МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«МОСКОВСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»



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 х 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.

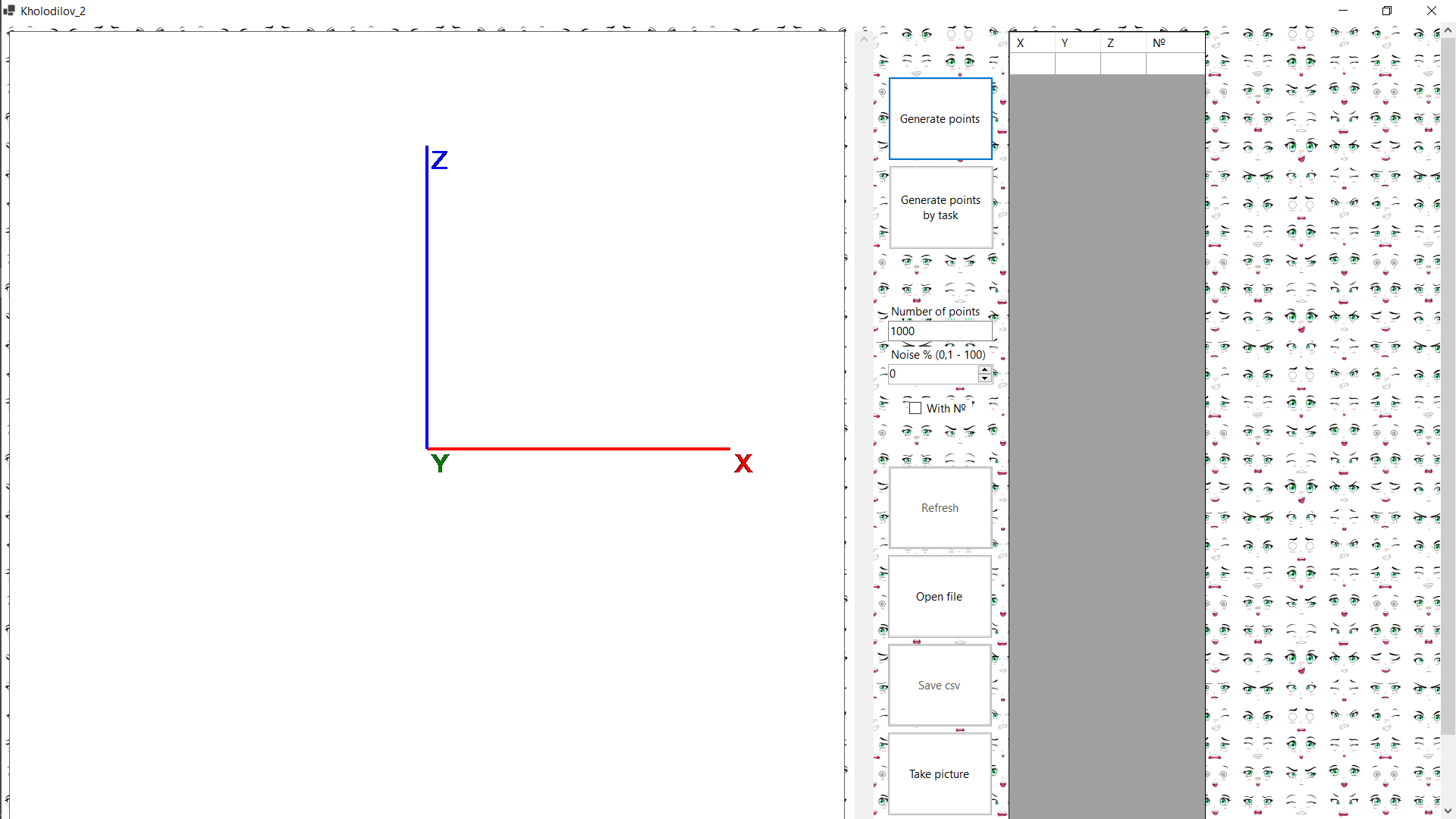


Рисунок 1 - Базовая форма с осями

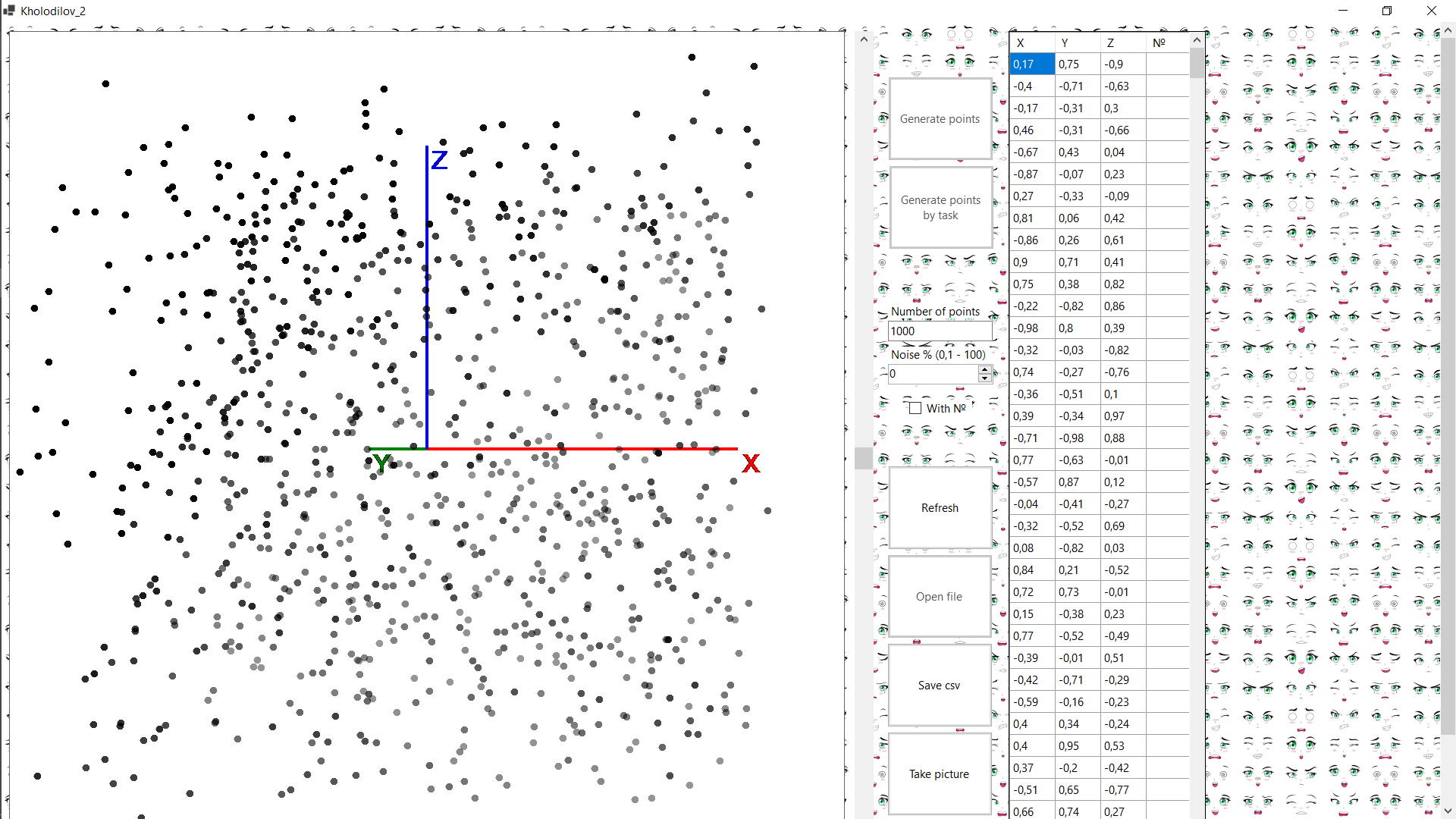


