

Systemy Wizyjne

Projekt – prosty edytor zdjęć

Realizujący projekt – Jakub Siejak, nr indeksu:-----

Poniższa dokumentacja opisuje funkcjonalność oraz wady stworzonego oprogramowania. Podczas implementacji nie zostały użyte biblioteki zewnętrzne, ani obrazy do których nie posiadam praw autorskich. W trakcie tworzenia dokumentacji posłużę się także fotografią wykonaną własnoręcznie.

Oprogramowanie (wersja z kodem) również dostępne jest pod wskazanymi poniżej dwoma linkami:

GitHub:

https://github.com/Kinoruu/Systemy_Wizyjne_projekt

Google Drive:

https://drive.google.com/drive/folders/1E42dqEzoIzAkY_ZV0l0tTf8SnB23dhc?usp=sharing

Projekt – oprogramowanie wraz z zrzutami ekranu:

W celu obróbki obrazu zostały zaimplementowane dwie wbudowane biblioteki:

```
10 10  using System.Drawing.Imaging;
11 11  using System.IO;
12 12
```

Cały kod projektu znajduje się w pliku Form1.cs dlatego zostaną tu wklejone poszczególne funkcje:

```
23  void openImage() //funkcja otwierająca obraz
24  {
25      DialogResult dr = openFileDialog1.ShowDialog();
26      if (dr == DialogResult.OK)
27      {
28          file = Image.FromFile(openFileDialog1.FileName);
29          newBitmap = new Bitmap(openFileDialog1.FileName);
30          pictureBox1.Image = file;
31          opened = true;
32      }
33  }
```

```
35     void saveImage() //funkcja zapisująca obraz
36     {
37         if (opened)
38         {
39             SaveFileDialog sfd = new SaveFileDialog();
40             sfd.Filter = "Images|*.png;*.bmp;*.jpg";
41             ImageFormat format = ImageFormat.Png; // z góry ustalony format to png gdyż nie tracimy wtedy na jakości
42             if (sfd.ShowDialog() == System.Windows.Forms.DialogResult.OK)
43             {
44                 string ext = Path.GetExtension(sfd.FileName);
45                 switch (ext)
46                 {
47                     case ".jpg":
48                         format = ImageFormat.Jpeg;
49                         break;
50                     case ".bmp":
51                         format = ImageFormat.Bmp;
52                         break;
53                 }
54                 pictureBox1.Image.Save(sfd.FileName, format);
55             }
56         }
57         else { MessageBox.Show("No image loaded, first upload image "); }
58     }
59 
```

```
60     void hue() //funkcja zmieniająca wartości RGB obrazu
61     {
62         float changered = trackBar1.Value * 0.1f;
63         float changegreen = trackBar2.Value * 0.1f;
64         float changeblue = trackBar3.Value * 0.1f;
65
66         label7.Text = (changered*10).ToString();
67         label9.Text = (changeblue*10).ToString();
68         label8.Text = (changegreen*10).ToString();
69
70         reload();
71         if (!opened)
72         {
73         }
74         else
75         {
76             Image img = pictureBox1.Image;
77             Bitmap bmpInverted = new Bitmap(img.Width, img.Height);
78
79             ImageAttributes ia = new ImageAttributes();
80             ColorMatrix cmPicture = new ColorMatrix(new float[][] {
81                 {
82                     new float[]{1+changeder, 0, 0, 0, 0},
83                     new float[]{0, 1-changegreen, 0, 0, 0},
84                     new float[]{0, 0, 1-changeblue, 0, 0},
85                     new float[]{0, 0, 0, 1, 0},
86                     new float[]{0, 0, 0, 0, 1}
87                 }
88             });
89             iaSetColorMatrix(cmPicture);
90             Graphics g = Graphics.FromImage(bmpInverted);
91
92             g.DrawImage(img, new Rectangle(0, 0, img.Width, img.Height), 0, 0, img.Width, img.Height, GraphicsUnit.Pixel, ia);
93             g.Dispose();
94             pictureBox1.Image = bmpInverted;
95         }
96     }
97 
```

```
98     void brightness() //funkcja zmieniająca jasność obrazu
99     {
100         float changebrightness = trackBar6.Value / 255.0f;
101
102         int val = (int)(changebrightness * 255);
103
104         label10.Text = val.ToString();
105
106         reload();
107         if (!opened)
108         {
109         }
110         else
111         {
112             Image img = pictureBox1.Image;
113             Bitmap bmpInverted = new Bitmap(img.Width, img.Height);
114
115             ImageAttributes ia = new ImageAttributes();
116             ColorMatrix cmPicture = new ColorMatrix(new float[][][]
117             {
118                 new float[]{1, 0, 0, 0, 0},
119                 new float[]{0, 1, 0, 0, 0},
120                 new float[]{0, 0, 1, 0, 0},
121                 new float[]{0, 0, 0, 1, 0},
122                 new float[]{changebrightness, changebrightness, changebrightness, 1, 1 }
123             });
124         };
125         iaSetColorMatrix(cmPicture);
126         Graphics g = Graphics.FromImage(bmpInverted);
127
128         g.DrawImage(img, new Rectangle(0, 0, img.Width, img.Height), 0, 0, img.Width, img.Height, GraphicsUnit.Pixel, ia);
129         g.Dispose();
130         pictureBox1.Image = bmpInverted;
131
132     }
133
134 }
135 }
```

```
136     void contrast() //funkcja zmieniająca kontrast obrazu
137     {
138         float changecontrast = trackBar5.Value * 0.041f;
139
140         int val = (int)(changecontrast * 24.4);
141
142         label11.Text = val.ToString();
143
144         reload();
145         if (!opened)
146         {
147         }
148         else
149         {
150             Image img = pictureBox1.Image;
151             Bitmap bmpInverted = new Bitmap(img.Width, img.Height);
152
153             ImageAttributes ia = new ImageAttributes();
154             ColorMatrix cmPicture = new ColorMatrix(new float[][][]
155             {
156                 new float[]{ changecontrast, 0, 0, 0, 0 },
157                 new float[]{0, changecontrast, 0, 0, 0 },
158                 new float[]{0, 0, changecontrast, 0, 0 },
159                 new float[]{0, 0, 0, 1f, 0 },
160                 new float[]{0.001f, 0.001f, 0.001f, 1, 1f }
161             });
162         };
163         iaSetColorMatrix(cmPicture);
164         Graphics g = Graphics.FromImage(bmpInverted);
165
166         g.DrawImage(img, new Rectangle(0, 0, img.Width, img.Height), 0, 0, img.Width, img.Height, GraphicsUnit.Pixel, ia);
167         g.Dispose();
168         pictureBox1.Image = bmpInverted;
169
170     }
171
172 }
```

```
174     void reload() //funkcja ładująca ponownie oryginalny obraz
175     {
176         if (!opened)
177         {
178             MessageBox.Show("Open an Image then apply changes");
179         }
180         else
181         {
182             if (opened)
183             {
184                 file = Image.FromFile(openFileDialog1.FileName);
185                 newBitmap = new Bitmap(openFileDialog1.FileName);
186                 pictureBox1.Image = file;
187                 opened = true;
188             }
189         }
190     }
```

```
192     void grayscale() //funkcja zmieniająca kolory w odcieniu szarości
193     {
194         if (!opened)
195         {
196             MessageBox.Show("Open an Image then apply changes");
197         }
198         else
199         {
200             trackBar1.Value = 0;
201             trackBar2.Value = 0;
202             trackBar3.Value = 0;
203             Image img = pictureBox1.Image;
204             Bitmap bmpInverted = new Bitmap(img.Width, img.Height);
205
206             ImageAttributes ia = new ImageAttributes();
207             ColorMatrix cmPicture = new ColorMatrix(new float[][][]
208             {
209                 new float[]{0.299f, 0.299f, 0.299f, 0, 0},
210                 new float[]{0.587f, 0.587f, 0.587f, 0, 0},
211                 new float[]{0.114f, 0.114f, 0.114f, 0, 0},
212                 new float[]{0, 0, 0, 1, 0},
213                 new float[]{0, 0, 0, 0, 0}
214             });
215             iaSetColorMatrix(cmPicture);
216             Graphics g = Graphics.FromImage(bmpInverted);
217
218             g.DrawImage(img, new Rectangle(0, 0, img.Width, img.Height), 0, 0, img.Width, img.Height, GraphicsUnit.Pixel, ia);
219
220             g.Dispose();
221             pictureBox1.Image = bmpInverted;
222         }
223     }
```

```
226     void blur() // funkcja rozmywająca obraz poprzez uśrednianie wartości pikseli
227     {
228         if (!opened)
229         {
230             MessageBox.Show("Open an Image then apply changes");
231         }
232         else
233         {
234             for (int x = 1; x < newBitmap.Width; x++)
235             {
236                 for (int y = 1; y < newBitmap.Height; y++)
237                 {
238                     try
239                     {
240                         Color prevX = newBitmap.GetPixel(x - 1, y);
241                         Color nextX = newBitmap.GetPixel(x + 1, y);
242                         Color prevY = newBitmap.GetPixel(x, y - 1);
243                         Color nextY = newBitmap.GetPixel(x, y + 1);
244
245                         int avgR = (int)((prevX.R + nextX.R + prevY.R + nextY.R) / 4);
246                         int avgG = (int)((prevX.G + nextX.G + prevY.G + nextY.G) / 4);
247                         int avgB = (int)((prevX.B + nextX.B + prevY.B + nextY.B) / 4);
248
249                         newBitmap.SetPixel(x, y, Color.FromArgb(avgR, avgG, avgB));
250                     }
251                     catch (Exception) { }
252                 }
253             }
254         }
255     }
256 }
```

```
258     void invert() // funkcja odwracająca kolory(negatyw)
259     {
260         for (int x = 1; x < newBitmap.Width; x++)
261         {
262             for (int y = 1; y < newBitmap.Height; y++)
263             {
264                 try
265                 {
266                     Color pixel = newBitmap.GetPixel(x, y);
267
268                     int red = pixel.R;
269                     int green = pixel.G;
270                     int blue = pixel.B;
271
272                     newBitmap.SetPixel(x, y, Color.FromArgb(255 - red, 255 - green, 255 - blue));
273                 }
274                 catch (Exception) { }
275             }
276         }
277     }
278 }
```

```
280     void edge() //funkcja znajdujaca krawedzie na podstawie rownicy wartosci pikseli( imat
281     {
282         Bitmap nB = new Bitmap(newBitmap.Width, newBitmap.Height);
283
284         for (int x = 1; x <= newBitmap.Width - 1; x++)
285         {
286             for (int y = 1; y <= newBitmap.Height - 1; y++)
287             {
288                 nB.SetPixel(x, y, Color.DarkGray);
289             }
290         }
291         for (int x = 1; x <= newBitmap.Width - 1; x++)
292         {
293             for (int y = 1; y <= newBitmap.Height - 1; y++)
294             {
295                 try
296                 {
297                     Color pixel = newBitmap.GetPixel(x, y);
298
299                     int colVal = (pixel.R + pixel.G + pixel.B);
300
301                     if(lastCol == 0) lastCol = (pixel.R + pixel.G + pixel.B);
302
303                     int diff;
304
305                     if(colVal > lastCol)
306                     {
307                         diff = colVal - lastCol;
308                     }
309                     else
310                     {
311                         diff = lastCol - colVal;
312                     }
313
314                     if(diff > 100)
315                     {
316                         nB.SetPixel(x, y, Color.Gray);
317                         lastCol = colVal;
318                     }
319
320                 }
321                 catch (Exception) { }
322             }
323         }
324         for (int y = 1; y <= newBitmap.Height - 1; y++)
325         {
326             try
327             {
328                 Color pixel = newBitmap.GetPixel(x, y);
329
330                 int colVal = (pixel.R + pixel.G + pixel.B);
331
332                 if (lastCol == 0) lastCol = (pixel.R + pixel.G + pixel.B);
333
334                 int diff;
335
336                 if (colVal > lastCol)
337                 {
338                     diff = colVal - lastCol;
339                 }
340                 else
341                 {
342                     diff = lastCol - colVal;
343                 }
344
345                 if (diff > 100)
346                 {
347                     nB.SetPixel(x, y, Color.Gray);
348                     lastCol = colVal;
349                 }
350
351             }
352             catch (Exception) { }
353         }
354     }
355 }
```

```
358     void mirror_l() //funkcja odbijająca lewą połowę obrazu
359     {
360         for (int xl = 1, xr = newBitmap.Width; xl < newBitmap.Width; xl++, xr--)
361         {
362             for (int y = 1; y < newBitmap.Height; y++)
363             {
364                 try
365                 {
366                     Color pixel = newBitmap.GetPixel(xl, y);
367
368                     newBitmap.SetPixel(xr, y, pixel);
369
370                 }
371                 catch (Exception) {}
372             }
373         }
374         pictureBox1.Image = newBitmap;
375     }
376 }
```

```
377     void mirror_r() //funkcja odbijająca prawą połowę obrazu
378     {
379         for (int xl = 1, xr = newBitmap.Width; xl < newBitmap.Width; xl++, xr--)
380         {
381             for (int y = 1; y < newBitmap.Height; y++)
382             {
383                 try
384                 {
385                     Color pixel = newBitmap.GetPixel(xr, y);
386
387                     newBitmap.SetPixel(xl, y, pixel);
388
389                 }
390                 catch (Exception) {}
391             }
392         }
393         pictureBox1.Image = newBitmap;
394     }
```

```
395     void mirror_t() ... //funkcja odbijająca górną połowę obrazu
396     {
397         for (int x = 1; x < newBitmap.Width; x++)
398         {
399             for (int yt = 1, yb = newBitmap.Height; yt < newBitmap.Height; yt++, yb--)
400             {
401                 try
402                 {
403                     Color pixel = newBitmap.GetPixel(x, yt);
404                     newBitmap.SetPixel(x, yb, pixel);
405                 }
406                 catch (Exception) { }
407             }
408         }
409     }
410     pictureBox1.Image = newBitmap;
411 }
```

```
413     void mirror_b() ... //funkcja odbijająca dolną połowę obrazu
414     {
415         for (int x = 1; x < newBitmap.Width; x++)
416         {
417             for (int yt = 1, yb = newBitmap.Height; yt < newBitmap.Height; yt++, yb--)
418             {
419                 try
420                 {
421                     Color pixel = newBitmap.GetPixel(x, yb);
422                     newBitmap.SetPixel(x, yt, pixel);
423                 }
424                 catch (Exception) { }
425             }
426         }
427     }
428     pictureBox1.Image = newBitmap;
429 }
```

