```
//Tilt Shift
//(c) 2012 by Tomasz Lubinski
//www.algorytm.org

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.Drawing.Imaging;

namespace tilt_shift
{
    public partial class Form1 : Form
    {

        /// <summary>
        /// Obraz zrodlowy
        /// </summary>
        private Image zrodlo;
        private int bytesPerPixel;

        /// <summary>
        /// Zainicjuj
        /// </summary>
        public Form1()
        {
            zrodlo = null;
            InitializeComponent();
        }

        /// <summary>
        /// Wczytaj obraz
        /// </summary>
        private void wczytaj_Click(object sender, EventArgs e)
        {
            OpenFileDialog dlg = new OpenFileDialog();
            dlg.Title = "Otwórz obraz";
            dlg.Filter = "pliki jpg (*.jpg)|*.jpg|pliki png
               (*.png)|*.png|wszystkie pliki (*.*)|*.*";
            if (dlg.ShowDialog() == DialogResult.OK)
            {
                zrodlo = new Bitmap(dlg.OpenFile());
                bytesPerPixel =
                   Image.GetPixelFormatSize(zrodlo.PixelFormat) / 8;
```

```
            obraz.Image = new Bitmap(dlg.OpenFile());
            obraz.Height = obraz.Image.Height;
            obraz.Width = obraz.Image.Width;
            this.ClientSize = new
               System.Drawing.Size(Math.Max(obraz.Width + 32,
               374), obraz.Height + 155);
            info.Text = "Wskaż na zdjęciu środek obszaru
               'in-focus'";
        }
        dlg.Dispose();
    }

    /// <summary>
    /// Utworz tablice ze wspolczynnikami do rozmycia
       Gaussowskiego
    /// </summary>
    /// <param name="factor">wspolczynnik rozmycia</param>
    /// <param name="width">maksymalna wielkosc tablicy</param>
    /// <returns>tablica ze wspolczynnikami do rozmycia
       Gaussowskiego</returns>
    private double[] gaussianArray(double factor, int width)
    {
        double[] result = new double[width];
        for (var i=0; i<width; i++)
        {
            result[i] = (1 / (Math.Sqrt(2.0 * Math.PI) *
               factor)) * Math.Exp(-(i * i) / (2.0 * factor *
               factor));
            if (result[i] < 0.003)
            {
                double[] subArray = new double[i];
                for (int j = 0; j < i; j++)
                {
                    subArray[j] = result[j];
                }
                return subArray;
            }
        }
        return result;
    }

    /// <summary>
    /// Rozmyj pixel w poziomie
    /// </summary>
    /// <param name="y">indeks linii do rozmycia</param>
    /// <param name="blurArray">tablica wspolczynnikow
       rozmycia</param>
```

```
/// <param name="pixelValues">tablica z wartosciami
   pikseli</param>
/// <param name="stride">szerokosc linii danych z
   pikselami</param>
/// <param name="width">szerokosc obrazu</param>
private void blurHorizontal(int y, double[] blurArray,
   byte[] pixelValues, int stride, int width)
{
    double[] pixelBytes = new double[bytesPerPixel];
    for (var x = 0; x < width; x++)
    {
        int index = y * stride + x * bytesPerPixel;
        for (int i = 0; i < bytesPerPixel; i++)
        {
            pixelBytes[i] = pixelValues[index + i] *
               blurArray[0];
        }
        double sum = blurArray[0];
        for (var k = 1; k < blurArray.Length; k++)
        {
            if (x + k >= width)
            {
                index = y * stride + (x + k + 1 - width) *
                   bytesPerPixel;
            }
            else
            {
                index = y * stride + (x + k) * bytesPerPixel;
            }
            for (int i = 0; i < bytesPerPixel; i++)
            {
                pixelBytes[i] += pixelValues[index + i] *
                   blurArray[k];
            }
            if (x - k < 0)
            {
                index = y * stride + Math.Abs(x - k) *
                   bytesPerPixel;
            }
            else
            {
                index = y * stride + (x - k) * bytesPerPixel;
            }
            for (int i = 0; i < bytesPerPixel; i++)
            {
                pixelBytes[i] += pixelValues[index + i] *
                   blurArray[k];
            }
```

```
                sum += 2 * blurArray[k];
            }
            index = y * stride + x * bytesPerPixel;
            for (int i = 0; i < bytesPerPixel; i++)
            {
                pixelValues[index + i] = (byte)(pixelBytes[i] /
                    sum);
            }
        }
    }

    /// <summary>
    /// Rozmyj pixel w pionie
    /// </summary>
    /// <param name="y">indeks linii do rozmycia</param>
    /// <param name="blurArray">tablica wspolczynnikow
        rozmycia</param>
    /// <param name="pixelValues">tablica z wartosciami
        pikseli</param>
    /// <param name="stride">szerokosc linii danych z
        pikselami</param>
    /// <param name="width">szerokosc obrazu</param>
    private void blurVertical(int y, double[] blurArray, byte[]
        pixelValues, int stride, int width, int height)
    {
