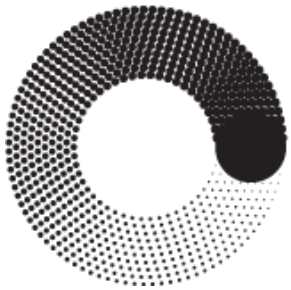
МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«МОСКОВСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»



МОСКОВСКИЙ ПОЛИТЕХ

Кафедра СМАРТ технологий

Лабораторная работа №2
«Применение проекций и матриц трансляции при визуализации данных»

По дисциплине «Технологии визуализации данных систем управления»

Группа	201-325
Студент	Холодилов И.В.
Дата	18.05.2023
Преподаватель	Идиатулов Т.Т.

2023

Цель работы

Подготовить приложение на языке C# для визуализации собранных наборов пространственных данных методом проекции на плоскость

Задачи

- Реализовать загрузку набора данных, заданных как тройки чисел (X, Y, Z) из файлов формата CSV (разделитель – точка с запятой) и генерацию заданного (через текстовое поле) количества случайных точек, где X,Y и Z – равномерно распределенные случайные величины на диапазоне [-1 ÷ 1]. Разработать систему отображения данных в виде облака точек на битовой карте (bitmap) с использованием прямой (ортогональной) проекции на координатные плоскости. Размер битовой карты – не менее 600 x 600 px. Применять изменение яркости точек для отображения удаленности точек от наблюдателя. Разработать систему визуализации с использованием преспективной проекции облака точек, полагая, что область распределения точек можно принять за куб с длиной грани в 1 м, размещенной на расстоянии 4 метров от наблюдателя. Разработать систему поворота базовой системы координат размещения облака точек (относительно плоскости экрана) с помощью матрицы трансляцииповорота. Реализовать инструмент редактирования матрицы трансляцииповорота через DataGrid или текстовые поля. Реализовать инструмент инкрементального поворота базовой системы коор- динат путем использования механизма Drag&Drop позиционным манипулятором. Каждое перетаскивание интерпретировать как локальное приращение угла поворота вокруг осей 0X и 0Y в системе координат, связанной с плоскостью экрана, но с центром, размещенным в начале системы координат набора точек. Реализовать отображение точек с учетом накопленного поворота, а также отображение базовых векторов системы координат.

