МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра САПР

ОТЧЕТ

по учебной практике (технологической (проектно-технологической) практике)

Тема: Разработка приложений с использованием графического интерфейса.

Студент гр. 9301 _____ Панасенко М.И.

Преподаватель _____ Калмычков В.А.

Санкт-Петербург
2021

ЗАДАНИЕ
УЧЕБНАЯ ПРАКТИКА

Студент Панасенко Е.А.

Группа 9301

Тема работы: Разработка приложений с использованием графического интерфейса.

Язык программирования: C#

Задание на практику: выполнить 6 общих ознакомительных, а также два индивидуальных задания, представляющих собой написание программ с графическим пользовательским интерфейсом.

Сроки прохождения практики: 01.07.2021 – 14.07.2021

Дата сдачи отчета:

Дата защиты отчета:

Студент(ка) _____ Панасенко М.И.

Преподаватель _____ Калмычков В.А.

АННОТАЦИЯ

Основная цель прохождения практики – приобретение навыков разработки программного обеспечения с использованием графического пользовательского интерфейса, а также умения использовать математические формулы и инструменты графических библиотек при написании программного кода.

Были разработаны приложения, позволяющие построить траекторию с движущимся по ней геометрическим объектом, а также построить фрактал до определённого, заданного пользователем, уровня.

SUMMARY

The main purpose of practice work is the purchase of software development skills using a graphical user interface and the ability to use mathematical formulas and tools of graphic libraries when developing programs.

Applications have been developed, allowing to build a path with a geometric object moving through it, and also to construct a fractal to a certain user-defined level.

Оглавление.

Цель работы

Целью практической работы является изучение и освоение базовых понятий, методов и приемов использования инструментальных средств и технологий программирования при решении практических задач с выбором различных структур данных и организацией программного графического интерфейса пользователя, закрепление и приобретение новых знаний и практических навыков программирования.

Введение

Практическая работа состоит из трех разделов, каждый из которых написан, соблюдая применение базовых практических понятий, лежащих в основе процесса разработки программного графического интерфейса пользователя. В работе ставятся и решаются такие задачи, как разработка графического интерфейса, математическая постановка задачи и её программная реализация.

Реализация каждого программного кода проводится с помощью языка программирования C#. Процесс разработки ПО включает в себя создание удобной графической оболочки с помощью системы Windows Forms.

ПЕРВЫЙ РАЗДЕЛ. ОЗНАКОМИТЕЛЬНЫЕ ЗАДАНИЯ.

Задание №1.

Формулировка задания.

Необходимо написать программу, которая копирует введенный пользователем текст и отображает его в строке выше.

Формализация задания.

Для реализации поставленной задачи необходимо создать элементы: Button, TextBox и Label. При нажатии на кнопку срабатывает событие, которое вызывает метод, помещающий текст, введённый пользователем в текстовое поле, в элемент Label.

Контрольный пример.

Примеры работы программы представлены на рис. 1.1.1 и рис. 1.1.2.

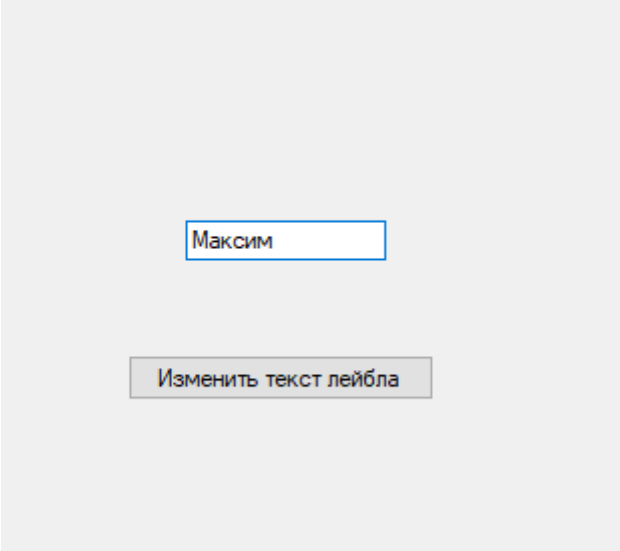

Рисунок 1.1.1-Работа программы до нажатии на кнопку.

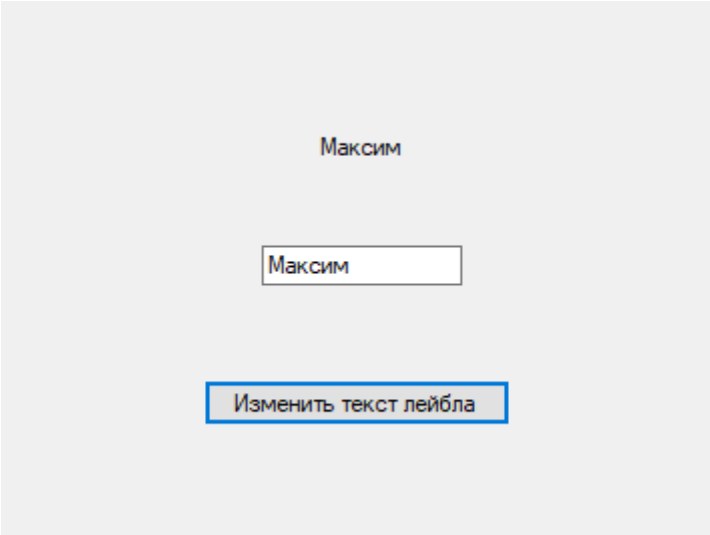

Рисунок 1.1.2-Работа программы после нажатия на кнопку.

Задание №2.

Формулировка задания.

Необходимо написать программу, которая выводит приветственное сообщение в отдельном окне.

Формализация задания.

Для реализации поставленной задачи необходимо создать элементы: Button, TextBox, Label и ToolTip. При нажатии на кнопку срабатывает событие, которое вызывает метод, при котором с помощью элемента MessageBox на экран выводится приветственное окно, содержащее в себе имя, которое ввёл пользователь ранее в текстовое поле. Также при наведении мыши на строку ввода имени всплывает подсказка.

Контрольный пример.

Примеры работы программы представлены на рис. 1.2.1 и рис. 1.2.2.

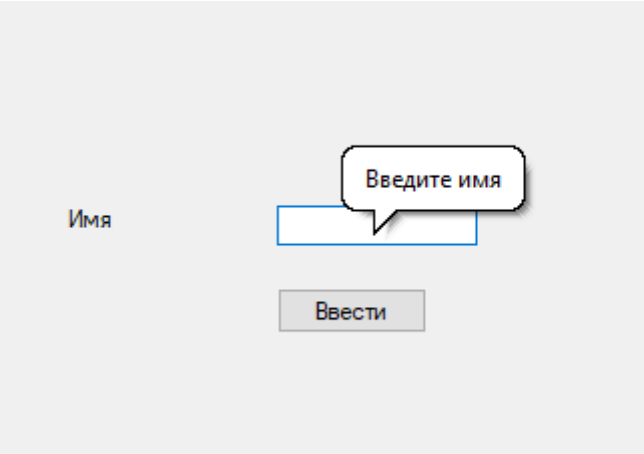

Рисунок 1.2.1-Работа программы до нажатии на кнопку.

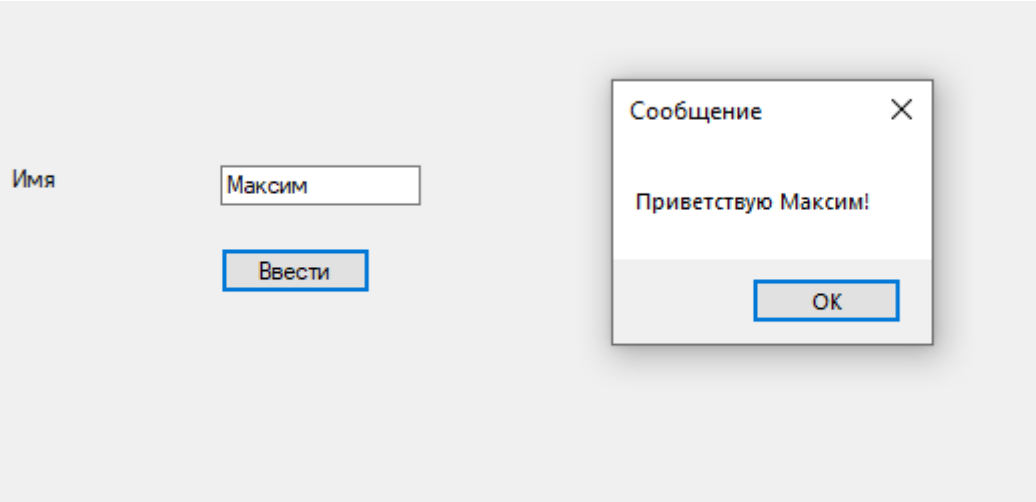

Рисунок 1.2.2-Работа программы после нажатия на кнопку.

