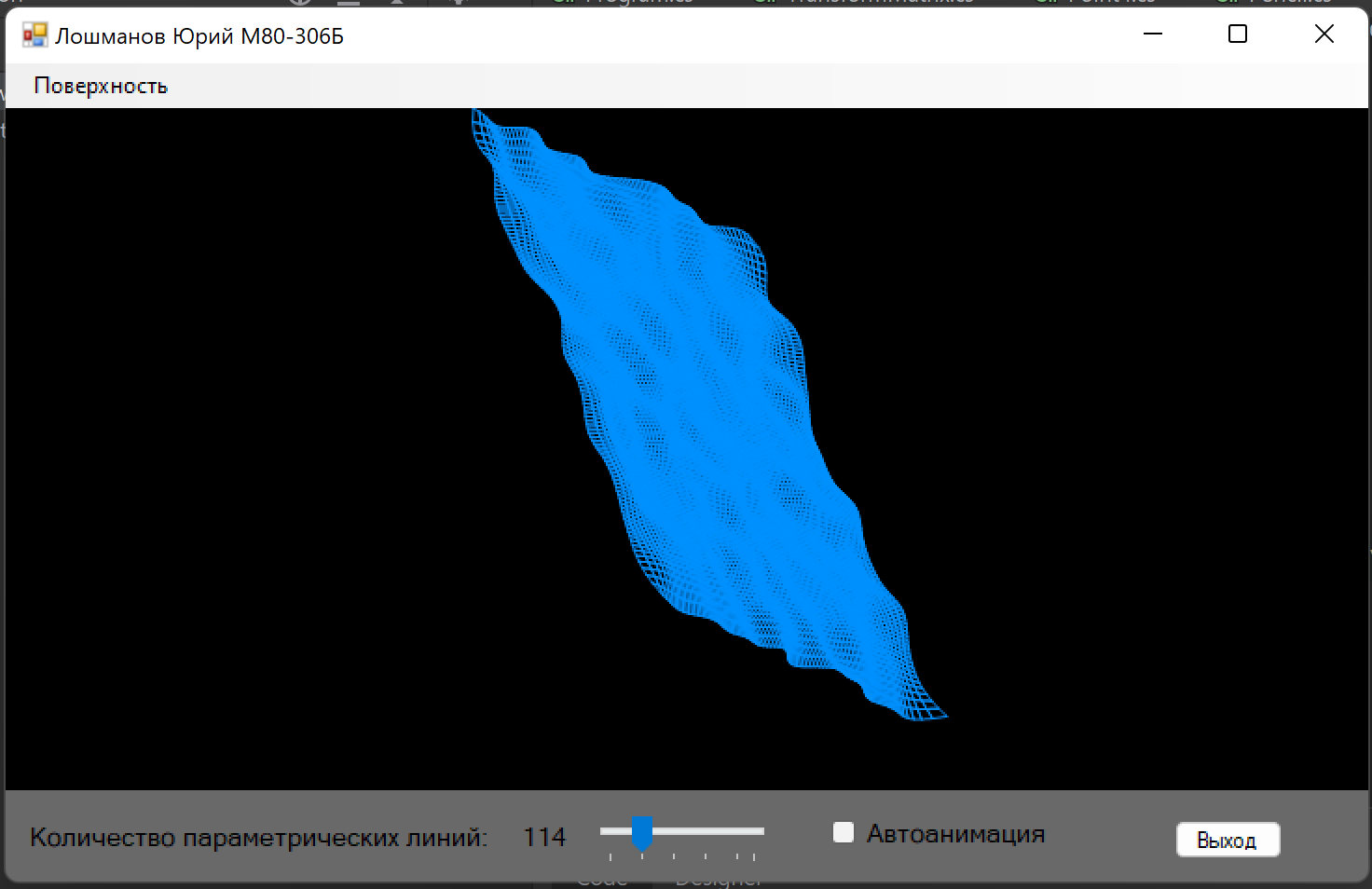


Рис. 4.5 Результат увеличения количества параметрических линий

6. Вращение поверхности.