        double[] pixelBytes = new double[bytesPerPixel];
        for (var x = 0; x < width; x++)
        {
            int index = y * stride + x * bytesPerPixel;
            for (int i = 0; i < bytesPerPixel; i++)
            {
                pixelBytes[i] = pixelValues[index + i] *
                    blurArray[0];
            }
            double sum = blurArray[0];
            for (var k = 1; k < blurArray.Length; k++)
            {
                if (y + k >= height)
                {
                    index = (y + k + 1 - height) * stride + x *
                        bytesPerPixel;
                }
                else
                {
                    index = (y + k) * stride + x * bytesPerPixel;
                }
                for (int i = 0; i < bytesPerPixel; i++)
                {
```

```
                    pixelBytes[i] += pixelValues[index + i] *
                        blurArray[k];
                }
                if (y - k < 0)
                {
                    index = Math.Abs(y - k) * stride + x *
                        bytesPerPixel;
                }
                else
                {
                    index = (y - k) * stride + x * bytesPerPixel;
                }
                for (int i = 0; i < bytesPerPixel; i++)
                {
                    pixelBytes[i] += pixelValues[index + i] *
                        blurArray[k];
                }
                sum += 2 * blurArray[k];
            }
            index = y * stride + x * bytesPerPixel;
            for (int i = 0; i < bytesPerPixel; i++)
            {
                pixelValues[index + i] = (byte)(pixelBytes[i] /
                    sum);
            }
        }
    }

    /// <summary>
    /// Efekt Tilt-Shift
    /// </summary>
    private void obraz_MouseDown(object sender, MouseEventArgs
        e)
    {
        //Nie rob nic jezeli obraz jest jeszcze niewczytany
        if (zrodlo == null)
        {
            return;
        }

        //Kopiuj obrazek zrodlowy
        Bitmap bitmap = (Bitmap)zrodlo.Clone();
        obraz.Image = bitmap;

        //Pobierz liczbe bajtow na punkt obrazu
        int bytesPerPixel =
            Image.GetPixelFormatSize(bitmap.PixelFormat) / 8;
        double[] pixelBytes = new double[bytesPerPixel];
```

```
        //Pobierz wartosc wszystkich punktow obrazu
        BitmapData bmpData = bitmap.LockBits(new Rectangle(0, 0,
           obraz.Width, obraz.Height), ImageLockMode.ReadWrite,
           bitmap.PixelFormat);
        byte[] pixelValues = new byte[Math.Abs(bmpData.Stride)
           * obraz.Height];
        System.Runtime.InteropServices.Marshal.Copy(bmpData.Scan
           0, pixelValues, 0, pixelValues.Length);

        //Dane obszarów
        int width = obraz.Width;
        int height = obraz.Height;
        int inFocusHeight = int.Parse(this.inFocusHeight.Text);
        int outFocusStartTop = Math.Max(e.Y - inFocusHeight / 2,
           0);
        int outFocusStartDown = Math.Min(e.Y + inFocusHeight /
           2, height);
        double outFocusHeight = e.Y > height / 2 ?
           outFocusStartTop : (height - outFocusStartDown);
        double outFocusBlur = double.Parse(blur.Text) - 0.1;

        //Rozmyj w poziomie obraz nad obszarem ostrym
        for (int y=0; y<outFocusStartTop; y++)
        {
            double factor = (outFocusStartTop-y)/outFocusHeight;
            double[] blurArray = gaussianArray(factor *
               outFocusBlur + 0.1, width);
            blurHorizontal(y, blurArray, pixelValues,
               bmpData.Stride, width);
        }

        //Rozmyj w poziomie obraz pod obszarem ostrym
        for (int y = outFocusStartDown; y < height; y++)
        {
            double factor = (y - outFocusStartDown) /
               outFocusHeight;
            double[] blurArray = gaussianArray(factor *
               outFocusBlur + 0.1, width);
            blurHorizontal(y, blurArray, pixelValues,
               bmpData.Stride, width);
        }

        //Rozmyj w pionie obraz nad obszarem ostrym
        for (int y = 0; y < outFocusStartTop; y++)
        {
            double factor = (outFocusStartTop - y) /
               outFocusHeight;
```

```
                double[] blurArray = gaussianArray(factor *
                    outFocusBlur + 0.1, width);
                blurVertical(y, blurArray, pixelValues,
                    bmpData.Stride, width, height);
            }

            //Rozmyj w pionie obraz pod obszarem ostrym
            for (int y = outFocusStartDown; y < height; y++)
            {
                double factor = (y - outFocusStartDown) /
                    outFocusHeight;
                double[] blurArray = gaussianArray(factor *
                    outFocusBlur + 0.1, width);
                blurVertical(y, blurArray, pixelValues,
                    bmpData.Stride, width, height);
            }

            System.Runtime.InteropServices.Marshal.Copy(pixelValues,
                0, bmpData.Scan0, pixelValues.Length);
            bitmap.UnlockBits(bmpData);
        }
    }
}
```