Grafika komputerowa

Sprawozdanie z laboratorium 1 i 2

Tematy: Przekształcenia obrazów rastrowych; Poprawa jakości obrazu

Student: Bartosz Baniak
Nr albumu: 80512
Nr w grupie: 1

Grupa: WCY21IJ2S1
Daty laboratoriów: 03.11.2023 r.; 10.11.2023 r.
Prowadzący: dr inż. Marek Salamon

1. Treść zadania

Laboratorium 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.

Oceny:
4 pkt. DST
5 pkt. DST+
6 pkt. DB
7 pkt. DB+
8 pkt. BDB

Zestaw 1.

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

Laboratorium 2

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.

   Razem: 4 pkt.

Oceny:
4 pkt. DST
5 pkt. DST+
6 pkt. DB
7 pkt. DB+
8 pkt. BDB

2. Sposób rozwiązania

Ćwiczenie 1

Skorzystano z funkcji:
ReadPixel(i,j); - procedura odczytuje i wysyła do urządzenia zobrazowania zawartość komórki mapy bitowej OBRAZ o adresie (i,j).
ReadTlo(N); - procedura wysyła do urządzenia zobrazowania piksel o kolorze N, gdzie N jest kolorem tła.

Algorytmy wykonane dla zestawu 1:

1.1. Metoda Efekt1() manipuluje obrazem, przesuwając go w prawo poprzez iterację po kolumnach. Wykorzystuje zmienne p, L, i K, gdzie p kontroluje przesunięcie, L określa liczbę kolumn obrazu, a K odpowiada liczbie wierszy. Wewnętrzne pętle wyświetlają tło (ReadTlo(N)) oraz piksele obrazu (ReadPixel(i, j)) z uwzględnieniem wartości p, symulując efekt przesuwania obrazu.

```
public void Efekt1()
{
    //efekt: przesuwanie się obrazu w kierunku prawej strony

    if (p >= L) p = 0;

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

1.2. Metoda Efekt2() symuluje przewijanie obrazu wzdłuż przekątnej ekranu w kierunku górnego lewego wierzchołka poprzez wyświetlanie tła i pikseli obrazu. Pierwsza część pętli przechodzi przez kolumny od p + 1 do L, wyświetlając piksele obrazu i tła. Druga część pętli wykonuje podobne operacje dla fragmentu przewijania od górnego lewego wierzchołka do p. W rezultacie kombinacja odczytu tła i pikseli obrazu w zależności od p symuluje przewijanie obrazu wzdłuż przekątnej ekranu.

```
public void Efekt2()
{
    //efekt: przewijanie się obrazu wzdłuż przekątnej ekranu w kierunku górnego lewego wierzchołka

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

1.3. Metoda Efekt3() realizuje efekt zasłaniania pionowego obrazu w kierunku górnej krawędzi ekranu poprzez wyświetlanie pikseli obrazu i tła. Sprawdza, czy wartość p przekracza liczbę kolumn L, resetując ją do zera w takim przypadku. Następnie iteruje przez piksele obrazu dla wierszy od 1 do L - p i wyświetla je. Kolejna część tej metody wyświetla tło dla p wierszy obrazu, symulując efekt zasłaniania pionowego obrazu.

```
public void Efekt3()
{
    //efekt: zasłanianie pionowego obrazu w kierunku górnej krawędzi ekranu

    if (p >= L) p = 0;

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

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

W celu powiązania ProgressBar z klatką należało ustawić jego wartość jako wartość parametru p w funkcji Work():

```
p++;
progressBar1.Value = p;
```

Ćwiczenie 2

Zadanie 1

a) Należy obliczyć wartość maksymalną oraz minimalną ze zbioru trzech składowych tonalnych (R, G, B), a następnie wyznaczyć wartość średnią z wartości maksymalnej i minimalnej. Uzyskaną wartość należy podstawić jako nowe R, G, B dla pikseli:

```
Color pixelColor = inputImage.GetPixel(i, j);
maxValue = Math.Max(pixelColor.R, Math.Max(pixelColor.G, pixelColor.B));
minValue = Math.Min(pixelColor.R, Math.Min(pixelColor.G, pixelColor.B));
int monoValue = (maxValue + minValue) / 2;
Color monoColor = Color.FromArgb(monoValue, monoValue, monoValue);
monochromeImage.SetPixel(i, j, monoColor);
```

b) Należy wyznaczyć wartość maksymalną z R, G, B i podstawić jako składowe tonalne:

```
Color pixelColor = inputImage.GetPixel(i, j);
int maxValue = Math.Max(pixelColor.R, Math.Max(pixelColor.G, pixelColor.B));
Color monoColor = Color.FromArgb(maxValue, maxValue, maxValue);
monochromeImage.SetPixel(i, j, monoColor);
```

c) Należy wyznaczyć średnią arytmetyczną składowych i podstawić jako składowe tonalne:

```
Color pixelColor = inputImage.GetPixel(i, j);
int averageValue = (int)((pixelColor.R + pixelColor.G + pixelColor.B) / 3);
Color monoColor = Color.FromArgb(averageValue, averageValue, averageValue);
monochromeImage.SetPixel(i, j, monoColor);
```

W celu policzenia jasności dla każdego z podpunktów należy skorzystać ze wzoru:

$$J = \frac{1}{MN} \sum_{i=1}^{M} \sum_{j=1}^{N} f(i,j)$$

Dla HLS:

```
int M = inputImage.Width;
int N = inputImage.Height;
int J = 0;
int maxValue = 0;
int minValue = 0;
int value = 0;
for (int i = 0; i < M; i++)
{
    for (int j = 0; j < N; j++)
    {
        Color pixelColor = inputImage.GetPixel(i, j);
        maxValue = Math.Max(pixelColor.R, Math.Max(pixelColor.G, pixelColor.B));
        minValue = Math.Min(pixelColor.R, Math.Min(pixelColor.G, pixelColor.B));
        value = (maxValue + minValue) / 2;
        J += value;
    }
}
return (double)(J / (M * N));
```

Dla HSV:

```
int M = inputImage.Width;
int N = inputImage.Height;
int J = 0;
for (int i = 0; i < M; i++)
{
    for (int j = 0; j < N; j++)
    {
        Color pixelColor = inputImage.GetPixel(i, j);
        J += (int)(Math.Max(pixelColor.R, Math.Max(pixelColor.G, pixelColor.B)));
    }
}
return (double)(J / (M * N));
```

Dla AVG:

```
int M = inputImage.Width;
int N = inputImage.Height;
int J = 0;
for (int i = 0; i < M; i++)
{
    for (int j = 0; j < N; j++)
    {
        Color pixelColor = inputImage.GetPixel(i, j);
        J += (int)((pixelColor.R + pixelColor.G + pixelColor.B) / 3);
    }
}
return (double)(J / (M * N));
```

W celu policzenia kontrastu dla każdego z podpunktów należy skorzystać ze wzoru:

$$C = \sqrt{\frac{1}{MN}\sum_{i=1}^{M}\sum_{j=1}^{N}[f(i,j) - J]^2}$$

```
int M = inputImage.Width;
int N = inputImage.Height;
double J = brightness;
double C = 0;
double temp = 0;

for (int i = 0; i < M; i++)
{
    for (int j = 0; j < N; j++)
    {
        Color pixelColor = inputImage.GetPixel(i, j);
        temp += Math.Pow(((int)(Math.Max(pixelColor.R, Math.Max(pixelColor.G, pixelColor.B))) - J), 2);
    }
}
temp = temp / (M * N);
C = Math.Sqrt(temp);
return C;
```

Zadanie 2

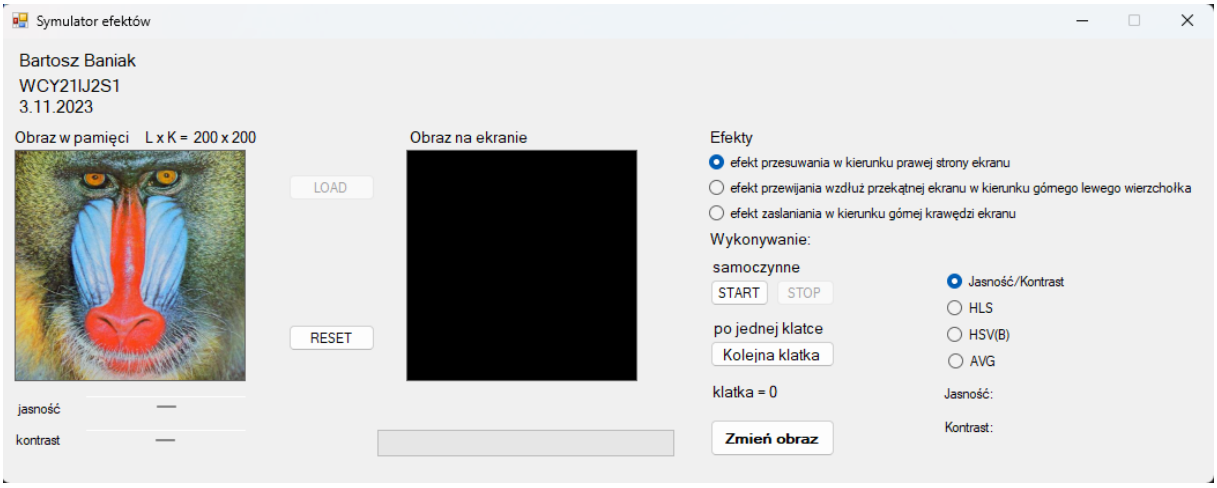
Należy wykorzystać algorytmy oraz wzory z zadania 1, w celu wyznaczenia nowych składowych tonalnych R,G,B dla wartości obecnej o wartość z ScrollBar jasności, korzystając ze wzoru: nowa_składowa = (tan(kontrast) * (poprzednia_wartość_składowej – 127.5) + 127.5), w celu wyznaczenia kontrastu i powiązania go ze zmianą wartości poprzez ScrollBar.

