

WOJSKOWA AKADEMIA TECHNICZNA



Grafika komputerowa

Sprawozdanie z pracy laboratoryjnej Nr 1-2

Temat: Przekształcanie obrazów rastrowych.

Prowadzący: dr inż. Marek Salamon

Autor: Mateusz Jasiński WCY20IJ1S1

Data wykonania: 29.11.2022

1. Treść zadania

Lab 1 - Zestaw 1S:

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:
 $K=L \neq 256$, 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 8:

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

Lab 2 – Zestaw 1:

1. Zamienić obraz kolorowy (RGB) o 24-bitowej strukturze piksela z ćwic.1 na obraz monochromatyczny reprezentowany przez skalę odcieni szarości (poziomy jasności) 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 umożliwiającą zmianę jasności w pełnym zakresie zmienności (od 0 do 100%):
 - a) wejściowego obrazu kolorowego (z ćwic. 1). **2 pkt.**
 - b) wyznaczyć jasność (J) i kontrast (K) uzyskiwanych obrazów. **0.5 pkt. i 1 pkt**Do zmiany jasności wykorzystać suwak ScrollBar (H/V) . **0.5 pkt.**
Razem: 4 pkt.

2. Wstęp teoretyczny

Laboratorium 1

Obraz rastrowy jest bitmapą, czyli dwuwymiarowa tablica, o L liniach (wierszach) i K kolumnach. Wyświetlanie obrazu (w przypadku monitorów CRT) polega na wyświetlaniu kolejnych pikseli od lewej do prawej, od góry do dołu. Do uzyskania efektów takich jak przemieszczanie, ukrywanie obrazu modyfikuje się miejsce, z którego zaczynamy/kończymy wczytywać piksele do wyświetlenia.

Laboratorium 2

Obraz RGB – jest modelem przestrzeni barw, w której konkretny kolor uzyskiwany jest poprzez zmieszanie w odpowiednich proporcjach kolorów R – czerwonego, G – zielonego, B – niebieskiego. Każdy piksel w modelu RGB jest określany przez 3 składowe przyjmujące wartości od 0 do 255, które są właśnie proporcjami trzech wspomnianych kolorów.

Obraz HSL – jest modelem opisu kolorów, w którym każdej barwie postrzeganej przez człowieka przyporządkowany jest jeden punkt w przestrzeni trójwymiarowej identyfikowany za pomocą trzech składowych:

- H – barwa (od 0 do 360 stopni),
- S – nasycenie koloru (z przedziału 0 - 1 lub 0% - 100%),
- L – jasność (z przedziału 0 - 1 lub 0% - 100%).

Model HSL nie jest w stanie przedstawić wszystkich barw, ponieważ nie wszystkie barwy można przedstawić jedną długością fali definiowaną przez składową H.

Obraz HSV – jest modelem opisu przestrzeni barw, który opisuje stożek oparty na kole barw. Składa się z trzech składowych:

- H – odcień światła widzialnego (od 1 do 360 stopni),
- S – nasycenie koloru (promień na kole barw)
- V – jasność (wysokość stożka).

Jasność - jest to cecha określająca ilość światła w kolorze. Jasność można obliczyć na podstawie wzoru:

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

Gdzie: M, N – wymiary obrazu; $f(i, j)$ - poziom jasności w punkcie (i, j)

Kontrast – jest to parametr opisujący różnice w jasnościach pikseli. Wyraża się wzorem:

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

Gdzie: J – jasność; M, N – wymiary obrazu; $f(i, j)$ - poziom jasności w punkcie (i, j)

3. Cel ćwiczenia i sposób rozwiązania

Laboratorium 1

Cel ćwiczenia

Głównym celem ćwiczenia było napisanie 3 algorytmów sterującym generowaniem adresu odczytu w celu uzyskania określonych efektów. Algorytmy zostały zaimplementowane w programie, który symuluje działanie urządzenia rastrowego. Program wczytuje plik graficzny, po czym wyświetla go w lewym okienku, zaś drugie pełni rolę symulowanego ekranu. Zadanie zostało wykonane w środowisku programistycznym Microsoft Visual Studio.