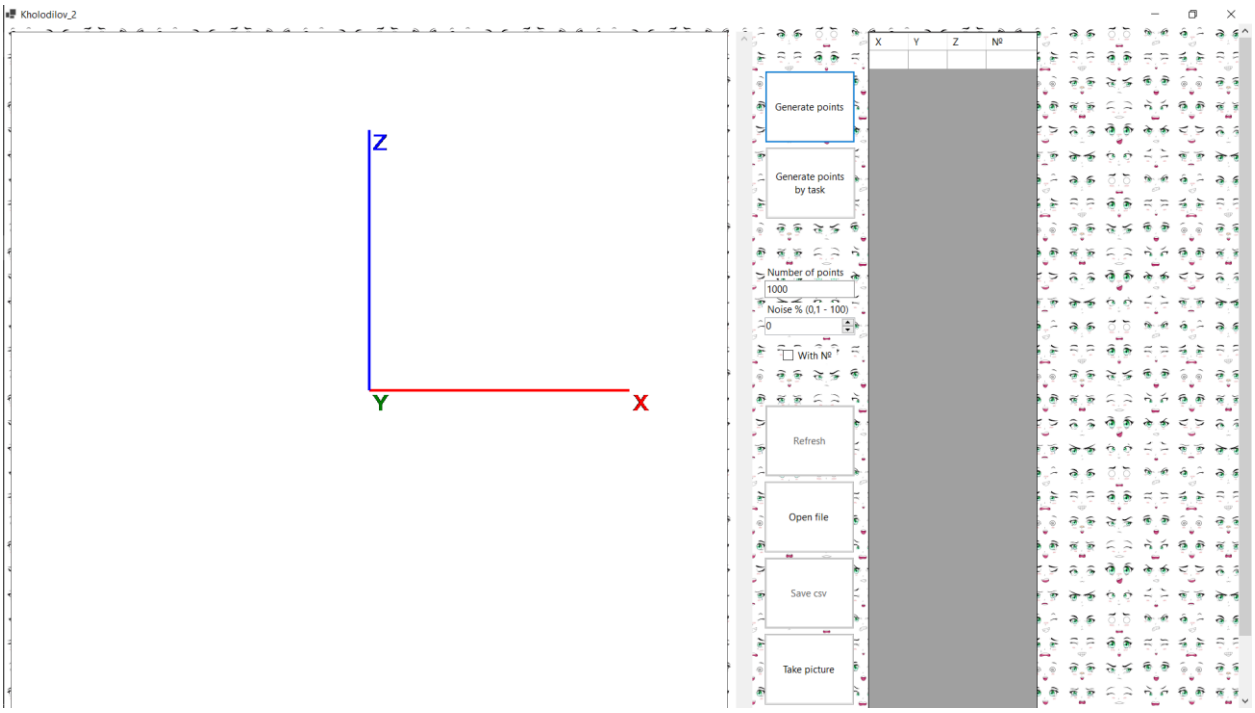


Рисунок 1 - Базовая форма с осями

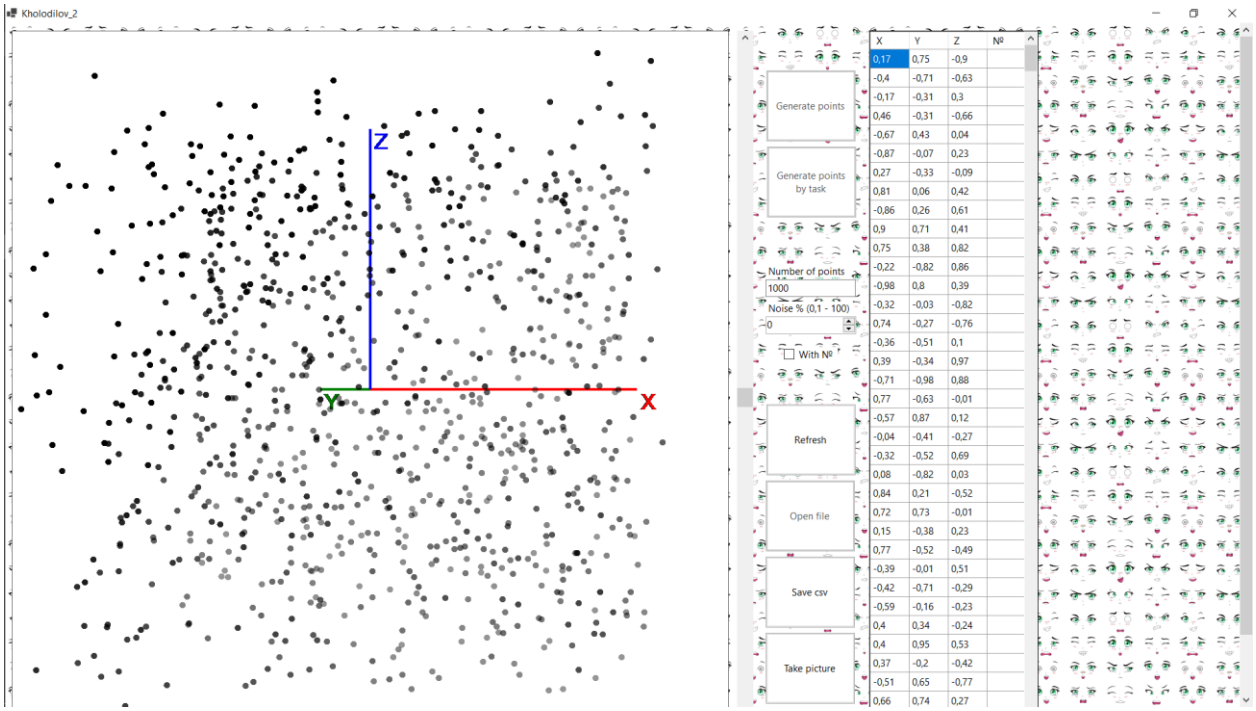


Рисунок 2 - Генерация точек

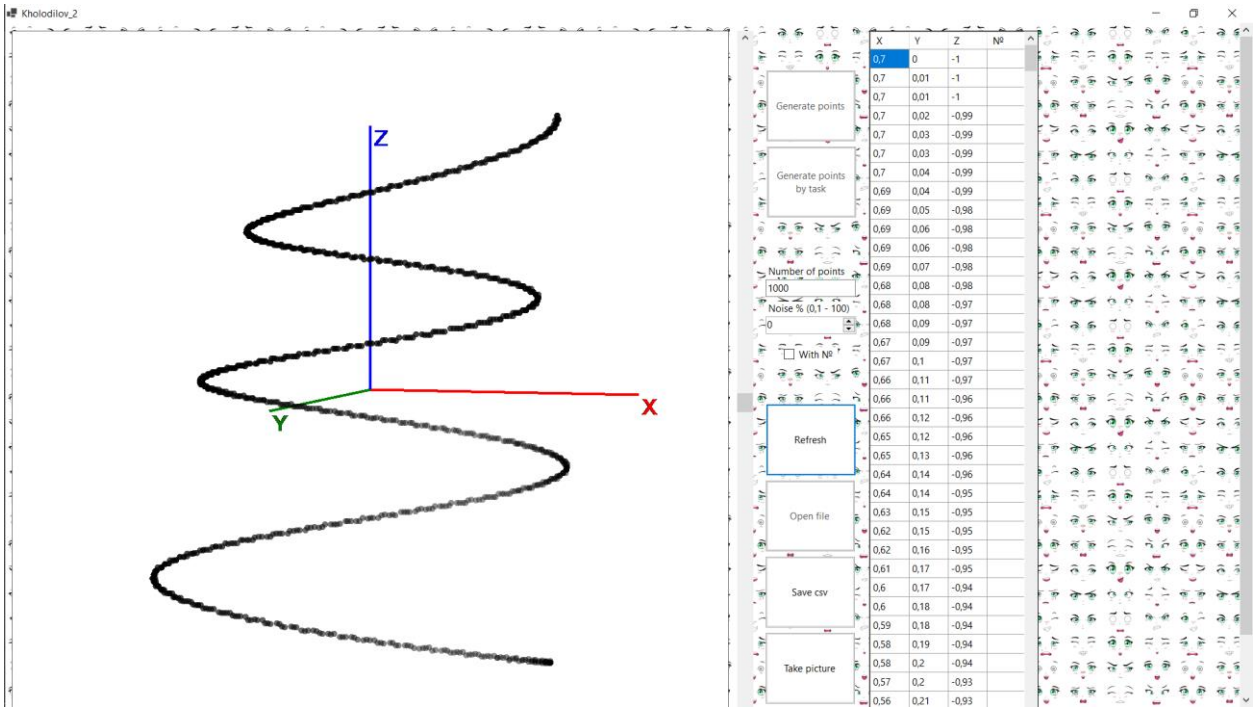


Рисунок 3 - Работа по варианту

- Разработать генератор случайных величин на основе пространственной параметрической функции (от номера точки) с добавлением случайного смещения по трем координатам, распределенного по нормальному закону с заданным параметром σ (СКО) [функция определяется индивидуальным заданием к лабораторной работе].

