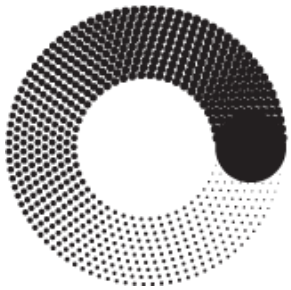
МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«МОСКОВСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»



московский политех

Кафедра СМАРТ технологий

Лабораторная работа №1
«Использование графических возможностей C# для визуализации данных стохастических процессов»

По дисциплине «Технологии визуализации данных систем управления»

| | |
|---|---|
| Группа | 201-325 |
| Студент | Холодилов И.В. |
| Дата | 11.05.2023 |
| Преподаватель | Идиатулов Т.Т. |

2023

## Цель работы

Разработать приложение по генерации стохастических данных с заданным профилем распределения и визуализации распределения случайных величин.

## Задачи

- Подготовить приложение на языке C# для статистической обработки и визуализации собранных наборов данных

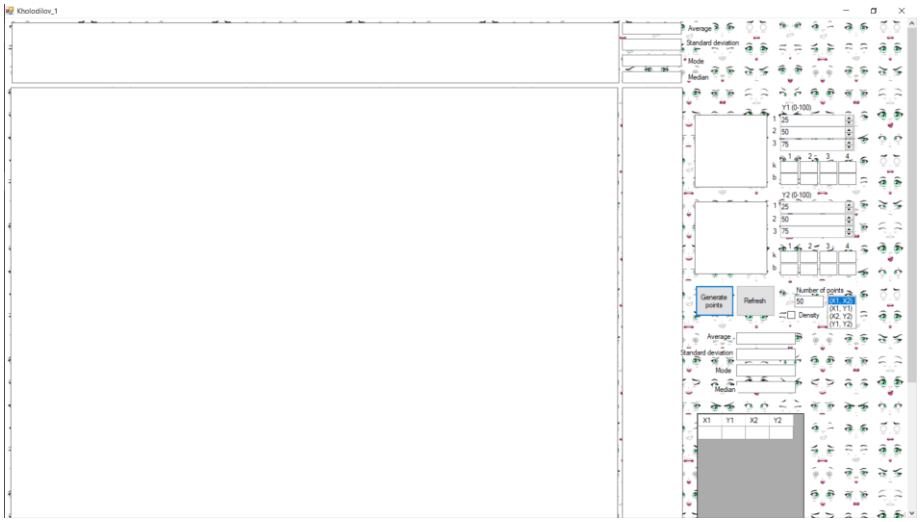


*Рисунок 1 - Form*

- Реализовать генерацию заданного (через текстовое поле) количества случайных точек (X1, X2), где X1 и X2 – равномерно распределенные случайные величины на диапазоне [0 ÷ 1]. Подготовить функционал для настройки профиля преобразования (пересчета) двух случайных величин (Y1, Y2) из равномерно распределенных случайных величин (X1, X2). Реализовать отрисовку наборов данных в виде облака точек, с возможностью выбора пар параметров, используемых как координаты точек.

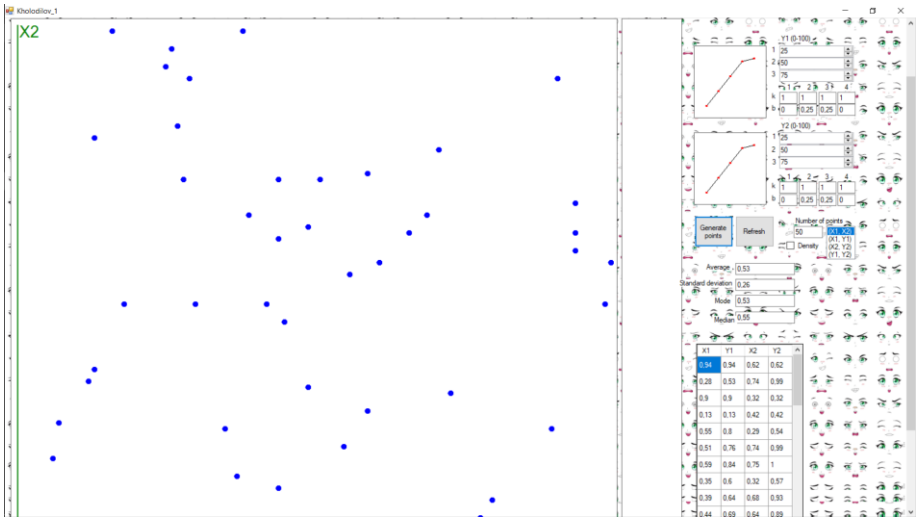



*Рисунок 2 - Генерация и отображение данных в виде изображения с соответствующими осями*

- Реализовать отображение профиля преобразования как кусочно-линейных функций (по пяти точкам – первая и последняя привязаны к границам диапазона генерации равномерно-распределенных случайных величин).

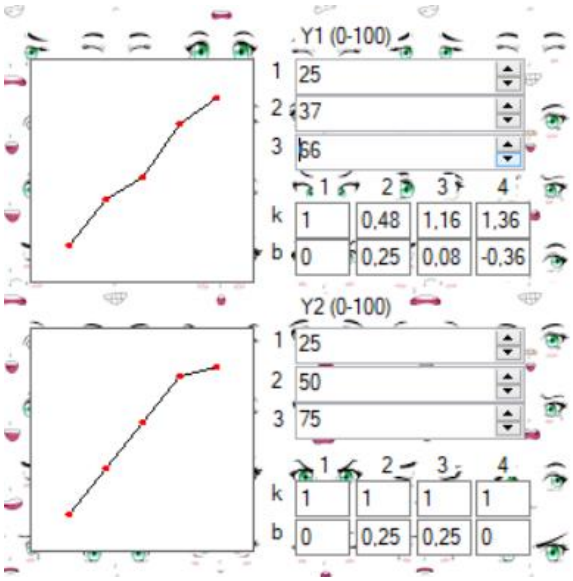



*Рисунок 3 - Генерация вторых значений данных с помощию numericupdown*

- Должна существовать возможность отобразить точки (X1, X2), (X1, Y1), (X2, Y2), (Y1, Y2).