Do wykonania poleceń konieczna była znajomość funkcji:

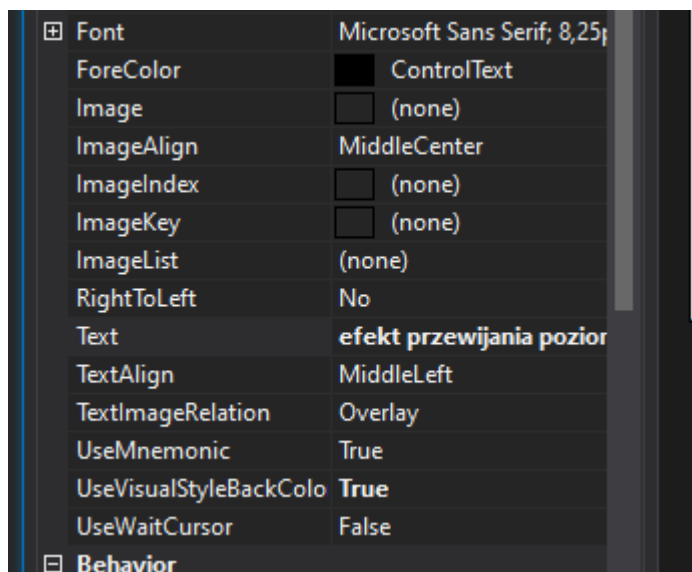
- ReadPixel(i, j) – odczytuje i wysyła do urządzenia zobrazowania wartość komórki mapy bitowej o adresie (i, j).
- ReadTlo(N) – wysyła do urządzenia zobrazowania piksel o kolorze określoną zmienną kolor.

W programie występują zmienne:

- K – szerokość ekranu.
- L – wysokość ekranu.
- N – kolor.
- p – zmienna pomocnicza

Modyfikacja okienka aplikacji

Modyfikację nazw funkcji ekranu dokonujemy poprzez kliknięcie PPM na jeden z elementów, po czym wybieramy „Properties” i zmieniamy wartość „Text”.



Dodanie informacji o wykonawcy

Z narzędzia Toolbox wybieramy element „Label” i przeciągamy go do okienka aplikacji. Następnie wchodzimy do „Properties” i modyfikujemy wartość „Text”.

Dodanie nowego obrazu

Tworzymy nowy obraz o dowolnych rozmiarach i zapisujemy go w formacie .bmp np. w programie Paint. Przenosimy zapisy obraz do \bin\Debug w projekcie. W kodzie modyfikujemy nazwę obrazu w funkcji wczytującej plik do programu.

```

public f_graf_kom()
{
    InitializeComponent();

    N = Color.Black;

    Bitmap im = new Bitmap("szop.bmp");

    m_obraz_w_pamieci = (Bitmap)im.Clone();

    L = m_obraz_w_pamieci.Height;
    K = m_obraz_w_pamieci.Width;

    m_ekran = (Bitmap)im.Clone();
}

```

Dodanie paska progresu „ProgressBar”

W narzędziu ToolBox wybieramy i przeciągamy do aplikacji element „ProgressBar”. Następnie ustawiamy jego wartości: minimalną, maksymalną, „skoku” oraz obecną. Można tego dokonać we właściwościach obiektu lub bezpośrednio w kodzie programu.

```

public f_graf_kom()
{
    InitializeComponent();

    N = Color.Black;

    Bitmap im = new Bitmap("szop.bmp");

    m_obraz_w_pamieci = (Bitmap)im.Clone();

    L = m_obraz_w_pamieci.Height;
    K = m_obraz_w_pamieci.Width;

    m_ekran = (Bitmap)im.Clone();

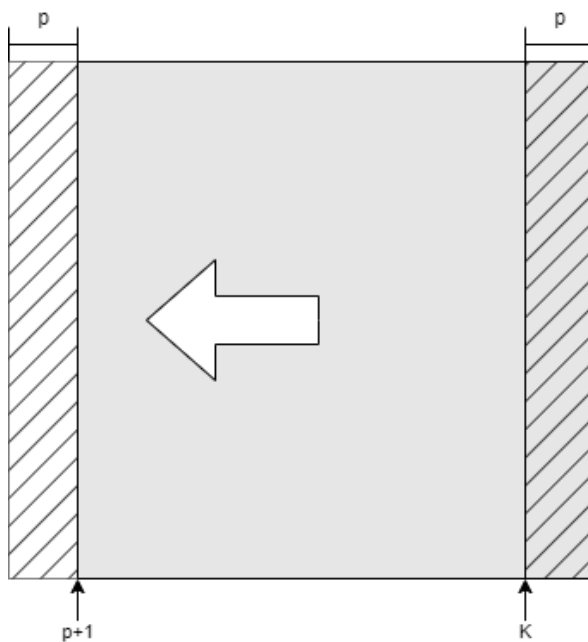
    progressBar1.Minimum = 1;
    progressBar1.Maximum = L;
    progressBar1.Step = 1;
    progressBar1.Value = 1;
}