Задание №3.

Формулировка задания.

Необходимо написать программу, которая позволяет пользователю выбирать изображение из имеющегося списка, а также выводить введенный пользователем текст в форму с помощью графических инструментов.

Формализация задания.

Для реализации поставленной задачи необходимо создать элементы: ComboBox, Label, PictureBox, Button, и TextBox. Выбор изображения пользователем осуществляется с помощью элемента ComboBox, после чего изображение загружается в элемент PictureBox. Ввод текста осуществляется в поле элемента TextBox, прорисовка этого текста осуществляется с помощью метода DrawString в поле элемента PictureBox.

Контрольный пример.

Примеры работы программы представлены на рис. 1.3.1.

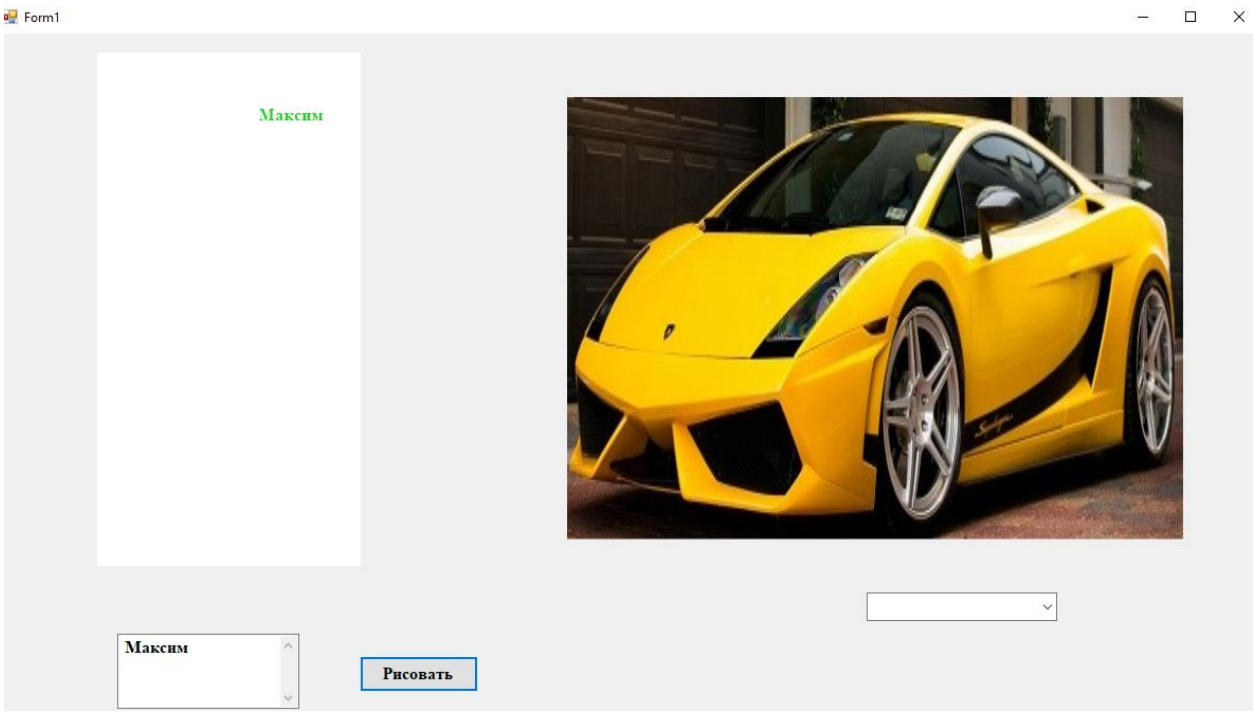

Рисунок 1.3.1-Результат работы программы.

Задание №4.

Формулировка задания.

Необходимо написать программу, которая позволяет сохранять введенный пользователем текст в текстовый файл, а также открывать другие текстовые файлы.

Формализация задания.

Для реализации поставленной задачи необходимо создать элементы: MenuStrip и TextBox. Ввод текста осуществляется в поле элемента TextBox, панель меню реализуется с помощью элемента MenuStrip.

Контрольный пример.

Примеры работы программы представлены на рис. 1.4.1.

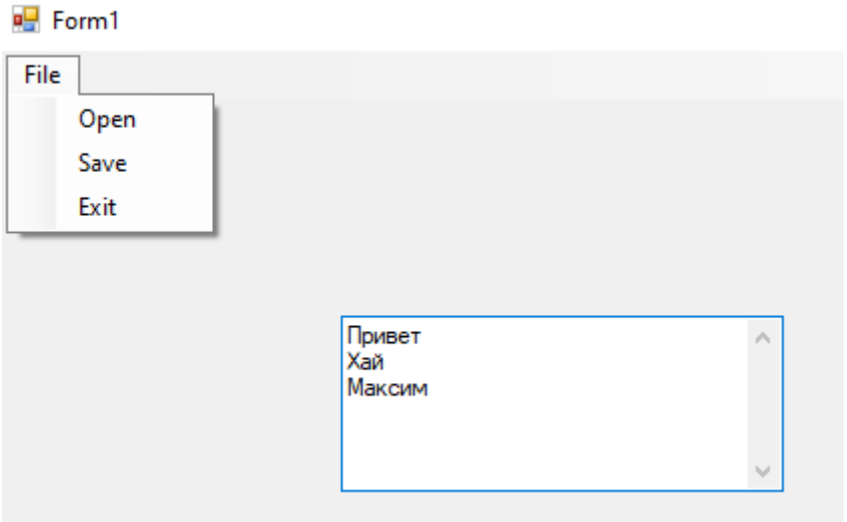

Рисунок 1.4.1-Результат работы программы.

Задание №5.

Формулировка задания.

Необходимо написать программу, которая позволяет нарисовать линию, треугольник и эллипс. Также программа позволяет залить фигуру цветом.

Формализация задания.

Для реализации поставленной задачи необходимо создать элементы: PictureBox, ComboBox и TextBox. Рисование производится в поле элемента PictureBox, координаты вводятся в поля элементов TextBox. Рисование фигур осуществляется с помощью методов DrawLine и DrawEllipse, заливка эллипса происходит с помощью метода FillEllipse.

Контрольный пример.

Примеры работы программы представлены на рис. 1.5.1.

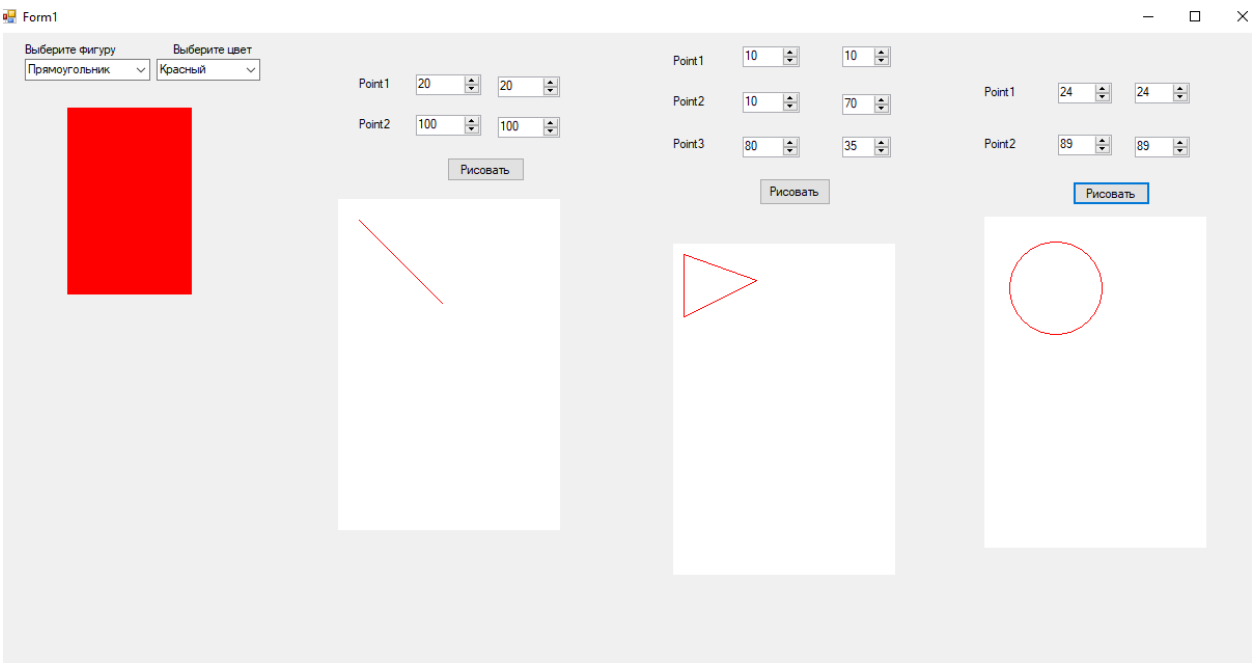

Рисунок 1.5.1-Результат работы программы.