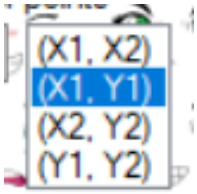



*Рисунок 4 - Выбос зависимости осей*

- Реализовать расчёт плотности распределения случайных точек и выполнить фоновую окраску области отрисовки случайных точек. При подсчете плотности разделить диапазон отображения по каждой оси на 10 интервалов.

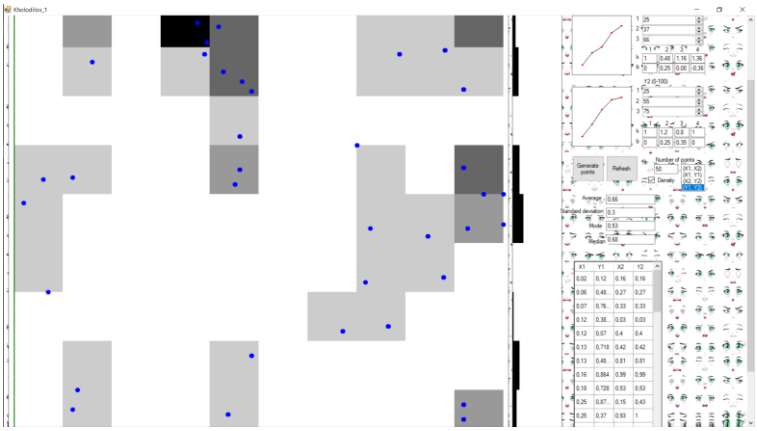


*Рисунок 5 - Создание гистограммы по выбранным данным*

- Добавлен в приложении расчёт статистических данных (описательной статистики) полученного распределения (среднее, средне-квадратическое отклонение, мода, медиана), а также отображения гистограммы распределения каждой из случайных величин. Расположить оси гистограмм вдоль соответствующих осей на диаграмме облака точек.

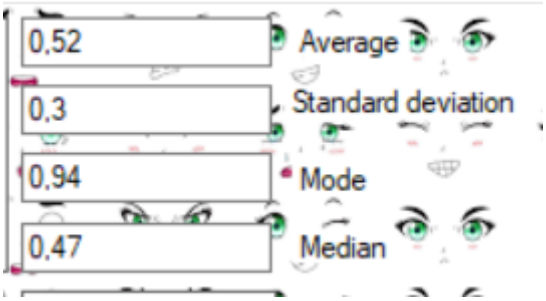



*Рисунок 6 - Подсчет всех мат. функций*

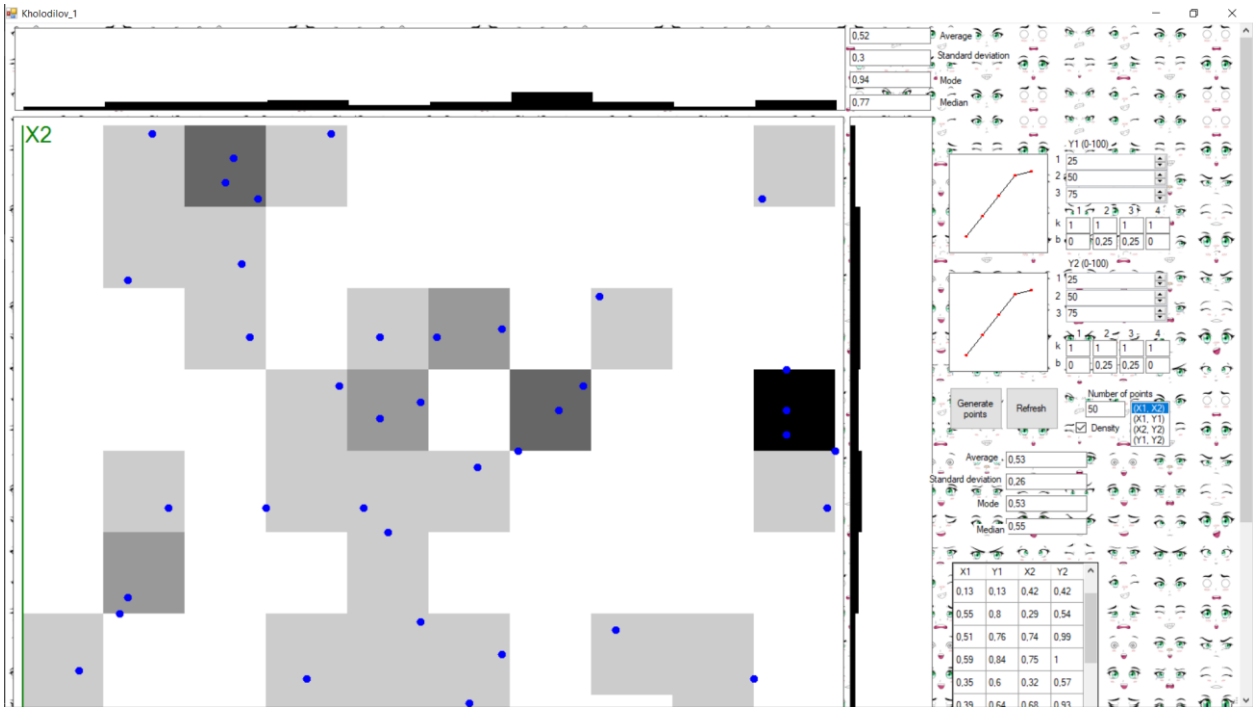



*Рисунок 7 - Итоговый вид проекта*

## Вывод

В ходе создания программы было написанно ПО для визуализации собранных наборов пространственных данных методом проекции на плоскость.