```

Efekt 1

Efekt przewijania poziomego obrazu w kierunku lewej strony ekranu.

W celu uzyskania efektu przewijania poziomego obrazu w kierunku lewej strony ekranu należy przesuwać kolumny pikseli w lewo, a te które wyjdą poza ekran przepisywać na prawą krawędź. Program składa się z pętli z zagnieżdżonymi dwiema pętlami. W pętli zewnętrznej bierzemy kolejne liczby reprezentujące linie. W pierwszej pętli wewnętrznej dokonujemy iteracji od kolumny $1 + p$ do K , a w drugiej od 1 do p . Obie pętle wewnętrzne przypisują nowe wartości pikseli za pomocą funkcji `ReadPixel(i, j)`. Po zakończeniu pętli zewnętrznej wartość zmiennej p jest zwiększana o 1 , co sprawia, że pierwszy piksel w każdym wierszu jest pomijany przy następnym wykonaniu pętli i zostaje przypisany dopiero na końcu wiersza.



Kod programu

```
public void Efekt1()
{
    //efekt: przewijanie poziome obrazu w kierunku lewej strony ekranu

    if (p >= L) p = 0;

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

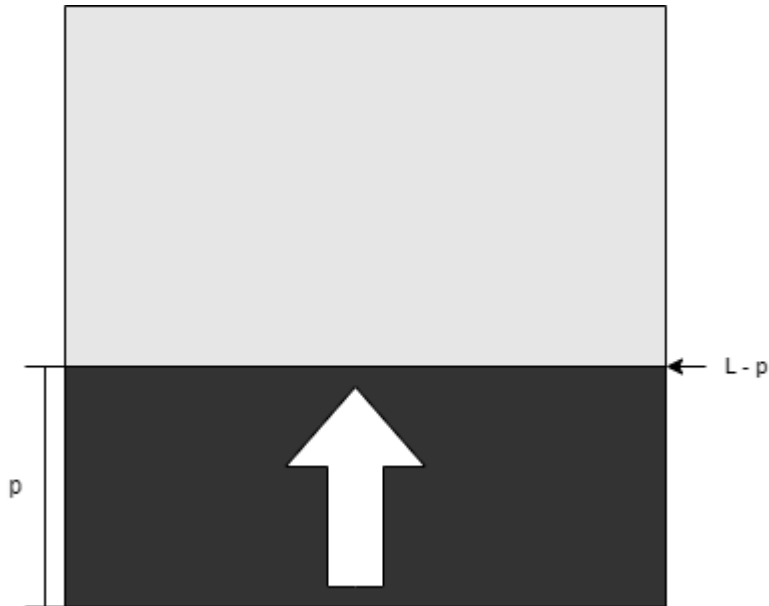
Efekt



Efekt 2

Efekt zasłaniania pionowego obrazu w kierunku górnej krawędzi ekranu.

W celu uzyskania efektu zasłaniania pionowego obrazu w kierunku górnej krawędzi ekranu należy podmieniać kolejne wiersze pikseli na wybrany kolor zaczynając od dolnego (ostatniego) wiersza, a kończąc na najwyższym (pierwszym). Program składa się z dwóch pętli. Pierwsza pętla odczytuje wiersze pikseli obrazu od 1 do $L - p$. Druga pętla wypełnia pozostałe p wierszy kolorem czarnym za pomocą funkcji `ReadTlo(N)`. Po wykonaniu obu pętli zmienna p jest zwiększana o 1, co powoduje, że wczytany zostaje o jeden mniej wiersz obrazu, a zamiast niego pojawia się wiersz czarnych pikseli.



Kod programu