Funkcjonalność zajmująca się reagowaniem na przyciski oraz suwaki zostanie wklejona ze względu na dużą objętość:

```
private void pictureBox1_Click(object sender, EventArgs e)
{
}

private void button9_Click(object sender, EventArgs e)
{
    openImage();
}

private void button10_Click(object sender, EventArgs e)
{
    saveImage();
}

private void button1_Click(object sender, EventArgs e)
{
    reload();
    trackBar1.Value = 0;
    trackBar2.Value = 0;
    trackBar3.Value = 0;
```

```
        trackBar5.Value = 0;
        trackBar6.Value = 0;
    }

    private void button2_Click(object sender, EventArgs e)
    {
        reload();
        grayscale();
    }

    private void trackBar1_ValueChanged(object sender, EventArgs e)
    {
        hue();
    }

    private void trackBar2_ValueChanged(object sender, EventArgs e)
    {
        hue();
    }

    private void trackBar3_ValueChanged(object sender, EventArgs e)
    {
        hue();
    }

    private void trackBar5_ValueChanged(object sender, EventArgs e)
    {
        contrast();
    }

    private void trackBar6_ValueChanged(object sender, EventArgs e)
    {
        brightness();
    }

    private void button3_Click(object sender, EventArgs e)
    {
        blur();
    }

    private void button4_Click(object sender, EventArgs e)
    {
        invert();
    }

    private void button5_Click(object sender, EventArgs e)
    {
        edge();
    }

    private void button6_Click(object sender, EventArgs e)
    {
        mirror_l();
    }

    private void button7_Click(object sender, EventArgs e)
    {
        mirror_r();
    }
```

```
private void button8_Click(object sender, EventArgs e)
{
    mirror_t();
}

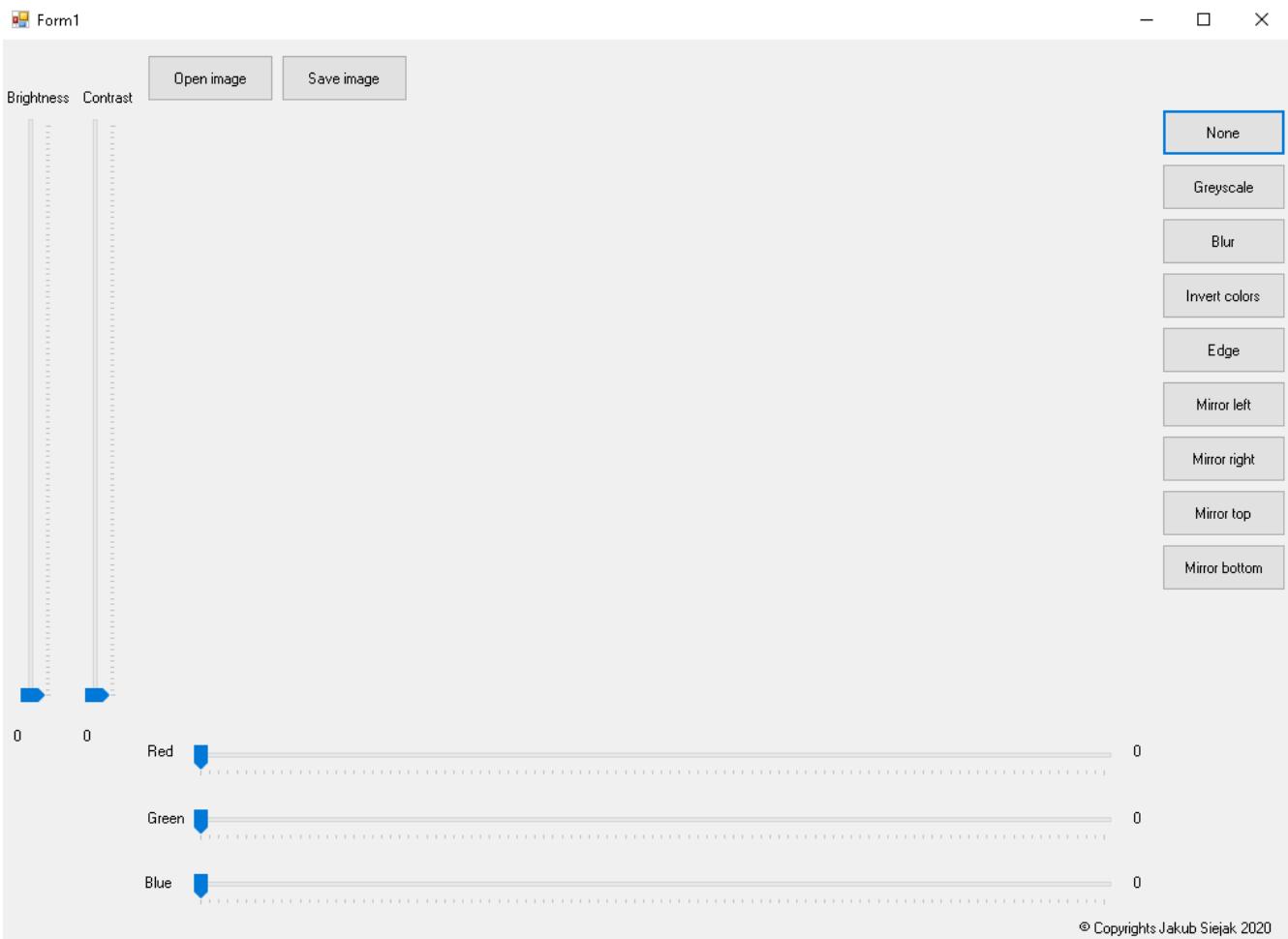
private void button11_Click(object sender, EventArgs e)
{
    mirror_b();
}
```

Krótki opis kodu:

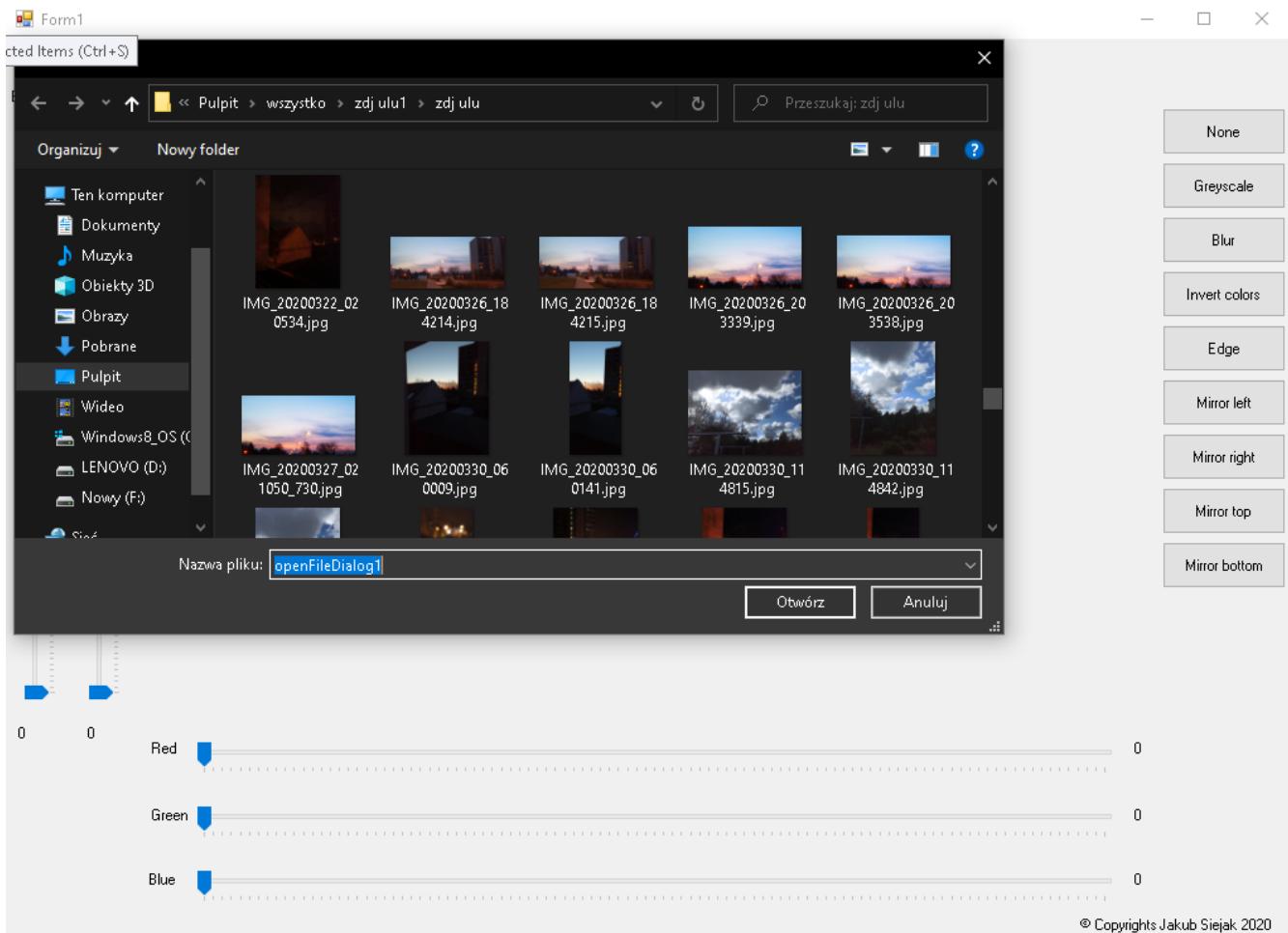
Form.cs

- implementacja bibliotek: “System.Drawing.Imaging” oraz “System.IO”,
- implementacja zmiennych: “newBitmap”, “file”, “lastCol”, “opened”,
- implementacja wszystkich funkcji,
- implementacja reakcji na przyciski oraz suwaki.