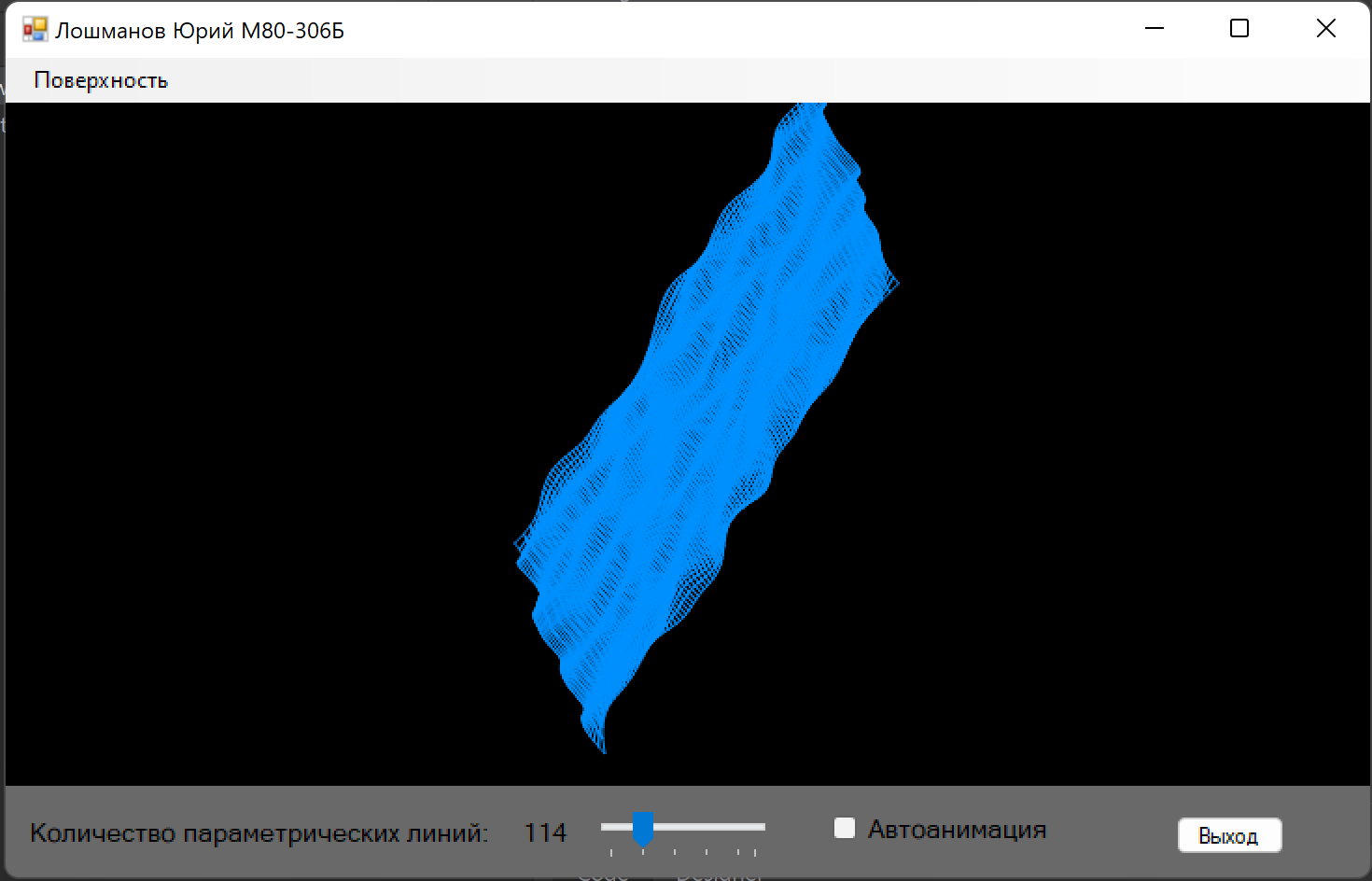


Рис. 4.6 Результат вращения плоскости

**5. Листинг программы**

**Program.cs**

// Лошманов Юрий Андреевич

// М8О-306Б-19

//

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace kp\_work

{

static class Program

{

/// <summary>

/// Главная точка входа для приложения.