```
public void Efekt2()
{
    //efekt: zasłanianie pionowe 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);
    }
}
```

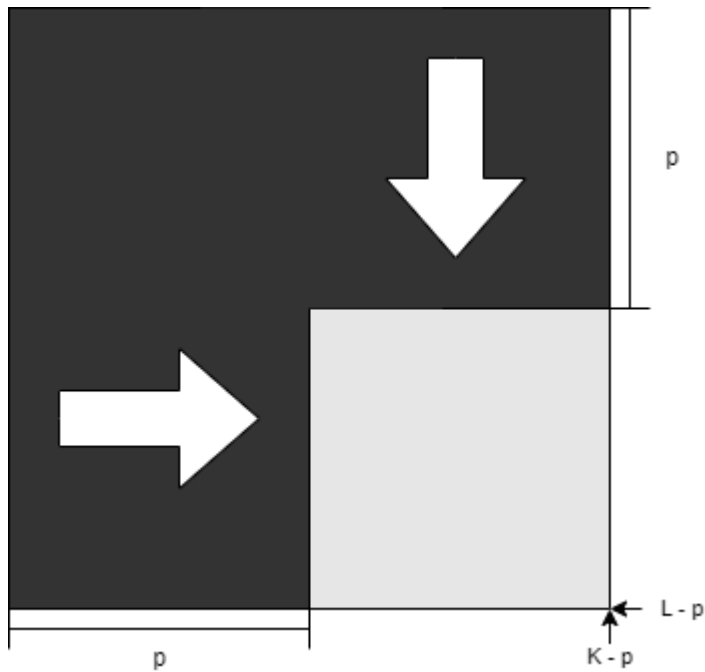
Efekt



Efekt 3

Efektu przesuwania obrazu wzdłuż przekątnej ekranu w kierunku dolnego prawego wierzchołka.

W celu uzyskania efektu przesuwania obrazu wzdłuż przekątnej ekranu w kierunku dolnego prawego wierzchołka, należy zwiększać liczbę kolumn i wierszy złożonych z czarnych pikseli, jednocześnie zmniejszając liczbę kolumn i wierszy wczytywanego obrazu. Program składa się z dwóch pętli. Pierwsza pętla wypełnia p pierwszych wierszy czarnymi pikselami. W drugiej pętli są zagnieżdżone dwie pętli, które wypełniają pozostałe $K - p$ wierszy. Pierwsza z nich wczytuje p czarnych pikseli w wierszu, a druga wczytuje $K - p$ początkowych pikseli z obrazu.



Kod programu

```
public void Efekt3()
{
    //efekt: przesuwanie obrazu wzdłuż przekątnej ekranu w kierunku dolnego prawego
    wierzchołka

    if (p >= L) p = 0;

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

Efekt



Laboratorium 2

Cel ćwiczenia

Celem ćwiczenia było rozbudowanie programu z poprzednich laboratoriów o zmianę kolorowego obrazu na monochromatyczny trzeba metodami, modyfikację i obliczenie jasności oraz kontrastu obrazu.

Wyznaczanie jasności i kontrastu

Jasność oraz kontrast wyliczane są na podstawie wzorów podanych we wstępie teoretycznym. Jasność to suma średniej arytmetycznej trzech składowych RGB każdego piksela podzielona przez liczbę pikseli w obrazie. Do uzyskania wyniku w procentach należy podzielić uzyskaną jasność przez maksymalną wartość składowej RGB, a następnie pomnożyć przez 100.

Kontrast możemy wyliczyć dopiero po wyznaczeniu jasności. Na początek sumujemy dla każdego piksela podniesioną do kwadratu różnicę średniej arytmetycznej składowych RGB i jasności. Następnie dzielimy ją przez liczbę pikseli, a na koniec pierwiastkujemy otrzymaną liczbę. Do uzyskania wyniku w procentach należy podzielić uzyskany kontrast przez połowę maksymalnej wartości składowej RGB, a następnie pomnożyć przez 100.

Kody obliczające jasność i kontrast uwzględnione są w kolejnych zadaniach.