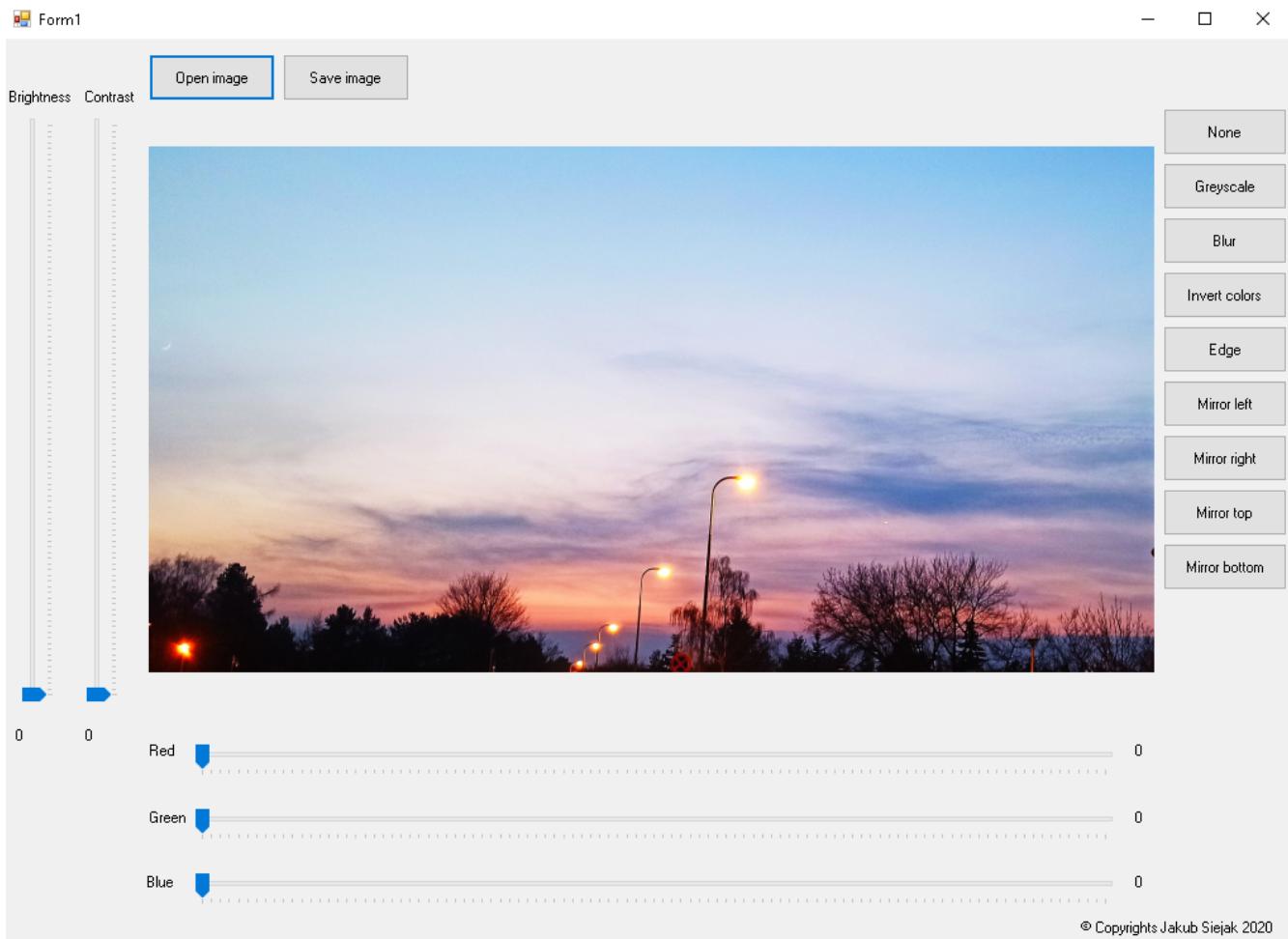
Zrzuty ekranu aplikacji wraz z opisem działania oraz jej wad:



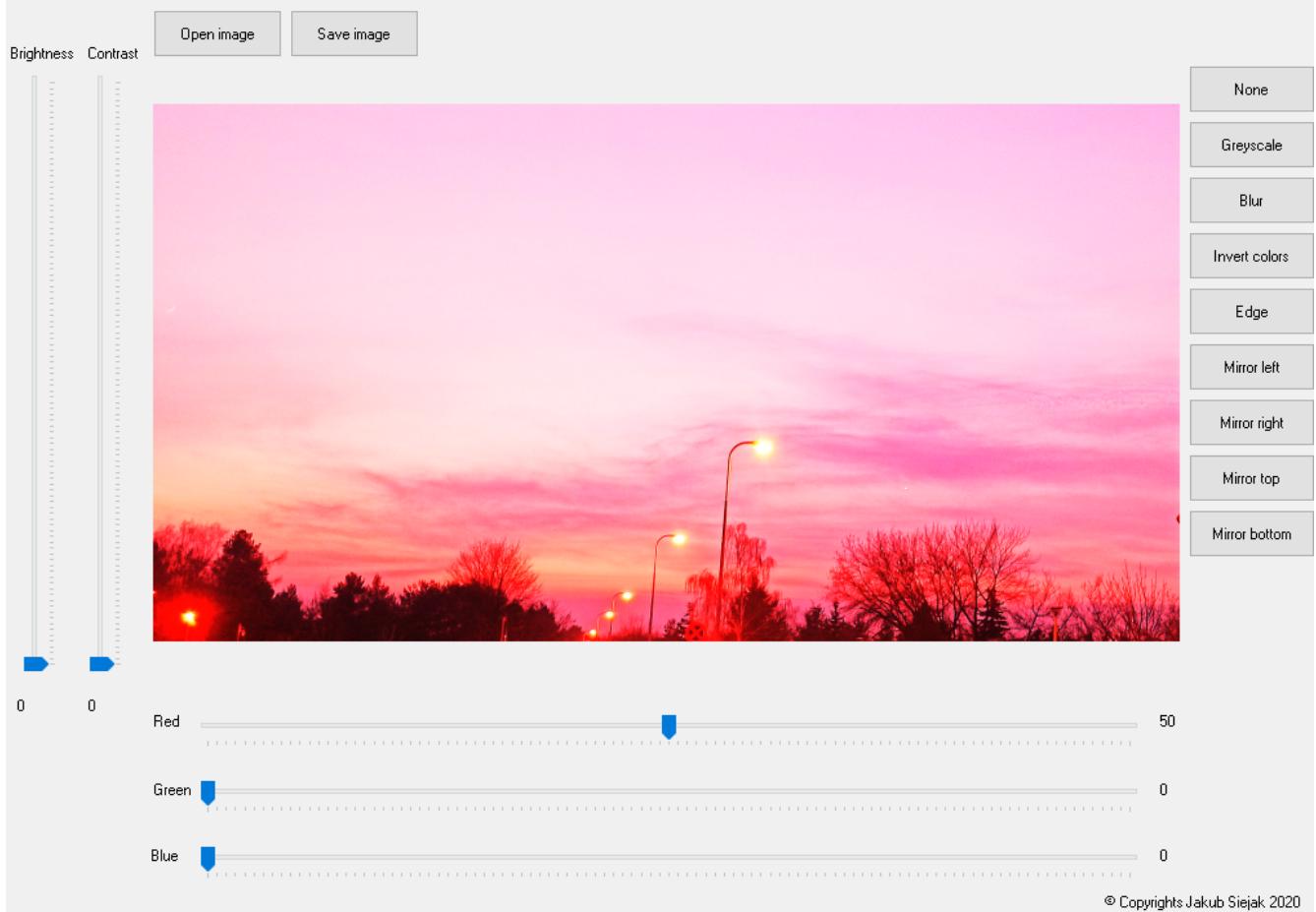
Ekran startowy



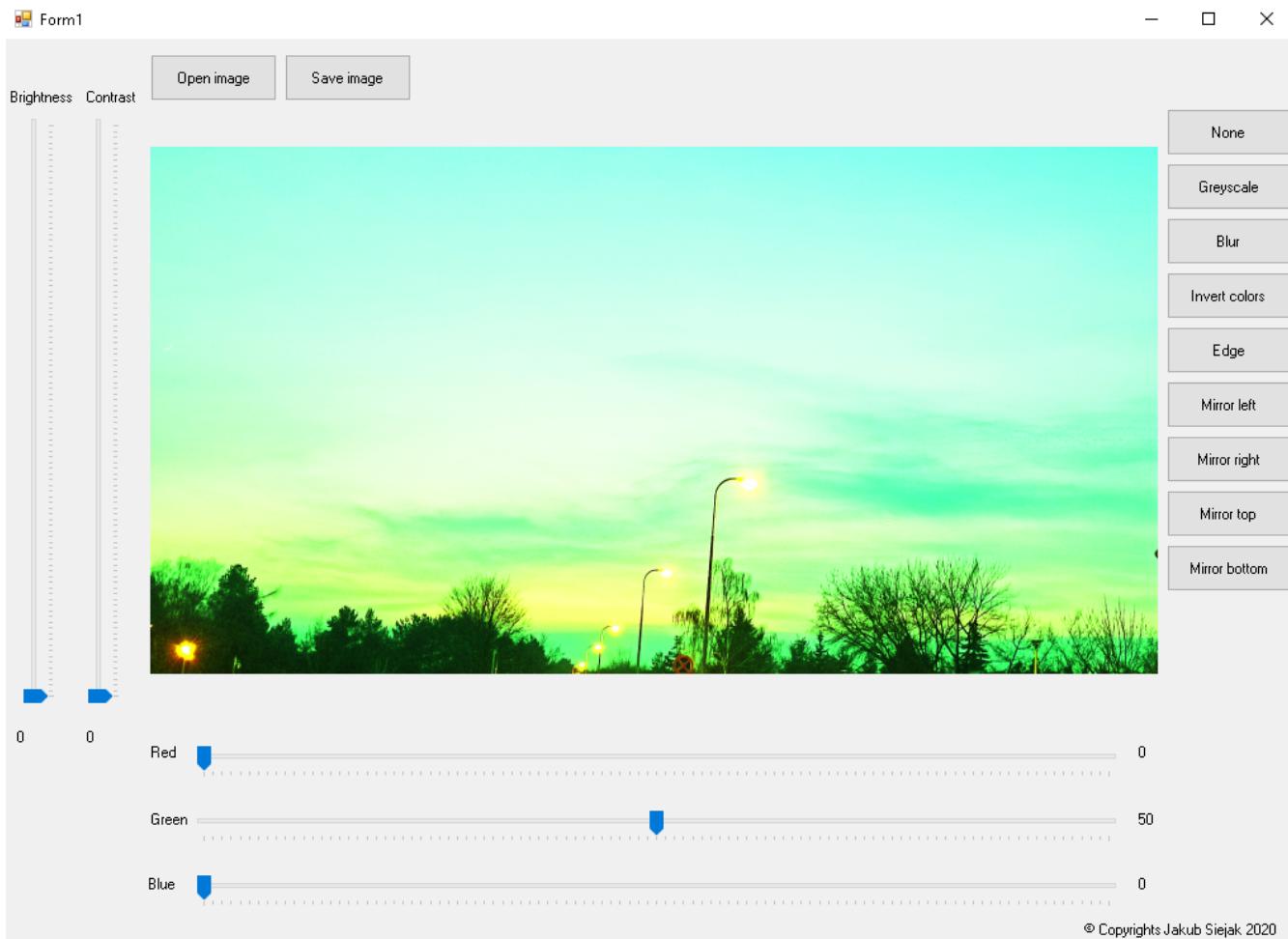
Ekran podczas wczytywania zdjęcia z komputera