```
newR = Math.Max(0, Math.Min((int)(Math.Tan((double)contrastBar.Value / 100) * ((double)newR - 127.5) + 127.5), 255));
newG = Math.Max(0, Math.Min((int)(Math.Tan((double)contrastBar.Value / 100) * ((double)newG - 127.5) + 127.5), 255));
newB = Math.Max(0, Math.Min((int)(Math.Tan((double)contrastBar.Value / 100) * ((double)newB - 127.5) + 127.5), 255));
```

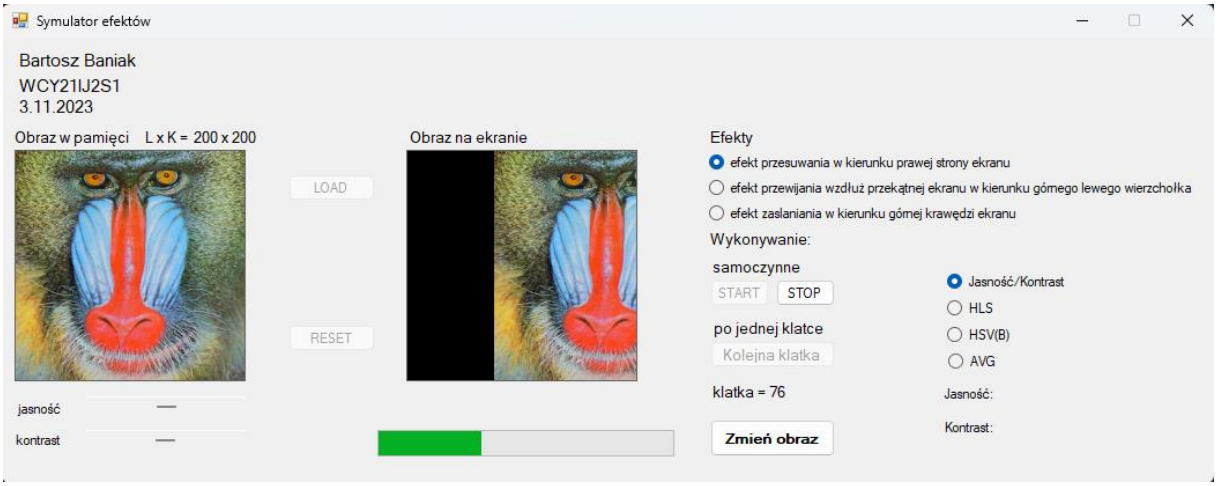