Kod programu:

```
System.Drawing.Color pixel;
double suma = 0, jasnoc, kontrast, avg;
for (int j = 1; j <= L; j++)
    for (int i = 1; i <= K; i++)
    {
        pixel = m_obraz_w_pamieci.GetPixel(i - 1, j - 1);
        avg = (pixel.R + pixel.G + pixel.B) / 3;
        suma += avg;
    }
jasnoc = suma / (L * K);
jasnoc_value.Text = Math.Round(jasnoc / 255 * 100, 0) + "%";
suma = 0;
for (int j = 1; j <= L; j++)
    for (int i = 1; i <= K; i++)
    {
        pixel = m_ekran.GetPixel(i - 1, j - 1);
        avg = (pixel.R + pixel.G + pixel.B) / 3;
        suma += (avg - jasnoc) * (avg - jasnoc);
    }
kontrast = Math.Sqrt(suma / (L * K));
kontrast_value.Text = Math.Round(kontrast / (255 / 2) * 100, 0) + "%";
```

Przejsie z modelu RGB na HLS

Zmiana RGB na HLS polega na wyznaczeniu średniej arytmetycznej z sumy największej i najmniejszej wartości ze składowych RGB, a następnie ustawienie otrzymanego wyniku jako nowej wartości wszystkich składowych RGB.

Kod programu

```
public void RGB_NA_HLS()
{
    System.Drawing.Color pixel;
    double suma = 0, jasnoc, kontrast, l_hls;
    for (int j = 1; j <= L; j++)
        for (int i = 1; i <= K; i++)
        {
            pixel = m_obraz_w_pamieci.GetPixel(i - 1, j - 1);
            l_hls = (Math.Max(Math.Max(pixel.R, pixel.G), pixel.B) + Math.Min(Math.Min(pixel.R, pixel.G), pixel.B)) / 2;
            pixel = System.Drawing.Color.FromArgb((int)l_hls, (int)l_hls, (int)l_hls);
            suma += l_hls;
            m_ekran.SetPixel(i - 1, j - 1, pixel);
        }
    jasnoc = suma / (L * K);
    jasnoc_value.Text = Math.Round(jasnoc / 255 * 100, 0) + "%";
    suma = 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);
            l_hls = (Math.Max(Math.Max(pixel.R, pixel.G), pixel.B) + Math.Min(Math.Min(pixel.R, pixel.G), pixel.B)) / 2;
            suma += (l_hls - jasnoc) * (l_hls - jasnoc);
        }
    kontrast = Math.Sqrt(suma / (L * K));
    kontrast_value.Text = Math.Round(kontrast / (255 / 2) * 100, 0) + "%";
    SetBitmap(ref m_ekran);
}
```

Efekt

Mateusz Jasiński WCY20U1S1 03.11.2022

Obraz w pamięci L x K = 300 x 300

LOAD

Obraz na ekranie

RESET

Efekty

- ☒ efekt przewijania poziomego w lewo
- ☐ efekt zasłaniania pionowego w górę
- ☐ efekt przesuwania przekątnego w dół-prawo

Wykonywanie:

samoczynne

START STOP

po jednej klatce

Kolejna klatka

klatka = 0

Zmień obraz

RGB na HLS

RGB na HSV(B)

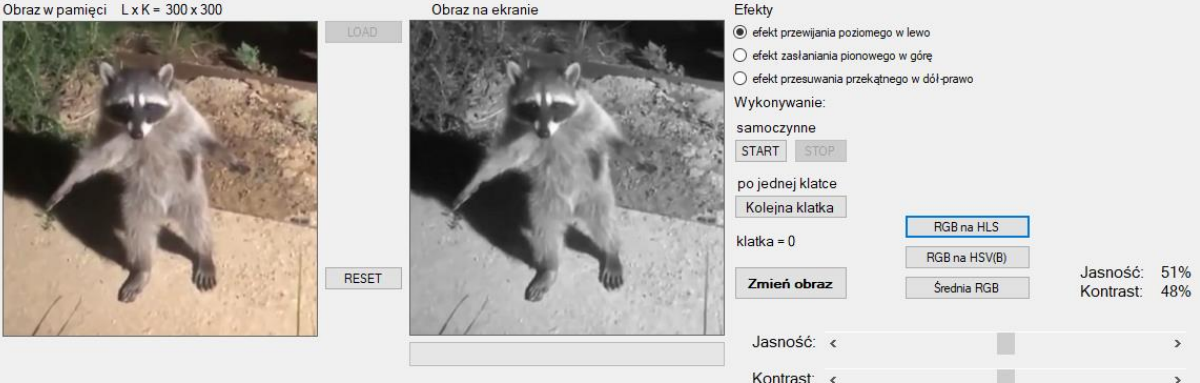
Średnia RGB

Jasność: 51%

Kontrast: 48%

Jasność: < >

Kontrast: < >



Przejście z modelu RGB na HSV

Zmiana RGB na HSV polega na wybraniu największej wartości ze składowych RGB, a następnie ustawienie jej w pozostałych składowych.