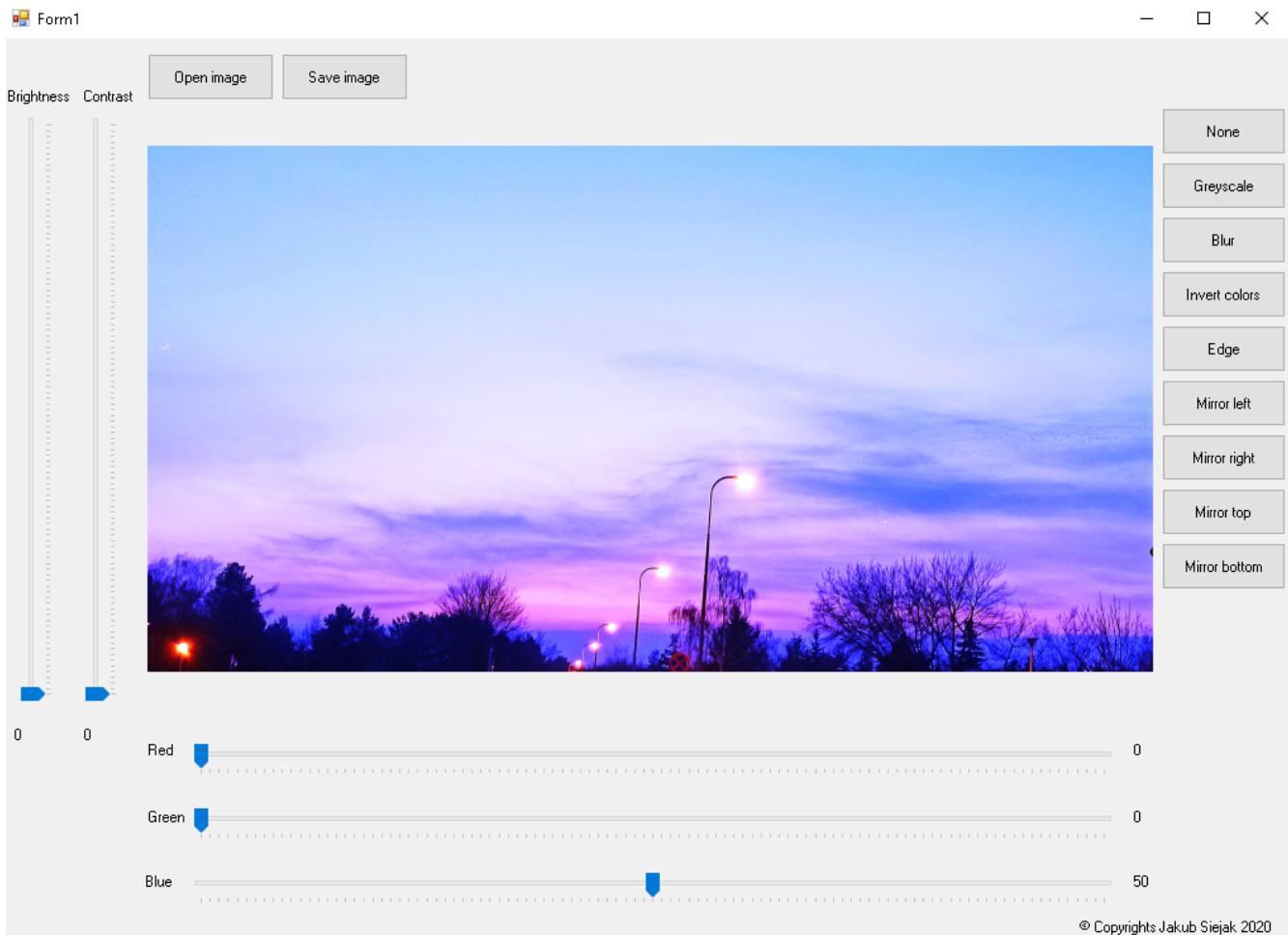
Ekran po wczytaniu gotowy do działania



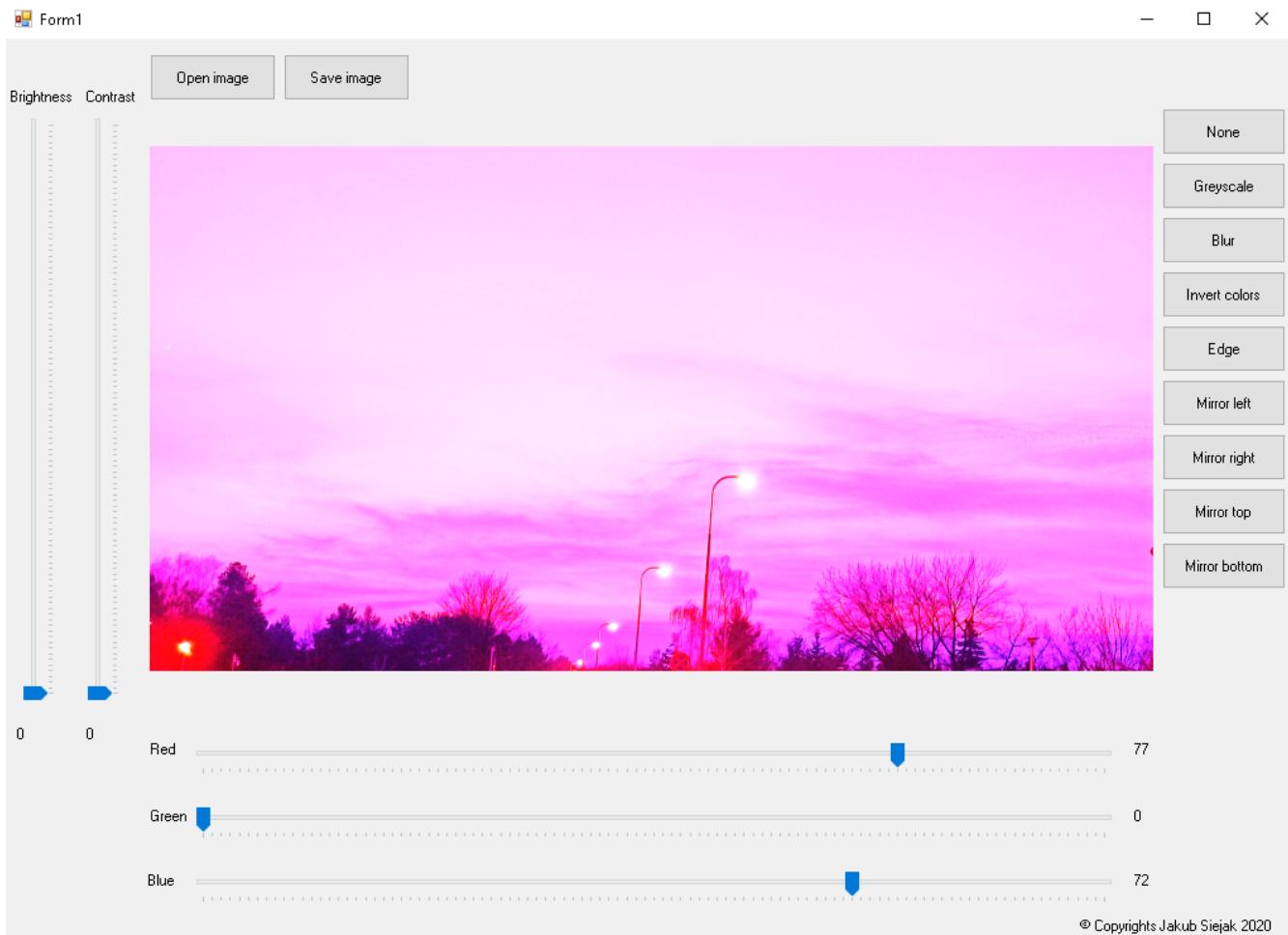
Zmiana wartości suwaka odpowiadającego za wartość czerwonej składowej RGB



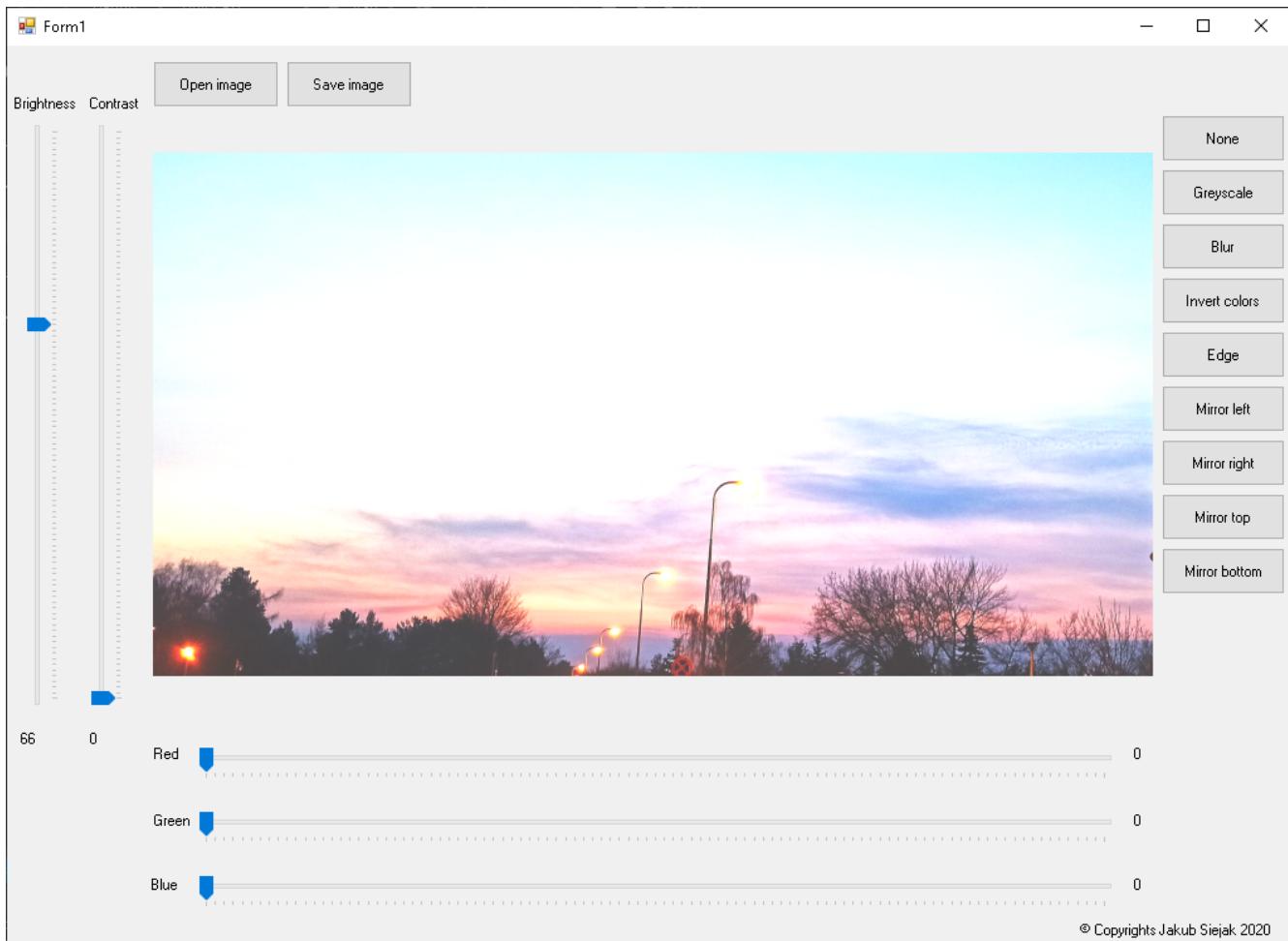
Zmiana wartości suwaka odpowiadającego za wartość zielonej składowej RGB



