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

namespace _2DTransformation
{
    public partial class Form1 : Form
    {

        private PointF[] points; // Polygon points

        public Form1()
        {
            InitializeComponent();
            this.points = new PointF[] {
                                                new PointF(0, 0),
                                                new PointF(110, 40),
                                                new PointF(140, 100),
                                                new PointF(200, 150)
                                                                    };

        }

        // This Pain method is called everytime where the Form loads or, this.Invalidate() or this.Refresh() is called. So, redraw the polygon on the Paint method and
        // call this.Refresh() on button clicks after points are adjusted using various Transformation functions.
        private void Form1_Paint(object sender, PaintEventArgs e)
        {
            Graphics grapics = e.Graphics;
            grapics.DrawPolygon(new Pen(Color.Red), this.points);
            grapics.Dispose();
        }

        private void btn_rotate_Click(object sender, EventArgs e)
        {
            int angle = int.Parse(this.txt_rotate_in.Text); // Get the input angle
            for (int j = 1; j <= angle; j++)
            {
                /* Rotate each point of the polygon by 1 degree around the point 'this.point[3]'  */
                for (int i = 0; i < this.points.Length; i++)
                {
                    rotate_point(ref this.points[i], this.points[3], 1);
                }
                this.Refresh();  // This will call the paint method (Form1_Paint())
                System.Threading.Thread.Sleep(10);
            }
        }


         /* This function rotates a point around a given point.
         * A point that rotates around the origin will have the matrix ((Cos(t) -sin(t)), (Sin(t) Cos(t))).
         * In order to rotate around a given point, first we must bring the piviot to the origin along with the point (bring the line to the origin so that piviot lies with the origin)
         * Then apply the rotation matrix, then move it back to where it was.
         * In order to bring it the origin,
         *         Move the point so that piviot lies on origin -> To do that, Apply the translation matrix T(x)
         *         Rotate around the piviot/origin -> To do that, Apply the rotation matrix R(x)
         *         Move the point back to where it was -> To do that, Apply the translation matrix.
         *
         *         So, if point P is (x, y) then, resulting point after above operations will be, (Remember that the Transformation matrices are applied in reverse order)
         *
         *              | x |        | 1   0  tx | | cos(a)   sin(a)  0 | | 1   0  -tx |
         *              | y |   =    | 0   1  ty |*| -sin(a)  cos(a)  0 |*| 0   1  -ty |
         *              | 1 |        | 0   0  1  | | 0        0       1 | | 0   0   1  |
         *
         *  Then, apply the 2D matrix. Then move the resulting cordinate back to original place by adding the Xp, Yp.
         */

         // 'ref' is to pass the pointer of the 'point'. piviot is the rotation point. 'angle' is the angle to rotate in degrees.
        private void rotate_point(ref PointF point, PointF piviot, int angle)
        {
            double angle_rad = (Math.PI / 180)*angle;    // Convert degree -> radians
            point.X = (float)((Math.Cos(angle_rad) * (point.X - piviot.X)) - (Math.Sin(angle_rad) * (point.Y - piviot.Y)) + piviot.X);
            point.Y = (float)((Math.Sin(angle_rad) * (point.X - piviot.X)) + (Math.Cos(angle_rad) * (point.Y - piviot.Y)) + piviot.Y);
        }

        private void scale_point(ref PointF point, float scale)
        {
            point.X = scale * point.X;
            point.Y = scale * point.Y;
        }

        private void btn_scale_Click(object sender, EventArgs e)
        {
            float scale = float.Parse(txt_scale_in.Text);

            // Scale each point of polygon by a 'scale'
            for (int i = 0; i < this.points.Length; i++)
            {
                scale_point(ref points[i], scale);
            }
            this.Refresh();  // This will call the paint method (Form1_Paint())
        }

        private void btn_translate_Click(object sender, EventArgs e)
        {
            float tx = float.Parse(txt_translate_x_in.Text);
            float ty = float.Parse(txt_translate_y_in.Text);

            // Translate each point of the polygon
            for (int i = 0; i < this.points.Length; i++)
            {
                translate_point(ref points[i], tx, ty);
            }
            this.Refresh();  // This will call the paint method (Form1_Paint())
        }

        private void translate_point(ref PointF point, float tx, float ty)
        {
            point.X += tx;
            point.Y += ty;
        }
    }
}
```