Задание №6.

Формулировка задания.

Необходимо написать программу, в ходе работы которой будет вызываться событие при наведении курсора на элемент области.

Формализация задания.

Для реализации поставленной задачи необходимо создать элемент Label. На форме будет расположен элемент Label, имеющий событие MouseHover. При вызове события на экране с помощью элемента MessageBox будет появляться окно с текстовым сообщением.

Контрольный пример.

Примеры работы программы представлены на рис. 1.6.1.

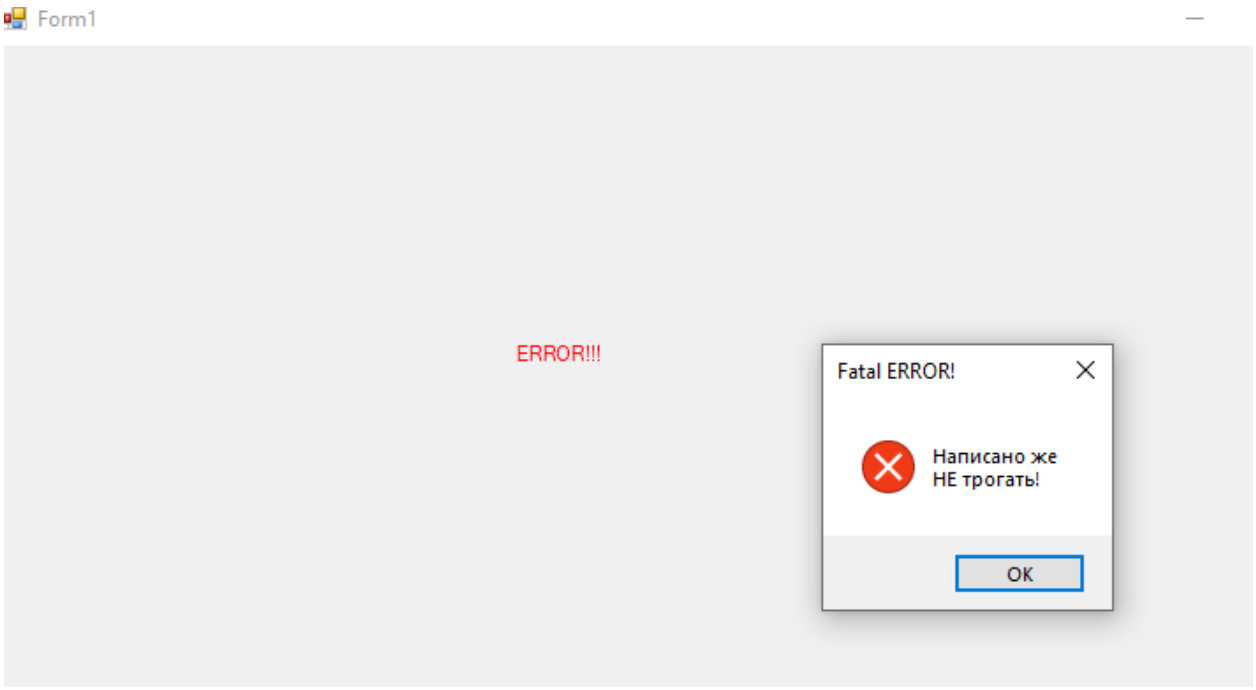

Рисунок 1.6.1-Результат работы программы

Задание №7.

Формулировка задания.

Необходимо написать программу, в ходе работы которой будет формироваться траектория для движения простого геометрического объекта.

Формализация задания.

Для реализации поставленной задачи необходимо создать элемент Button и pictureBox. Создается массив точек в зависимости от заданного количества шагов. Координаты этих точек формируются с помощью уравнения кривой, которая представляет собой траекторию. Точки соединяются между собой линиями с помощью метода DrawLines. В каждый момент времени заданный геометрический объект (в данном случае – окружность) располагается на кривой с центром в одной из сформированных точек. При каждом добавлении точек в массив изображение перестраивается, создавая эффект движения фигуры по траектории. Изображение эллипса формируется с помощью метода DrawEllipse.

Контрольный пример.

Примеры работы программы представлены на рис. 1.7.1.

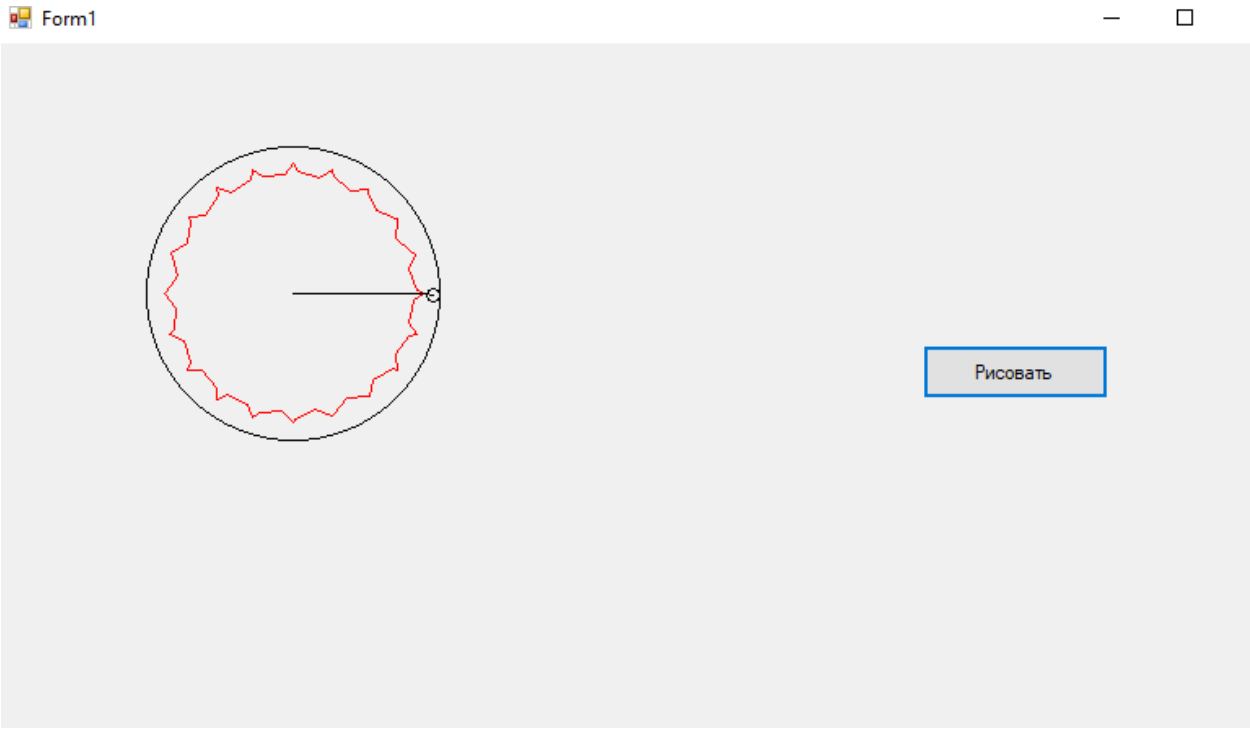

Рисунок 1.7.1-Результат работы программы

ВТОРОЙ РАЗДЕЛ. ИНДИВИДУАЛЬНОЕ ЗАДАНИЕ №1.