/// </summary>

[STAThread]

static void Main()

{

Application.EnableVisualStyles();

Application.SetCompatibleTextRenderingDefault(false);

Application.Run(new Form1());

}

}

}

**NURBS.cs**

// Фамилия: Лошманов Ю.А.

// Группа: M80-306Б

// Программа, реализующая NURBS поверхность порядка 4x4

using System;

using System.Collections.Generic;

using System.Diagnostics;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace kp\_work

{

public class NURBS

{

private int k; //порядок B-сплайна по направлению u

private int q; //порядок B-сплайна по направлению v

private int m; //кол-во контрольных точек по направлению u

private int n; //кол-во контрольных точек по направлению v

private List<List<Point>> contrPts; //Массив контрольных точек

private int[] uPts; //Массив узловых точек по направлению u

private int[] vPts; //Массив узловых точек по направлению v

/// <summary>

/// Конструктор поверхности NURBS

/// </summary>

/// <param name="k">порядок B-сплайна по направлению u</param>

/// <param name="q">порядок B-сплайна по направлению v</param>

/// <param name="p">массив контрольных точек (каждая -- три координаты и вес)</param>

public NURBS(int k, int q, List<List<Point>> p)

{

this.k = k;

this.q = q;

contrPts = p;

m = p.Count;

n = p[0].Count;

uPts = new int[m + k+1];

vPts = new int[n + q+1];

for (int j = 0; j < m + k + 1; j++)

if (j < k)

uPts[j] = 0;

else if (j >= m)

uPts[j] = m - k + 1;

else

uPts[j] = j - k + 1;

for (int i = 0; i < n + q + 1; i++)

if (i < q)

vPts[i] = 0;

else if (i >= n)

vPts[i] = n - q + 1;

else

vPts[i] = i - q + 1;

}

//формула Кокса – де Бура

private float CoxDeBoor(float p, int i, int k, int[] pts)

{

if (k == 1)

return (pts[i] - 0.001f <= p && p <= pts[i + 1] + 0.001f) ? 1 : 0;

float div1 = pts[i + k - 1] - pts[i];

float div2 = pts[i + k] - pts[i + 1];

float res1 = (div1 > 0) ? (p - pts[i]) / div1 \* CoxDeBoor(p, i, k - 1, pts) : 0;

float res2 = (div2 > 0) ? (pts[i + k] - p) / div2 \* CoxDeBoor(p, i + 1, k - 1, pts) : 0;

return res1 + res2;

}

//вычисление радиус-вектора

private Point r(float u, float v)

{

Point res = new Point(0, 0, 0, 1);

for (int i = 0; i < n; i++)

for (int j = 0; j < m; j++)

res = Point.Sum(res, Point.Mul(contrPts[i][j], CoxDeBoor(v, i, q, vPts) \* CoxDeBoor(u, j, k, uPts)));

return res;

}

//генерация координат поверхности в заданном числе узловых точек

public List<List<Point>> getSurfacePts(int k\_grid)

{

float u\_step = (m - k + 1) / (float)(k\_grid - 1);

float v\_step = (n - q + 1) / (float)(k\_grid - 1);

List<List<Point>> res = new List<List<Point>>();

float u, v; int i, j;

for (i = 0, u = 0; i < k\_grid; i++, u += u\_step)

{

List<Point> row = new List<Point>();

for (j = 0, v = 0; j < k\_grid; j++, v += v\_step)

row.Add(r(u, v));

res.Add(row);

}

return res;