# Приложение А

Листинг А-1 – программный код:

Листинг А-1 – программный код:

```
using System;
using System.Collections.Generic;
using System.Drawing;
using System.Windows.Forms;
//using System.Windows.Media;

namespace Lab_1
{
    public partial class Kholodilov_1 : Form
    {
        Random random = new Random();
        List<PointF> points_x = new List<PointF>();
        List<PointF> points_y = new List<PointF>();
        bool start_work = false;
        bool is_y1_param_change = false;
        bool is_y2_param_change = false;
        bool is_axes_change = true;
        bool is_density_change = false;
        int[,] density = new int[10, 10];
        int[] density_up = new int[10];
        int[] density_right = new int[10];
        float summ_up = 0;
        float summ_right = 0;

        public Kholodilov_1()
        {
            InitializeComponent();
            timer1.Enabled = true;
            dataGridView1.RowHeadersVisible = false;
            dataGridView1.ColumnCount = 4;
            int size_grid = 40;
            dataGridView1.Columns[0].Width = size_grid;
            dataGridView1.Columns[1].Width = size_grid;
```

```
            dataGridView1.Columns[2].Width = size_grid;
            dataGridView1.Columns[3].Width = size_grid;
            dataGridView1.Width = size_grid * 4 + 20;
            listBox1.SelectedIndex = 0;
            X1Y1_box.Image = new Bitmap(120, 120);
            X2Y2_box.Image = new Bitmap(120, 120);
            Main_box.Image = new Bitmap(1020, 1020);
            Up_box.Image = new Bitmap(1020, 100);
            Right_box.Image = new Bitmap(100, 1020);
        }
        private void Form1_Load_1(object sender, EventArgs e)
        {
        }
        public float check_max_min_point (float d)
        {
            if (d <= 0) return 0.0f;
            else if (d >= 1) return 1.0f;
            else return d;
        }
        private void Form1_FormClosed_1(object sender, FormClosedEventArgs e)
        {
            timer1.Stop();
        }
        public int check_id_density (int d)
        {
            if (d < 0) return 0;
            else if (d > 9) return 9;
            else return d;
        }
        public void draw_main_img ()
        {
            density = new int[10, 10] { { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 }, { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 }, { 0, 0, 0, 0,
0, 0, 0, 0, 0, 0 }, { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 }, { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 }, { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 }, {
0, 0, 0, 0, 0, 0, 0, 0, 0, 0 }, { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 }, { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 }, { 0, 0, 0, 0, 0, 0, 0,
0, 0, 0 } };
            density_up =    new int[10] { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 };
            density_right = new int[10] { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 };
            summ_right = 0;
```

```
        summ_up = 0;
        Graphics graphics = Graphics.FromImage(Main_box.Image);
        Graphics graphics_up = Graphics.FromImage(Up_box.Image);
        Graphics graphics_righ  = Graphics.FromImage(Right_box.Image);
        graphics.FillRectangle(Brushes.White, new Rectangle(0, 0, Main_box.Width,
Main_box.Height));
        graphics_up.FillRectangle(Brushes.White, new Rectangle(0, 0, Up_box.Width,
Up_box.Height));
        graphics_righ.FillRectangle(Brushes.White, new Rectangle(0, 0, Right_box.Width,
Right_box.Height));
        if (is_density.Checked)
        {
           int max = 0;
           for (int i = 0; i < points_x.Count; i++)
           {
              switch (listBox1.SelectedIndex)
              {
                 case 1:
                    density[check_id_density((int)Math.Ceiling(points_x[i].X * 10 - 1)),
check_id_density((int)Math.Ceiling(points_y[i].X * 10 - 1))]++;
                    if (density[check_id_density((int)Math.Ceiling(points_x[i].X * 10 - 1)),
check_id_density((int)Math.Ceiling(points_y[i].X * 10 - 1))] > max) max =
density[check_id_density((int)Math.Ceiling(points_x[i].X * 10 - 1)),
check_id_density((int)Math.Ceiling(points_y[i].X * 10 - 1))];
                    break;
                 case 2:
                    density[check_id_density((int)Math.Ceiling(points_x[i].Y * 10 - 1)),
check_id_density((int)Math.Ceiling(points_y[i].Y * 10 - 1))]++;
                    if (density[check_id_density((int)Math.Ceiling(points_x[i].Y * 10 - 1)),
check_id_density((int)Math.Ceiling(points_y[i].Y * 10 - 1))] > max) max =
density[check_id_density((int)Math.Ceiling(points_x[i].Y * 10 - 1)),
check_id_density((int)Math.Ceiling(points_y[i].Y * 10 - 1))];
                    break;
                 case 3:
                    density[check_id_density((int)Math.Ceiling(points_y[i].X * 10 - 1)),
check_id_density((int)Math.Ceiling(points_y[i].Y * 10 - 1))]++;
                    if (density[check_id_density((int)Math.Ceiling(points_y[i].X * 10 - 1)),
check_id_density((int)Math.Ceiling(points_y[i].Y * 10 - 1))] > max) max =
```

```
density[check_id_density((int)Math.Ceiling(points_y[i].X * 10 - 1)),
check_id_density((int)Math.Ceiling(points_y[i].Y * 10 - 1))];
                    break;
                default:
                    density[check_id_density((int)Math.Ceiling(points_x[i].X * 10 - 1)),
check_id_density((int)Math.Ceiling(points_x[i].Y * 10 - 1))]++;