Формулировка задания.

Разработать приложение, суть которого заключается в движении геометрического объекта по заданной траектории. Геометрические и визуальные параметры объекта задаются пользователем.

Траектория движения объекта: косинус

Движущийся объект: ромб (цвет заливки задаётся пользователем)

Математическая постановка задачи.

Движение заданного объекта задается косинусом.

Косинус (траектория) задается уравнением: $y = \cos(x)$.

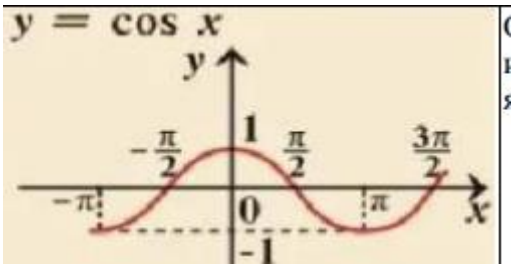

Рисунок 2.1 – График функции косинуса.

Для вычисления точек берутся значения X в интервале от 0 до 600 с определенным шагом. Все точки интервала подставляются в уравнение и полученные координаты записываются в массив точек.

Описание пользовательского интерфейса.

Программа имеет три кнопки:

1. «Рисовать» - запускает программу и рисует движение объекта по траектории.

Также пользователь может задавать различные значения параметров и выбирать цвет траектории и объекта.

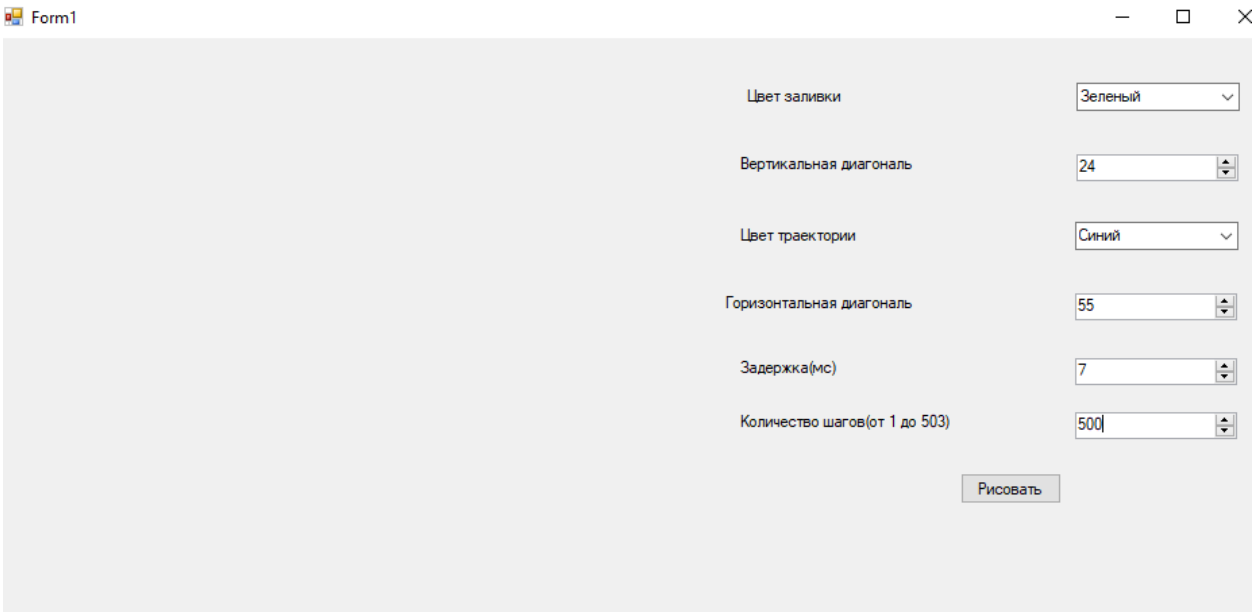

Рисунок 2.2 – Интерфейс программы.

Описание работы программы.

Исходя из заданного пользователем количества шагов создаётся массив точек, координаты которых формируются с помощью уравнения косинуса, которое представляет собой траекторию. Точки соединяются между собой линиями с помощью метода DrawLines. В каждый момент времени заданный геометрический объект (ромб) располагается на графике с крайней левой точкой в одной из сформированных точек. Через определённый промежуток времени в массив добавляется новая точка, после чего изображение перестраивается, создавая эффект движения фигуры по траектории. Изображение ромба формируется с помощью метода FillPoligon. Цвет заливки фигуры задаётся с помощью элемента ComboBox.

Контрольный пример.

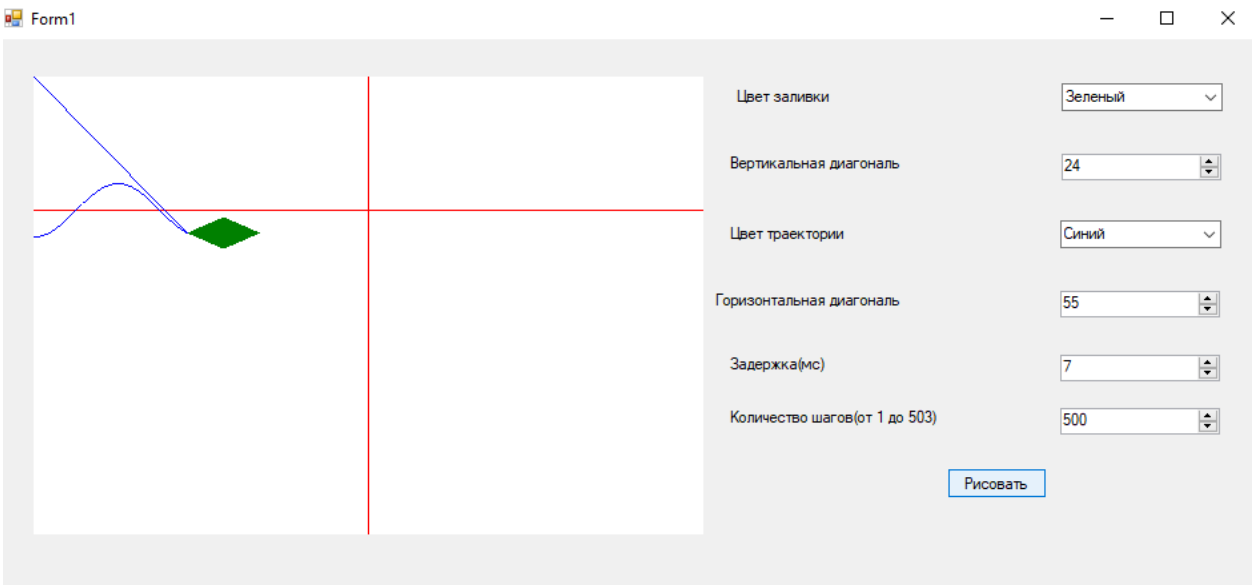

Рисунок 2.3 – Результат работы программы.