}

}

}

**Form1.cs**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.IO;

using System.Diagnostics;

using System.Globalization;

namespace kp\_work

{

public partial class Form1 : Form

{

Graphics g;

int N = 0;

int oriN = 0;

float zoomFactor = 2;

public float height, width;

List<List<Point>> contrPts;

List<List<Point>> surfacePts;

TransformMatrix matrix = new TransformMatrix();

NURBS nurbs;

float xStart;

float yStart;

float xEnd;

float yEnd;

Timer timerAnim = new Timer();

public Form1()

{

InitializeComponent();

timerAnim.Interval = 40;

timerAnim.Tick += TimerAnim\_Tick;

ResizeRedraw = true;

this.MouseWheel += new System.Windows.Forms.MouseEventHandler(this.form\_MouseWheel);

}

//автоанимация

private void TimerAnim\_Tick(object sender, EventArgs e)

{

matrix.Mult(TransformMatrix.RotationY(6));

this.Refresh();

}

//прокрутка колёсиком мыши

private void form\_MouseWheel(object sender, System.Windows.Forms.MouseEventArgs e)

{

if (e.Delta > 0)

zoomFactor \*= 1.2f;

else

zoomFactor /= 1.2f;

this.Refresh();

}

public void getPoints()

{

var filePath = string.Empty;

filePath = @"\\Mac\Home\Documents\Computer Graphics\cp\test1.txt";

ParseFile(filePath);

RecomputeSurface();

this.Refresh();

}

//"распознавание" файла

private void ParseFile(string filePath)

{

using (StreamReader reader = new StreamReader(filePath))

{

CultureInfo ci = CultureInfo.InvariantCulture;

string[] parsed;

int n, m;

parsed = reader.ReadLine().Split();

n = int.Parse(parsed[0]);

m = int.Parse(parsed[1]);

oriN = Math.Max(n, m);

N = 2 \* oriN;

contrPts = new List<List<Point>>();

for (int i = 0; i < n; i++)

{

List<Point> row = new List<Point>();

parsed = reader.ReadLine().Trim().Split(" \t".ToCharArray(), StringSplitOptions.RemoveEmptyEntries);

for (int j = 0; j < m; j++)

{

row.Add(new Point(

float.Parse(parsed[j \* 3], ci),

float.Parse(parsed[j \* 3 + 1], ci),

float.Parse(parsed[j \* 3 + 2], ci), 1

));

}

contrPts.Add(row);

}

labelN.Text = N.ToString();

nurbs = new NURBS(4, 4, contrPts);

}

}

//перевычисление поверхностей при изменении её параметров

private void RecomputeSurface()

{

surfacePts = nurbs.getSurfacePts(N);

this.Refresh();

}

private void Form1\_Shown(object sender, EventArgs e)

{

getPoints();

labelN.Text = N.ToString();

}

private void Form1\_Paint(object sender, PaintEventArgs e)

{

Graphics g = e.Graphics;

g.Clear(Color.Black);

g.SmoothingMode = System.Drawing.Drawing2D.SmoothingMode.AntiAlias;

this.height = this.Height - 100;

this.width = this.Width;

matrix.matrix[3, 3] = (int)(Math.Min(width, height) / zoomFactor);

Pencil.DrawSurface(surfacePts, matrix, g, width, height);

}

//перетаскивание

private void Form1\_MouseDown(object sender, MouseEventArgs e)

{

xStart = xEnd = e.X;

yStart = yEnd = e.Y;

}

//слайдер

private void trackBar1\_ValueChanged(object sender, EventArgs e)

{

N = 2\*trackBarN.Value\*oriN;

labelN.Text = N.ToString();

RecomputeSurface();

}

//загрузить

private void загрузитьToolStripMenuItem\_Click(object sender, EventArgs e)

{

using (OpenFileDialog openFileDialog = new OpenFileDialog())