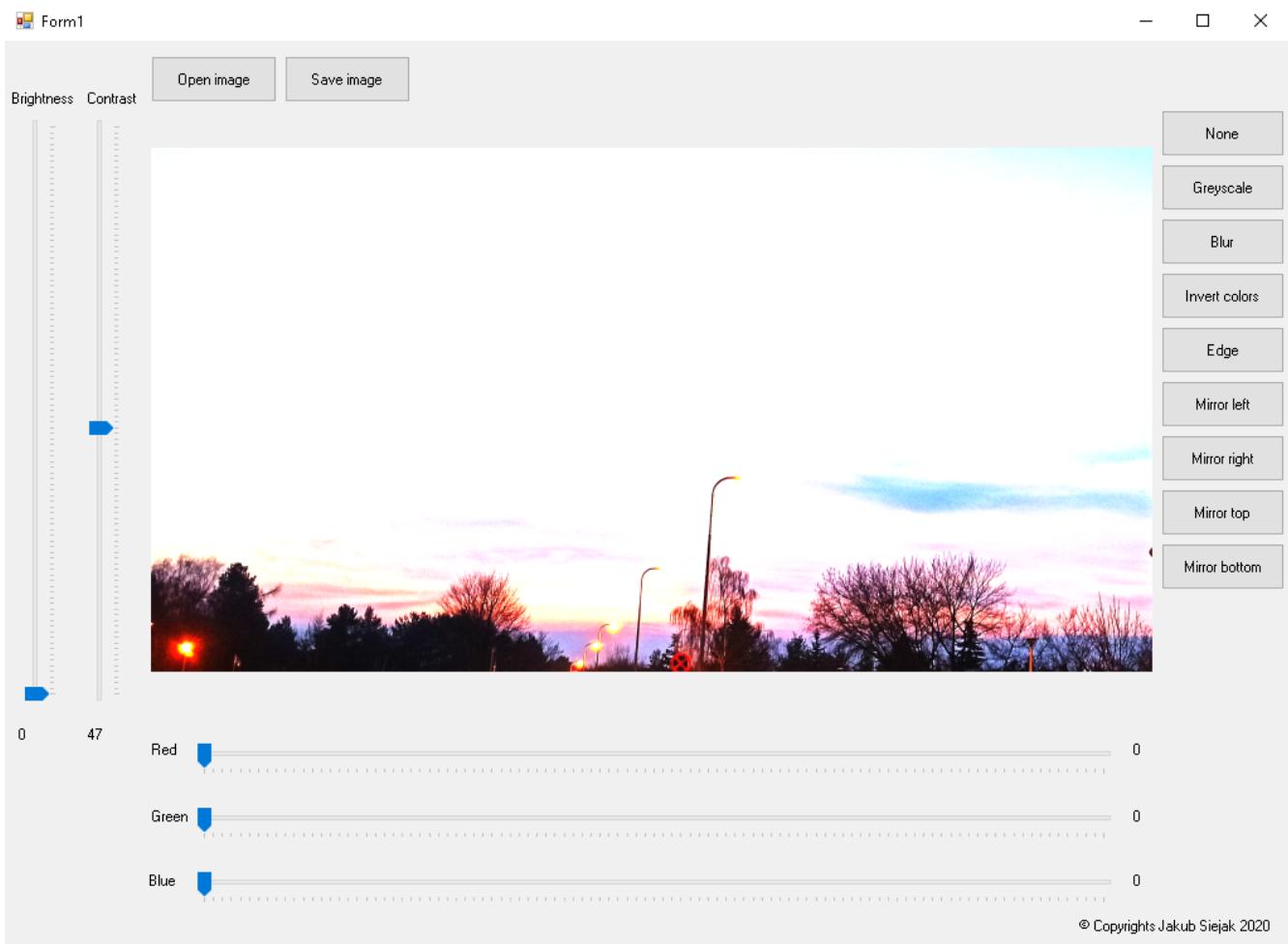
Zmiana wartości suwaka odpowiadającego za wartość niebieskiej składowej RGB



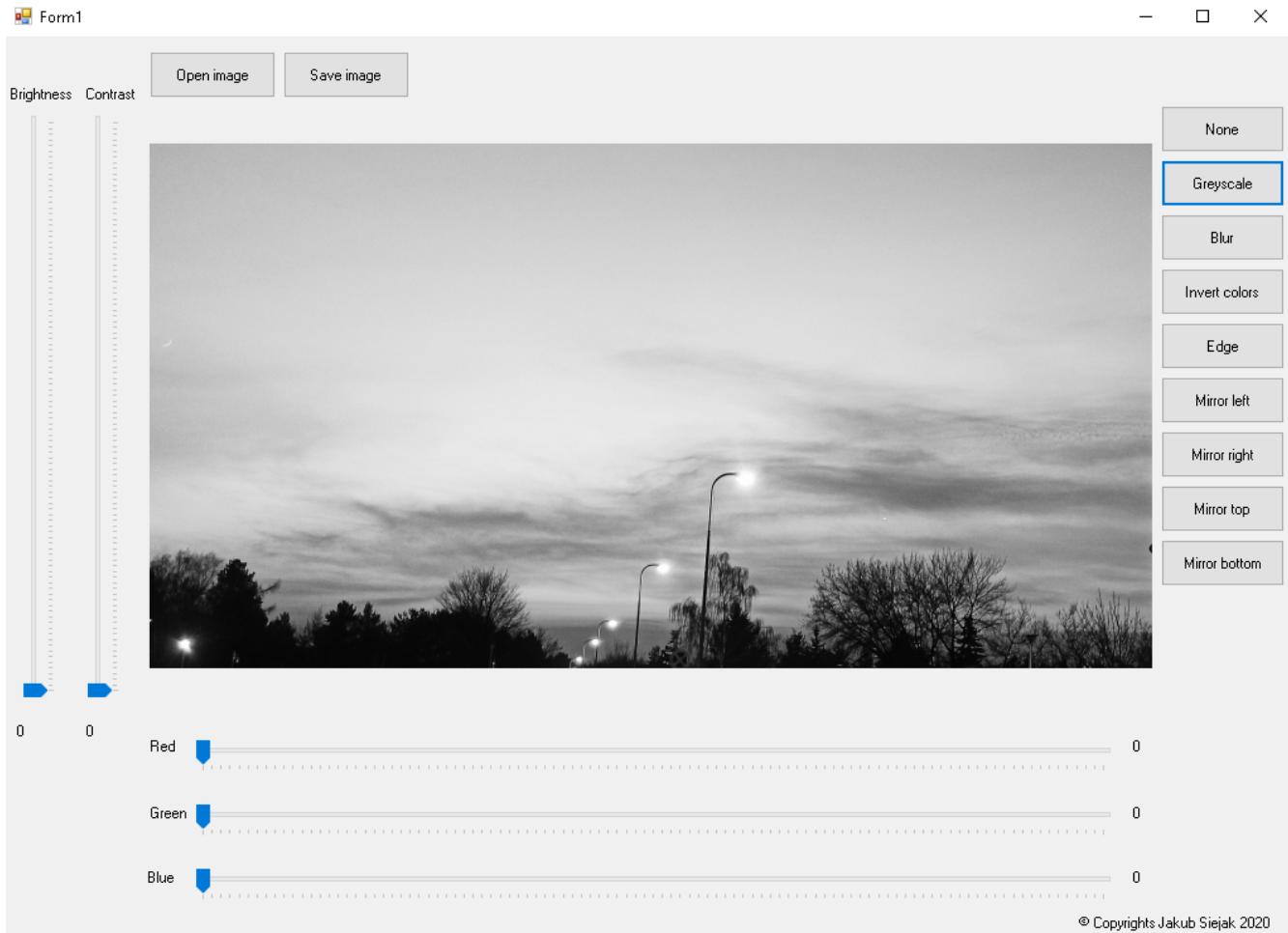
Zmiana wartości suwaków odpowiadających za wartości czerwonej i niebieskiej składowej RGB



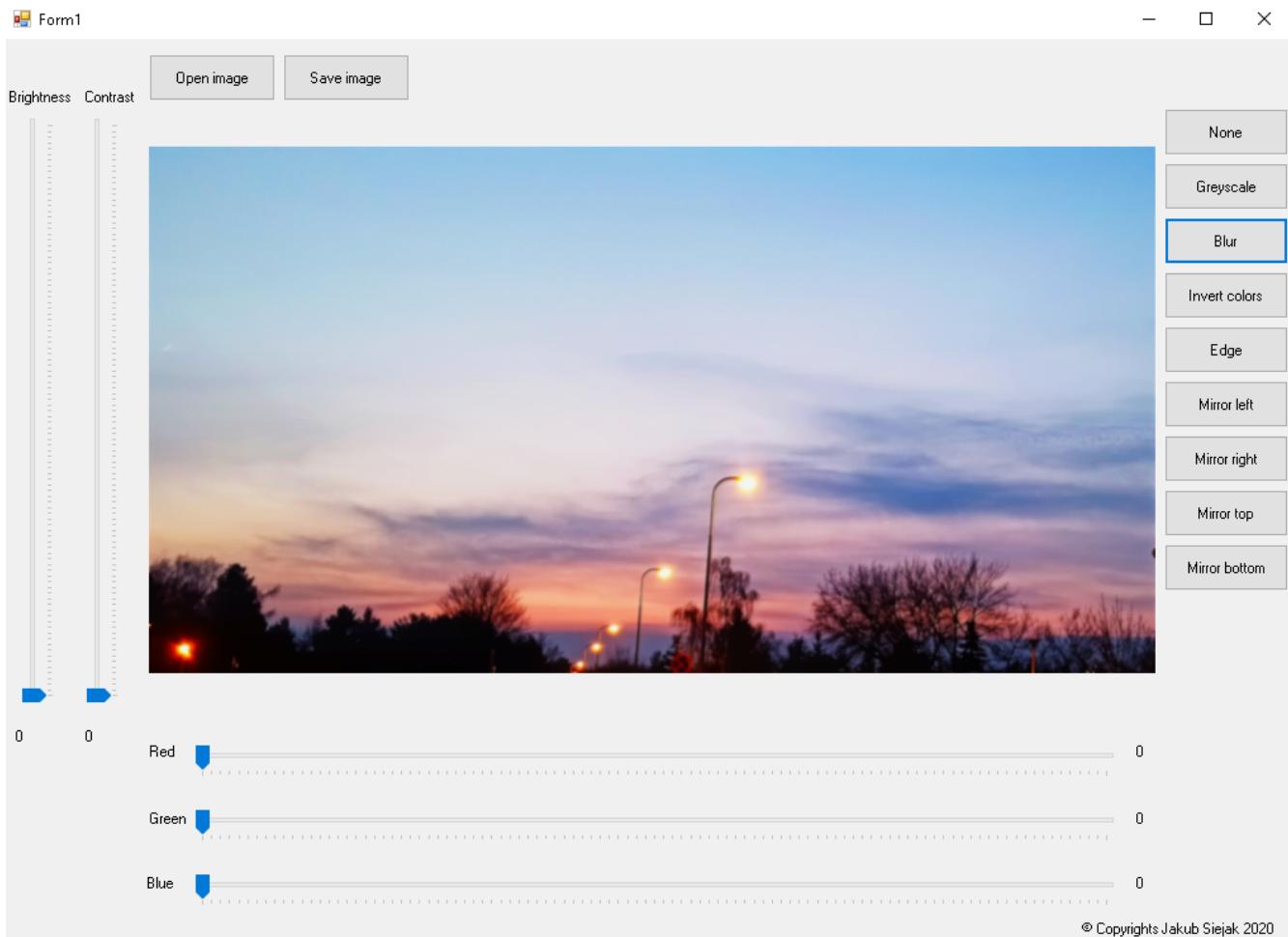
Zmiana wartości suwaka odpowiadającego za wartość jasności