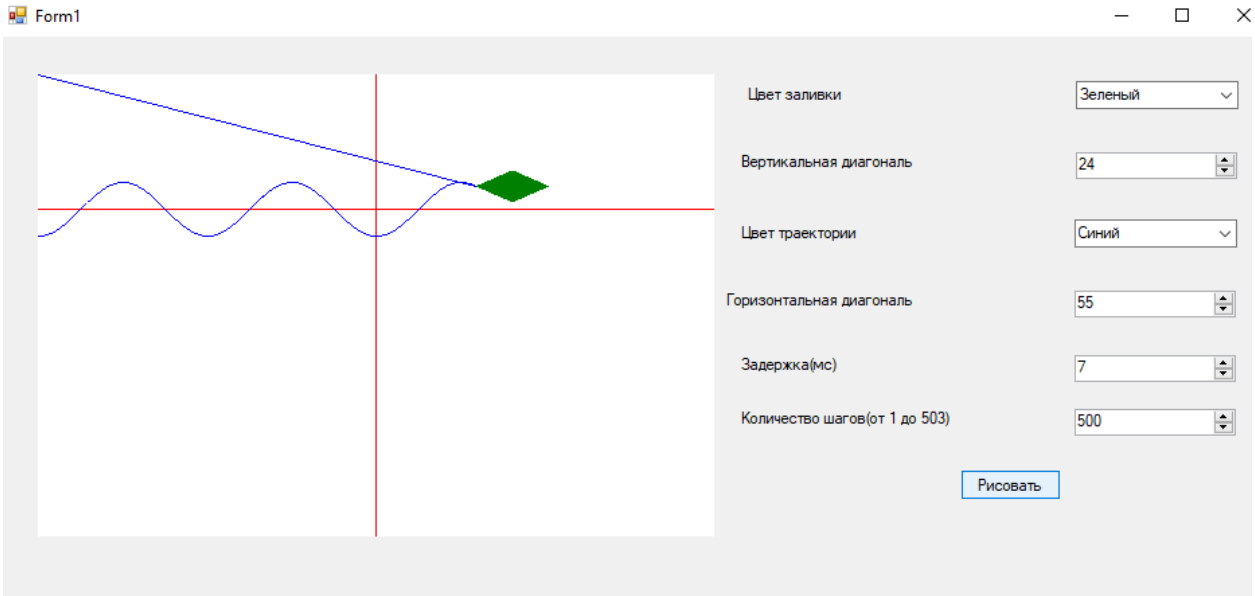

Рисунок 2.4 – Пример работы программы.

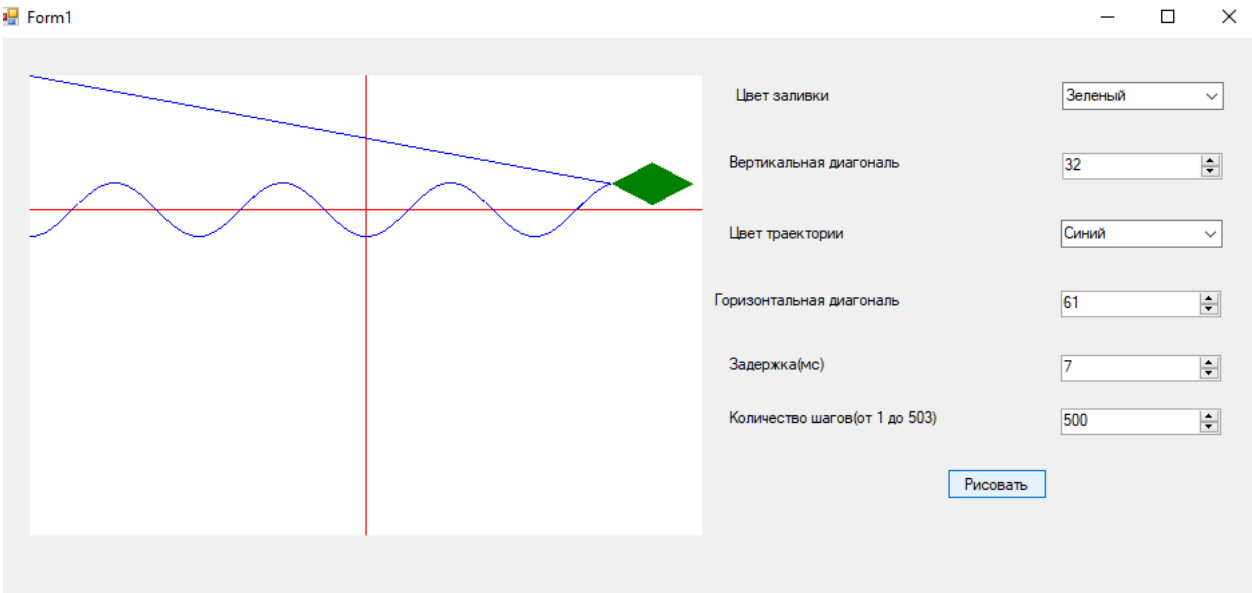

Рисунок 2.5-Пример работы программы.

Текст программы.

Текст программы представлен в приложении 1.

ТРЕТИЙ РАЗДЕЛ. ИНДИВИДУАЛЬНОЕ ЗАДАНИЕ №2.

Формулировка задания.

Разработать приложение, в ходе выполнения которого строятся определенного уровня n фракталов, а также дерево уровней. n фракталов означает прорисовку на одном изображении в разных его участках сразу всех получающихся на разных шагах (от 1 до n-1) фракталов, начиная с базовой образующей под номером 0. Уровень, до которого необходимо построить фрактал, задается пользователем (от 1 до 6).

Фрактал: Треугольники Серпинского на квадрате

Математическая постановка задачи.

Середины сторон равностороннего треугольника T0 соединяются отрезками. Получаются 4 новых треугольника. Из исходного треугольника удаляется внутренность срединного треугольника. Получается множество T1, состоящее из 3 оставшихся треугольников «первого ранга». Поступая точно так же с каждым из треугольников первого ранга, получим множество T2, состоящее

из 9 равносторонних треугольников второго ранга.

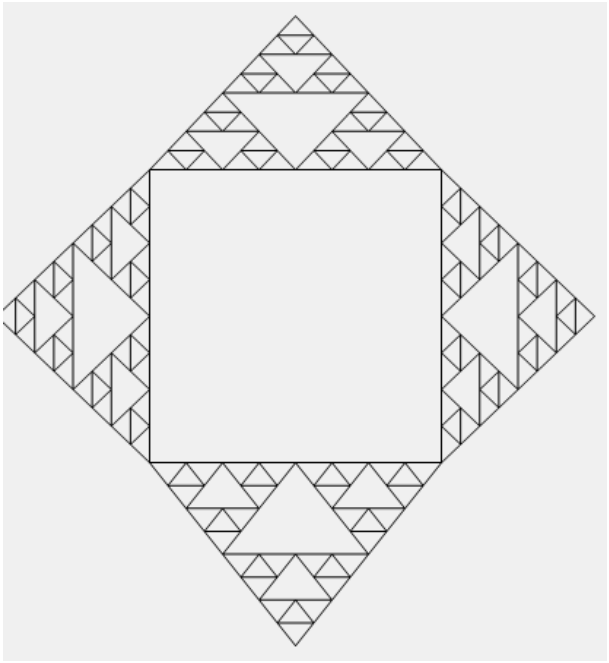

Рисунок 3.1 – Четвертый шаг построения треугольника Серпинского

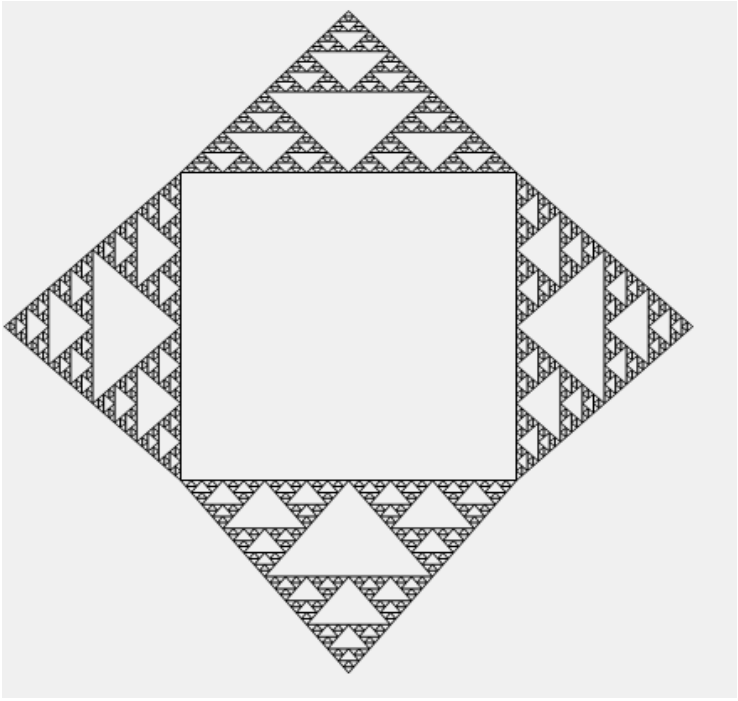

Рисунок 3.2 - Шестая стадия треугольника Серпинского.

В программе построение кривой реализовано при помощи рекурсии и двоичного дерева.

Описание пользовательского интерфейса.

Пример интерфейса программы при открытии представлен на рис. 3.5.

