**МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ (НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)**

**Факультет №8 «Информационных технологий и прикладной математики»**

**Кафедра вычислительной математики и программирования**

**Курсовой проект**

**по курсу**

**«Компьютерная графика»**

|  |  |
| --- | --- |
| **Студент:** | Сорокин Д.М. |
| **Группа:** | 8О-304Б |
| **Преподаватель:** | Филлипов Г.С. |
|  |  |
| **Оценка:** |  |
| **Дата:** |  |

3 курс, 5 семестр

Москва 2016

# Задание

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

4 – Линейчатая поверхность (направляющие – кубические кривые Безье 3D).

Линейчатой поверхностью называют поверхность, образованную движением прямой линии. Каждую прямую, принадлежащую этой поверхности называют прямолинейными образующими, а кривые пересекающие все прямолинейные образующие – направляющими кривыми.

# Модули программы

Программа состоит из 4 основных классов:

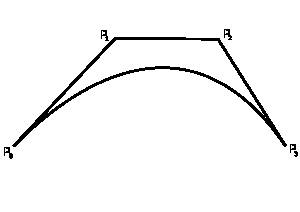
## 1,2 – Matrix.cs, MyPoint.cs

Данные классы были взяты из моих лабораторных, в них ничего не изменилось. В классе Matrix реализованы матрицы поворота, масштабирования, перемещения, а также стандартные операции над матрицами. В MyPoint реализован вектор и операции над ним.

## 3. BezierCurve.cs

В этом классе реализовано построение кривой Безье.

Кривая Безье

Она задается многоугольником и имеет математическое параметрическое представление вида:   
4   
где базис Безье или Бернштейна, или функция аппроксимации   
6   
8   
10 -  это i-тая функция базиса Бернштейна порядка n. Здесь i — порядковый номер опорной вершины, n — порядок определяющей функции базиса Бернштейна — и, следовательно, сегмента полиномиальной кривой, на единицу меньше количества точек определяющего многоугольника.   
  
Рис. 2.1. Кривая Безье и, образующий ее, многоугольник  
14 - биномиальные коэффициенты.   
16 - функция компонент векторов опорных вершин (координаты вершин многоугольника Безье).

using System;

using System.Drawing;

namespace CpCG

{

public class BezierCurve

{

private int NumberOfDrawPoints;

private MyPoint[] dataPoints;

public BezierCurve(MyPoint[] points, int NumberOfDrawPoints)

{

this.NumberOfDrawPoints = NumberOfDrawPoints;

dataPoints = points;

Invalidate();

}

public MyPoint[] DrawingPoints { get; private set; }

public MyPoint[] DataPoints //4 штуки

{

get { return dataPoints; }

set

{

dataPoints = value;

Invalidate();

}

}

public MyPoint this[int i]

{

get { return dataPoints[i]; }

set

{

dataPoints[i] = value;

Invalidate();

}

}

static public BezierCurve operator \*(Matrix m, BezierCurve b)

{

MyPoint[] dataPoints1 = new MyPoint[4];

for (int i = 0; i < 4; i++)

dataPoints1[i] = m \* b.DataPoints[i];

return new BezierCurve(dataPoints1, b.NumberOfDrawPoints);

}

public void Invalidate()

{

DrawingPoints = new MyPoint[NumberOfDrawPoints + 1];

double dt = 1f / NumberOfDrawPoints;

double t = 0f;

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

{

DrawingPoints[i] = B(t);

t += dt;

}

}

private MyPoint B(double t)

{

double c0 = (1 - t) \* (1 - t) \* (1 - t);

double c1 = (1 - t) \* (1 - t) \* 3 \* t;

double c2 = (1 - t) \* t \* 3 \* t;

double c3 = t \* t \* t;

double x = c0 \* dataPoints[0].x + c1 \* dataPoints[1].x + c2 \* dataPoints[2].x + c3 \* dataPoints[3].x;

double y = c0 \* dataPoints[0].y + c1 \* dataPoints[1].y + c2 \* dataPoints[2].y + c3 \* dataPoints[3].y;

double z = c0 \* dataPoints[0].z + c1 \* dataPoints[1].z + c2 \* dataPoints[2].z + c3 \* dataPoints[3].z;

return new MyPoint(x, y, z, 1);

}

public void Draw(Matrix preobr, Graphics g)

{

Pen pen = new Pen(Color.Aqua, 2f);

BezierCurve b;

b = preobr \* this;

for (int i = 0; i < DrawingPoints.Length - 1; i++)

{

g.DrawLine(pen, (int)b.DrawingPoints[i].x, (int)b.DrawingPoints[i].y, (int)b.DrawingPoints[i + 1].x, (int)b.DrawingPoints[i + 1].y);

}

}

public void DrawMarkers(Matrix preobr, Graphics g)

{

BezierCurve b;

b = preobr \* this;

for (int i = 0; i < 4; i++)

{

RectangleF rectangle = new RectangleF((int)b.dataPoints[i].x - 5, (int)b.dataPoints[i].y - 5, (float)10, (float)10);

g.FillEllipse(Brushes.Black, rectangle);

}

}

}

}

## 4. LinearSurface.cs

Задает линейчатую поверхность по двум кривым Безье.