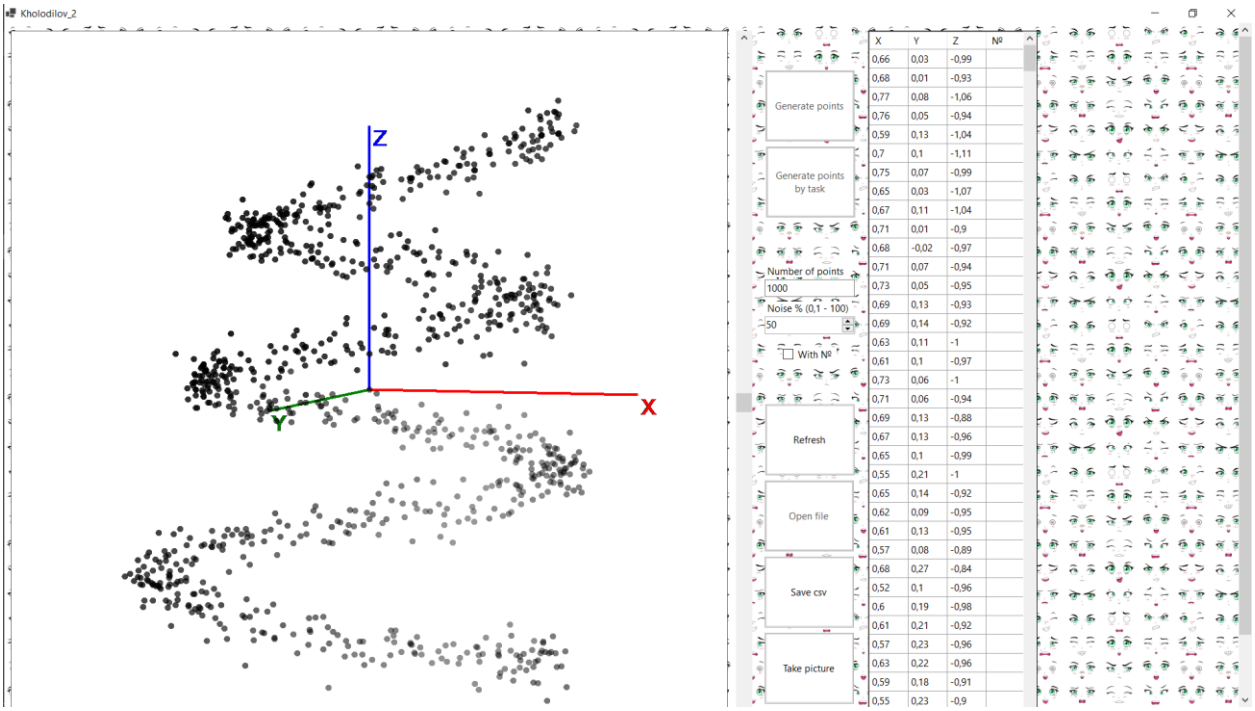


Рисунок 4 - Шумы для данных

- Реализовать сохранение полученного изображения в файл.