                    if (density[check_id_density((int)Math.Ceiling(points_x[i].X * 10 - 1)),
check_id_density((int)Math.Ceiling(points_x[i].Y * 10 - 1))] > max) max =
density[check_id_density((int)Math.Ceiling(points_x[i].X * 10 - 1)),
check_id_density((int)Math.Ceiling(points_x[i].Y * 10 - 1))];
                    break;
            }
        }
        for (int i = 0; i < 10; i++)
        {
            for (int j = 0; j < 10; j++)
            {
                System.Drawing.Brush brush = new
SolidBrush(System.Drawing.Color.FromArgb((int)(density[i,j]*255/max), 0, 0, 0));
                graphics.FillRectangle(brush, 10 + 100 * i, 10 + 900 - (100 * j), 100, 100);
                density_up[j] += density[i, j];
                density_right[i] += density[i, j];
            }
        }
        for (int i = 0; i < 10; i++)
        {
            graphics_up.FillRectangle(Brushes.Black, 10 + 100 * i, 100 - (int)((density_up[i] * 100) /
Int32.Parse(num_point.Text)), 100, (int)((density_up[i] * 100) / Int32.Parse(num_point.Text)));
            graphics_righ.FillRectangle(Brushes.Black, 0, 10 + 100 * i, (int)((density_right[i] * 100) /
Int32.Parse(num_point.Text)), 100);
            //graphics_up.FillRectangle(Brushes.Black, 10 + 100 * i, 100 - ((density_up[i] * 100) /
Int32.Parse(num_point.Text)), 100, 10);
            //MessageBox.Show((100 - (density_up[i] / Int32.Parse(num_point.Text) *
100)).ToString());
        }
    }
    Font drawFont = new Font("Arial", 20);
    StringFormat drawFormat = new StringFormat();
```

```
        SolidBrush drawBrush = new SolidBrush(Color.Black);
        graphics.DrawString(listBox1.Items[listBox1.SelectedIndex].ToString().Substring(5,          2),
drawFont, Brushes.Green, 10, 5, drawFormat);
        graphics.DrawString(listBox1.Items[listBox1.SelectedIndex].ToString().Substring(1,          2),
drawFont, Brushes.Red, Main_box.Width - 47, Main_box.Height - 40, drawFormat);
        graphics.FillRectangle(Brushes.Green,   10, 10, 2, Main_box.Height - 20);
        graphics.FillRectangle(Brushes.Red,     10, Main_box.Height - 10, Main_box.Width - 20, 2);

        for (int i = 0; i < points_x.Count; i++)
        {
           switch (listBox1.SelectedIndex)
           {
              case 1:
                 graphics.FillEllipse(Brushes.Blue,  10  +  points_x[i].X  *  1000  -  5,  10  +  1000  -
points_y[i].X * 1000 - 5, 5 + 5, 5 + 5);
                 summ_right += points_y[i].X;
                 summ_up += points_x[i].X;
                 break;
              case 2:
                 graphics.FillEllipse(Brushes.Blue,  10  +  points_x[i].Y  *  1000  -  5,  10  +  1000  -
points_y[i].Y * 1000 - 5, 5 + 5, 5 + 5);
                 summ_right += points_y[i].Y;
                 summ_up += points_x[i].Y;
                 break;
              case 3:
                 graphics.FillEllipse(Brushes.Blue,  10  +  points_y[i].X  *  1000  -  5,  10  +  1000  -
points_y[i].Y * 1000 - 5, 5 + 5, 5 + 5);
                 summ_right += points_y[i].Y;
                 summ_up += points_y[i].X;
                 break;
              default:
                 graphics.FillEllipse(Brushes.Blue,  10  +  points_x[i].X  *  1000  -  5,  10  +  1000  -
points_x[i].Y * 1000 - 5, 5 + 5, 5 + 5);
                 summ_right += points_x[i].Y;
                 summ_up += points_x[i].X;
                 break;
           }
        }
```

```
if (is_y1_param_change || is_y2_param_change || is_axes_change)
{
    float ave_right = ((float)(summ_right / points_x.Count));
    float ave_up = ((float)(summ_up / points_x.Count));

    double st_deviation_up = 0;
    double st_deviation_right = 0;
    for (int i = 0; i < points_x.Count; i++)
    {
        switch (listBox1.SelectedIndex)
        {
            case 1:
                //summ_right += points_y[i].X;
                //summ_up += points_x[i].X;
                st_deviation_up += Math.Pow(points_x[i].X - ave_up, 2);
                st_deviation_right += Math.Pow(points_y[i].X - ave_right, 2);
                break;
            case 2:
                //summ_right += points_y[i].Y;
                //summ_up += points_x[i].Y;
                st_deviation_up += Math.Pow(points_x[i].Y - ave_up, 2);
                st_deviation_right += Math.Pow(points_y[i].Y - ave_right, 2);
                break;
            case 3:
                //summ_right += points_y[i].Y;
                //summ_up += points_y[i].X;
                st_deviation_up += Math.Pow(points_y[i].X - ave_up, 2);
                st_deviation_right += Math.Pow(points_y[i].Y - ave_right, 2);
                break;
            default:
                //summ_right += points_x[i].Y;
                //summ_up += points_x[i].X;
                st_deviation_up += Math.Pow(points_x[i].X - ave_up, 2);
                st_deviation_right += Math.Pow(points_x[i].Y - ave_right, 2);
                break;
        }
    }
    average_right.Text = Math.Round(ave_right, 2).ToString();
```

```
            average_up.Text = Math.Round(ave_up, 2).ToString();
            standard_deviation_right.Text = Math.Round(Math.Sqrt((float)(st_deviation_right / points_x.Count)), 2).ToString();
            standard_deviation_up.Text = Math.Round(Math.Sqrt((float)(st_deviation_up / points_x.Count)), 2).ToString();

            // Mode
            int m_right = 0;
            int m_up = 0;