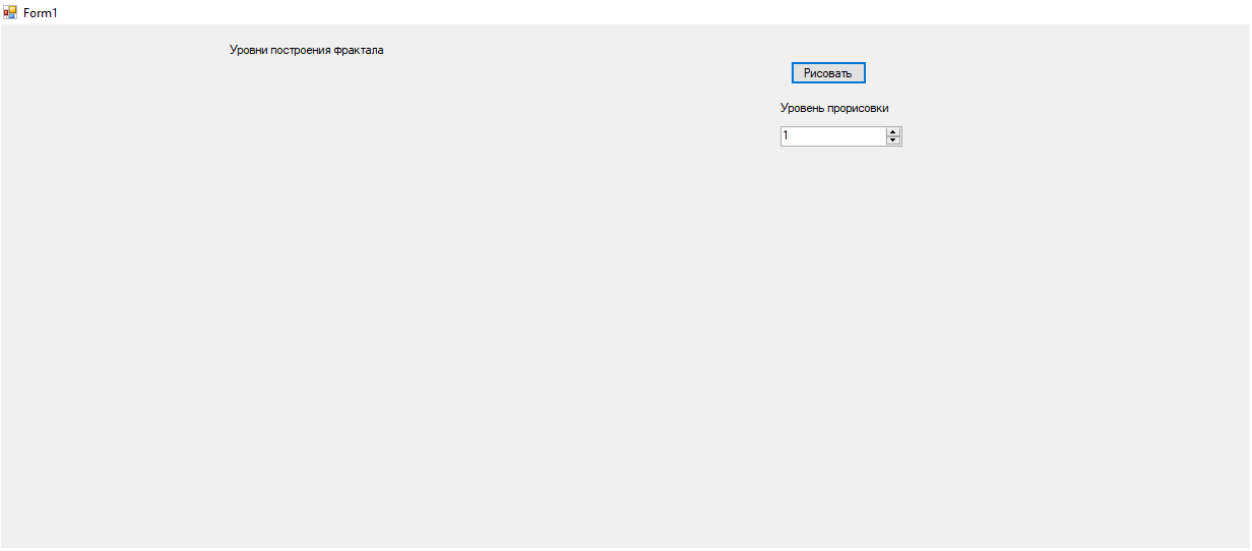

Рисунок 3.4 – Пример интерфейса программы.

Пользователю необходимо выбрать уровень построения фрактала от 1 до 6 и нажать на кнопку «Рисовать». Затем нарисуется дерево уровней и фрактал.

Описание работы программы.

При нажатии на кнопку “Рисовать” сначала идет формирование фрактала в памяти(структуре дерево). Построение фрактала осуществляется рекурсивно. На каждом этапе формируется по три новых координат по заданным правилам. Построение прямых происходит при помощи метода AddLine.

Далее строится дерево уровней фрактала. При построении дерева используются методы DrawLine, FillEllipse. Для построения очередного уровня дерева необходимо знать координаты листа предыдущего уровня, а также длину ребра.

Контрольный пример.

Примеры работы программы при разных входных данных представлены на рисунках 3.6-3.8.

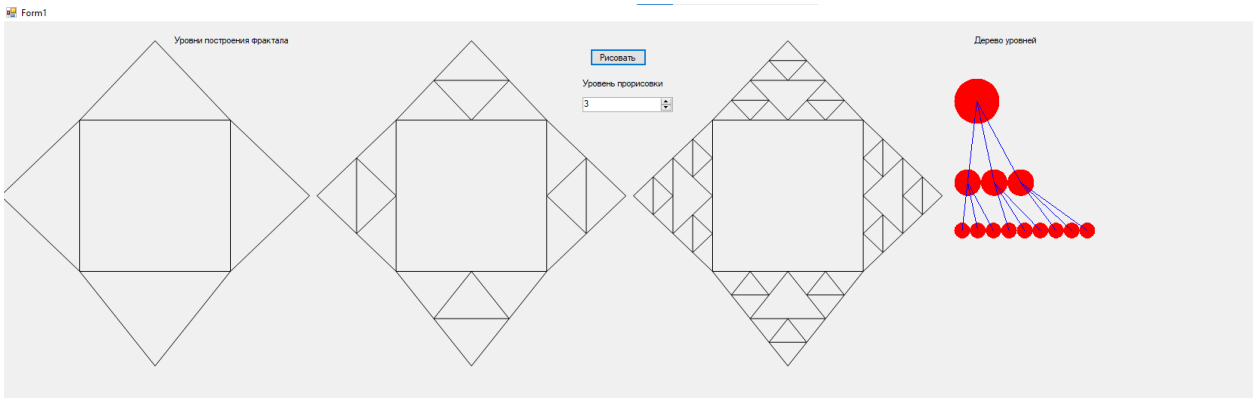

Рисунок 3.5 – Пример работы программы при выборе 3 уровня факториала.

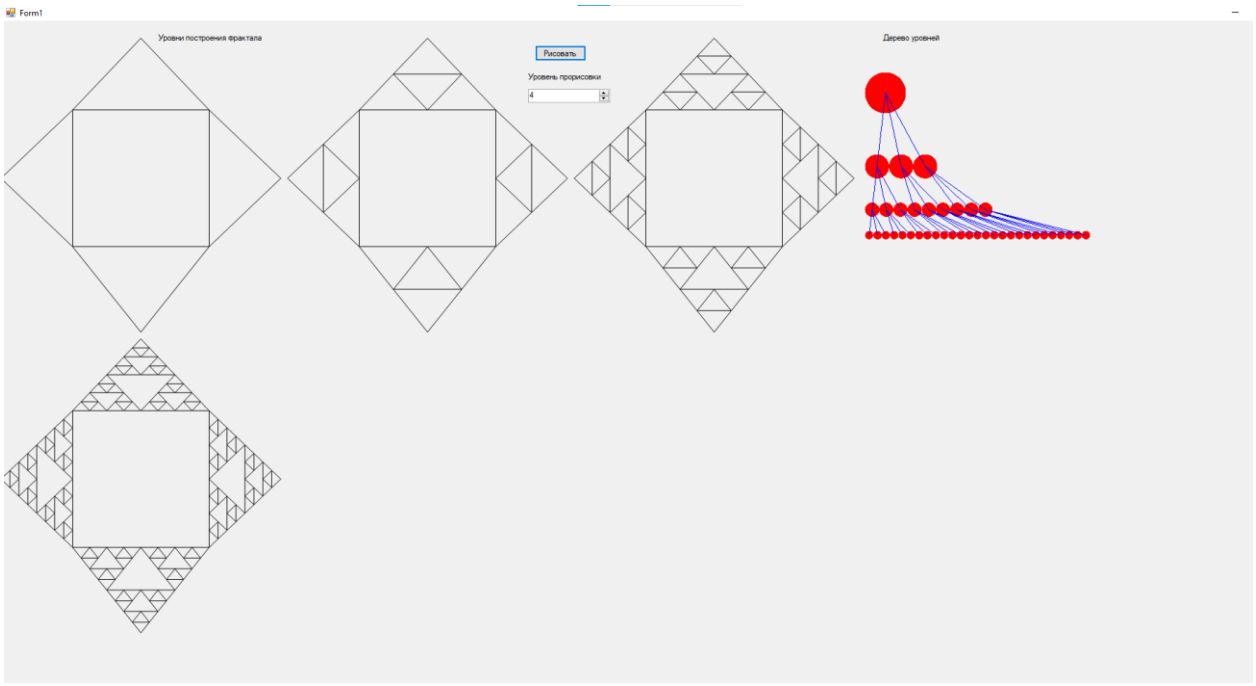

Рисунок 3.6 – Пример работы программы при выборе 4 уровня факториала.