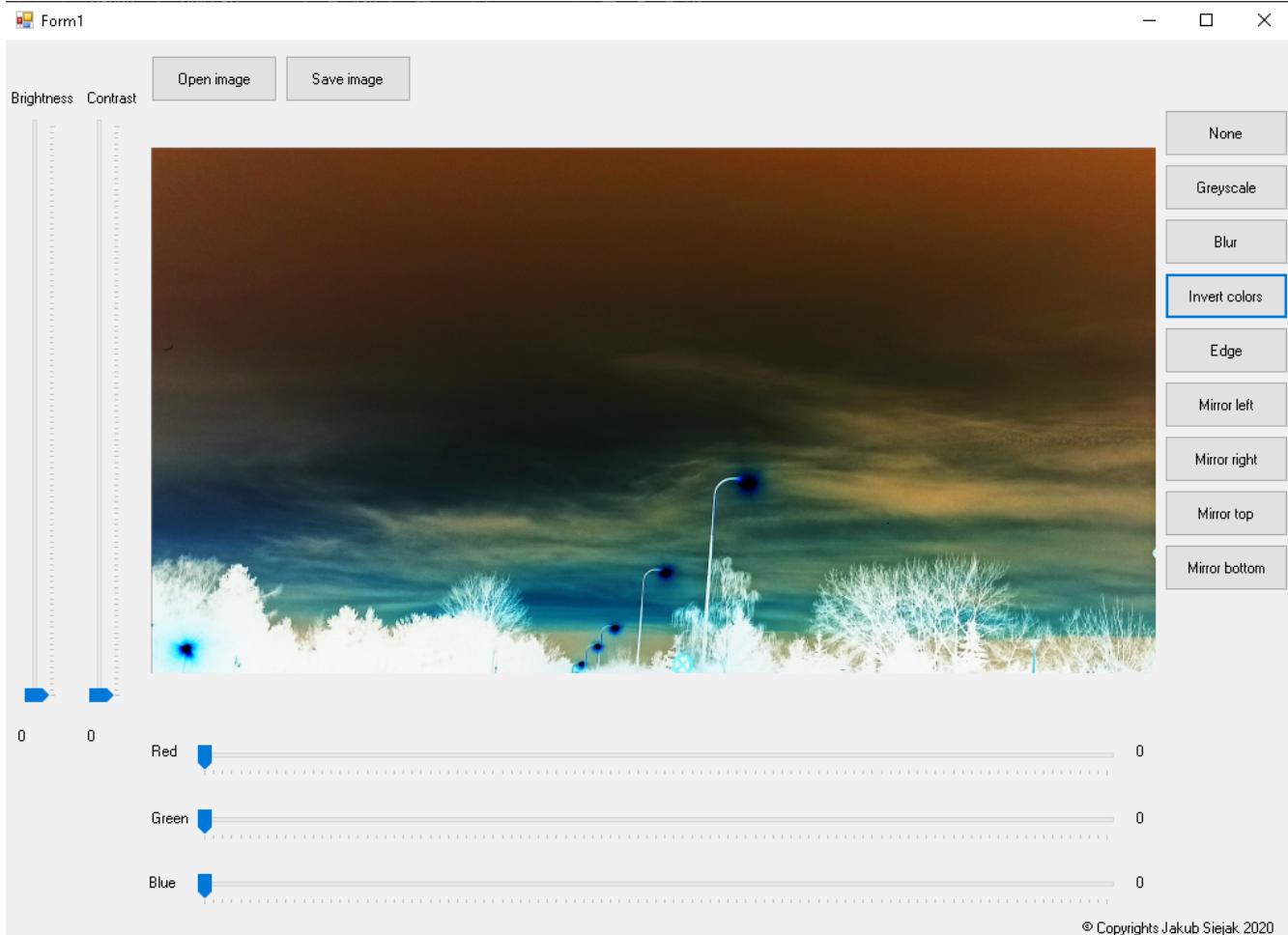
Zmiana wartości suwaka odpowiadającego za wartość kontrastu



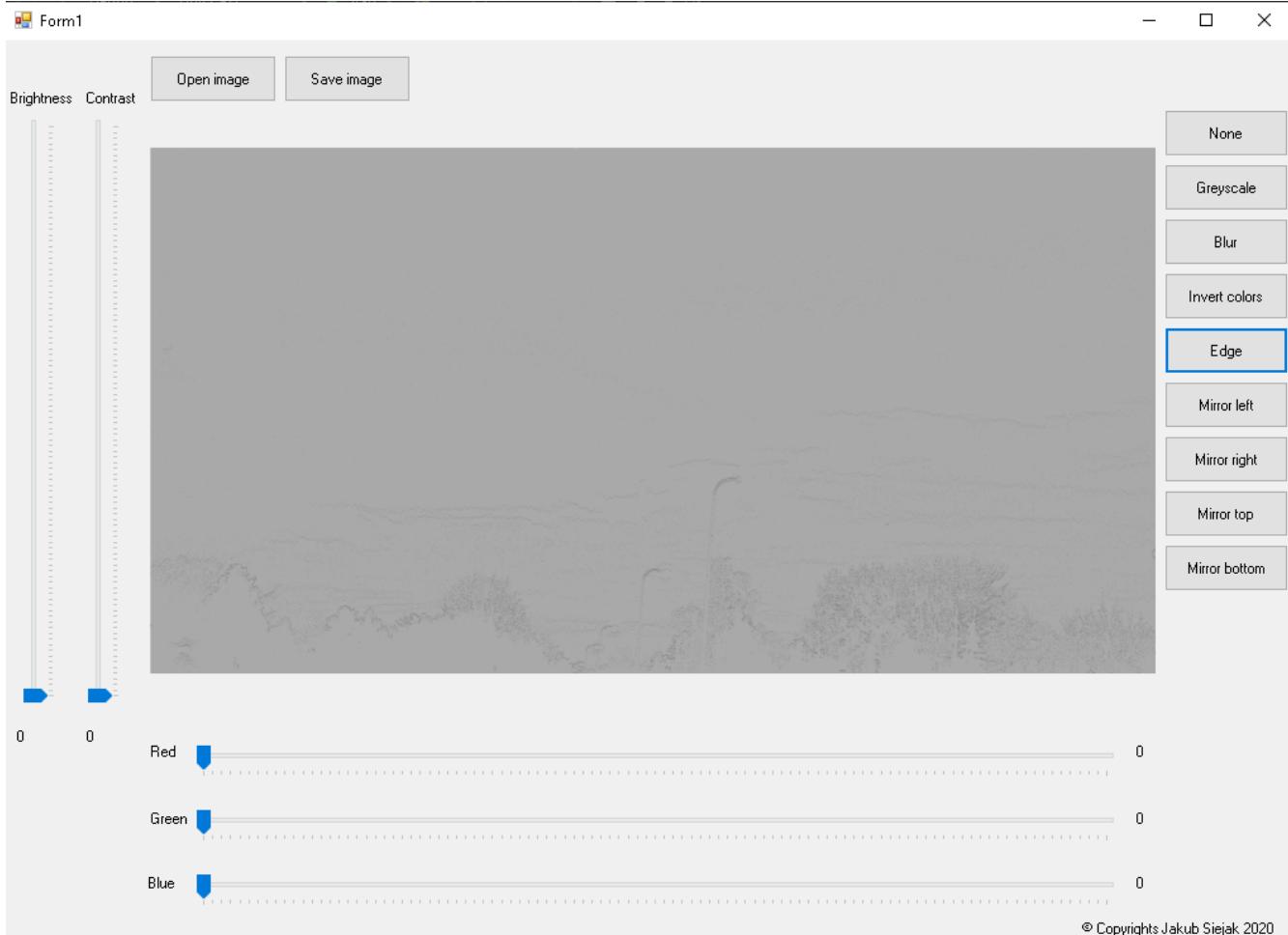
Efekt użycia filtru grayscale(na zrzutach widowcowych błąd językowy usunięty w kolejnej poprawce)



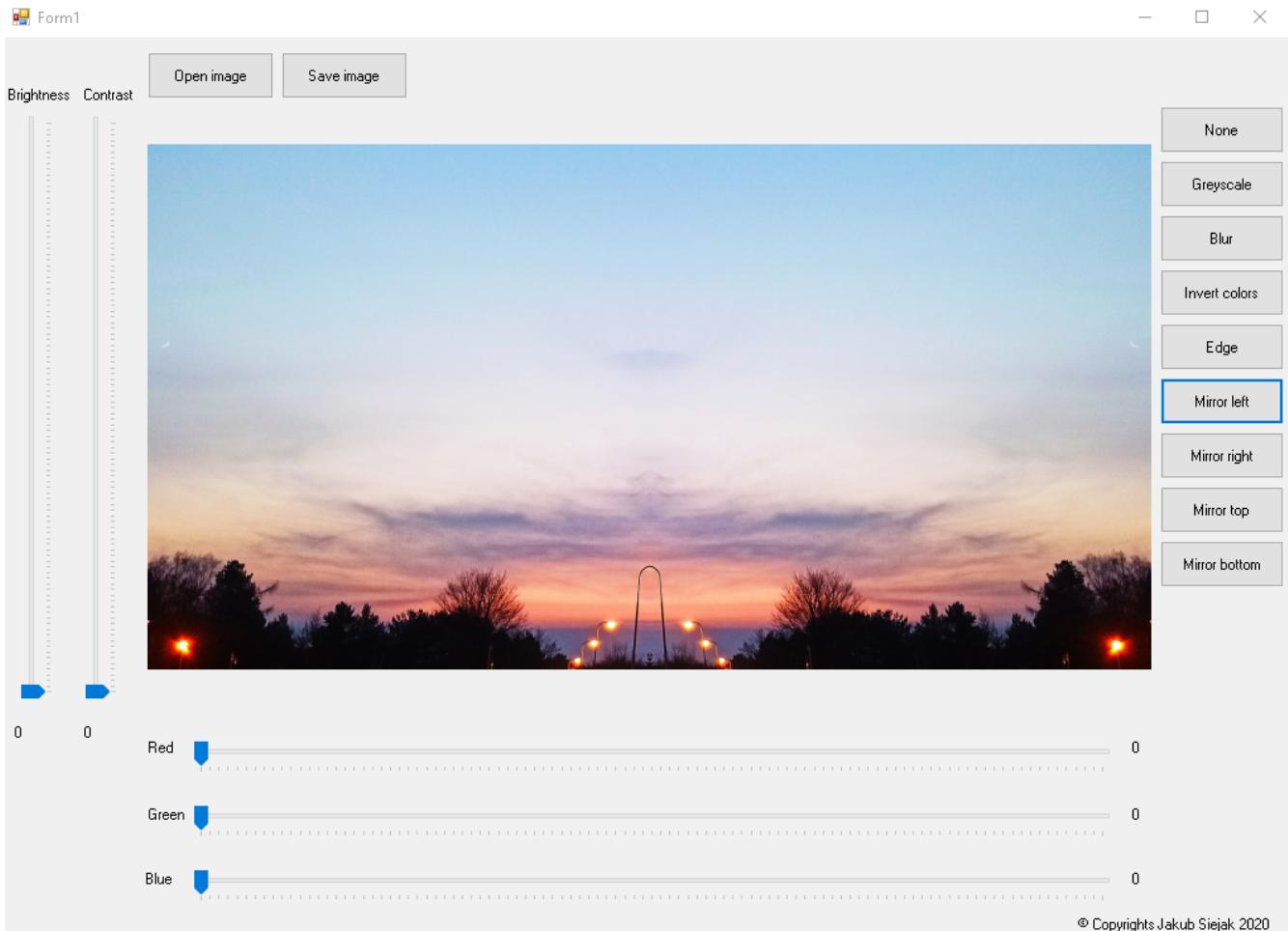
Efekt użycia filtru blur(na wysokiej jakości zdjęciach należy użyć filtru kilkukrotnie na powyższym przykładzie użyto dziesięciokrotnie efekt jest zauważalny jedynie na lini drzew) filtr ten mocno obciąża komputer I wymaga wielu obliczeń dlatego należy zaczekać na efekt(nie jest wtedy możliwa inna działalność w aplikacji)