            float m_data_right = -1;
            float m_data_up = -1;
            float m_last_right = -1;
            float m_last_up = -1;

            switch (listBox1.SelectedIndex)
            {
               case 1:
                  //summ_right += points_y[i].X;
                  //summ_up += points_x[i].X;
                  points_y = Sort_list(points_y, true);
                  points_x = Sort_list(points_x, true);
                  for (int i = 0; i < points_x.Count; i++)
                  {
                     if (points_y[i].X == m_last_right)
                     {
                        m_right++;
                     }
                     else
                     {
                        if (m_data_right < m_right)
                        {
                           mode_right.Text = m_last_right.ToString();
                           m_data_right = m_right;
                        }
                        m_right = 0;
                     }
                     m_last_right = points_y[i].X;
```

```
            if (points_x[i].X == m_last_up)
            {
                m_up++;
            }
            else
            {
                if (m_data_up < m_up)
                {
                    mode_up.Text = m_last_up.ToString();
                    m_data_up = m_up;
                }
                m_up = 0;
            }
            m_last_up = points_x[i].X;
        }
        break;
    case 2:
        //summ_right += points_y[i].Y;
        //summ_up += points_x[i].Y;
        points_y = Sort_list(points_y, false);
        points_x = Sort_list(points_x, false);
        for (int i = 0; i < points_x.Count; i++)
        {
            if (points_y[i].Y == m_last_right)
            {
                m_right++;
            }
            else
            {
                if (m_data_right < m_right)
                {
                    mode_right.Text = m_last_right.ToString();
                    m_data_right = m_right;
                }
                m_right = 0;
            }
            m_last_right = points_y[i].Y;
```

```
            if (points_x[i].Y == m_last_up)
            {
                m_up++;
            }
            else
            {
                if (m_data_up < m_up)
                {
                    mode_up.Text = m_last_up.ToString();
                    m_data_up = m_up;
                }
                m_up = 0;
            }
            m_last_up = points_x[i].Y;
        }

        break;
    case 3:
        //summ_right += points_y[i].Y;
        //summ_up += points_y[i].X;
        points_y = Sort_list(points_y, false);
        for (int i = 0; i < points_x.Count; i++)
        {
            if (points_y[i].Y == m_last_right)
            {
                m_right++;
            }
            else
            {
                if (m_data_right < m_right)
                {
                    mode_right.Text = m_last_right.ToString();
                    m_data_right = m_right;
                }
                m_right = 0;
            }
            m_last_right = points_y[i].Y;
        }
```

```
            points_y = Sort_list(points_y, true);
            for (int i = 0; i < points_x.Count; i++)
            {
                if (points_y[i].X == m_last_up)
                {
                    m_up++;
                }
                else
                {
                    if (m_data_up < m_up)
                    {
                        mode_up.Text = m_last_up.ToString();
                        m_data_up = m_up;
                    }
                    m_up = 0;
                }
                m_last_up = points_y[i].X;
            }
            break;
        default:
            //summ_right += points_x[i].Y;
            //summ_up += points_x[i].X;
            points_x = Sort_list(points_x, false);
            for (int i = 0; i < points_x.Count; i++)
            {
                if (points_x[i].Y == m_last_right)
                {
                    m_right++;
                }
                else
                {
                    if (m_data_right < m_right)
                    {
                        mode_right.Text = m_last_right.ToString();
                        m_data_right = m_right;
                    }
                    m_right = 0;
                }
```

```
                    m_last_right = points_x[i].Y;
                }
                points_x = Sort_list(points_x, true);
                for (int i = 0; i < points_x.Count; i++)
                {
                    if (points_x[i].X == m_last_up)
                    {
                        m_up++;
                    }
                    else
                    {
                        if (m_data_up < m_up)
                        {
                            mode_up.Text = m_last_up.ToString();
                            m_data_up = m_up;
                        }
                        m_up = 0;
                    }
                    m_last_up = points_x[i].X;
                }
                break;
            }
        }
        // Median

        switch (listBox1.SelectedIndex)
        {
            case 1:
                //summ_right += points_y[i].X;
                //summ_up += points_x[i].X;
                if (points_x.Count % 2 == 0)
                {