{

openFileDialog.Filter = "txt files (\*.txt)|\*.txt|All files (\*.\*)|\*.\*";

openFileDialog.FilterIndex = 2;

openFileDialog.RestoreDirectory = true;

if (openFileDialog.ShowDialog() == DialogResult.OK)

{

string filePath = openFileDialog.FileName;

ParseFile(filePath);

RecomputeSurface();

this.Refresh();

}

}

}

//включить/отключить автоанимацию

private void checkBox1\_CheckedChanged(object sender, EventArgs e)

{

if (checkBox1.Checked)

timerAnim.Start();

else

timerAnim.Stop();

}

//выход

private void button1\_Click(object sender, EventArgs e)

{

this.Close();

}

//сохранить

private void сохранитьToolStripMenuItem\_Click(object sender, EventArgs e)

{

CultureInfo ci = CultureInfo.InvariantCulture;

using (SaveFileDialog saveFileDialog = new SaveFileDialog())

{

saveFileDialog.Filter = "txt files (\*.txt)|\*.txt|All files (\*.\*)|\*.\*";

saveFileDialog.FilterIndex = 2;

saveFileDialog.RestoreDirectory = true;

if (saveFileDialog.ShowDialog() == DialogResult.OK)

{

string filePath = saveFileDialog.FileName;

TextWriter txt = new StreamWriter(filePath);

int n = contrPts.Count, m = contrPts[0].Count;

txt.WriteLine("{0} {1}", n, m);

for (int i = 0; i < n; i++)

{

for (int j = 0; j < m; j++)

{

txt.Write("{0} {1} {2} ", contrPts[i][j].x.ToString(ci), contrPts[i][j].y.ToString(ci), contrPts[i][j].z.ToString(ci));

}

txt.WriteLine();

}

txt.Close();

}

}

}

//сгенерировать

private void сгенерироватьToolStripMenuItem\_Click(object sender, EventArgs e)

{

Random rnd = new Random((int)DateTime.Now.Ticks);

int n = rnd.Next(10, 20);

int m = n;

oriN = Math.Max(n, m);

N = 2 \* oriN;

contrPts = new List<List<Point>>();

float dz = rnd.Next(1, 5);

for (int i = 0; i < n; i++)

{

List<Point> row = new List<Point>();

for (int j = 0; j < m; j++)

{

row.Add(new Point((float)(i - n / 2), (float)j - m / 2, (float)rnd.NextDouble() \* dz - dz / 2, 1));

}

contrPts.Add(row);

}

labelN.Text = N.ToString();

nurbs = new NURBS(4, 4, contrPts);

RecomputeSurface();

this.Refresh();

}

//двигаем мышкой

private void Form1\_MouseMove(object sender, MouseEventArgs e)

{

if (e.Button == MouseButtons.Left)

{

xEnd = e.X;

yEnd = e.Y;

float dX = xEnd - xStart;

float dY = yEnd - yStart;

matrix.Mult(TransformMatrix.RotationY(dX));

matrix.Mult(TransformMatrix.RotationX(dY));

xStart = xEnd;

yStart = yEnd;

this.Refresh();

}

}

}

}

**Point4.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace kp\_work

{

public class Point

{

public float x;

public float y;

public float z;

public float w;

public Point(float x, float y, float z, float w)

{

this.x = x;

this.y = y;

this.z = z;

this.w = w;

}

public Point(Point p)

{

this.x = p.x;

this.y = p.y;

}

public static Point Sum(Point v1, Point v2)

{

return new Point(v1.x + v2.x, v1.y + v2.y, v1.z + v2.z, v2.w);

}

public static Point Sub(Point v1, Point v2)

{

return new Point(v2.x - v1.x, v2.y - v1.y, v2.z - v1.z, v2.w);

}

public static Point Mul(Point v1, float m)

{

return new Point(v1.x \* m, v1.y \* m, v1.z \* m, v1.w);

}

public Point Scale()

{

return new Point(x \* w, y \* w, z \* w, 1);

}

}

}

**Pencil.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace kp\_work

{

public class Pencil

{

Pencil()