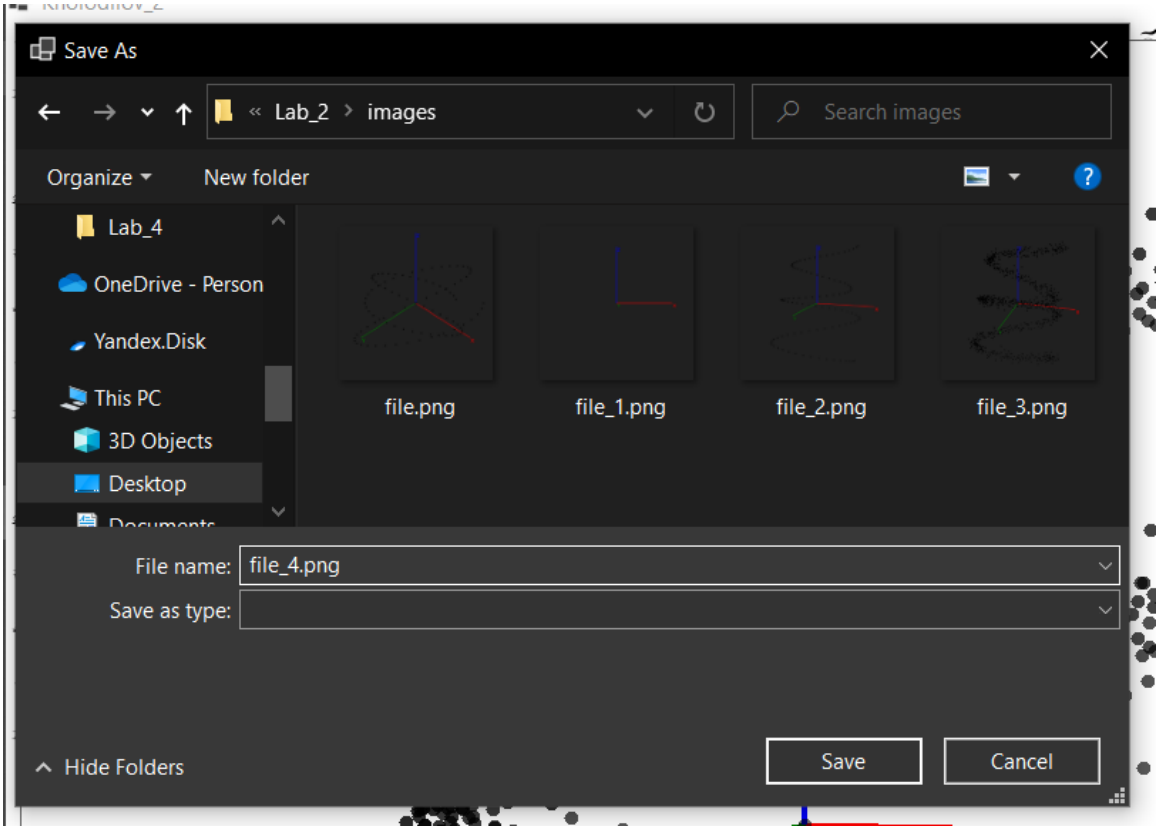


Рисунок 5 - Сохранение изображения

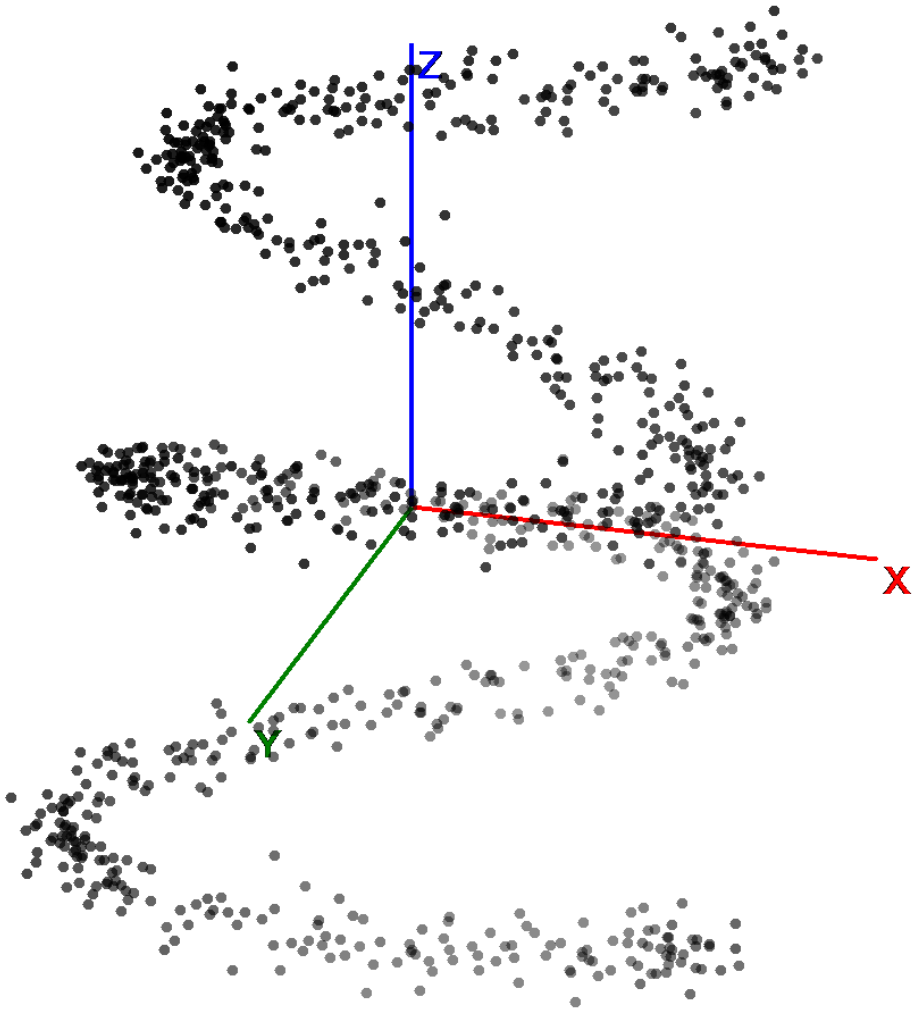


Рисунок 6 - Сохранненое изображение

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:

```
using System;
using System.Linq;
namespace test
{
    public partial class Kholodilov_2 : Form
    {
        Random random = new Random();
        bool is_start_work = false;
        bool is_noise_change = false;
        bool is_generate_task = false;
        bool is_rotate_change = false;
        bool is_rotate_scroll_change = false;
        bool is_n_change = false;
        float[,] generated_point;
        int count_point;
        int axis_line_size = 4;
        int point_size = 5;
        int picture_size = 1100;
        int size_grid = 60;
        private _3d_transform_point tr_3D = new _3d_transform_point();
        int[] start_mouse_pose;

        public Kholodilov_2()
        {
            InitializeComponent();
            timer1.Enabled = true;
            dataGridView1.RowHeadersVisible = false;
            dataGridView1.ColumnCount = 4;
            dataGridView1.Columns[0].Width = size_grid;
            dataGridView1.Columns[1].Width = size_grid;
            dataGridView1.Columns[2].Width = size_grid;
            dataGridView1.Width = size_grid * dataGridView1.ColumnCount + 20;
            Main_box.Image = new Bitmap(picture_size, picture_size);
            tr_3D.half_picture_size = (picture_size - 300) / 2;
```

```
            angleY_bar.Enabled = false;
            angleX_bar.Enabled = false;
            Refr_but.Enabled = false;
            Save_but.Enabled = false;
        }
        public void draw_main_img()
        {
            // Draw axis
            Bitmap bmp = new Bitmap(picture_size, picture_size);
            Graphics g = Graphics.FromImage(bmp);
            g.TranslateTransform(picture_size / 2, picture_size / 2);

            int[] XAxe0Point = tr_3D.Transform_point(new float[,]  { { 0f },  { 0f },  { 0f } }); // X: start
            int[] XAxeEndPoint = tr_3D.Transform_point(new float[,] { { 1f },  { 0f },  { 0f } }); // X: end
            int[] YAxe0Point = tr_3D.Transform_point(new float[,]  { { 0f },  { 0f },  { 0f } }); // Y: start
            int[] YAxeEndPoint = tr_3D.Transform_point(new float[,] { { 0f},  { 0f},  { -1f} });  // Y: end
            int[] ZAxe0Point = tr_3D.Transform_point(new float[,]  { { 0f },  { 0f },  { 0f } }); // Z: start
            int[] ZAxeEndPoint = tr_3D.Transform_point(new float[,] { { 0f },  { -1f }, { 0f } });  // Z: end

            g.DrawLine(new Pen(Color.Red, axis_line_size), XAxe0Point[0], XAxe0Point[1], XAxeEndPoint[0], XAxeEndPoint[1]);
            g.DrawLine(new Pen(Color.Green, axis_line_size), YAxe0Point[0], YAxe0Point[1], YAxeEndPoint[0], YAxeEndPoint[1]);
            g.DrawLine(new Pen(Color.Blue, axis_line_size), ZAxe0Point[0], ZAxe0Point[1], ZAxeEndPoint[0], ZAxeEndPoint[1]);

            g.DrawString("X", new Font("Aria", 20, FontStyle.Bold), Brushes.Red, XAxeEndPoint[0], XAxeEndPoint[1]);
            g.DrawString("Y", new Font("Aria", 20, FontStyle.Bold), Brushes.Green, YAxeEndPoint[0], YAxeEndPoint[1]);
            g.DrawString("Z", new Font("Aria", 20, FontStyle.Bold), Brushes.Blue, ZAxeEndPoint[0], ZAxeEndPoint[1]);

            //Draw points
            if (is_start_work)
