Grafika komputerowa – sprawozdanie L1-2

Dominik Matusiak WCY22IJ3S1

LAB 1

Treść zadania:

Zadanie laboratoryjne – ćwicz.1

1. Używając narzędzia ToolBox/Przybornik zmodyfikować graficzne okno aplikacji wprowadzając:
   a) Modyfikację opisów funkcji programu zgodnie z zadaniem indywidualnym
   b) Informację o wykonawcy (imię i nazwisko, grupa, data wykonania)
   0.5 pkt.
2. Przygotować nowy obraz (kolorowy)o parametrach:
   K=L ≠ 256, plik BMP 24 bitowy. Wprowadzić nowy obraz do okna aplikacji.
   0.5 pkt.
3. Zmodyfikować algorytmy (3) przekształceń obrazu zgodnie z zadaniem indywidualnym
   6 pkt. (2 pkt./efekt)
4. Używając narzędzia ToolBox wprowadzić pasek ProgressBar i powiązać z kolejną klatką wykonywanego przekształcenia obrazu
   1 pkt.

Zestaw 11.

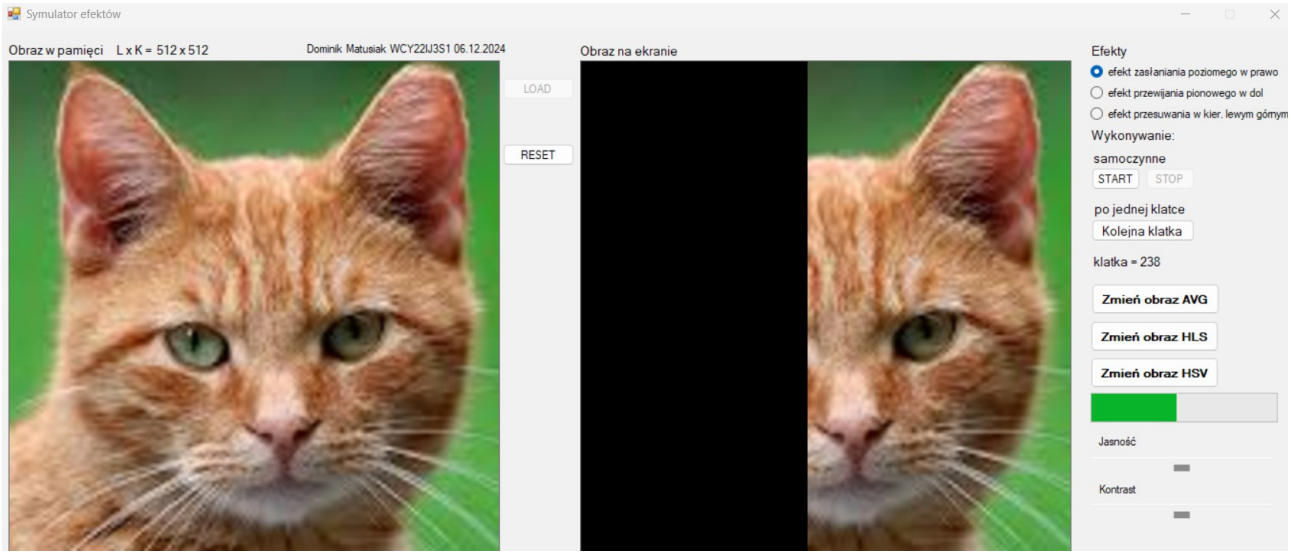
1. Napisać algorytm sterujący generatorem adresu odczytu w celu uzyskania efektu zasłaniania poziomego obrazu w kierunku prawej strony ekranu
2. Napisać algorytm sterujący generatorem adresu odczytu w celu uzyskania efektu przewijania pionowego obrazu w kierunku dolnej krawędzi ekranu.
3. Napisać algorytm sterujący generatorem adresu odczytu w celu uzyskania efektu przesuwania obrazu wzdłuż przekątnej ekranu w kierunku górnego lewego wierzchołka

efekt 1 - Rozwiązanie:

```
public void Efekt1()
{
    ////efekt: zaslanianie w prawo
    if (p >= L) p = 0;

    for (int j = 1; j <= L; j++)
    {
        for (int i = 1; i <= p; i++)
            ReadTlo(N);

        for (int i = p + 1; i <= K; i++)
            ReadPixel(i, j);
    }
}
```

Efekt zasłaniania w prawo osiągnięto przez wczytywanie do każdej linii p pixeli tła a następnie (K-p-1) pierwszych pikseli obrazu odpowiadających danej linii.

Wynik:

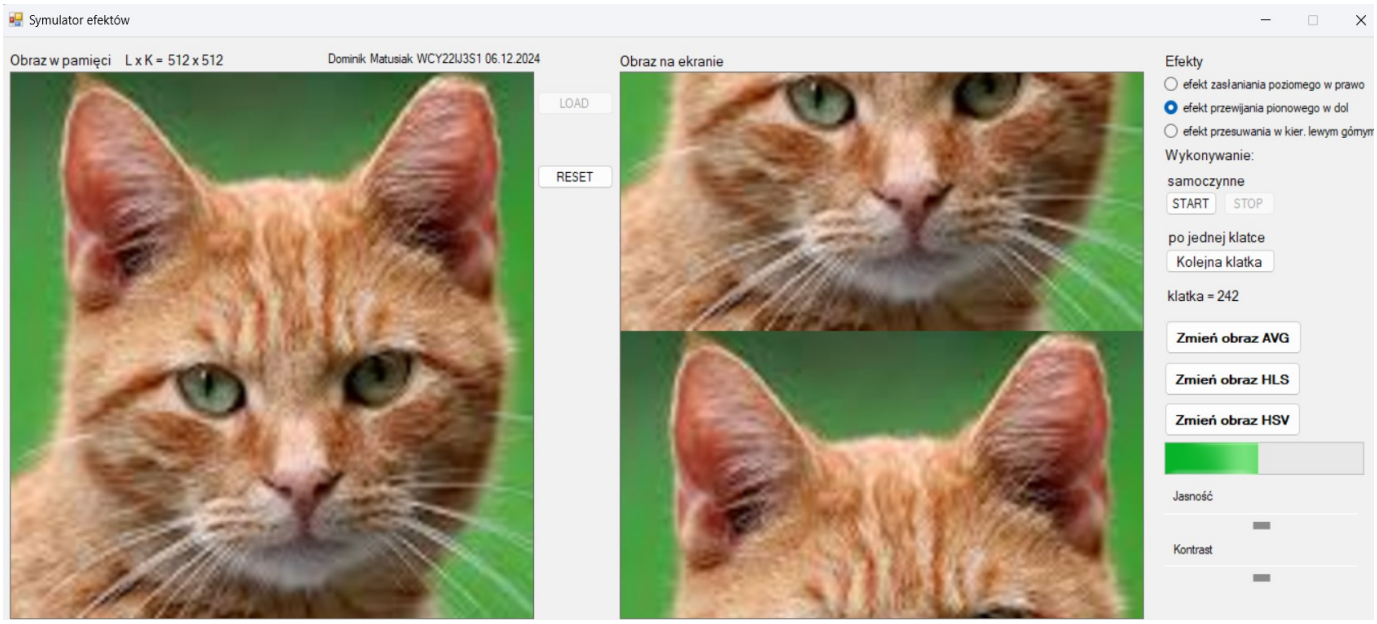


efekt 2 – Rozwiązanie

```
public void Efekt2()
{
    //efekt: przewijanie się obrazu w dol

    if (p >= L) p = 0;


    for (int j = L - (p); j <= L; j++)
    {

        for (int i = 1; i <= K; i++)
            ReadPixel(i, j);
    }
    for (int j = 1; j <= L - (p + 1); j++)
    {
        for (int i = 1; i <= K; i++)
            ReadPixel(i, j);
    }
}
```

efekt przewijania obrazu w dół osiągnięto przez wczytywanie najpierw p ostatnich linii obrazu, a następnie L-(p+1) linii od początku (od 1 linii oryginalnego obrazu)

Wynik:

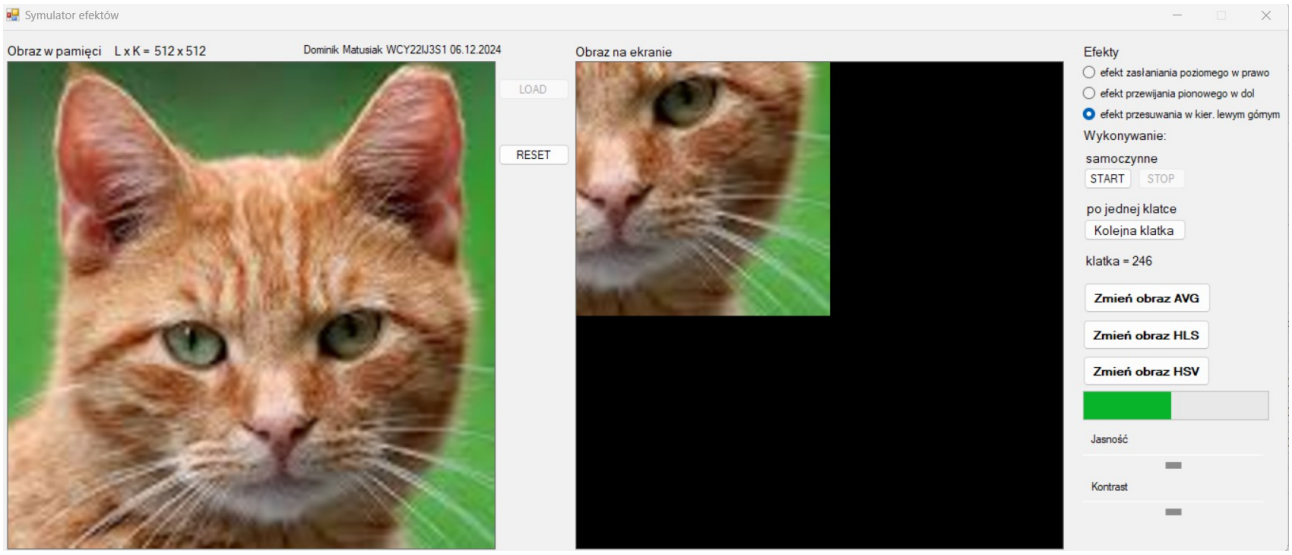


efekt 3 – rozwiązanie

```
public void Efekt3()
{
    //efekt: przesuwanie do lewego gornego rogu

    if (p >= L) p = 0;

    for (int j = p + 1; j <= L; j++)
    {
        for (int i = p + 1; i <= K; i++)
        {
            ReadPixel(i, j);
        }
        for (int i = 1; i <= p; i++)
        {
            ReadTlo(N);
        }

    }
    for (int j = 1; j <= p; j++)
    {
        for (int i = 1; i <= K; i++)
        {
            ReadTlo(N);
        }
    }

}
```

efekt przesuwania do lewego górnego rogu uzyskano przez wczytywanie obrazu od p+1 linii, i dla każdej linii obraz jest wczytywany od p+1 kolumny, a reszta linii (p kolumn) jest zapełniane tłem. Po wczytaniu L-(p+1) linii reszta jest zapełniana tłem w oddzielnej pętli.

Wynik:



Wnioski z lab 1 :

Widać, że wczytywanie kolumn i linii obrazu w odpowiedniej kolejności pozwala uzyskać różne efekty. Potrzebne do tego są znajomość wymiarów obrazu, oraz wczytywanie go pixel po pixelu z wykorzystaniem odpowiedniego algorytmu.

Lab 2:

Treść zadania:

Zadanie laboratoryjne –zestaw 3

1. Zamienić obraz kolorowy (RGB) o 24-bitowej strukturze piksela (z ćwicz.1) na obraz monochromatyczny reprezentowany przez skalę odcieni szarości wykorzystując:
   a) równanie przejścia z modelu RGB na HLS; 1pkt.
   b) równanie przejścia z modelu RGB na HSV(B); 1pkt.
   c) średnią arytmetyczną składowych R, G, B. 0.5 pkt.
   Wyznaczyć jasność (J) i kontrast (K) uzyskanych obrazów. 1.5 pkt. (0.5pkt i 1 pkt.)
   Razem: 4 pkt.
2. Wykorzystując liniową korekcję tonalną napisać program umożliwiający zmianę jasności (1 pkt.) i kontrastu (2 pkt.) obrazu kolorowego (z ćwicz. 1) w pełnym zakresie zmienności. Do zmiany jasności/kontrastu wykorzystać suwaki ScrollBar (H/V)
   Wyznaczyć jasność (J) i kontrast (K) uzyskiwanych obrazów. 0.5 pkt. i 0.5 pkt.