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Drawing;

namespace CpCG

{

public class LinearSurface

{

public LinearSurface(BezierCurve b1, BezierCurve b2)

{

this.b1 = b1;

this.b2 = b2;

}

public void Draw(Matrix preobr, Graphics g, int dataPointCount, bool withMarkers)

{

BezierCurve tempb1, tempb2;

tempb1 = preobr \* b1;

tempb2 = preobr \* b2;

Pen pen = new Pen(Color.Green, 2f);

Pen pen1 = new Pen(Color.Red, 3f);

for (int i = 0; i < tempb1.DrawingPoints.Length; i++)

{

if (i == 0 || i == tempb1.DrawingPoints.Length - 1) g.DrawLine(pen1, (int)tempb1.DrawingPoints[i].x, (int)tempb1.DrawingPoints[i].y, (int)tempb2.DrawingPoints[i].x, (int)tempb2.DrawingPoints[i].y);

else g.DrawLine(pen, (int)tempb1.DrawingPoints[i].x, (int)tempb1.DrawingPoints[i].y, (int)tempb2.DrawingPoints[i].x, (int)tempb2.DrawingPoints[i].y);

}

b1.Draw(preobr, g);

b2.Draw(preobr, g);

if (withMarkers)

{

b1.DrawMarkers(preobr, g);

b2.DrawMarkers(preobr, g);

Pen pen2 = new Pen(Color.Blue, 1f);

for (int i = 0; i < 8; i++)

{

if (i < 3) g.DrawLine(pen2, (int)tempb1.DataPoints[i].x, (int)tempb1.DataPoints[i].y, (int)tempb1.DataPoints[i + 1].x, (int)tempb1.DataPoints[i + 1].y);

if (i > 4) g.DrawLine(pen2, (int)tempb2.DataPoints[i - 5].x, (int)tempb2.DataPoints[i - 5].y, (int)tempb2.DataPoints[i - 5 + 1].x, (int)tempb2.DataPoints[i - 5 + 1].y);

}

if (dataPointCount != 8)

{

RectangleF rectangle;

if (dataPointCount < 4)

{

rectangle = new RectangleF((int)tempb1.DataPoints[dataPointCount].x - 5, (int)tempb1.DataPoints[dataPointCount].y - 5, (float)10, (float)10);

}

else

{

rectangle = new RectangleF((int)tempb2.DataPoints[dataPointCount - 4].x - 5, (int)tempb2.DataPoints[dataPointCount - 4].y - 5, (float)10, (float)10);

}

g.FillEllipse(Brushes.Red, rectangle);

}

}

}

BezierCurve b1;

BezierCurve b2;

}

}

# Описание работы программы

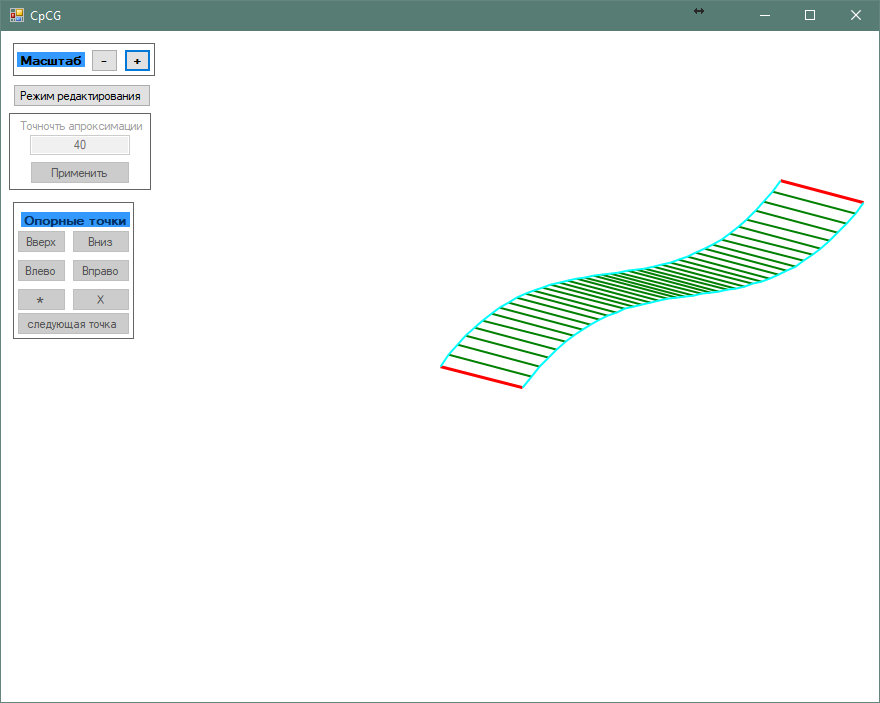
Моя программа позволяет наглядно увидеть и пощупать линейчатую поверхность, построенную по двум направляющим кривым Безье.