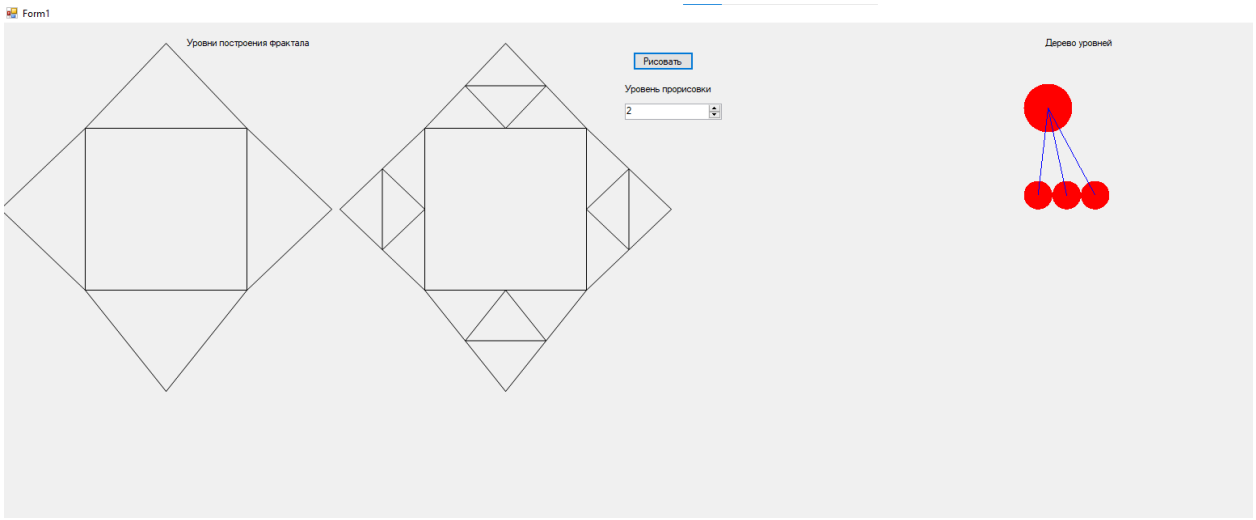

Рисунок 3.7 – Пример работы программы при выборе 2 уровня факториала.

ЗАКЛЮЧЕНИЕ.

Поставленные задачи были выполнены, в ходе прохождения практики разработаны два приложения с графическим пользовательским интерфейсом. Приложения работают корректно благодаря правильным математическим расчётам и корректному графическому отображению в окне программы при помощи соответствующих инструментов.

В первом приложении показано движение геометрического объекта по заданной траектории. Геометрические и визуальные параметры задаются пользователем. Объектом была фигура – ромб, а траекторией - косинус.

Во втором приложении строится заданное пользователем количество фракталов по уровням с визуализацией иерархии уровней в виде дерева. Количество уровней также задаются пользователем. Уровни отображаются друг за другом в порядке возрастания.

Таким образом, был приобретён опыт разработки приложений в среде Visual Studio при помощи интерфейса Windows Forms, а также работы с графической библиотекой языка C#.

ПРИЛОЖЕНИЯ.

Приложение 1.

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace IZ1
{
    public partial class Form1 : Form
    {

        public Form1()
        {
            InitializeComponent();
            comboBox1.SelectedIndex = 0;
            comboBox3.SelectedIndex = 0;
        }

        private void button1_Click(object sender, EventArgs e)
        {
            PointF[] points = new PointF[1000];
            Graphics graphics;
            float x = 0;
            float y;
            int i = 0;
            float step = (float)numericUpDown4.Value;
            float incrementX = pictureBox1.Width / step;
            while (x < pictureBox1.Size.Width)
            {
                y = (float)(20 * Math.Cos(x / 20)) + 100;
                points[i] = new PointF(x, y);
                DrawFigure(x, y);
                graphics = pictureBox1.CreateGraphics();
                DrawCoordinate();
                graphics.DrawLines(GetColorFromComboBox3(), points);
                x+= incrementX;
                i++;
                Thread.Sleep((int)numericUpDown3.Value);
            }

            y = (float)(20 * Math.Cos(x / 20)) + 100;
            points[i] = new PointF(x, y);
            DrawFigure(x, y);
            graphics = pictureBox1.CreateGraphics();
            DrawCoordinate();
            graphics.DrawLines(GetColorFromComboBox3(), points);
            x += incrementX;
            i++;
            Thread.Sleep((int)numericUpDown3.Value);
        }

        private void DrawFigure(float x, float y)
        {
            Graphics graphics = pictureBox1.CreateGraphics();
            graphics.Clear(Color.White);
```

```
            float verticalDiagonal = (float)numericUpDown1.Value;
            float horizontalDiagonal = (float)numericUpDown2.Value;

            PointF leftPoint = new PointF(x, y);
            PointF rightPoint = new PointF(leftPoint.X + horizontalDiagonal,
leftPoint.Y);
            PointF topPoint = new PointF(leftPoint.X + horizontalDiagonal / 2,
leftPoint.Y - verticalDiagonal / 2);
            PointF bottomPoint = new PointF(leftPoint.X + horizontalDiagonal / 2,
leftPoint.Y + verticalDiagonal / 2);

            graphics.FillPolygon(GetColorFromComboBox1(), new[] { leftPoint, topPoint,
rightPoint, bottomPoint });
        }

        private void DrawCoordinate()
        {
            Graphics graphics = pictureBox1.CreateGraphics();
            Point point1 = new Point(0, 100);
            Point point2 = new Point(pictureBox1.Size.Width, point1.Y);
            graphics.DrawLine(new Pen(Color.Red), point1, point2);
            point1 = new Point(pictureBox1.Size.Width / 2, 0);
            point2 = new Point(point1.X, pictureBox1.Size.Height);
            graphics.DrawLine(new Pen(Color.Red), point1, point2);
        }

        Brush GetColorFromComboBox1()
        {
            switch (comboBox1.SelectedIndex)
            {
                case 0:
                    return new SolidBrush(Color.Green);
                case 1:
                    return new SolidBrush(Color.Yellow);
                case 2:
                    return new SolidBrush(Color.Blue);
                default:
                    return new SolidBrush(Color.Black);
            }
        }

        Pen GetColorFromComboBox3()
        {
            switch (comboBox3.SelectedIndex)
            {
                case 0:
                    return new Pen(Color.Green);
                case 1:
                    return new Pen(Color.Yellow);
                case 2:
                    return new Pen(Color.Blue);
                default:
                    return new Pen(Color.Black);
            }

        }
    }
   }
```

Приложение 2.

Form1.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Drawing.Drawing2D;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace IZ2
{
    public partial class Form1 : Form
    {
        private GraphicsPath path1 = new GraphicsPath();
        private GraphicsPath path2 = new GraphicsPath();
        private GraphicsPath path3 = new GraphicsPath();
        private GraphicsPath path4 = new GraphicsPath();

        int rX = -200;

        int rY = 0;
         int x1 = -100;
         int y1 = 30;
         int x2 = 0;
         int y2 = -75;
         int x3 = 100;
         int y3 = 30;
         int x4 = 100;
         int y4 = 230;
         int x5 = 0;
         int y5 = 355;
         int x6 = -100;
         int y6 = 230;
         int x7 = 100;
         int y7 = 30;
         int x8 = 205;
         int y8 = 130;
         int x9 = 100;
         int y9 = 230;
         int x10 = -100;
         int y10 = 230;
         int x11 = -205;
         int y11 = 130;
         int x12 = -100;
         int y12 = 30;
        public Form1()
        {
            InitializeComponent();

            SetStyle(ControlStyles.AllPaintingInWmPaint |
ControlStyles.OptimizedDoubleBuffer | ControlStyles.ResizeRedraw |
ControlStyles.UserPaint, true);
        }

        protected override void OnPaint(PaintEventArgs e)
        {
            e.Graphics.Clear(BackColor);
            e.Graphics.SmoothingMode = SmoothingMode.HighQuality;
            e.Graphics.TranslateTransform(400, 100);
            e.Graphics.DrawPath(Pens.Black, path1);
```

```
    e.Graphics.DrawPath(Pens.Black, path2);
    e.Graphics.DrawPath(Pens.Black, path3);
    e.Graphics.DrawPath(Pens.Black, path4);

    DrawTreeCircles((int)numericUpDown1.Value);


}

FractalTree GetFractalTree(PointF p1, PointF p2, PointF p3, int level)
{
    FractalTree fractalTree = new FractalTree(level);

    FractalNode fractalNode = new FractalNode(p1, p2, p3, 1);

    fractalTree.CreateFractalTree(fractalNode);

    return fractalTree;
}

private void button1_Click(object sender, EventArgs e)
{
    path1 = new GraphicsPath();
    path2 = new GraphicsPath();
    path3 = new GraphicsPath();
    path4 = new GraphicsPath();

    int level = (int)numericUpDown1.Value;

    DrawFractals(level);

    Refresh();
}

void DrawTreeCircles(int level)
{
    CircleTree circleTree = new CircleTree(level);

    Graphics graphics = pictureBox1.CreateGraphics();

    CircleNode root = new CircleNode(new PointF(0, 30), 60, 1);

    circleTree.CreateCircleTree(root);

    circleTree.DrawCircles(graphics);

    circleTree.DrawLines(graphics);
}

void DrawFractals(int level)
{
    for (int i = 1; i <= Math.Min(3, level); i++)
    {
        DrawFractal(i, rX + (i - 1) * 420, rY);
    }

    for (int i = 4; i <= level; i++)
    {
        DrawFractal(i, rX + (i - 4) * 420, rY + 440);
    }
}

void DrawFractal(int iteration, int rX, int rY)
{
```

```
            FractalTree topFractalTree = GetFractalTree(new PointF(x1 + rX, y1 + rY), new
PointF(x2 + rX, y2 + rY), new PointF(x3 + rX, y3 + rY), iteration);
            FractalTree bottomFractalTree = GetFractalTree(new PointF(x4 + rX, y4 + rY),
new PointF(x5 + rX, y5 + rY), new PointF(x6 + rX, y6 + rY), iteration);
            FractalTree rightFractalTree = GetFractalTree(new PointF(x7 + rX, y7 + rY),
new PointF(x8 + rX, y8 + rY), new PointF(x9 + rX, y9 + rY), iteration);
            FractalTree leftFractalTree = GetFractalTree(new PointF(x10 + rX, y10 + rY),
new PointF(x11 + rX, y11 + rY), new PointF(x12 + rX, y12 + rY), iteration);

            leftFractalTree.DrawFractal(path1);

            topFractalTree.DrawFractal(path2);

            rightFractalTree.DrawFractal(path3);

            bottomFractalTree.DrawFractal(path4);
        }
    }
}
```