                    median_right.Text = ((points_y[(int)Math.Ceiling((double)points_x.Count / 2)].X + points_y[(int)Math.Ceiling((double)points_x.Count / 2) - 1].X) / 2).ToString();
                    median_up.Text = ((points_x[(int)Math.Ceiling((double)points_x.Count / 2)].X + points_x[(int)Math.Ceiling((double)points_x.Count / 2) - 1].X) / 2 ).ToString();
                }
                else
```

```
            {
                median_right.Text = points_y[ (int)Math.Ceiling((double)points_x.Count / 2) ].X.ToString();
                median_up.Text = points_x[(int)Math.Ceiling((double)points_x.Count / 2)].X.ToString();
            }
            break;
        case 2:
            //summ_right += points_y[i].Y;
            //summ_up += points_x[i].Y;
            if (points_x.Count % 2 == 0)
            {
                median_right.Text = ((points_y[(int)Math.Ceiling((double)points_x.Count / 2)].Y + points_y[(int)Math.Ceiling((double)points_x.Count / 2) - 1].Y) / 2).ToString();
                median_up.Text = ((points_x[(int)Math.Ceiling((double)points_x.Count / 2)].Y + points_x[(int)Math.Ceiling((double)points_x.Count / 2) - 1].Y) / 2).ToString();
            }
            else
            {
                median_right.Text = points_y[(int)Math.Ceiling((double)points_x.Count / 2)].Y.ToString();
                median_up.Text = points_x[(int)Math.Ceiling((double)points_x.Count / 2)].Y.ToString();
            }
            break;
        case 3:
            //summ_right += points_y[i].Y;
            //summ_up += points_y[i].X;
            if (points_x.Count % 2 == 0)
            {
                median_up.Text = ((points_y[(int)Math.Ceiling((double)points_x.Count / 2)].X + points_y[(int)Math.Ceiling((double)points_x.Count / 2) - 1].X) / 2).ToString();
                points_y = Sort_list(points_y, false);
                median_right.Text = ((points_y[(int)Math.Ceiling((double)points_x.Count / 2)].Y + points_y[(int)Math.Ceiling((double)points_x.Count / 2) - 1].Y) / 2).ToString();
            }
            else
            {
```

```
                median_up.Text = points_y[(int)Math.Ceiling((double)points_x.Count / 2)].X.ToString();
                points_y = Sort_list(points_y, false);
                median_right.Text = points_y[(int)Math.Ceiling((double)points_x.Count / 2)].Y.ToString();
            }
            break;
        default:
            //summ_right += points_x[i].Y;
            //summ_up += points_x[i].X;
            if (points_x.Count % 2 == 0)
            {
                median_up.Text = ((points_x[(int)Math.Ceiling((double)points_x.Count / 2)].X + points_x[(int)Math.Ceiling((double)points_x.Count / 2) - 1].X) / 2).ToString();
                points_x = Sort_list(points_x, false);
                median_right.Text = ((points_x[(int)Math.Ceiling((double)points_x.Count / 2)].Y + points_x[(int)Math.Ceiling((double)points_x.Count / 2) - 1].Y) / 2).ToString();
            }
            else
            {
                median_up.Text = points_x[(int)Math.Ceiling((double)points_x.Count / 2)].X.ToString();
                points_x = Sort_list(points_x, false);
                median_right.Text = points_x[(int)Math.Ceiling((double)points_x.Count / 2)].Y.ToString();
            }
            break;
    }
    median_right.Text = Math.Round(float.Parse(median_right.Text), 2).ToString();
    median_up.Text = Math.Round(float.Parse(median_up.Text), 2).ToString();
    Main_box.Refresh();
    Up_box.Refresh();
    Right_box.Refresh();
}
private void timer1_Tick(object sender, EventArgs e)
{
    if (start_work && (is_y1_param_change || is_y2_param_change))
    {
```

```
            if (is_y1_param_change) generate_y_points(2);
            else generate_y_points(1);
            view_datagreed();
            draw_main_img();
            is_y1_param_change = false;
            is_y2_param_change = false;
        }
        else if (start_work && (is_axes_change || is_density_change))
        {
            draw_main_img();
            is_axes_change = false;
            is_density_change = false;
        }
    }
    public List<PointF> Sort_list(List<PointF> list, bool is_fist = true)
    {
        float[,] massive_data = new float[2, list.Count];
        for (int i = 0; i < list.Count; i++)
        {
            massive_data[0, i] = list[i].X;
            massive_data[1, i] = list[i].Y;
        }
        int n = list.Count;
        for (int i = 0; i < n - 1; i++)
            for (int j = 0; j < n - i - 1; j++)
                if (is_fist)
                {
                    if (massive_data[0, j] > massive_data[0, j + 1])
                    {
                        var tempVar = massive_data[0, j];
                        massive_data[0, j] = massive_data[0, j + 1];
                        massive_data[0, j + 1] = tempVar;
                        tempVar = massive_data[1, j];
                        massive_data[1, j] = massive_data[1, j + 1];
                        massive_data[1, j + 1] = tempVar;
                    }
                }
                else
```

```
            {
                if (massive_data[1, j] > massive_data[1, j + 1])
                {
                    var tempVar = massive_data[1, j];
                    massive_data[1, j] = massive_data[1, j + 1];
                    massive_data[1, j + 1] = tempVar;
                    tempVar = massive_data[0, j];
                    massive_data[0, j] = massive_data[0, j + 1];