            {
                for (int i = 0; i < count_point; i++)
                {
```

```
            int[] Projected = tr_3D.Transform_point(new float[,] { { generated_point[i, 0] }, { -generated_point[i, 2] }, { -generated_point[i, 1] } });
            // Calculate color for points
            Color col = Color.FromArgb(tr_3D.Calculate_color(new float[,] { { generated_point[i, 0] }, { -generated_point[i, 2] }, { -generated_point[i, 1] } }), 0, 0, 0);
            g.FillEllipse(new SolidBrush(col), Projected[0] - point_size, Projected[1] - point_size, point_size * 2, point_size * 2);
            if (checkBox_n.Checked) g.DrawString((i+1).ToString(), new Font("Aria", 20, FontStyle.Bold), Brushes.Pink, Projected[0], Projected[1]);
          }
        }
        Main_box.Image = bmp;
      }
      public void view_datagreed()
      {
        if (count_point > 0)
        {
          dataGridView1.RowCount = 1;
          for (int i = 0; i < count_point; i++)
          {
            dataGridView1.RowCount += 1;
            dataGridView1.Rows[i].Cells[0].Value = Math.Round(generated_point[i, 0], 2);
            dataGridView1.Rows[i].Cells[1].Value = Math.Round(generated_point[i, 1], 2);
            dataGridView1.Rows[i].Cells[2].Value = Math.Round(generated_point[i, 2], 2);
            if (checkBox_n.Checked) dataGridView1.Rows[i].Cells[3].Value = i+1;
          }
        }
      }
      public void generate_points(bool is_task = false, bool is_noise = false)
      {
        generated_point = new float[count_point, 3];
        float [] sum_generated_point = new float[3];
        for (int i = 0; i < count_point; i++)
        {
          if (!is_task) for (int j = 0; j < 3; j++) generated_point[i, j] = (float)Math.Round((double)random.Next(-100, 100) / 100, 2);
          else
          {
```

```
                //x = 0.7 * cos(6 * pi * (i / count_point))
                generated_point[i, 0] = check_point((float)(0.7f * Math.Cos(6d * Math.PI * (((double)i / count_point)))));
                //y = 0.5 * sin(4 * pi * (i / count_point))
                generated_point[i, 1] = check_point((float)(0.5f * Math.Sin(4d * Math.PI * ((double)i / count_point))));
                //z = -1 + 2 * i / N
                generated_point[i, 2] = check_point((float)(-1f + 2f * (float)i / count_point));
              }
              if (is_noise) for (int j = 0; j < 3; j++) sum_generated_point[j] += generated_point[i, j];
            }
            if (is_noise) for (int i = 0; i < count_point; i++) for (int j = 0; j < 3; j++) generated_point[i, j] = normal_distribution(generated_point[i, j], (float)(sum_generated_point[j]/count_point));
        }
        public float check_point(float x)
        {
            if (x < -1f) return -1f;
            else if (x > 1f) return 1f;
            else return (float)Math.Round(x, 2);
        }
        public float normal_distribution(float a, float mean)
        {
            double u = random.NextDouble();
            double v = random.NextDouble();
            float std_normal = (float)(Math.Sqrt(-2.0d * Math.Log(u)) * Math.Sin(2.0d * Math.PI * v));
            float gen_noise = (float)((float)(noise.Value / 1000) * std_normal) + a + mean;
            return gen_noise;
        }
        public void start_work(bool is_task = false)