Рисунок 2 - Генерация точек

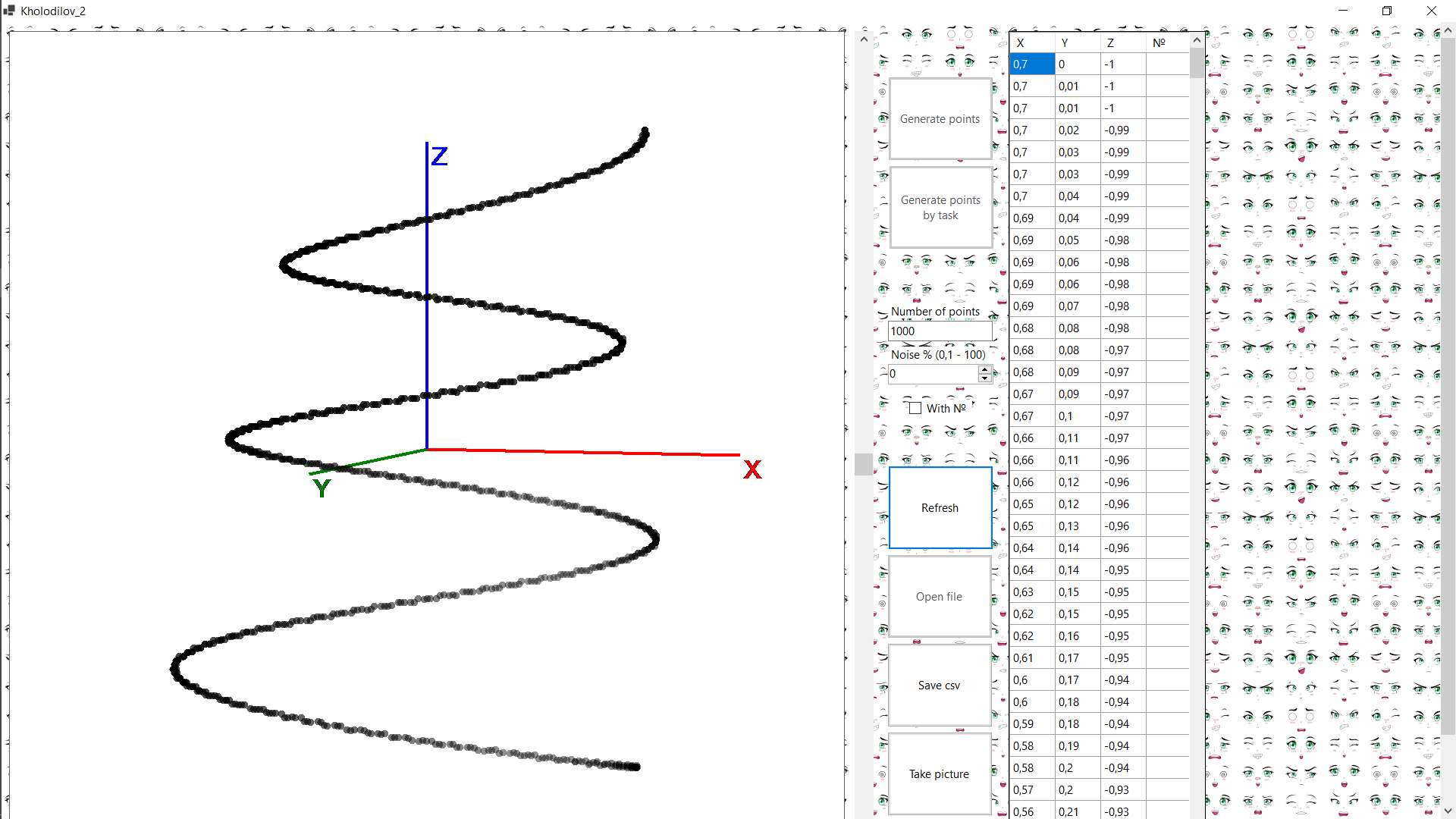


Рисунок 3 - Работа по варианту

* 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].