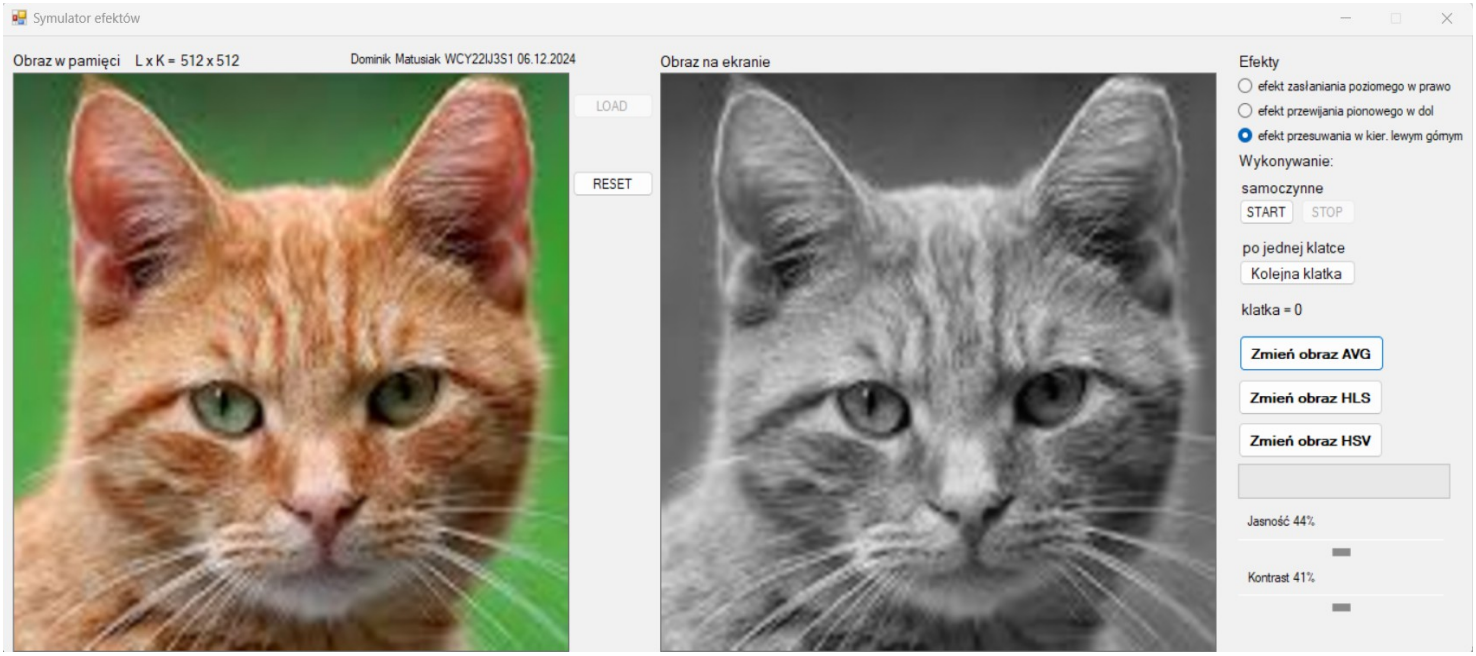
Zadanie 1:

AVG:

```
public void Zmien_obraz()
{
    System.Drawing.Color pixel;
    int J = 0;
    for (int j = 1; j <= L; j++)
        for (int i = 1; i <= K; i++)
        {
            pixel = m_obraz_w_pamieci.GetPixel(i - 1, j - 1);

            //--------------------------------------//
            int jas = (int)(pixel.R + pixel.G + pixel.B) / 3;
            pixel = System.Drawing.Color.FromArgb(jas, jas, jas);
            //--------------------------------------//
            m_ekran.SetPixel(i - 1, j - 1, pixel);
        }

    //labelSc1.Text = ("Jasność " + Math.Round((double)J * 100.0 / 255.0) + "%");
    SetBitMap(ref m_ekran);
    LiczJC(m_ekran);
}
```

dla każdego pixela liczymy jasność jako średnią intensywności każdego jego koloru składowego, a następnie zapisujemy intensywność każdego koloru jako obliczoną jasność.

Wynik:

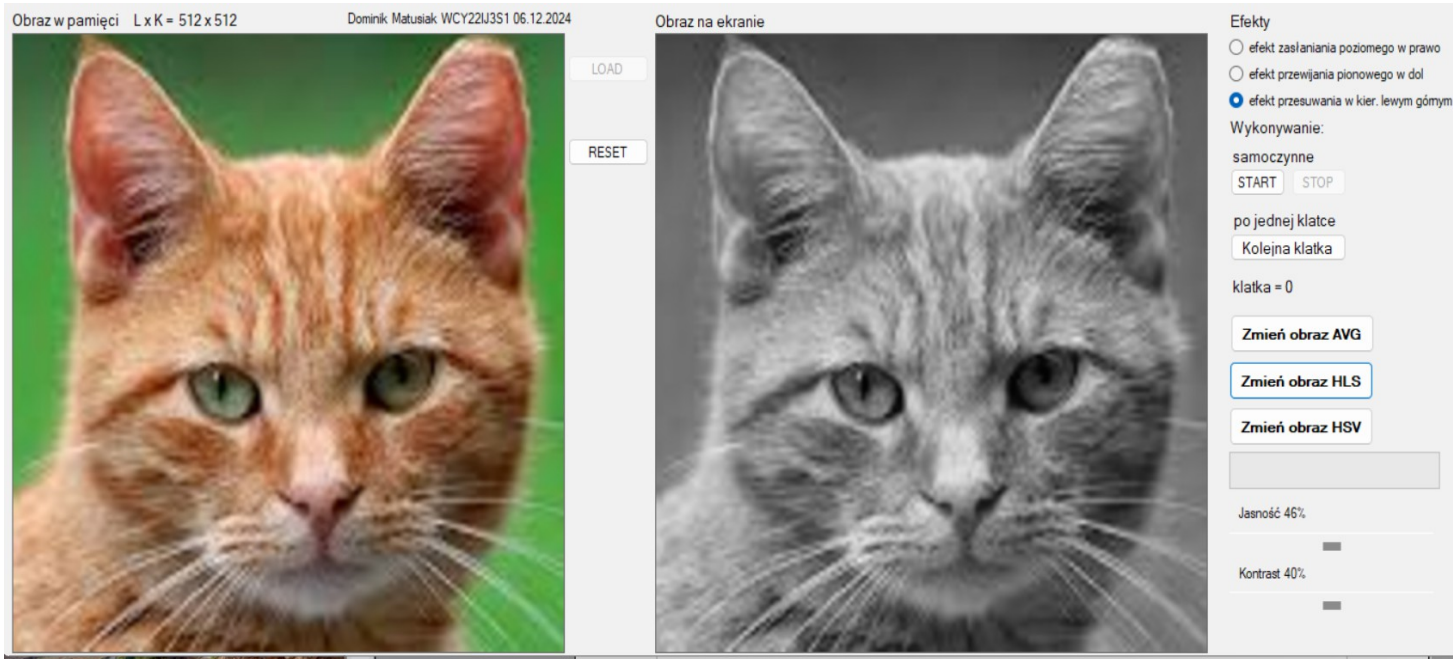


HLS:

```
public void Zmien_obrazHLS()
{
    System.Drawing.Color pixel;
    int J = 0;
    for (int j = 1; j <= L; j++)
        for (int i = 1; i <= K; i++)
        {
            pixel = m_obraz_w_pamieci.GetPixel(i - 1, j - 1);

            double r = pixel.R;
            double g = pixel.G;
            double b = pixel.B;
            double Cmax = Math.Max(Math.Max(r, g), Math.Max(g, b));
            double Cmin = Math.Min(Math.Min(r, g), Math.Min(g, b));
            double l = (Cmax + Cmin) / 2;
            int l2 = (int)(l);

            int jas = (int)(pixel.R + pixel.G + pixel.B) / 3;
            pixel = System.Drawing.Color.FromArgb(l2, l2, l2);
            //--------------------------------------//

            m_ekran.SetPixel(i - 1, j - 1, pixel);
        }
    SetBitMap(ref m_ekran);
    LiczJC(m_ekran);
}
```

dla każdego piksela znajdujemy jego jasność jako średnią intensywności najmniej i najbardziej intesywnego koloru. Następnie każdy kolor generowanego piksela jest ustawiany na tę jasność.

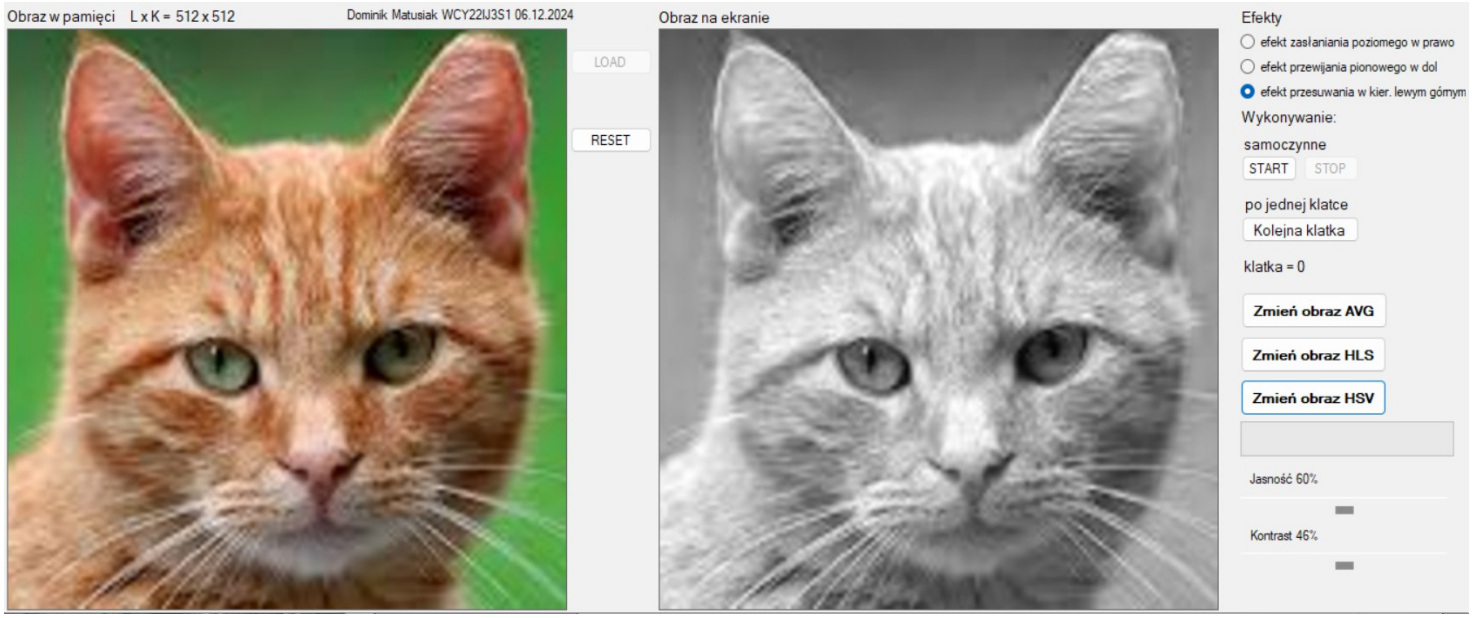
Wynik:



HSV:

```
public void Zmien_obrazHSV()
{
    System.Drawing.Color pixel;
    int J = 0;
    for (int j = 1; j <= L; j++)
        for (int i = 1; i <= K; i++)
        {
            pixel = m_obraz_w_pamieci.GetPixel(i - 1, j - 1);

            double r = pixel.R;
            double g = pixel.G;
            double b = pixel.B;
            double Cmax = Math.Max(Math.Max(r, g), Math.Max(g, b));
            int V = (int)Cmax;
            pixel = System.Drawing.Color.FromArgb(V, V, V);
            //------------------------------------//
            m_ekran.SetPixel(i - 1, j - 1, pixel);
        }
    SetBitMap(ref m_ekran);
    LiczJC(m_ekran);
}
```

dla każdego piksela jest wyszukiwana wartość koloru o największej intensywności, a następnie kolory generowanego piksela są ustawiane na tę wartość.