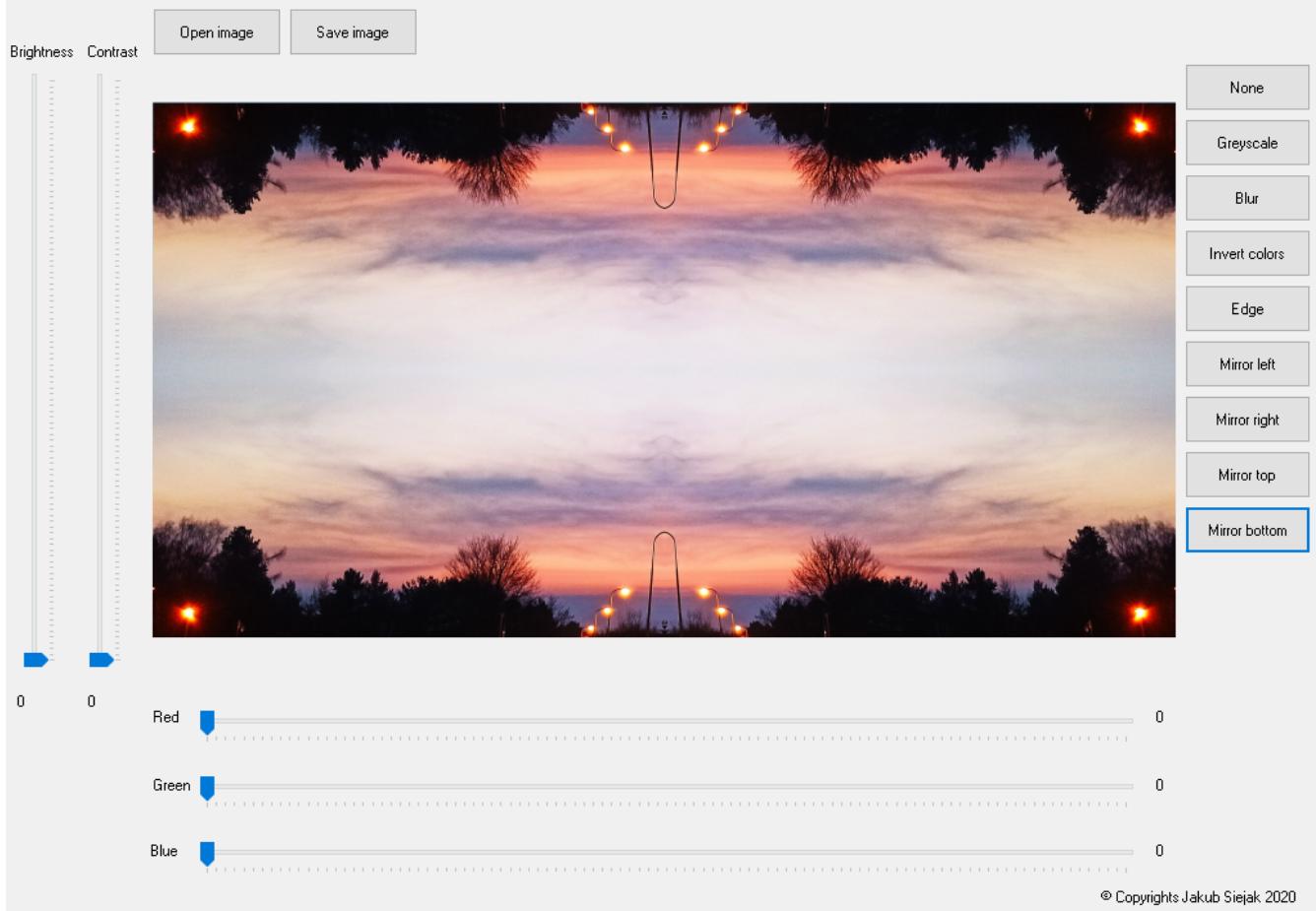
Efekt użycia filtra invert colors(negatyw)



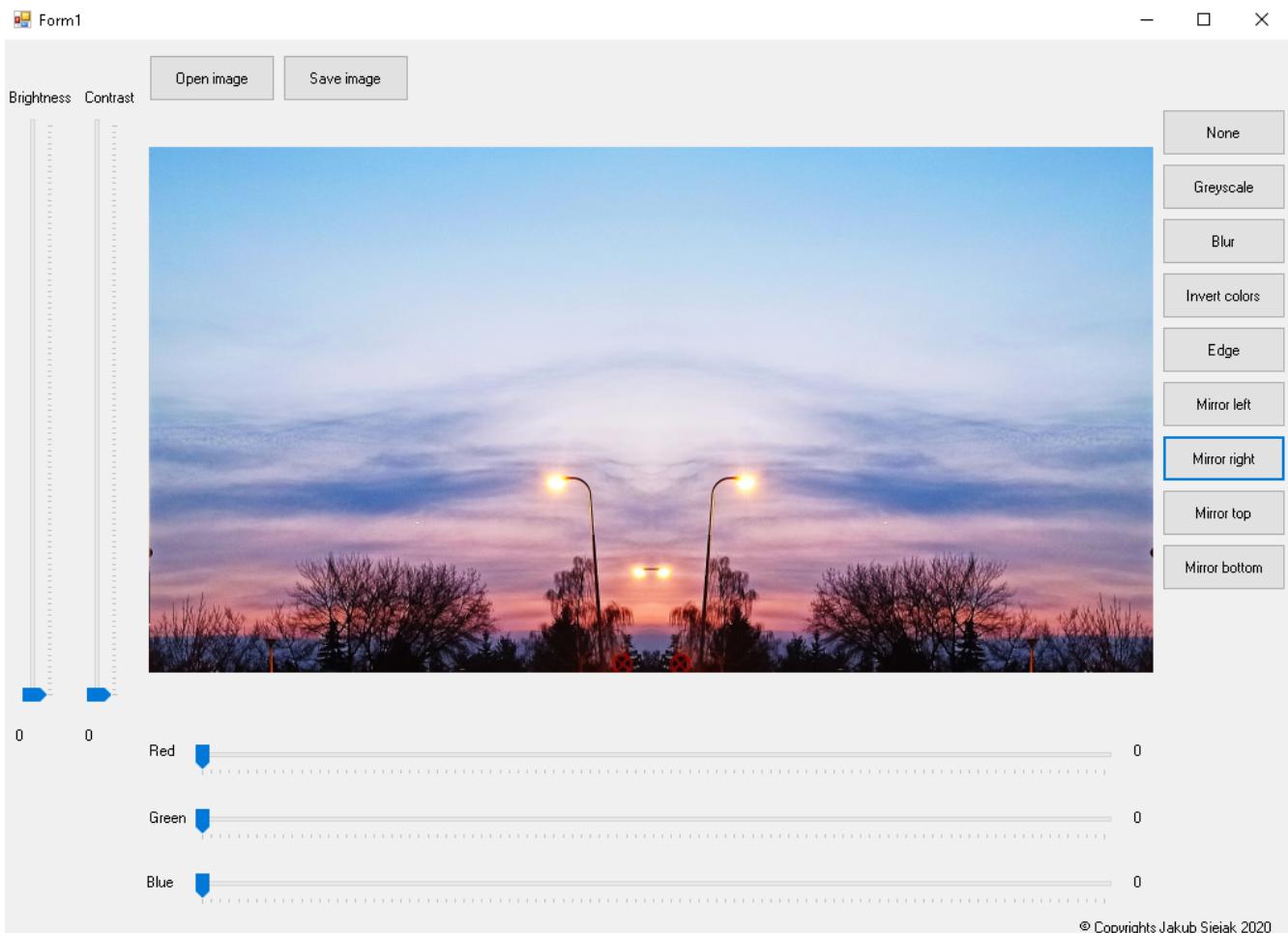
Efekt użycia filtru edge(filtr imitujący emboss)