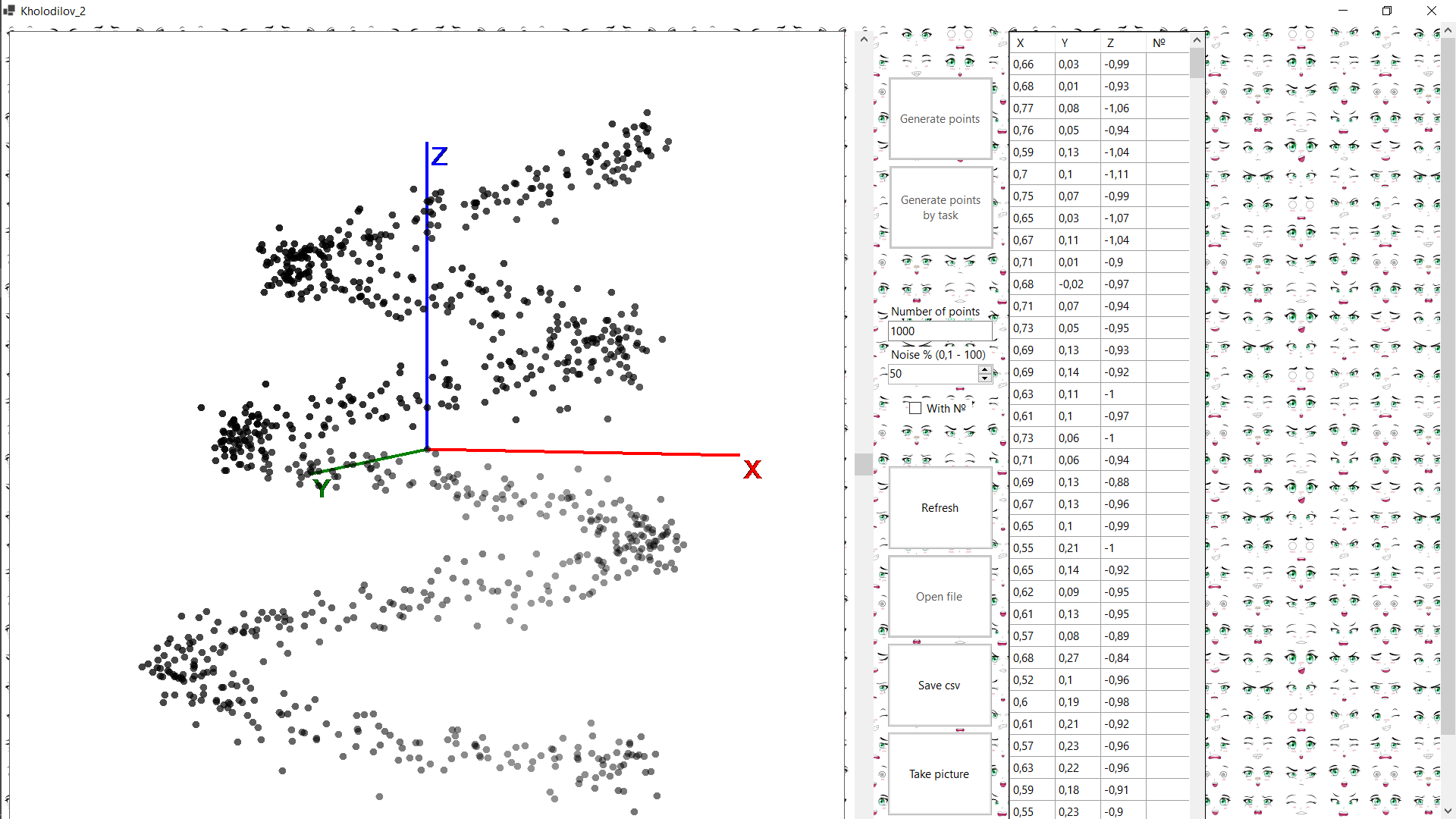


Рисунок 4 - Шумы для данных

* Реализовать сохранение полученного изображения в файл.