{

}

public static void DrawList(List<Point> points, TransformMatrix matrix, System.Drawing.Graphics canvas, float width, float height)

{

List<Point> transformed = new List<Point>();

for (int i = 0; i < points.Count; i++)

{

transformed.Add(matrix.Transform(points[i]).Scale());

}

float xCenter = width / 2;

float yCenter = height / 2;

for (int i = 0; i < transformed.Count - 1; i++)

{

Point p1 = transformed[i];

Point p2 = transformed[i + 1];

canvas.DrawLine(System.Drawing.Pens.Red,

xCenter + p1.x, yCenter - p1.y, xCenter + p2.x, yCenter - p2.y);

}

}

public static void DrawSurface(List<List<Point>> surface, TransformMatrix matrix, System.Drawing.Graphics canvas, float width, float height)

{

List<List<Point>> points = new List<List<Point>>();

for (int i = 0; i < surface.Count; i++)

{

List<Point> tmp = new List<Point>();

List<Point> current = surface[i];

for (int j = 0; j < surface[i].Count; j++)

{

tmp.Add(matrix.Transform(current[j]).Scale());

}

points.Add(tmp);

}

float xCenter = width / 2;

float yCenter = height / 2;

for (int i = 0; i < points.Count - 1; i++)

{

List<Point> cur = points[i];

List<Point> next = points[i + 1];

for (int j = 0; j < cur.Count - 1; j++)

{

DrawRectangle(cur[j], cur[j + 1], next[j + 1], next[j], canvas, xCenter, yCenter,

System.Drawing.Color.DodgerBlue);

}

}

}

private static void DrawRectangle(Point p1, Point p2, Point p3, Point p4,

System.Drawing.Graphics g, float xCenter, float yCenter, System.Drawing.Color color)

{

System.Drawing.Pen p = new System.Drawing.Pen(color, 1.5f);

g.DrawLine(p, xCenter + p1.x, yCenter - p1.y, xCenter + p2.x, yCenter - p2.y);

g.DrawLine(p, xCenter + p2.x, yCenter - p2.y, xCenter + p3.x, yCenter - p3.y);

g.DrawLine(p, xCenter + p3.x, yCenter - p3.y, xCenter + p4.x, yCenter - p4.y);

g.DrawLine(p, xCenter + p4.x, yCenter - p4.y, xCenter + p1.x, yCenter - p1.y);

}

private static void DrawTriangle(Point p1, Point p2, Point p3, System.Drawing.Graphics g,

float xCenter, float yCenter)

{

g.DrawLine(System.Drawing.Pens.Green, xCenter + p1.x, yCenter - p1.y, xCenter + p2.x, yCenter - p2.y);

g.DrawLine(System.Drawing.Pens.Green, xCenter + p1.x, yCenter - p1.y, xCenter + p3.x, yCenter - p3.y);

g.DrawLine(System.Drawing.Pens.Green, xCenter + p3.x, yCenter - p3.y, xCenter + p2.x, yCenter - p2.y);

}

}

}

**6. Выводы**

Выполнив данный курсовой проект я закрепил и усовершенствовал знания компьютерной графики. Смог применить знания всех предыдущих ЛР по курсе, научился строить сложные параметрические поверхности и изменять их. Узнал как строить поверхности по сплайнам, а не точкам.

**ЛИТЕРАТУРА**

1. NURBS [Электронный ресурс]URL:<https://ru.wikipedia.org/wiki/NURBS>

(Дата обращения: 20.12.2021).

2. What are NURBS? [Электронный ресурс]URL: <https://www.rhino3d.com/features/nurbs/> (Дата обращения: 20.12.2021).

3. NURBS-поверхности, полигональные поверхности и поверхности с иерархическим разбиением [Электронный ресурс]URL: <https://3d.demiart.ru/book/Maya-6/Glava_04/Index02.htm> (Дата обращения: 20.12.2021).