3. Wyniki

Ćwiczenie 1

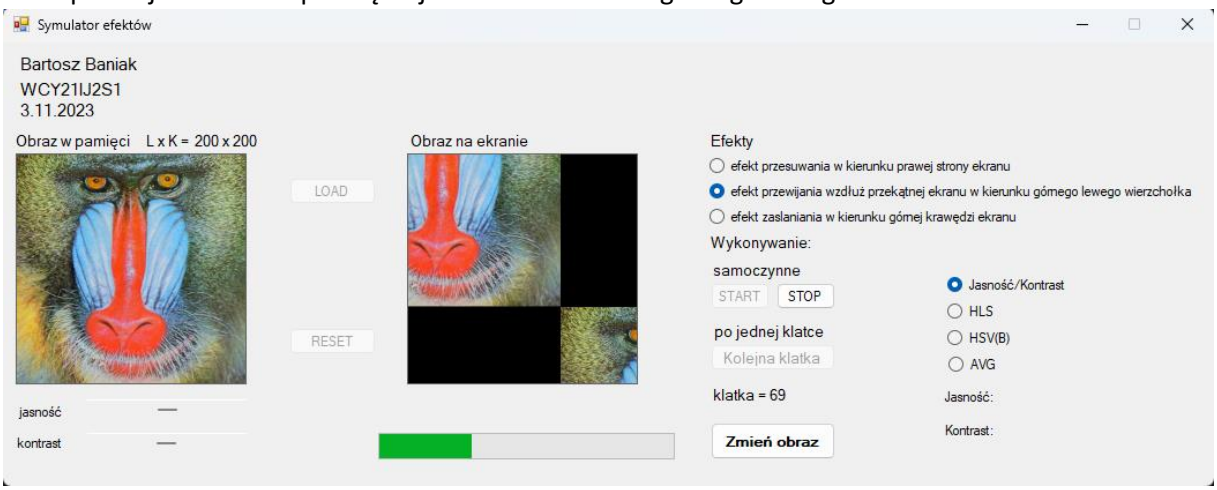
Zmodyfikowane okno aplikacji:



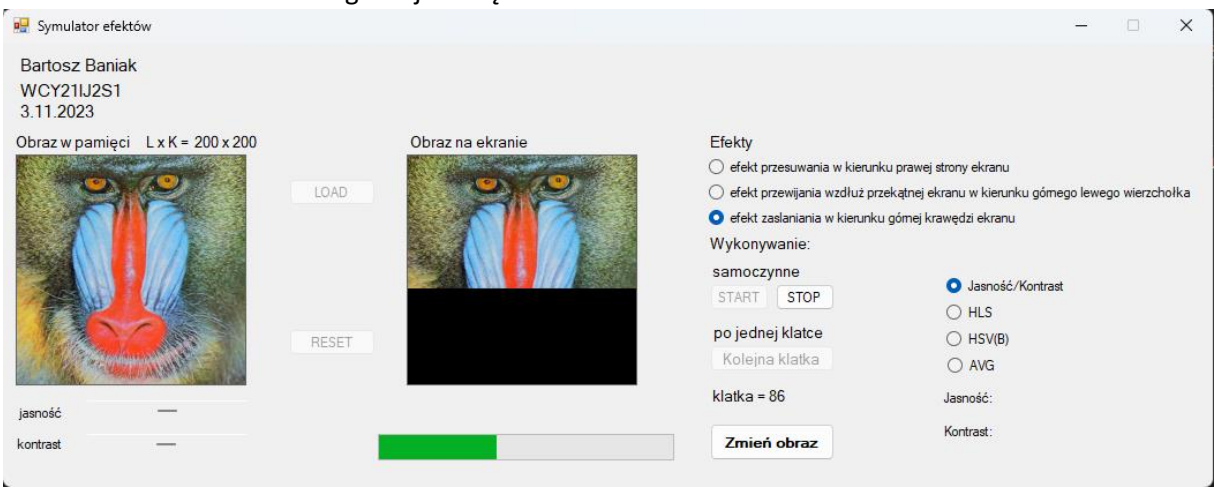
Efekt przesuwania w kierunku prawej strony ekranu



Efekt przewijania wzdłuż przekątnej ekranu w kierunku górnego lewego wierzchołka:



Efekt zasłaniania w kierunku górnej krawędzi ekranu:



Ćwiczenie 2

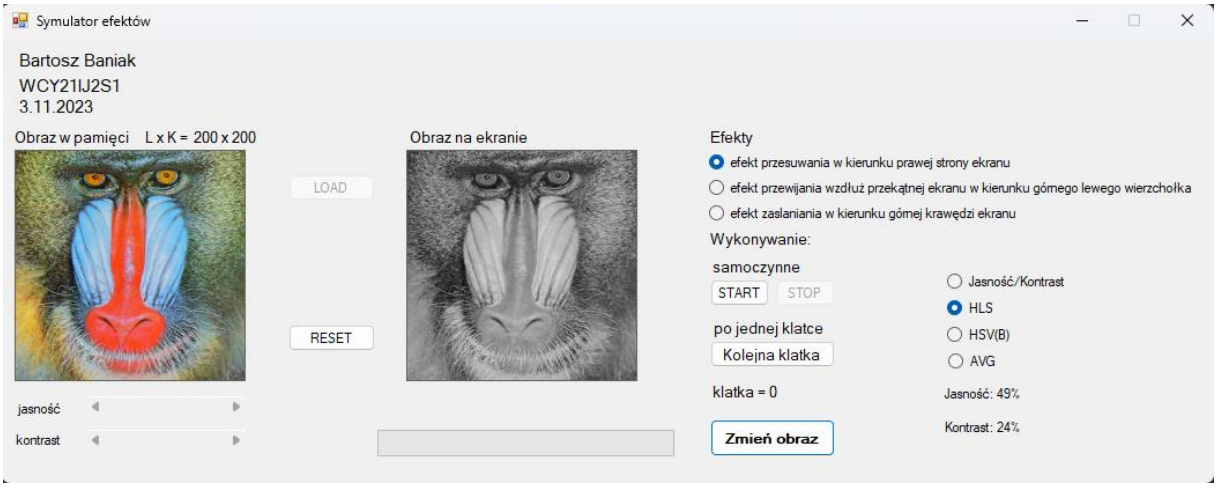
Zadanie 1

HLS

Kod:

```
Bitmap monochromeImage = new Bitmap(K, L);
int maxValue = 0;
int minValue = 0;


for (int i = 0; i < K; i++)
{
    for (int j = 0; j < L; j++)
    {
        Color pixelColor = inputImage.GetPixel(i, j);
        maxValue = Math.Max(pixelColor.R, Math.Max(pixelColor.G, pixelColor.B));
        minValue = Math.Min(pixelColor.R, Math.Min(pixelColor.G, pixelColor.B));
        int monoValue = (maxValue + minValue) / 2;
        Color monoColor = Color.FromArgb(monoValue, monoValue, monoValue);
        monochromeImage.SetPixel(i, j, monoColor);
    }
}
return monochromeImage;
```

Działanie:

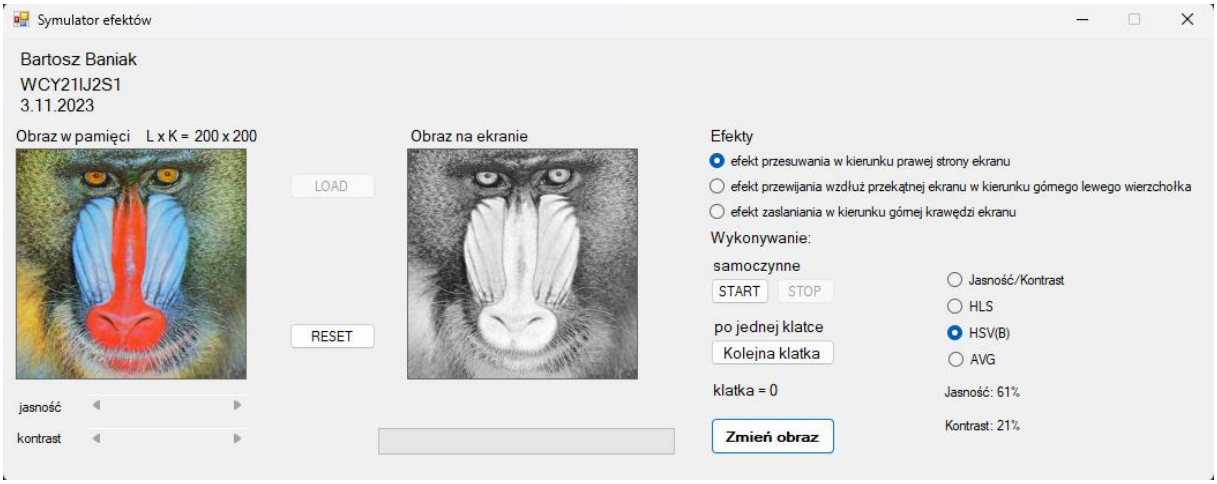


HSV

Kod:

```
Bitmap monochromeImage = new Bitmap(K, L);

for (int i = 0; i < K; i++)
{
    for (int j = 0; j < L; j++)
    {
        Color pixelColor = inputImage.GetPixel(i, j);
        int maxValue = Math.Max(pixelColor.R, Math.Max(pixelColor.G, pixelColor.B));
        Color monoColor = Color.FromArgb(maxValue, maxValue, maxValue);
        monochromeImage.SetPixel(i, j, monoColor);
    }
}
return monochromeImage;
```

Działanie:

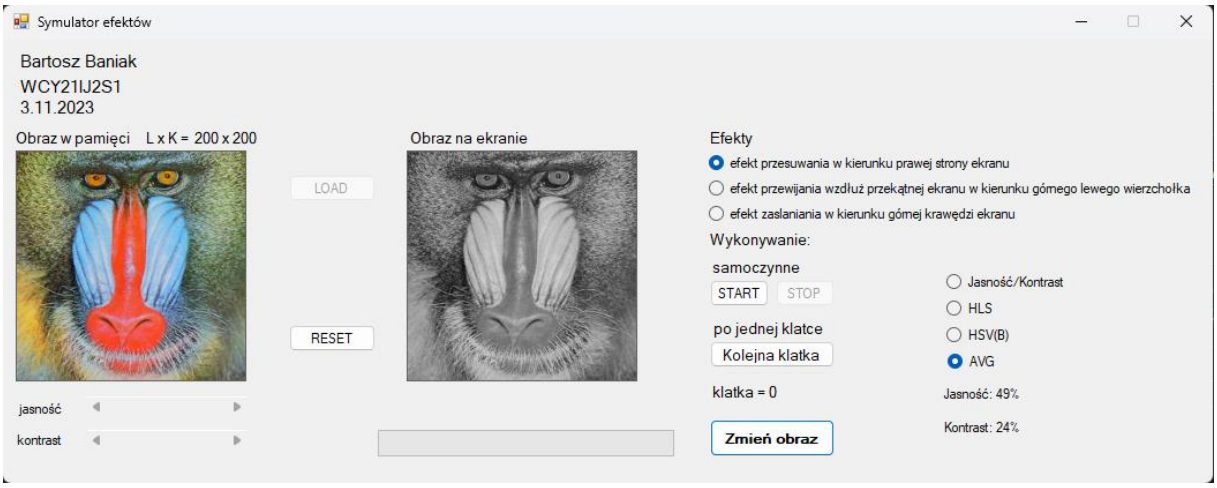


AVG

Kod:

```
Bitmap monochromeImage = new Bitmap(K, L);

for (int i = 0; i < K; i++)
{
    for (int j = 0; j < L; j++)
    {
        Color pixelColor = inputImage.GetPixel(i, j);
        int averageValue = (int)((pixelColor.R + pixelColor.G + pixelColor.B) / 3);
        Color monoColor = Color.FromArgb(averageValue, averageValue, averageValue);
        monochromeImage.SetPixel(i, j, monoColor);
    }
}
return monochromeImage;
```

Działanie:



Zadanie 2

Kod:

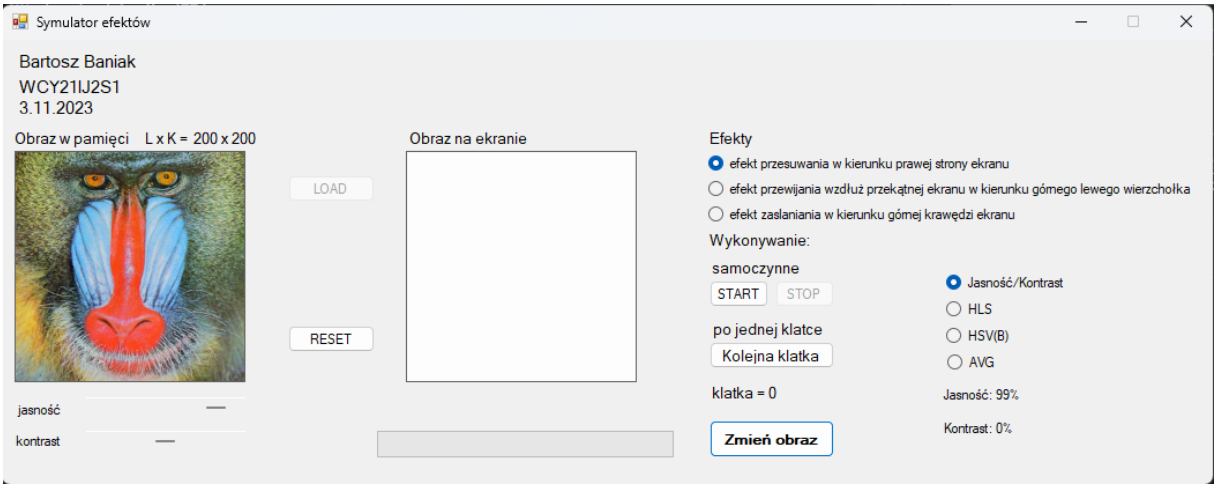
```
private void brightnessBar_ValueChanged(object sender, EventArgs e)
{
    contrastBar.Value = 78;
    if (radioButtonBC.Checked)
    {
        m_ekran = changeBrightness(m_obraz_w_pamieci);
        int brightness = (int)DetermineBrightnessRGB(m_ekran);
        int contrast = (int)DetermineContrastRGB(m_ekran, brightness);

        double brightnessP = Percentage(brightness);
        double contrastP = Percentage(contrast);

        brightnessLabel.Text = "Jasność: " + brightnessP + "%";
        contrastLabel.Text = "Kontrast: " + contrastP + "%";
    }

        SetBitMap(ref m_ekran);
}
```

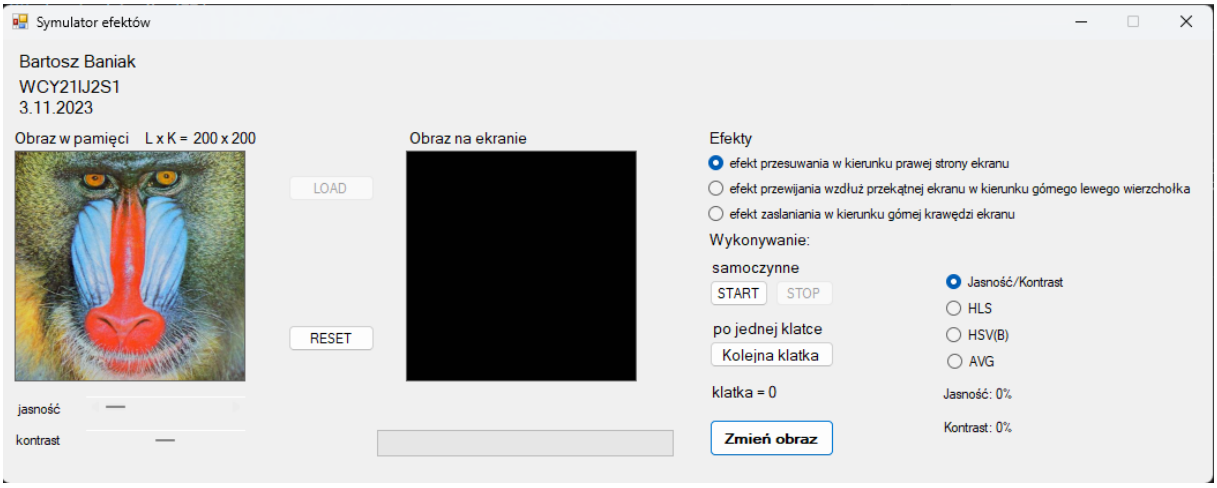
```
private void contrastBar_ValueChanged(object sender, EventArgs e)
{
    brightnessBar.Value = 0;
    if (radioButtonBC.Checked)
    {
        m_ekran = changeBrightness(m_obraz_w_pamieci);
        int brightness = (int)DetermineBrightnessRGB(m_ekran);
        int contrast = (int)DetermineContrastRGB(m_ekran, brightness);

        double brightnessP = Percentage(brightness);
        double contrastP = Percentage(contrast);

        brightnessLabel.Text = "Jasność: " + brightnessP + "%";
        contrastLabel.Text = "Kontrast: " + contrastP + "%";
    }