Wyznaczanie jasności i kontrastu:

```
public void LiczJC(Bitmap map)
{
    int J = 0;
    System.Drawing.Color pixel;
    for (int j = 1; j <= L; j++)
        for (int i = 1; i <= K; i++)
        {
            pixel = map.GetPixel(i - 1, j - 1);
            //J += Licz_jasn(pixel);
            J += Licz_jasn(pixel);
        }

    J /= (L * K);
    labelSc1.Text = ("Jasność " + Math.Round((double)J * 100.0 / 255.0) + "%");

    int liczba = 0;
    int roznica = 0;

    for (int j = 1; j <= L; j++)
        for (int i = 1; i <= K; i++)
        {
            pixel = map.GetPixel(i - 1, j - 1);
            //J += Licz_jasn(pixel);
            roznica += (int)Math.Pow((double)(Licz_jasn(pixel) - J), 2.0);
        }
    roznica /= (L * K);
    roznica = (int)Math.Sqrt(roznica);
    labelSc2.Text = ("Kontrast " + Math.Round((double)roznica ) + "%");

}
```

aby obliczyć jasność obrazu liczona jest średnia jasność każdego piksela, gdzie jasność 1 piksela jest wyznaczana jako średnia intensywności jego najmniej i najbardziej intensywnego koloru.

Oby obliczyć kontrast obliczamy sumę kwadratów różnic jasności piksela i jasności obrazu, a następnie dzielimy przez liczbę pikseli i pierwiastkujemy.

Wyniki widać na wynikach poprzednich zadań (wartości procentowe przy scrollbarach)

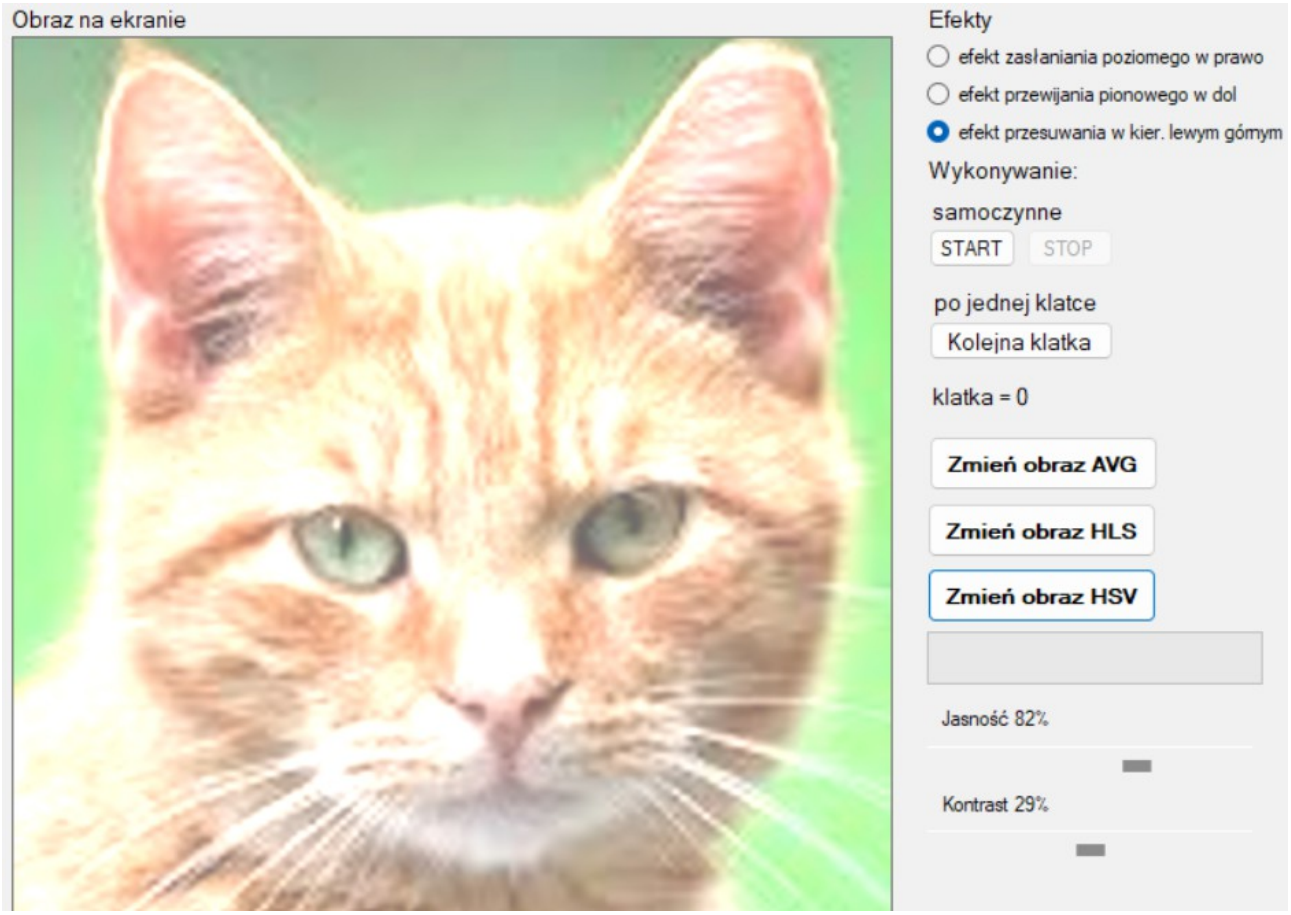
Zadanie 2:

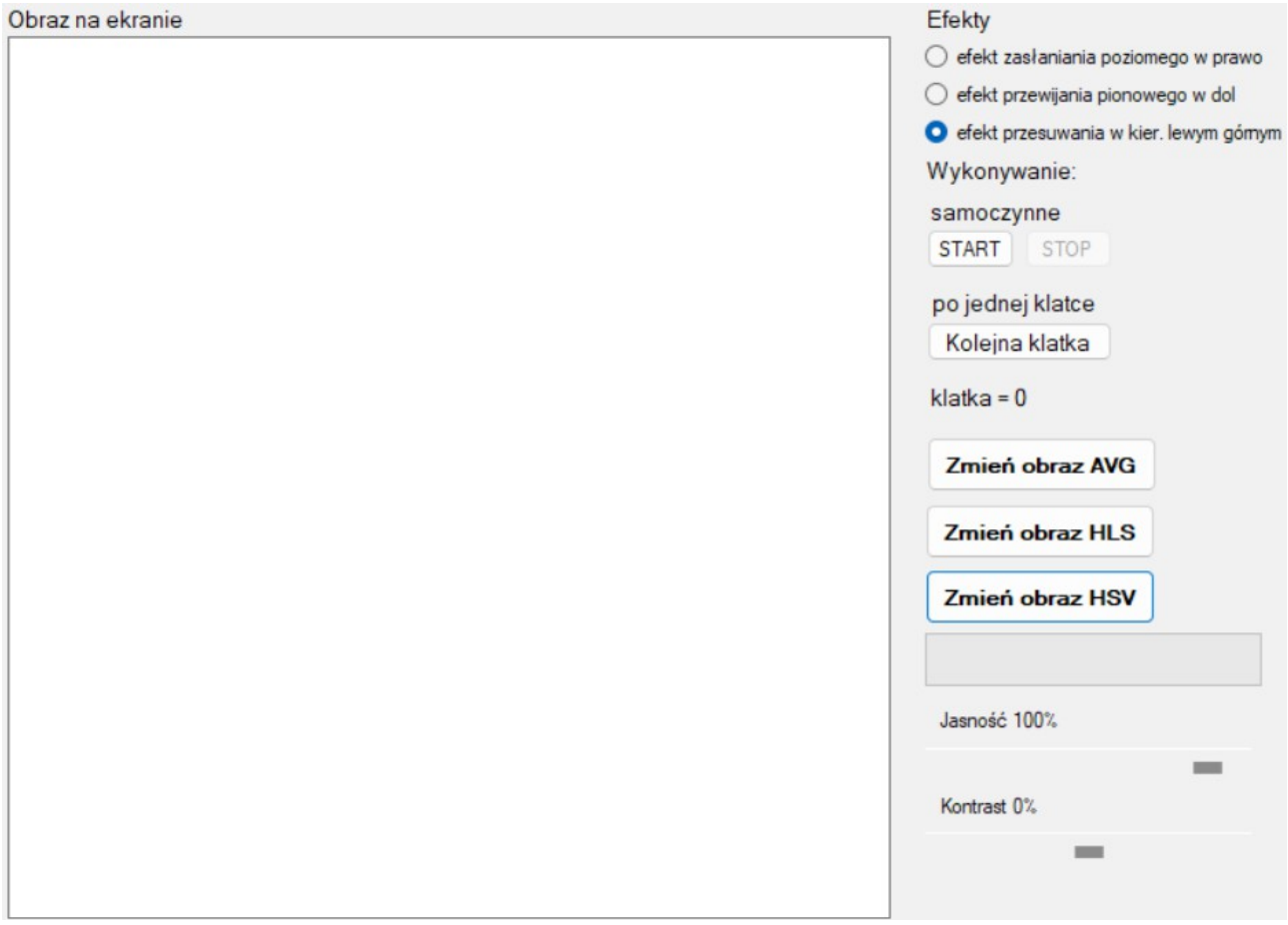
Zmiana jasności:

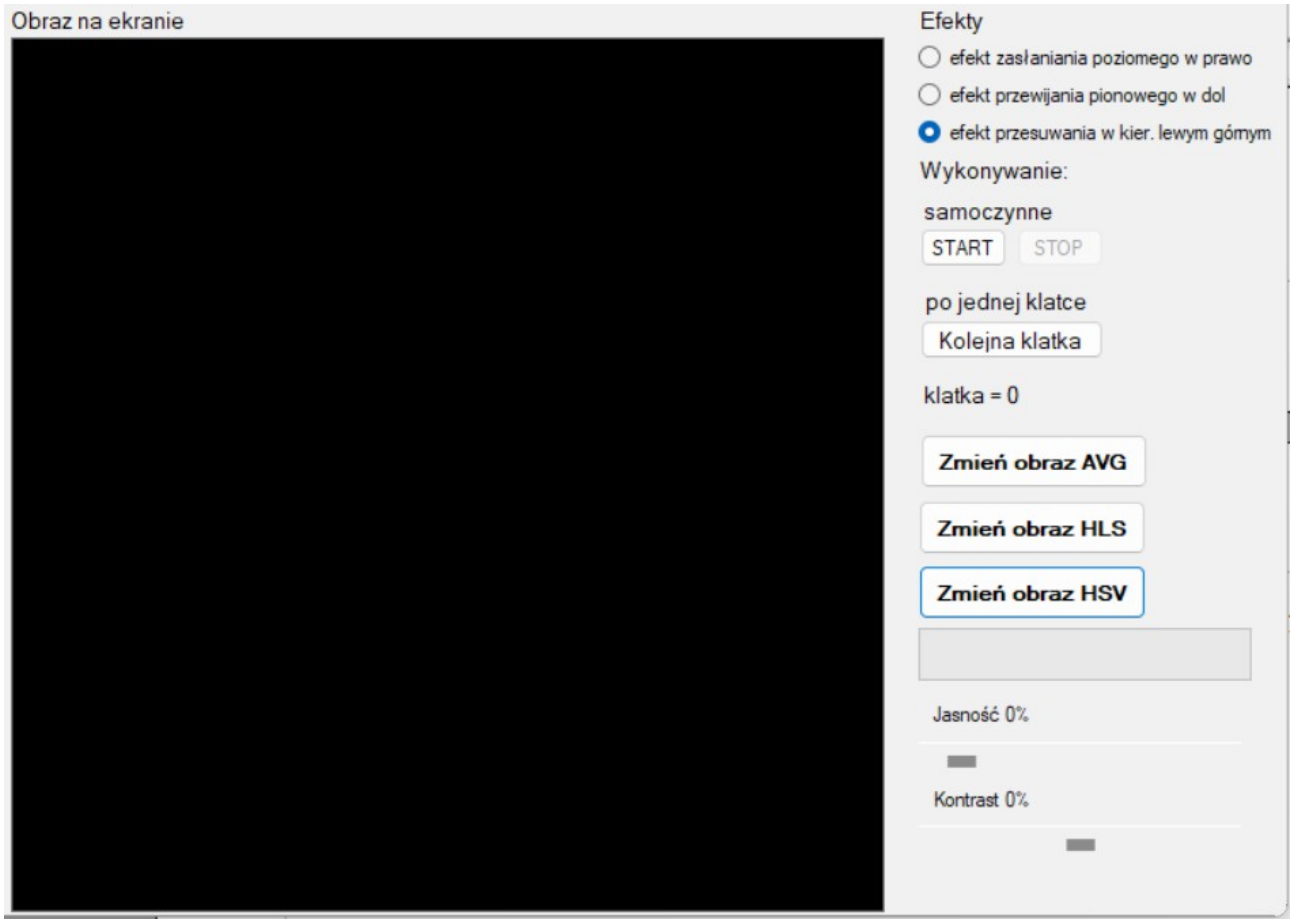
```
public void set_JC(object sender, EventArgs e)
{
    System.Drawing.Color pixel;
    int r, g, b;
    for (int j = 1; j <= L; j++)
        for (int i = 1; i <= K; i++)
        {
            pixel = m_obraz_w_pamieci.GetPixel(i - 1, j - 1);
            //set J
            r= pixel.R+hScrollBarJ.Value;
            if (r < 0) r = 0;
            if (r > 255) r = 255;
            g = pixel.G + hScrollBarJ.Value;
            if (g < 0) g = 0;
            if (g > 255) g = 255;
            b = pixel.B + hScrollBarJ.Value;
            if (b < 0) b = 0;
            if (b > 255) b = 255;


            pixel = System.Drawing.Color.FromArgb(r, g, b);
            m_ekran.SetPixel(i - 1, j - 1, pixel);
        }
    SetBitMap(ref m_ekran);
    LiczJC(m_ekran);
}
```