        {
            count_point = Int32.Parse(num_point.Text);
            generate_points(is_task, is_noise_change);
            is_start_work = true;
            angleY_bar.Enabled = true;
            angleX_bar.Enabled = true;
            view_datagreed();
            draw_main_img();
        }
```

```
private void Form1_Load(object sender, EventArgs e)
{
   draw_main_img();
}
private void Start_but_Click_1(object sender, EventArgs e)
{
   if (Int32.Parse(num_point.Text) > 0)
   {
      is_generate_task = false;
      start_work();
      timer1.Start();
      generate_var_but.Enabled = false;
      Start_but.Enabled = false;
      Refr_but.Enabled = true;
      Open_but.Enabled = false;
      Save_but.Enabled = true;
   }
}
private void generate_var_but_Click(object sender, EventArgs e)
{
   if (Int32.Parse(num_point.Text) > 0)
   {
      is_generate_task = true;
      start_work(true);
      timer1.Start();
      Start_but.Enabled = false;
      generate_var_but.Enabled = false;
      Refr_but.Enabled = true;
      Open_but.Enabled = false;
      Save_but.Enabled = true;
   }
}
private void noise_ValueChanged(object sender, EventArgs e)
{
   is_noise_change = true;
}
private void Refr_but_Click(object sender, EventArgs e)
{
```

```
        dataGridView1.Rows.Clear();
        dataGridView1.Refresh();
        is_start_work = false;
        tr_3D.angle_y = 0;
        tr_3D.angle_x = 0;
        angleY_bar.Enabled = false;
        angleX_bar.Enabled = false;
        generate_var_but.Enabled = true;
        Start_but.Enabled = true;
        Open_but.Enabled = true;
        Save_but.Enabled = false;
        noise.Value = 0;
        draw_main_img();
    }
    private void timer1_Tick_1(object sender, EventArgs e)
    {
        if (is_start_work && is_noise_change)
        {
            generate_points(is_generate_task, is_noise_change);
            view_datagreed();
            draw_main_img();
            is_noise_change = false;
        }
        else if (is_start_work && (is_rotate_change || is_rotate_scroll_change))
        {
            draw_main_img();
            is_rotate_scroll_change = false;
        }
        else if (is_start_work && is_n_change)
        {
            view_datagreed();
            draw_main_img();
            is_n_change = false;
        }
    }
    private void picture_but_Click(object sender, EventArgs e)
    {
        DialogResult res = saveFileDialog1.ShowDialog();
```

```
            try
            {
                if (res == DialogResult.OK) Main_box.Image.Save(saveFileDialog1.FileName);
            }
            catch
            {
                MessageBox.Show("Something wrong with your picture");
            }
        }
        private void Main_box_MouseDown(object sender, MouseEventArgs e)
        {
            is_rotate_change = true;
            start_mouse_pose = new int[]  {e.X, e.Y};
        }
        private void Main_box_MouseMove(object sender, MouseEventArgs e)
        {
            if (is_rotate_change)
            {
                tr_3D.angle_y  -=  (float)((((double)(start_mouse_pose[0]  -  e.X))  *  (Math.PI  /  6))  /
tr_3D.half_picture_size);
                tr_3D.angle_x  -=  (float)((((double)(start_mouse_pose[1]  -  e.Y))  *  (Math.PI  /  6))  /
tr_3D.half_picture_size);