    SetBitMap(ref m_ekran);
}
```

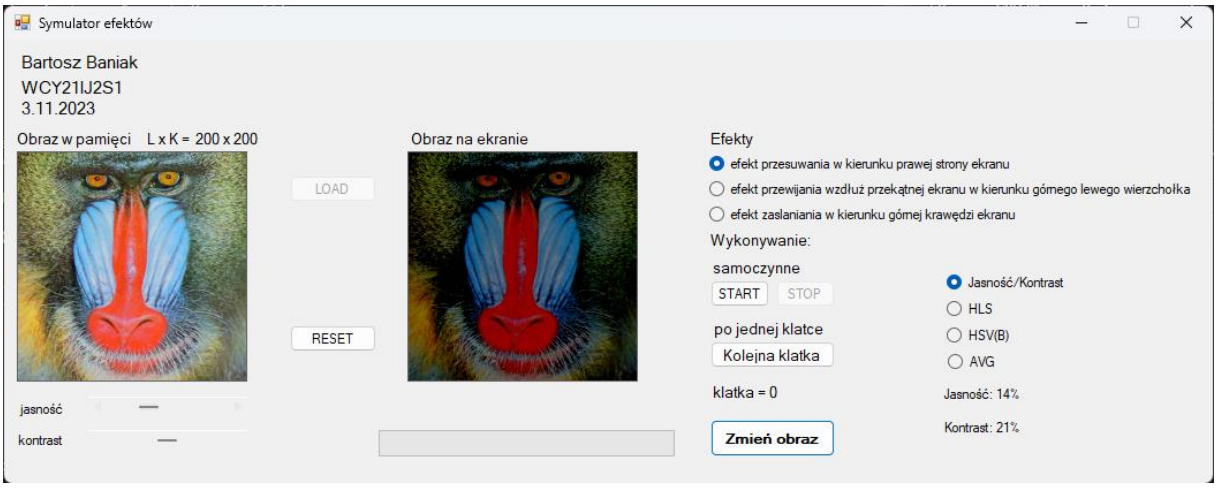
Funkcja pomocnicza wyznaczająca wartość jasności i kontrastu w procentach:

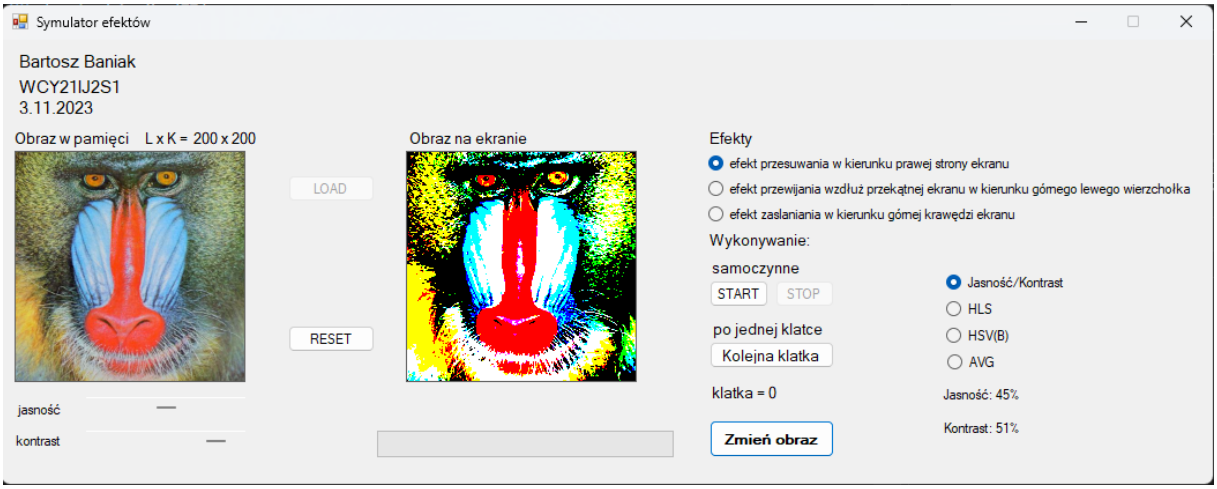
```
public double Percentage (int value)
{
    if(value < -255 || value > 255)
    {
        throw new ArgumentOutOfRangeException();
    }

    double percentage = (value) / 255.0 * 100;