do kolorów pikseli dodawana jest wartość z zakresu <-255,255> pochodząca ze scrollbara. Następnie zapewnione jest, że wartość każdego koloru należy do przedziału <0,255> , po czym generowany jest piksel wynikowy z wyliczonym nowym kolorem.

Wyniki:







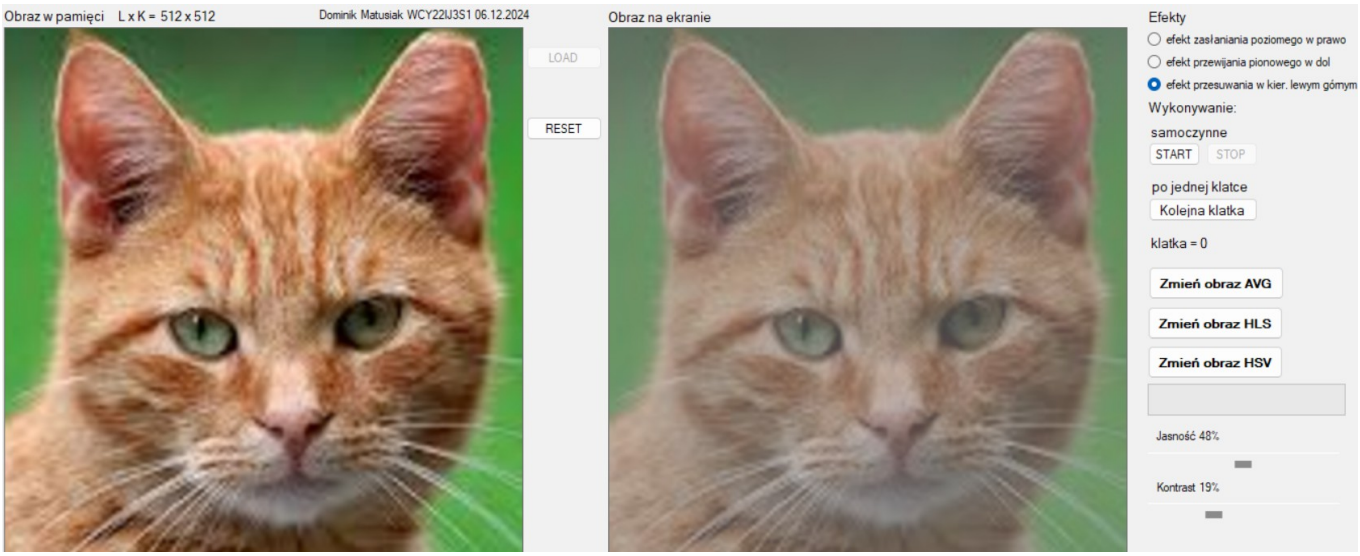
Zmiana kontrastu:

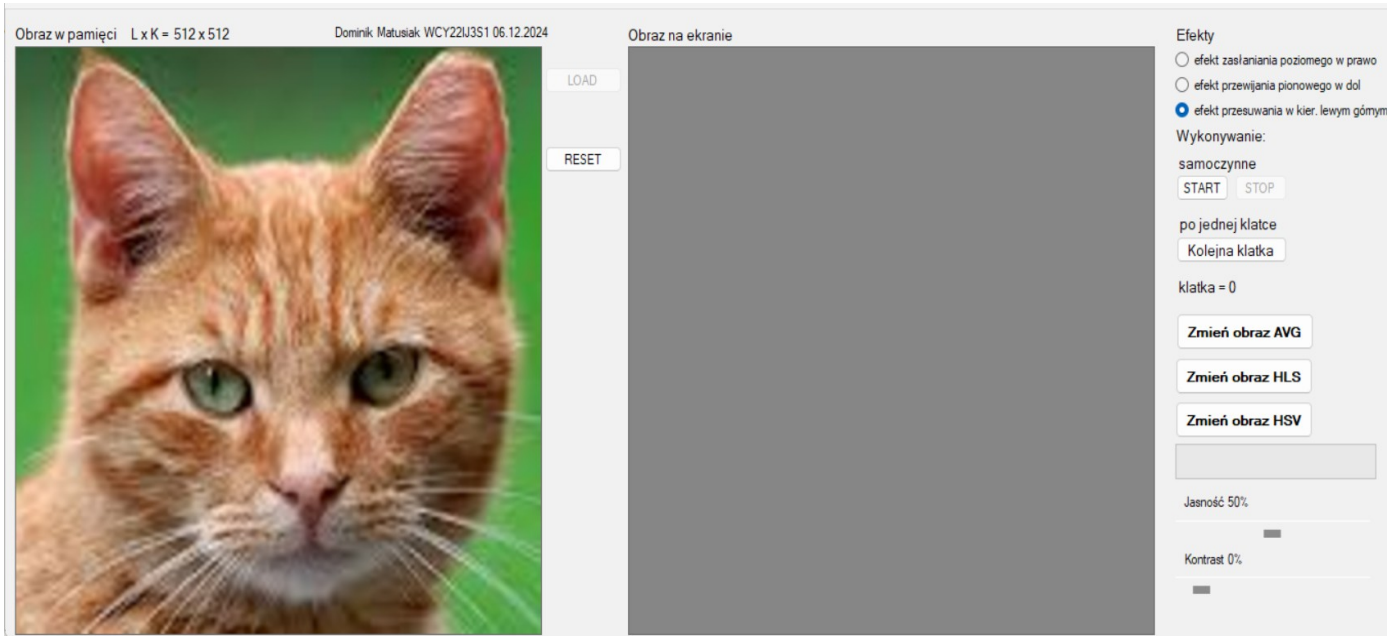
```
public void set_C(object sender, EventArgs e)
{
    System.Drawing.Color pixel;
    int r, g, b;
    double alfa = (double)hScrollBarC.Value/1000.0;
    double a = Math.Tan(alfa);
    for (int j = 1; j <= L; j++)
        for (int i = 1; i <= K; i++)
        {
            pixel = m_obraz_w_pamieci.GetPixel(i - 1, j - 1);
            //set J
            r = pixel.R;
            r = (int)((a * (r - 128.0)) + 128);
            if (r < 0) r = 0;
            if (r > 255) r = 255;
            g = pixel.G;
           g = (int)((a * (g - 128.0)) + 128);
            if (g < 0) g = 0;
            if (g > 255) g = 255;
            b = pixel.B;
            b = (int)((a * (b - 128.0)) + 128);
            if (b < 0) b = 0;
            if (b > 255) b = 255;

            pixel = System.Drawing.Color.FromArgb(r, g, b);
            m_ekran.SetPixel(i - 1, j - 1, pixel);
        }
    SetBitMap(ref m_ekran);
    LiczJC(m_ekran);
}
```

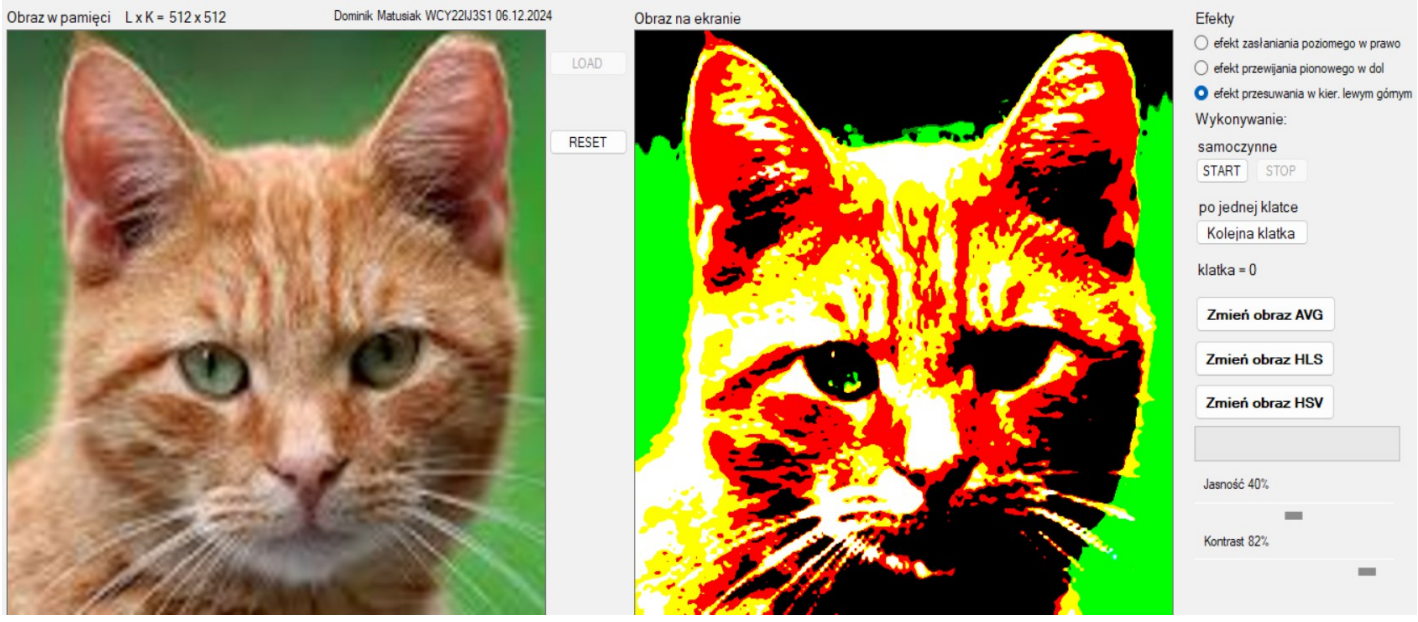
ze scrollbara pobierany jest kąt nachylenia krzywej odwzorowania tonów, wyliczany z niego jest współczynnik kierunkowy a krzywej, po czym zgodnie z nową krzywą są aktualizowane kolory

każdego piksela. Następnie upewniono się że intensywności każdego koloru należą do przedziału <0,255>, i wygenerowane zostały piksele nowego obrazu.

Wyniki:







wyliczanie jasności i kontrastu zrealizowano tą samą metodą jak w zadaniu 1.

Wnioski z Lab 2:

Widać, że metody zamiany obrazu kolorowego na monochromatyczny dają różne obrazy wynikowe, ze względu na różne metody wyliczania jasności obrazu oryginalnego. Ponadto widać, że zmiany jasności można dokonać, dodając lub odejmując pewną wartość do kolorów piksela oryginalnego, a zmiany kontrastu wyliczając nową krzywą odwzorowania tonów.