FractalTree.cs

```
using System;
using System.Collections.Generic;
using System.Drawing;
using System.Drawing.Drawing2D;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace IZ2
{
    class FractalNode
    {
        public PointF Point1 { get; set; }
        public PointF Point2 { get; set; }
        public PointF Point3 { get; set; }
        public int Level { get; set; }
        public FractalNode LeftChild { get; set; }
        public FractalNode MidlChild { get; set; }
        public FractalNode RightChild { get; set; }

        public FractalNode(PointF point1, PointF point2, PointF point3, int level)
        {
            Point1 = point1;
            Point2 = point2;
            Point3 = point3;
            Level = level;
        }
    }

    class CircleNode
    {
        public PointF Point { get; set; }
        public float Diameter { get; set; }
        public int Level { get; set; }
        public CircleNode LeftChild { get; set; }
        public CircleNode MidlChild { get; set; }
        public CircleNode RightChild { get; set; }
        public CircleNode(PointF point1, float diameter, int level)
        {
            Point = point1;
            Diameter = diameter;
```

```
            Level = level;
        }
    }
    class FractalTree
    {
        public FractalNode Root { get; private set; }

        int level;

        public FractalTree(int level)
        {
            this.level = level;
        }

        public void DrawFractal(GraphicsPath path)
        {
            DrawNode(Root, path);
        }

        public void CreateFractalTree(FractalNode node)
        {
            Root = node;

            AddNode(node);
        }

        void DrawNode(FractalNode node, GraphicsPath path)
        {
            PointF p1 = node.Point1;
            PointF p2 = node.Point2;
            PointF p3 = node.Point3;

            path.AddLine(p1, p2);
            path.AddLine(p2, p3);
            path.CloseFigure();

            if(node.LeftChild != null)
            {
                DrawNode(node.LeftChild, path);
            }

            if(node.MidlChild != null)
            {
                DrawNode(node.MidlChild, path);
            }

            if(node.RightChild != null)
            {
                DrawNode(node.RightChild, path);
            }
        }

        void AddNode(FractalNode node)
        {
            int nextLevel = node.Level + 1;

            if (nextLevel <= level)
            {
                var p12 = new PointF(node.Point1.X + (node.Point2.X - node.Point1.X) / 2,
node.Point1.Y + (node.Point2.Y - node.Point1.Y) / 2);
                var p23 = new PointF(node.Point2.X + (node.Point3.X - node.Point2.X) / 2,
node.Point2.Y + (node.Point3.Y - node.Point2.Y) / 2);
                var p13 = new PointF(node.Point1.X + (node.Point3.X - node.Point1.X) / 2,
node.Point1.Y + (node.Point3.Y - node.Point1.Y) / 2);
```

```
                FractalNode leftChild = new FractalNode(node.Point1, p12, p13,
nextLevel);
                FractalNode midlChild = new FractalNode(p12, node.Point2, p23,
nextLevel);
                FractalNode rightChild = new FractalNode(p13, p23, node.Point3,
nextLevel);

                node.LeftChild = leftChild;
                node.MidlChild = midlChild;
                node.RightChild = rightChild;

                AddNode(leftChild);
                AddNode(midlChild);
                AddNode(rightChild);
            }
        }
    }

    class CircleTree
    {
        public CircleNode Root { get; private set; }

        int level;

        float[] lastXForLevel;

        public CircleTree(int level)
        {
            this.level = level;
            lastXForLevel = new float[level];
        }

        public void DrawCircles(Graphics graphics)
        {
            DrawCircle(Root, graphics);
        }

        public void CreateCircleTree(CircleNode node)
        {
            Root = node;

            AddNode(node);
        }

        public void DrawLines(Graphics graphics)
        {
            DrawLine(Root, graphics);
        }

        void DrawLine(CircleNode node, Graphics graphics)
        {
            if(node.LeftChild == null)
            {
                return;
            }

            graphics.DrawLine(new Pen(Color.Blue), new PointF(node.Point.X +
node.Diameter/2, node.Point.Y + node.Diameter / 2), new PointF(node.LeftChild.Point.X +
node.LeftChild.Diameter/2, node.LeftChild.Point.Y + node.LeftChild.Diameter/2));
            graphics.DrawLine(new Pen(Color.Blue), new PointF(node.Point.X +
node.Diameter / 2, node.Point.Y + node.Diameter / 2), new PointF(node.MidlChild.Point.X +
node.MidlChild.Diameter / 2, node.MidlChild.Point.Y + node.MidlChild.Diameter / 2));
```

```
            graphics.DrawLine(new Pen(Color.Blue), new PointF(node.Point.X +
node.Diameter / 2, node.Point.Y + node.Diameter / 2), new PointF(node.RightChild.Point.X
+ node.RightChild.Diameter / 2, node.RightChild.Point.Y + node.RightChild.Diameter / 2));

            DrawLine(node.LeftChild, graphics);
            DrawLine(node.MidlChild, graphics);
            DrawLine(node.RightChild, graphics);
        }

        void DrawCircle(CircleNode node, Graphics graphics)
        {
            graphics.FillEllipse(Brushes.Red, node.Point.X, node.Point.Y, node.Diameter,
node.Diameter);

            if (node.LeftChild != null)
            {
                DrawCircle(node.LeftChild, graphics);
            }

            if (node.MidlChild != null)
            {
                DrawCircle(node.MidlChild, graphics);
            }

            if (node.RightChild != null)
            {
                DrawCircle(node.RightChild, graphics);
            }
        }

        void AddNode(CircleNode node)
        {
            int nextLevel = node.Level + 1;

            if (nextLevel <= level)
            {
                float lastX = lastXForLevel[node.Level - 1];
                float newDiameter = node.Diameter / (float)1.7;
                CircleNode leftChild = new CircleNode(new PointF(lastX, node.Point.Y + 2
* node.Diameter), newDiameter, nextLevel);
                lastX += newDiameter;
                CircleNode midlChild = new CircleNode(new PointF(lastX, node.Point.Y + 2
* node.Diameter), newDiameter, nextLevel);
                lastX += newDiameter;
                CircleNode rightChild = new CircleNode(new PointF(lastX, node.Point.Y + 2
* node.Diameter), newDiameter, nextLevel);
                lastX += newDiameter;
                lastXForLevel[node.Level - 1] = lastX;

                node.LeftChild = leftChild;
                node.MidlChild = midlChild;
                node.RightChild = rightChild;

                AddNode(leftChild);
                AddNode(midlChild);
                AddNode(rightChild);
            }
        }
    }
}
```