    return (int)percentage;
}
```

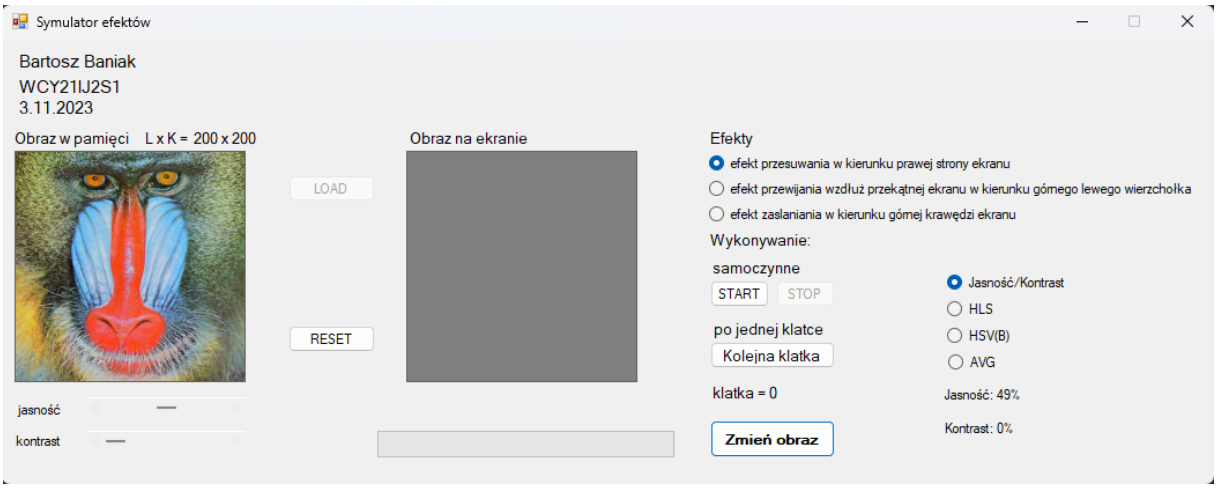
Działanie:






Symulator efektów
Bartosz Baniak
WCY21IJ2S1
3.11.2023
Obraz w pamięci L x K = 200 x 200
LOAD
RESET
Obraz na ekranie
Efekty
efekt przesuwania w kierunku prawej strony ekranu
efekt przewijania wzdłuż przekątnej ekranu w kierunku gómego lewego wierzchołka
efekt zaslaniania w kierunku gómej krawędzi ekranu
Wykonywanie:
samoczynne
START
STOP
po jednej klatce
Kolejna klatka
klatka = 0
Zmień obraz
Jasność/Kontrast
HLS
HSV(B)
AVG
Jasność: 14%
Kontrast: 21%
jasność
kontrast


Symulator efektów
Bartosz Baniak
WCY21IJ2S1
3.11.2023
Obraz w pamięci L x K = 200 x 200
LOAD
RESET
Obraz na ekranie
Efekty
efekt przesuwania w kierunku prawej strony ekranu
efekt przewijania wzdłuż przekątnej ekranu w kierunku gómego lewego wierzchołka
efekt zaslaniania w kierunku gómej krawędzi ekranu
Wykonywanie:
samoczynne
START
STOP
po jednej klatce
Kolejna klatka
klatka = 0
Zmień obraz
Jasność/Kontrast
HLS
HSV(B)
AVG
Jasność: 45%
Kontrast: 51%
jasność
kontrast


Symulator efektów
Bartosz Baniak
WCY21IJ2S1
3.11.2023
Obraz w pamięci L x K = 200 x 200
LOAD
RESET
Obraz na ekranie
Efekty
efekt przesuwania w kierunku prawej strony ekranu
efekt przewijania wzdłuż przekątnej ekranu w kierunku gómego lewego wierzchołka
efekt zaslaniania w kierunku gómej krawędzi ekranu
Wykonywanie:
samoczynne
START
STOP
po jednej klatce
Kolejna klatka
klatka = 0
Zmień obraz
Jasność/Kontrast
HLS
HSV(B)
AVG
Jasność: 49%
Kontrast: 0%
jasność
kontrast

4. Wnioski

Funkcja Percentage jest zaprojektowana do obliczania procentowej wartości na podstawie przekazanej wartości value. Problemem, który wpływa na wynik funkcji równy 99% jest sposób przekształcenia wartości procentowej. Wynik operacji double percentage = (value) / 255.0 * 100 jest obcięty do typu int, co skutkuje utratą wartości po przecinku. W ten sposób zwracanie wartości typu int prowadzi do zaokrąglenia liczby zmiennoprzecinkowej w dół.
Podczas tych ćwiczeń w środowisku VS eksplorowano projektowanie aplikacji okienkowych oraz dodawanie interakcji do elementów takich jak przyciski i suwaki. Poznano zasady transformacji obrazów, takie jak przesuwanie, zasłanianie i przewijanie, a także trzy metody konwersji obrazów kolorowych na obrazy w odcieniach szarości. Dodatkowo, zdobyto również wiedzę o sposobach manipulacji jasności i kontrastu obrazów, co pozwoliło lepiej zrozumieć jak wpływać na ich wygląd i właściwości.