**Московский авиационный институт**

**(Национальный исследовательский университет)**

Факультет прикладной математики и физики

Кафедра вычислительной математики и программирования

**Курсовая работа**

по курсу «Компьютерная графика»

Тема: Каркасная визуализация поверхности

Студент: Сахарин Н.А

Группа: 80-308

Преподаватель: Чернышов Л.Н.

Оценка:

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.

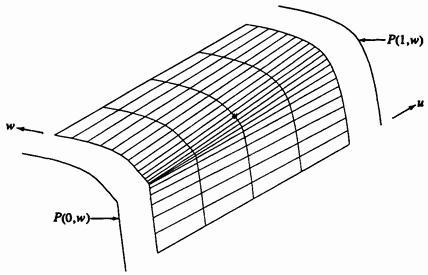
1. **Решение задачи**

Курсовая работа выполнена на языке программирования С#. При запуске программы пользователь видит окно, в котором содержится несколько дополнительных окон для масштабирования, задания точности апроксимации, и возможность изменения положения опорных точек направляющих кривых. Повороты поверхности производятся с помощью курсора, окно масштабируется.

Линейчатая поверхность в дифференциальной геометрии ― поверхность, образованная движением прямой линии. Прямые, принадлежащие этой поверхности, называются прямолинейными образующими, а каждая кривая, пересекающая все прямолинейные образующие, направляющей кривой.

Если ― радиус-вектор направляющей, a ― единичный вектор образующей, проходящей через , то радиус-вектор линейчатой поверхности есть где ― координата точки на образующей.

В терминах отображения параметрического пространства  в объектное пространство линейчатая поверхность получается с помощью линейного интерполирования между двумя известными граничными кривыми, ассоциированными с противоположными сторонами единичного квадрата в параметрическом пространстве, скажем, между кривыми  и . Поверхность задается уравнениями:



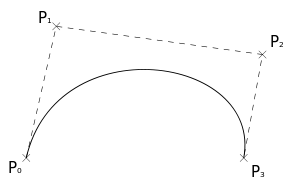
*Рисунок 1 Пример линейчатой поверхности*

Кривая Безье — параметрическая кривая, задаваемая выражением

Где P – функция компонент векторов опорных вершин, а b – базисные функции кривой Безье.

Кубические кривые Безье - В параметрической форме кубическая кривая Безье (n = 3) описывается следующим уравнением:

B(t)=(1-t)^3\*P\_0+3t(1-t)^2\*P\_1+3t^2\*(1-t)P\_2+t^3\*P\_3, где 0<t<1.