                start_mouse_pose = new int[] { e.X, e.Y };
                angleY_bar.Value = (int)((180f / (float)Math.PI) * tr_3D.angle_x);
                angleX_bar.Value = (int)((180f / (float)Math.PI) * tr_3D.angle_y);
            }
        }
        private void Main_box_MouseUp(object sender, MouseEventArgs e)
        {
            is_rotate_change = false;
        }
        private void checkBox1_Click(object sender, EventArgs e)
        {
            is_n_change = true;
        }
        private void angleY_bar_ValueChanged(object sender, EventArgs e)
        {
            tr_3D.angle_x = (float)(((float)Math.PI / 180f) * angleY_bar.Value);
```

```
        is_rotate_scroll_change = true;
    }
    private void angleX_bar_ValueChanged(object sender, EventArgs e)
    {
        tr_3D.angle_y = (float)(((float)Math.PI / 180f) * angleX_bar.Value);
        is_rotate_scroll_change = true;
    }
    private void Save_but_Click(object sender, EventArgs e)
    {
        if (count_point > 0)
        {
            DialogResult res = saveFileDialog1.ShowDialog();
            try
            {
                if (res == DialogResult.OK) SaveCSV(saveFileDialog1.FileName);
            }
            catch
            {
                MessageBox.Show("Something wrong with your points");
            }
        }
    }
    public void SaveCSV(string csvPath)
    {
        string data = "X;Y;Z\n";
        for (int i = 0; i < dataGridView1.RowCount - 1; i++)
        {
            data += dataGridView1.Rows[i].Cells[0].Value.ToString() + ";";
            data += dataGridView1.Rows[i].Cells[1].Value.ToString() + ";";
            data += dataGridView1.Rows[i].Cells[2].Value.ToString();
            data += "\n";
        }
        File.WriteAllText(csvPath, data);
    }
    public void OpenCSV(string csvPath)
    {
        string csvContentStr = File.ReadAllText(csvPath);
        string[] vs = csvContentStr.Split('\n');
```

```
            string[] vs2;
            count_point = vs.Length - 2;
            num_point.Text = count_point.ToString();
            generated_point = new float[count_point, 3];
            for (int i = 1; i < count_point + 1; i++)
            {
                vs2 = vs[i].Split(';');
                for (int j = 0; j < 3; j++) generated_point[i-1, j] = float.Parse(vs2[j]);
            }
        }
        private void Open_but_Click(object sender, EventArgs e)
        {
            try
            {
                DialogResult res = openFileDialog1.ShowDialog();
                if (res == DialogResult.OK)
                {
                    OpenCSV(openFileDialog1.FileName);
                    is_start_work = true;
                    angleY_bar.Enabled = true;
                    angleX_bar.Enabled = true;
                    draw_main_img();
                    view_datagreed();
                    timer1.Start();
                    generate_var_but.Enabled = false;
                    Save_but.Enabled = true;
                    Refr_but.Enabled = true;
                }
                else MessageBox.Show("Error, you don't take any file.");
            }
            catch (Exception ex)
            {
                MessageBox.Show("Error, your file have incorrect type. You must take .csv.");
                MessageBox.Show(ex.Message);
            }
        }
    }
}
```