                    massive_data[0, j + 1] = tempVar;
                }
            }
        list.Clear();
        for (int i = 0; i < Int32.Parse(num_point.Text); i++)
        {
            list.Add(new PointF(massive_data[0, i], massive_data[1, i]));
        }
        return list;
    }
    private void Start_but_Click(object sender, EventArgs e)
    {
        if (Int32.Parse(num_point.Text) > 0)
        {
            for (int i = 0; i < Int32.Parse(num_point.Text); i++)
            {
                PointF generate = new Point();
                generate.X = (float)Math.Round(random.NextDouble(), 2);
                generate.Y = (float)Math.Round(random.NextDouble(), 2);
                points_x.Add(generate);
                points_y.Add(generate);
            }
            start_work = true;
            //Sort_x();
            generate_y_points();
            view_datagreed();
            draw_main_img();
            timer1.Start();
        }
    }
```

```
    public void generate_y_points(int case_y = 0)
    {
        double k1_y1_p = 0.0f, k2_y1_p = 0.0f, k3_y1_p = 0.0f, k4_y1_p = 0.0f;
        double b2_y1_p = 0.0f, b3_y1_p = 0.0f, b4_y1_p = 0.0f;
        double k1_y2_p = 0.0f, k2_y2_p = 0.0f, k3_y2_p = 0.0f, k4_y2_p = 0.0f;
        double b2_y2_p = 0.0f, b3_y2_p = 0.0f, b4_y2_p = 0.0f;
        if (case_y == 2 || case_y == 0)
        {
            // For X1
            // generate param k and b for x < 0.25
            k1_y1_p = (((double)(y1_1_param.Value) / 100)) / (0.25 - 0);
            k1_y1.Text = k1_y1_p.ToString();
            b1_y1.Text = "0";
            // generate param k and b for x >= 0.25 && x < 0.5
            k2_y1_p = ((((double)(y1_2_param.Value) / 100) - ((double)(y1_1_param.Value)) / 100)) /
(0.5 - 0.25);
            b2_y1_p = ((double)(y1_2_param.Value) / 100) - k2_y1_p * 0.25;
            k2_y1.Text = k2_y1_p.ToString();
            b2_y1.Text = b2_y1_p.ToString();
            // generate param k and b for x >= 0.5 && x < 0.75
            k3_y1_p = ((((double)(y1_3_param.Value) / 100) - ((double)(y1_2_param.Value)) / 100)) /
(0.75 - 0.5);
            b3_y1_p = ((double)(y1_3_param.Value) / 100) - k3_y1_p * 0.5;
            k3_y1.Text = k3_y1_p.ToString();
            b3_y1.Text = b3_y1_p.ToString();
            // generate param k and b for x >= 0.75
            k4_y1_p = (1 - ((double)(y1_3_param.Value)) / 100) / (1 - 0.75);
            b4_y1_p = ((double)(y1_3_param.Value) / 100) - k4_y1_p * 0.75;
            k4_y1.Text = k4_y1_p.ToString();
            b4_y1.Text = b4_y1_p.ToString();

            Graphics graphics = Graphics.FromImage(X1Y1_box.Image);
            Pen pen = new Pen(Color.Black);
            graphics.FillRectangle(Brushes.White, new Rectangle(0, 0, X1Y1_box.Width,
X1Y1_box.Height));
            graphics.DrawLine(pen, 20, 100,                 40, 100 - (int)y1_1_param.Value);
            graphics.DrawLine(pen, 40, 100 - (int)y1_1_param.Value, 60, 100 - (int)y1_2_param.Value);
            graphics.DrawLine(pen, 60, 100 - (int)y1_2_param.Value, 80, 100 - (int)y1_3_param.Value);
```

```
            graphics.DrawLine(pen, 80, 100 - (int)y1_3_param.Value, 100, 20);
            graphics.FillEllipse(Brushes.Red, 20 - 2,  100 - 2, 2 + 2, 2 + 2);
            graphics.FillEllipse(Brushes.Red, 40 - 2,  100 - (int)y1_1_param.Value - 2, 2 + 2, 2 + 2);
            graphics.FillEllipse(Brushes.Red, 60 - 2,  100 - (int)y1_2_param.Value - 2, 2 + 2, 2 + 2);
            graphics.FillEllipse(Brushes.Red, 80 - 2,  100 - (int)y1_3_param.Value - 2, 2 + 2, 2 + 2);
            graphics.FillEllipse(Brushes.Red, 100 - 2, 20 - 2, 2 + 2, 2 + 2);
            X1Y1_box.Refresh();
        }
        if (case_y == 1 || case_y == 0)
        {
            // For X2 and Y2
            // generate param k and b for x < 0.25
            k1_y2_p = (((double)(y2_1_param.Value) / 100)) / (0.25 - 0);
            k1_y2.Text = k1_y2_p.ToString();
            b1_y2.Text = "0";
            // generate param k and b for x >= 0.25 && x < 0.5
            k2_y2_p = ((((double)(y2_2_param.Value) / 100) - ((double)(y2_1_param.Value)) / 100)) /
(0.5 - 0.25);
            b2_y2_p = ((double)(y2_2_param.Value) / 100) - k2_y2_p * 0.25;
            k2_y2.Text = k2_y2_p.ToString();
            b2_y2.Text = b2_y2_p.ToString();
            // generate param k and b for x >= 0.5 && x < 0.75
            k3_y2_p = ((((double)(y2_3_param.Value) / 100) - ((double)(y2_2_param.Value)) / 100)) /
(0.75 - 0.5);
            b3_y2_p = ((double)(y2_3_param.Value) / 100) - k3_y2_p * 0.5;
            k3_y2.Text = k3_y2_p.ToString();
            b3_y2.Text = b3_y2_p.ToString();
            // generate param k and b for x >= 0.75
            k4_y2_p = (1 - ((double)(y2_3_param.Value)) / 100) / (1 - 0.75);
            b4_y2_p = ((double)(y2_3_param.Value) / 100) - k4_y2_p * 0.75;
            k4_y2.Text = k4_y2_p.ToString();
            b4_y2.Text = b4_y2_p.ToString();

            Graphics graphics = Graphics.FromImage(X2Y2_box.Image);
            Pen pen = new Pen(Color.Black);
            graphics.FillRectangle(Brushes.White,    new    Rectangle(0,    0,    X2Y2_box.Width,
X2Y2_box.Height));
            graphics.DrawLine(pen, 20, 100,                    40, 100 - (int)y2_1_param.Value);
```

```
            graphics.DrawLine(pen, 40, 100 - (int)y2_1_param.Value, 60, 100 - (int)y2_2_param.Value);