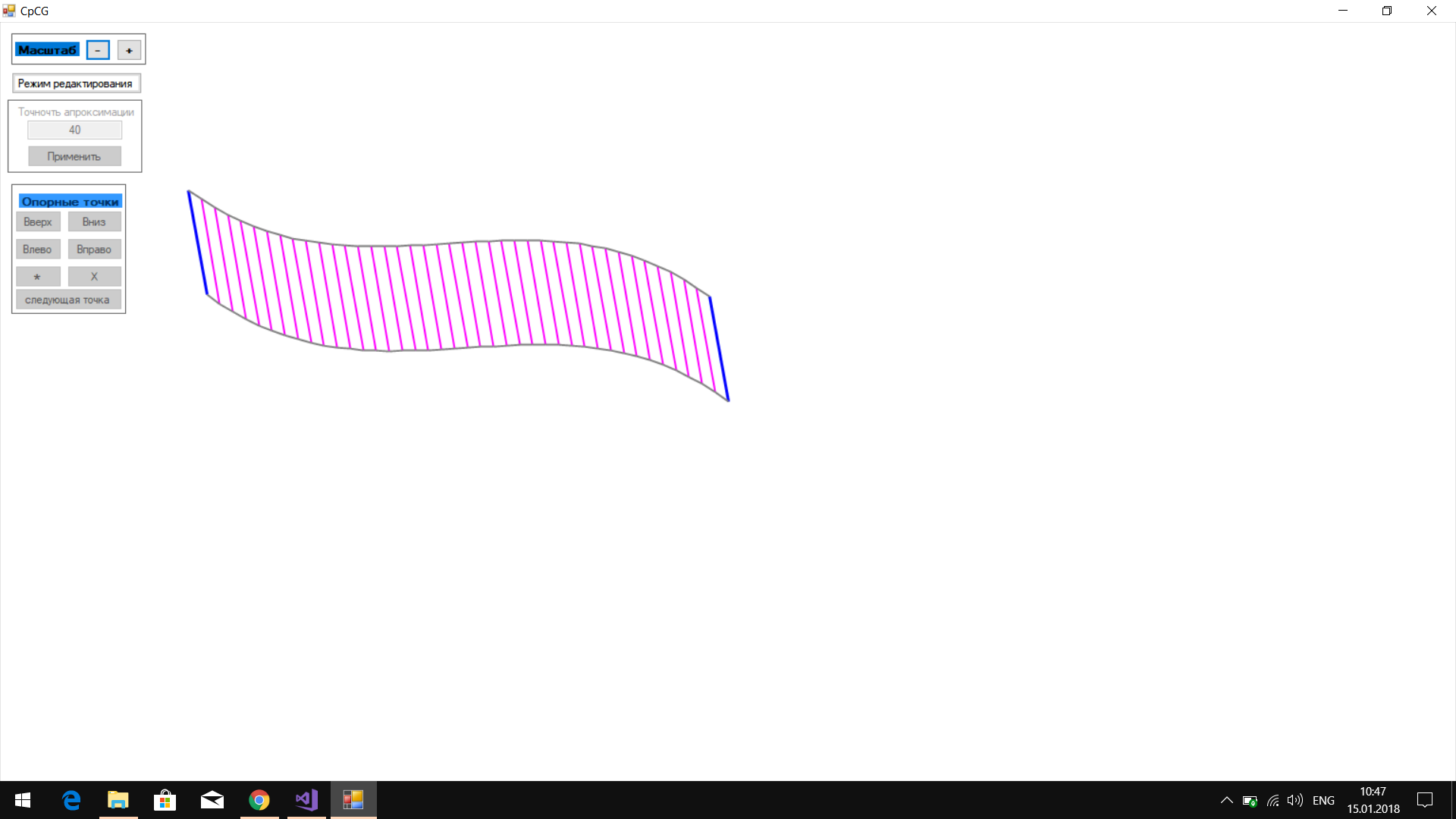


*Рисунок 2 Кубическая кривая Безье*

Благодаря простоте задания и манипуляции, кривые Безье нашли широкое применение в компьютерной графике для моделирования гладких линий. Кривая целиком лежит в выпуклой оболочке своих опорных точек. Это свойство кривых Безье с одной стороны значительно облегчает задачу нахождения точек пересечения кривых (если не пересекаются выпуклые оболочки опорных точек, то не пересекаются и сами кривые), а с другой стороны позволяет осуществлять интуитивно понятное управление параметрами кривой в графическом интерфейсе с помощью её опорных точек. Кроме того, аффинные преобразования кривой (перенос, масштабирование, вращение и др.) также могут быть осуществлены путём применения соответствующих трансформаций к опорным точкам.