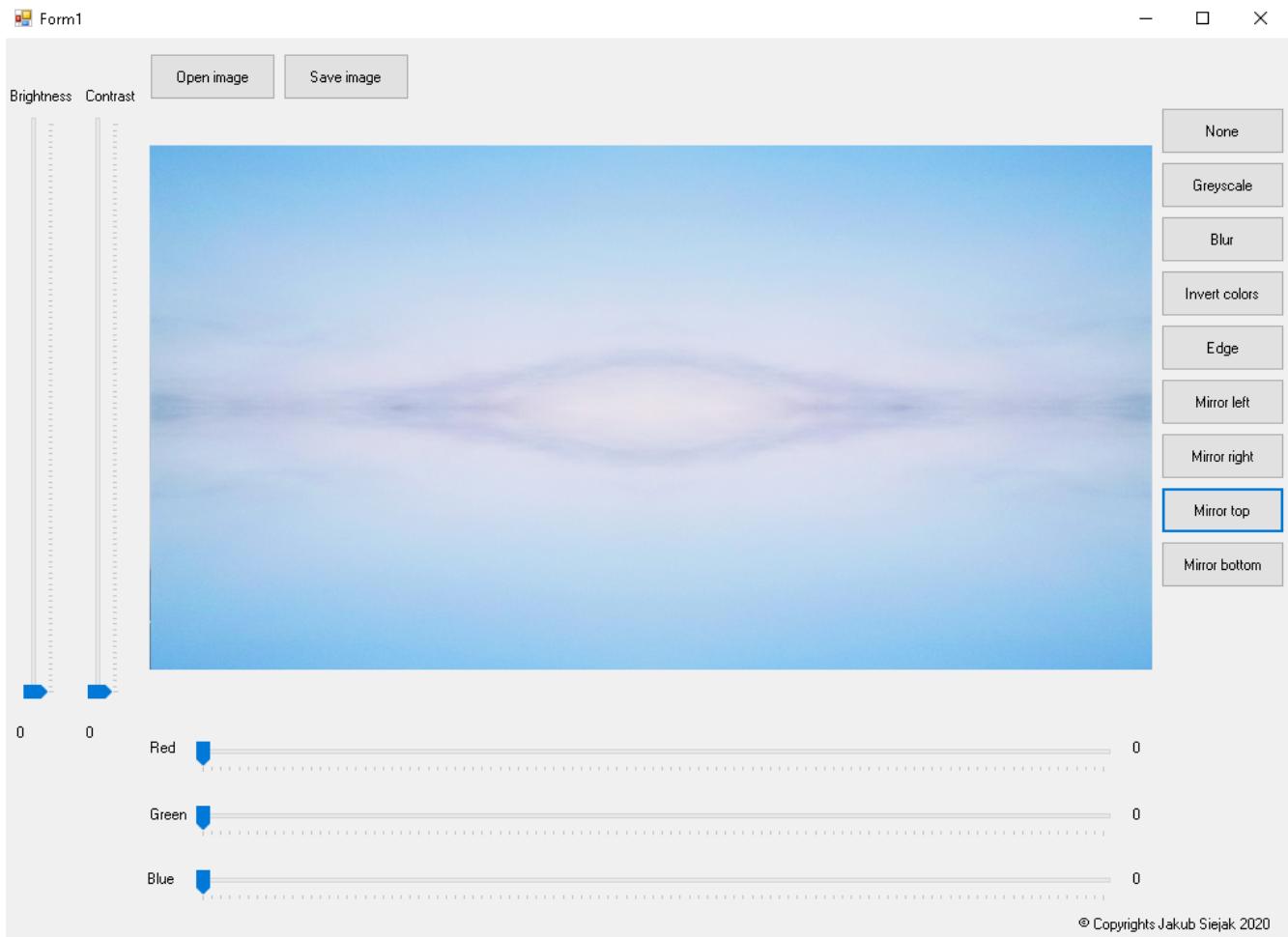
Efekt działania filtru mirror left



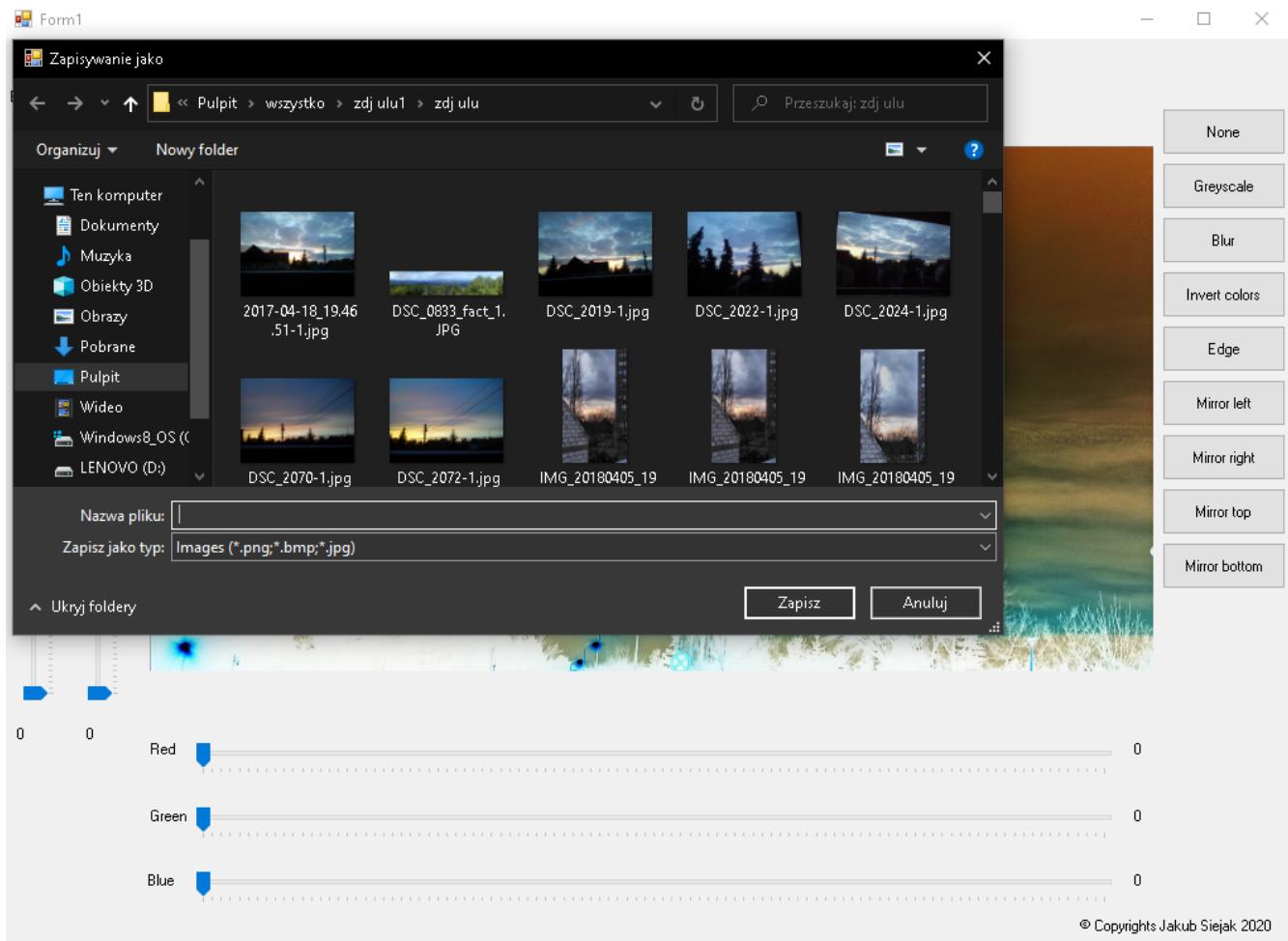
Nałożenie efektu mirror bottom na efekt mirror left



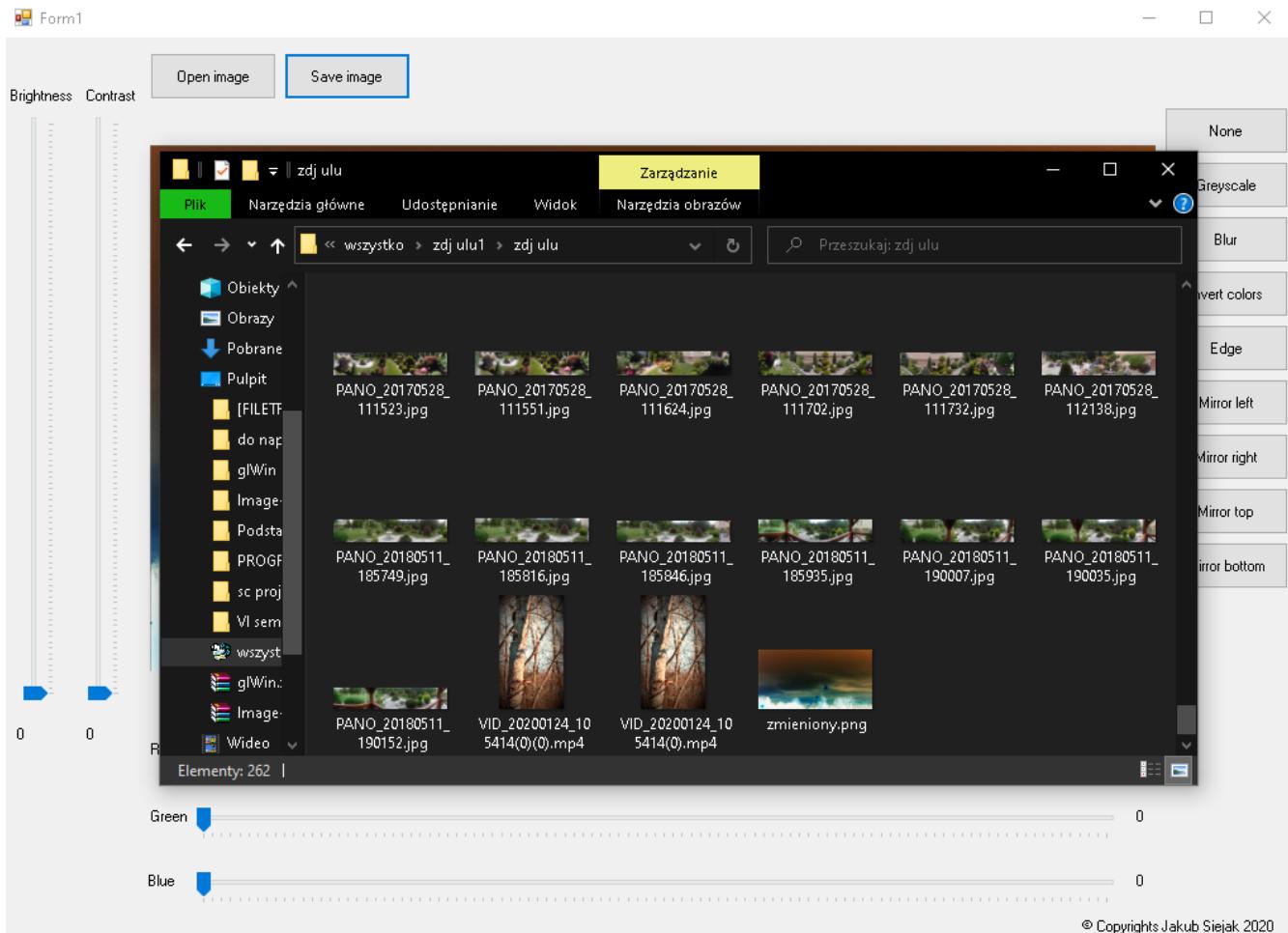
Efekt filtru mirror right



Nałożenie efektu mirror top na efekt mirror right

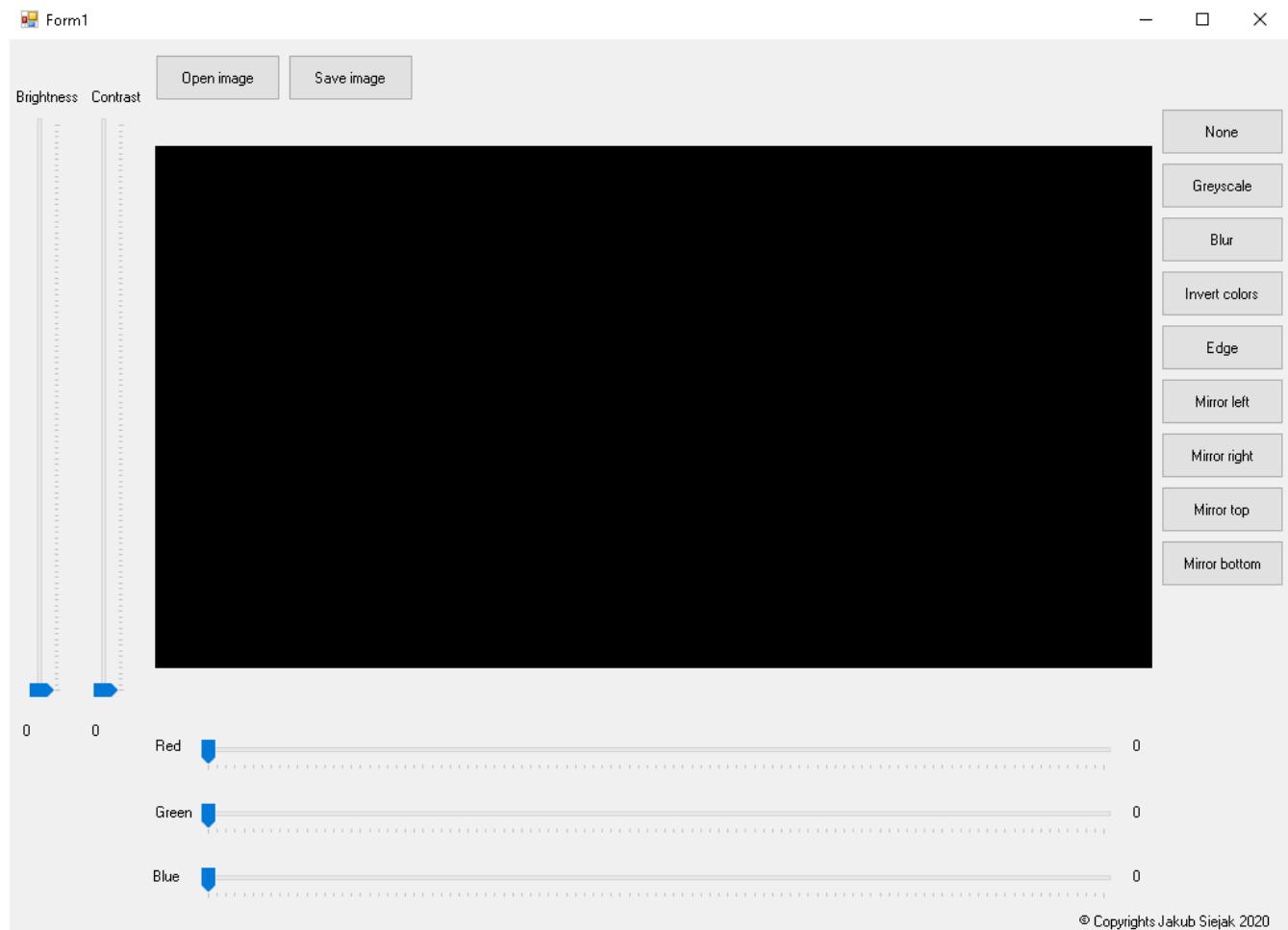


Ekran podczas zapisywania zmienionego obrazu



Widok zapisanego obrazu w formacie domyślnym

Błędy:



Efekt uzyskany po zmienieniu wartości suwaka kontrast na inną niż domyślną i powrót do wartości "0" jest to błąd wynikający ze złego zaimplementowania domyślnej wartości

Ze względu na brak możliwości poprawnego przedstawienia błędu opisany zostanie poniżej:

Kolejną dość dużą wadą jest nienakładanie się efektów zmiany suwaków wraz z gotowymi filtrami, i a także nienakładnie się efektu pracy filtrów grayscale, blur, invert, colors oraz edge między sobą oraz z efektami mirror.