            graphics.DrawLine(pen, 60, 100 - (int)y2_2_param.Value, 80, 100 - (int)y2_3_param.Value);
            graphics.DrawLine(pen, 80, 100 - (int)y2_3_param.Value, 100, 20);
            graphics.FillEllipse(Brushes.Red, 20 - 2, 100 - 2, 2 + 2, 2 + 2);
            graphics.FillEllipse(Brushes.Red, 40 - 2, 100 - (int)y2_1_param.Value - 2, 2 + 2, 2 + 2);
            graphics.FillEllipse(Brushes.Red, 60 - 2, 100 - (int)y2_2_param.Value - 2, 2 + 2, 2 + 2);
            graphics.FillEllipse(Brushes.Red, 80 - 2, 100 - (int)y2_3_param.Value - 2, 2 + 2, 2 + 2);
            graphics.FillEllipse(Brushes.Red, 100 - 2, 20 - 2, 2 + 2, 2 + 2);
            X2Y2_box.Refresh();
        }

        for (int i = 0; i < Int32.Parse(num_point.Text); i++)
        {
            if (points_x[i].X < 0.25)
            {
                if (case_y == 1) points_y[i] = new PointF(points_y[i].X, check_max_min_point((float)(points_x[i].Y * k1_y2_p)));
                else if (case_y == 2) points_y[i] = new PointF(check_max_min_point((float)(points_x[i].X * k1_y1_p)), points_y[i].Y);
                else points_y[i] = new PointF(check_max_min_point((float)(points_x[i].X * k1_y1_p)), check_max_min_point((float)(points_x[i].Y * k1_y2_p)));
            }
            else if (points_x[i].X >= 0.25 && points_x[i].X < 0.5)
            {
                if (case_y == 1) points_y[i] = new PointF(points_y[i].X, check_max_min_point((float)(points_x[i].Y * k2_y2_p + b2_y2_p)));
                else if (case_y == 2) points_y[i] = new PointF(check_max_min_point((float)(points_x[i].X * k2_y1_p + b2_y1_p)), points_y[i].Y);
                else points_y[i] = new PointF(check_max_min_point((float)(points_x[i].X * k2_y1_p + b2_y1_p)), check_max_min_point((float)(points_x[i].Y * k2_y2_p + b2_y2_p)));
            }
            else if (points_x[i].X >= 0.5 && points_x[i].X < 0.75)
            {
                if (case_y == 1) points_y[i] = new PointF(points_y[i].X, check_max_min_point((float)(points_x[i].Y * k3_y2_p + b3_y2_p)));
                else if (case_y == 2) points_y[i] = new PointF(check_max_min_point((float)(points_x[i].X * k3_y1_p + b3_y1_p)), points_y[i].Y);
                else points_y[i] = new PointF(check_max_min_point((float)(points_x[i].X *
```

```
k3_y1_p + b3_y1_p)), check_max_min_point((float)(points_x[i].Y * k3_y2_p + b3_y2_p)));
            }
            else
            {
               if (case_y == 1) points_y[i] = new PointF(points_y[i].X, check_max_min_point((float)(points_x[i].Y * k4_y2_p + b4_y2_p)));
               else if (case_y == 2) points_y[i] = new PointF(check_max_min_point((float)(points_x[i].X * k4_y1_p + b4_y1_p)), points_y[i].Y);
               else points_y[i] = new PointF(check_max_min_point((float)(points_x[i].X * k4_y1_p + b4_y1_p)), check_max_min_point((float)(points_x[i].Y * k4_y2_p + b4_y2_p)));
            }
         }
      }
      public void view_datagreed()
      {
         if (Int32.Parse(num_point.Text) > 0)
         {
            dataGridView1.RowCount = 1;
            for (int i = 0; i < Int32.Parse(num_point.Text); i++)
            {
               dataGridView1.RowCount += 1;
               dataGridView1.Rows[i].Cells[0].Value = points_x[i].X;
               dataGridView1.Rows[i].Cells[1].Value = points_y[i].X;
               dataGridView1.Rows[i].Cells[2].Value = points_x[i].Y;
               dataGridView1.Rows[i].Cells[3].Value = points_y[i].Y;
            }
         }
      }
      private void Send_Click(object sender, EventArgs e)
      {
         points_x.Clear();
         points_y.Clear();
         dataGridView1.Rows.Clear();
         dataGridView1.Refresh();
         k4_y2.Clear(); k3_y2.Clear(); k2_y2.Clear(); k1_y2.Clear(); k4_y1.Clear(); k3_y1.Clear(); k2_y1.Clear(); k1_y1.Clear();
         b4_y1.Clear(); b3_y1.Clear(); b2_y1.Clear(); b1_y1.Clear(); b4_y2.Clear(); b3_y2.Clear(); b2_y2.Clear(); b1_y2.Clear();
```

```
            Graphics graphics = Graphics.FromImage(X1Y1_box.Image);
            graphics.FillRectangle(Brushes.White, new Rectangle(0, 0, X1Y1_box.Width,
X1Y1_box.Height));
            X1Y1_box.Refresh();
            graphics = Graphics.FromImage(X2Y2_box.Image);
            graphics.FillRectangle(Brushes.White, new Rectangle(0, 0, X2Y2_box.Width,
X2Y2_box.Height));
            X2Y2_box.Refresh();
            graphics = Graphics.FromImage(Main_box.Image);
            graphics.FillRectangle(Brushes.White, new Rectangle(0, 0, Main_box.Width,
Main_box.Height));
            Main_box.Refresh();
            graphics = Graphics.FromImage(Up_box.Image);
            graphics.FillRectangle(Brushes.White, new Rectangle(0, 0, Up_box.Width, Up_box.Height));
            Up_box.Refresh();
            graphics = Graphics.FromImage(Right_box.Image);
            graphics.FillRectangle(Brushes.White, new Rectangle(0, 0, Right_box.Width,
Right_box.Height));
            Right_box.Refresh();
        }
        private void y1_1_param_ValueChanged_1(object sender, EventArgs e)
        {
            is_y1_param_change = true;
        }
        private void y2_1_param_ValueChanged(object sender, EventArgs e)
        {
            is_y2_param_change = true;
        }
        private void listBox1_Click(object sender, EventArgs e)
        {
            is_axes_change = true;
        }

        private void is_density_CheckedChanged(object sender, EventArgs e)
        {
            is_density_change = true;
        }
    }
```

```
}
```