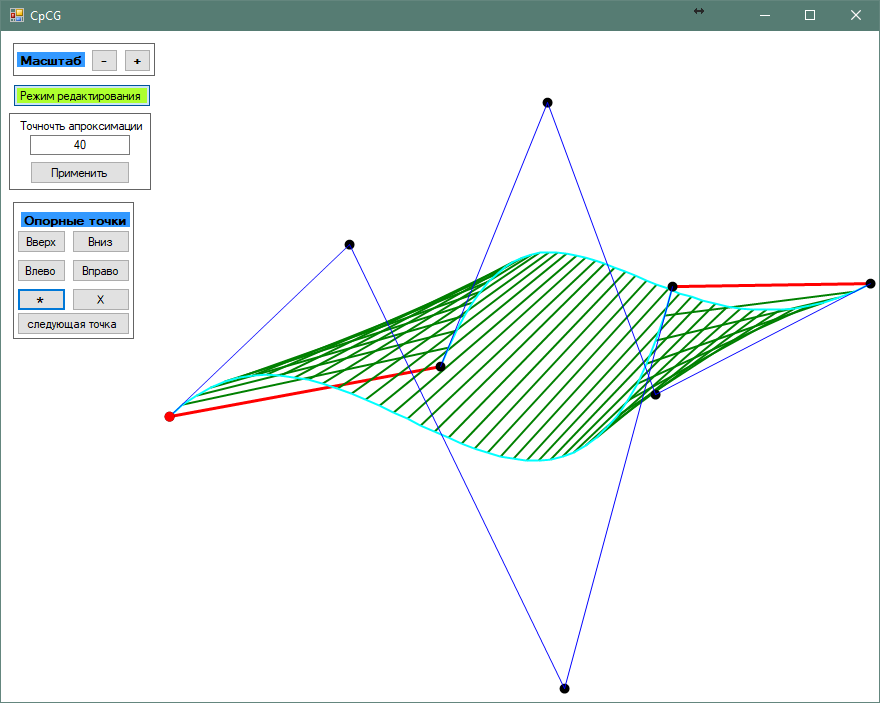
Чтобы уменьшить/увеличить масштаб, нужно соответственно нажать на кнопки -/+.

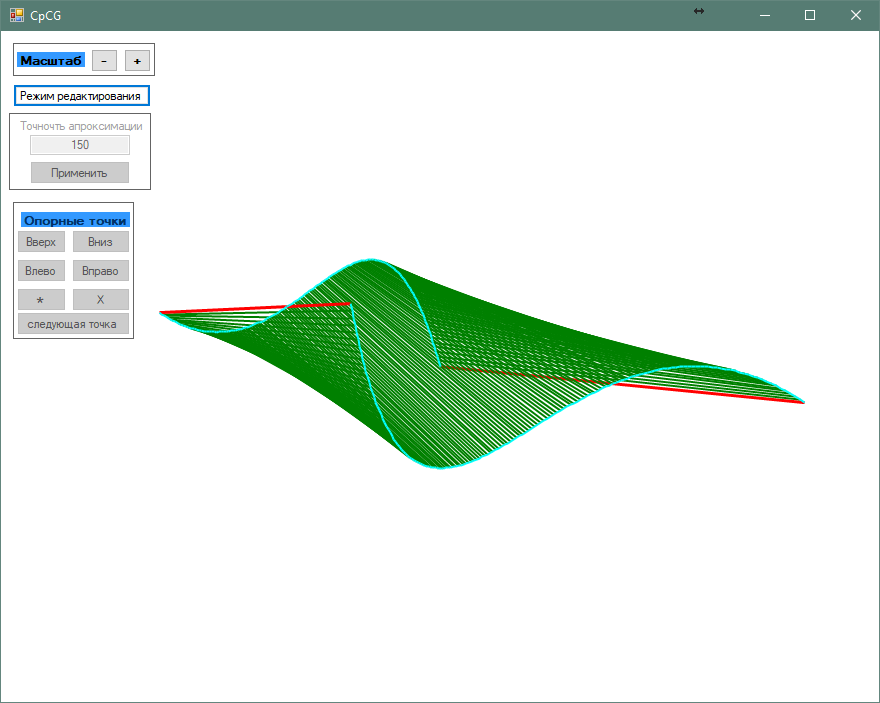
Чтобы изменить направления кривых, и саму поверхность соответственно, нужно войти в режим редактирования, выбрать точки и передвигать их с помощью кнопок на панели управления.

Можно также поменять точность аппроксимации поверхности. Она отражается в количестве прямолинейных образующих, соединяющих кривые.

# Пример работы программы







# Выводы

Этим курсовым проектом завершается увлекательное приключение по дисциплине «Компьютерная графика». Этот курс был очень интересным. Воссоздавать собственноручно все эти фигуры, поверхности, кривые на машине – просто фантастика! С помощью этого можно наглядно увидеть работу математики, просмотреть зависимости и прочие интересные закономерности.

Данный курс очень сильно помог мне разобраться во многих вещах, связанных с машинной графикой. Теперь общение с ней облегчится, и мне станет проще применять её в своих работах и проектах.