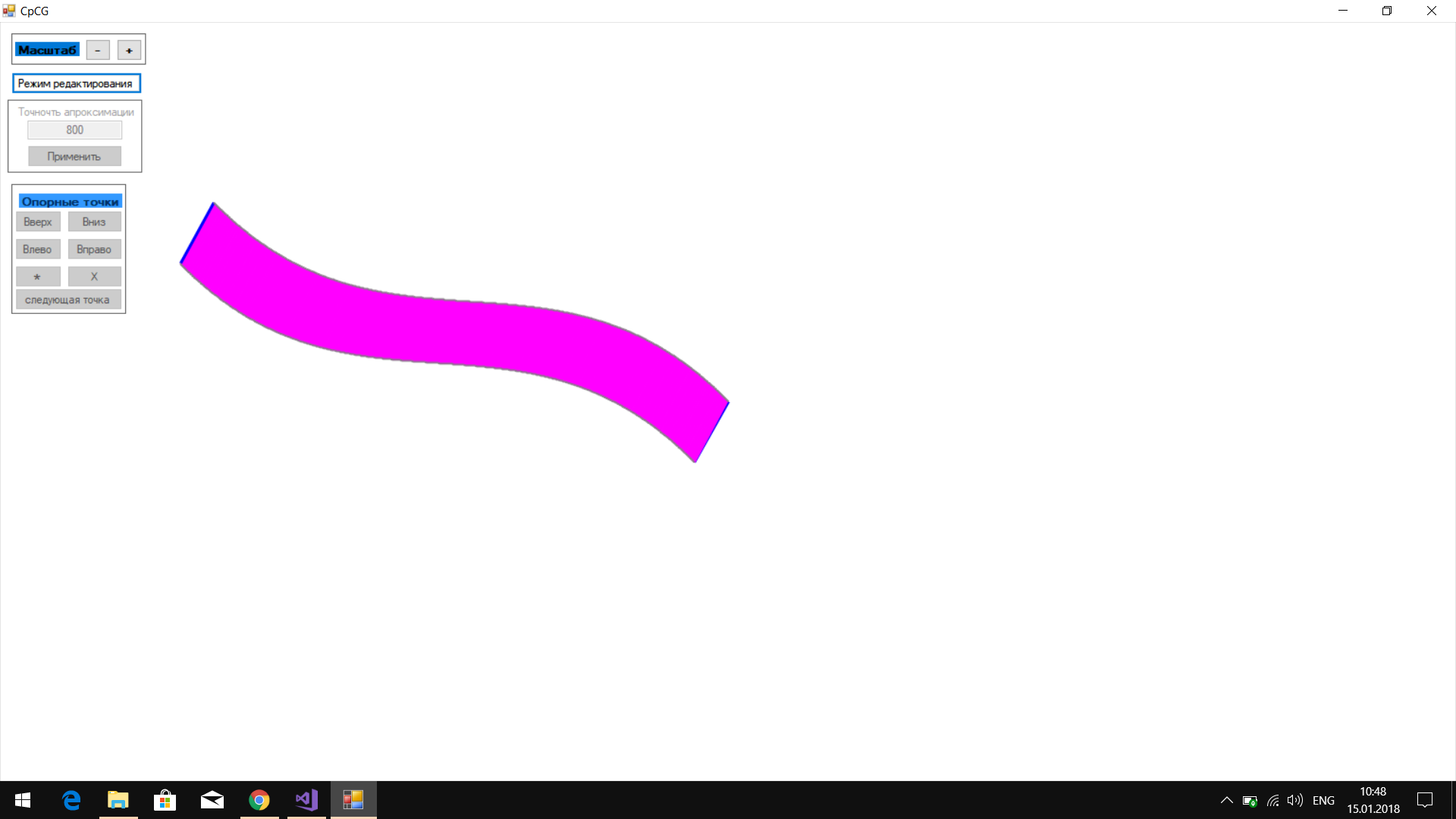
Наибольшее значение имеют кривые Безье второй и третьей степеней (квадратичные и кубические).

1. **Руководство по использованию программы**



При запуске пользователь видит данное окно.

При изменении точности апроксимации



4. **Листинг программы**

using System;

using System.Drawing;

namespace CP\_Sakharin

{

public class BezierCurve

{

private int N = 40;

private PointF[] dataPoints;

public BezierCurve(PointF[] points)

{

dataPoints = points;

Invalidate();

}

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

public PointF[] DataPoints //4 штуки

{

get { return dataPoints; }

set

{

dataPoints = value;

Invalidate();

}

}

public PointF this[int i]

{

get { return dataPoints[i]; }

set

{

dataPoints[i] = value;

Invalidate();

}

}

public void Invalidate()

{

DrawingPoints = new PointF[N + 1];

float dt = 1f / N;

float t = 0f;

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

{

DrawingPoints[i] = B(t);

t += dt;

}

}

private PointF B(float t)

{

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

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

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

float c3 = t \* t \* t;

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

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

return new PointF(x, y);

}

public void Draw(Graphics g)

{

Pen pen = new Pen(System.Drawing.SystemColors.Highlight, 2f);

g.DrawLines(pen, DrawingPoints);

}

}

}

namespace CP\_Sakharin

{

public class Marker

{

public Marker(int x, int y)

{

rectangle = new RectangleF(x - Radius / 2f, y - Radius / 2f, Radius, Radius);

}

public Action<PointF> OnDrag { get; set; }

public Action<PointF> OnMouseDown { get; set; }

public PointF Location

{

get { return new PointF(rectangle.X + Radius / 2f, rectangle.Y + Radius / 2f); }

}

public void Draw(Graphics g)

{

g.FillEllipse(Brushes.Black, rectangle);

}

public void MouseDown(MouseEventArgs e)

{

if (rectangle.Contains(e.Location))

{

drag = true;

if (OnMouseDown != null)

{

OnMouseDown(e.Location);

}

}

}

public void MouseUp()

{

drag = false;

}

public void MouseMove(MouseEventArgs e)

{

if (drag)

{

rectangle.X = e.X - Radius / 2f;

rectangle.Y = e.Y - Radius / 2f;

if (OnDrag != null)

{

OnDrag(e.Location);

}

}

}

private int Radius = 10;

private bool drag;

private RectangleF rectangle;

}

}

**Выводы**

При выполнении курсового проекта была реализована программа, которая строит каркасную модель заданной поверхности. Были изучены такие понятия как линейчатая поверхность, кубические кривые Безье. Также при выполнении проекта были получены навыки создания графических приложений на языке С#.