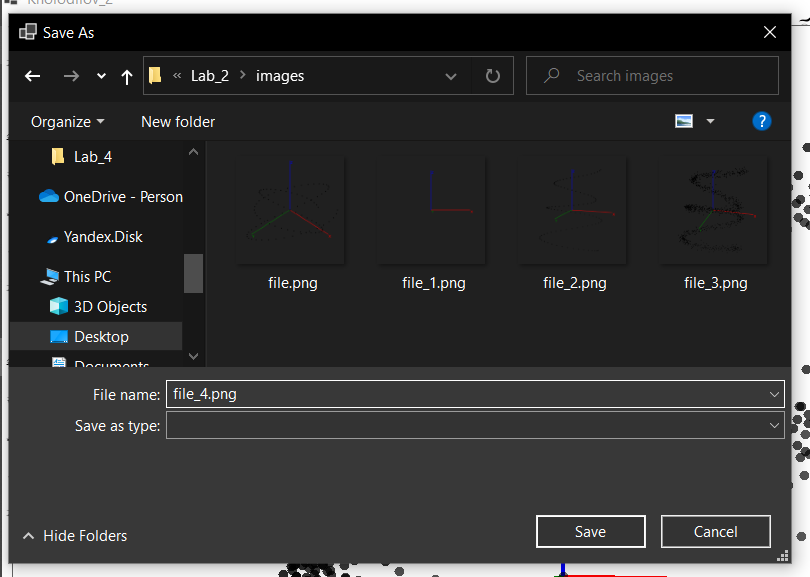


Рисунок 5 - Сохранение изображения

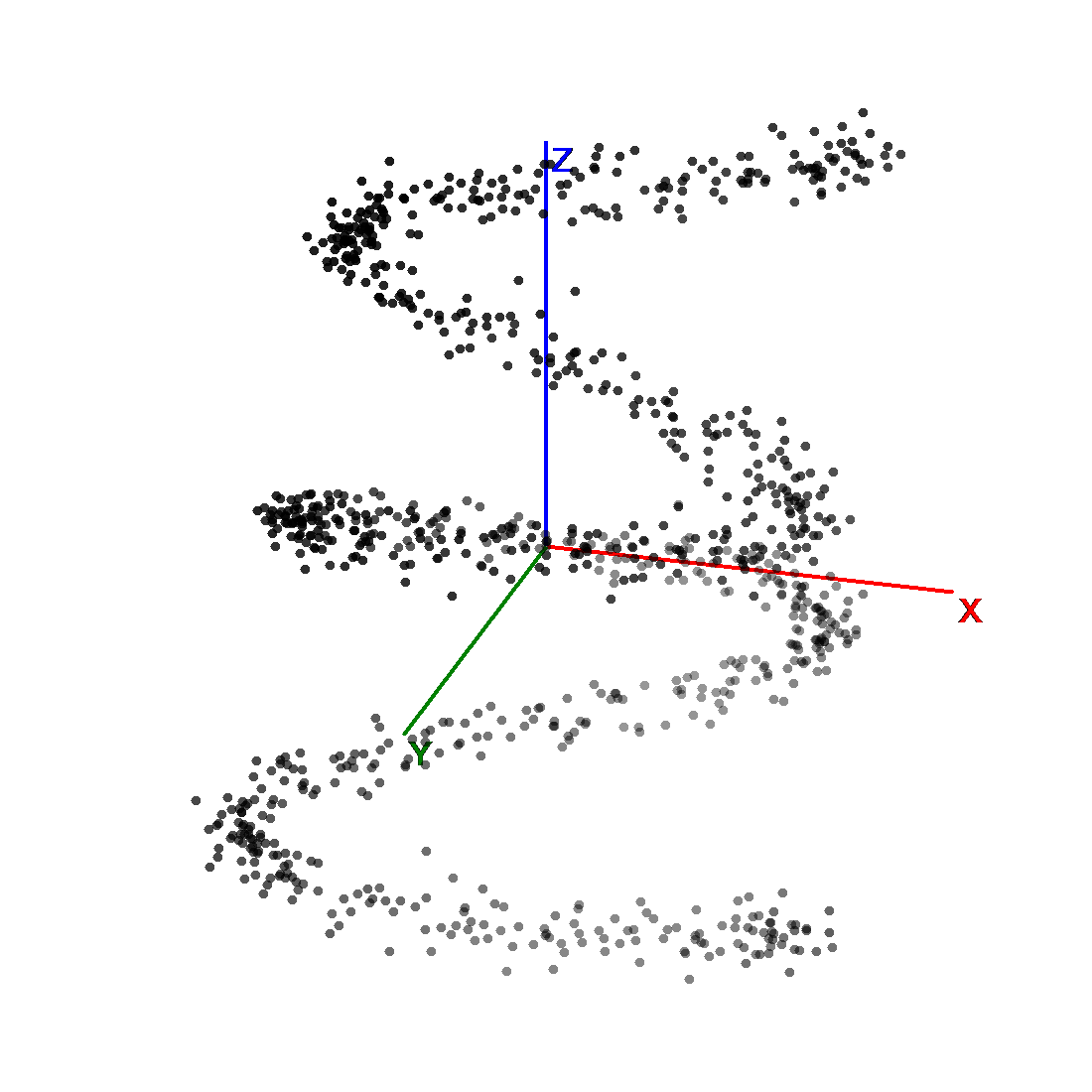


Рисунок 6 - Сохранненое изображение

**Вывод**

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

# Приложение A

Листинг A-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);  }  }  }  } |