Kod programu

```
public void RGB_NA_HSV()
{
    System.Drawing.Color pixel;
    double suma = 0, jasnoc, kontrast, l_hsv;
    for (int j = 1; j <= L; j++)
        for (int i = 1; i <= K; i++)
        {
            pixel = m_obraz_w_pamieci.GetPixel(i - 1, j - 1);
            l_hsv = Math.Max(Math.Max(pixel.R, pixel.G), pixel.B);
            pixel = System.Drawing.Color.FromArgb((int)l_hsv, (int)l_hsv, (int)l_hsv);
            suma += l_hsv;
            m_ekran.SetPixel(i - 1, j - 1, pixel);
        }
    jasnoc = suma / (L * K);
    jasnoc_value.Text = Math.Round(jasnoc / 255 * 100, 0) + "%";
    suma = 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);
            l_hsv = Math.Max(Math.Max(pixel.R, pixel.G), pixel.B);
            suma += (l_hsv - jasnoc) * (l_hsv - jasnoc);
        }
    kontrast = Math.Sqrt(suma / (L * K));
    kontrast_value.Text = Math.Round(kontrast / (255 / 2) * 100, 0) + "%";
    SetBitmap(ref m_ekran);
}
```

Efekt

Mateusz Jasiński WCY20U1S1 03.11.2022

Obraz w pamięci L x K = 300 x 300

Obraz na ekranie

LOAD

RESET

Efekty

- ☒ efekt przewijania poziomego w lewo
- ☐ efekt zasłaniania pionowego w górę
- ☐ efekt przesuwania przekątnego w dół-prawo

Wykonywanie:

samoczynne

START STOP

po jednej klatce

Kolejna klatka

klatka = 0

Zmień obraz

RGB na HLS

RGB na HSV(B)

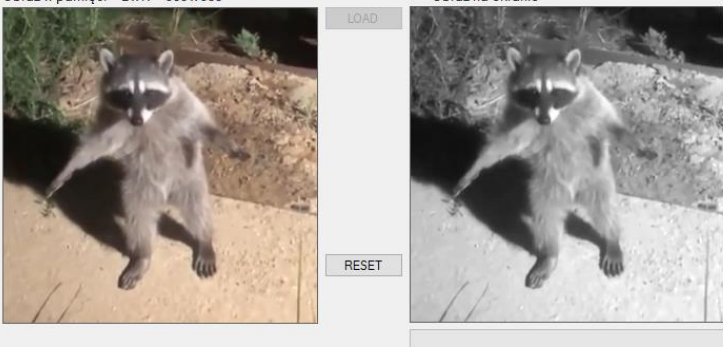
Średnia RGB

Jasność: 58%

Kontrast: 54%

Jasność: < >

Kontrast: < >



Średnia arytmetyczna składowych RGB

Metoda średniej arytmetycznej składowych RGB polega na zastąpieniu składowych RGB średnią arytmetyczną ich wartości.

Kod programu

```
public void AVG_RGB()
{
    System.Drawing.Color pixel;
    double suma = 0, jasnoc, kontrast, avg;
    for (int j = 1; j <= L; j++)
        for (int i = 1; i <= K; i++)
        {
            pixel = m_obraz_w_pamieci.GetPixel(i - 1, j - 1);
            avg = (pixel.R + pixel.G + pixel.B) / 3;
            pixel = System.Drawing.Color.FromArgb((int)avg, (int)avg, (int)avg);
            suma += avg;
            m_ekran.SetPixel(i - 1, j - 1, pixel);
        }
    jasnoc = suma / (L * K);
    jasnoc_value.Text = Math.Round(jasnoc / 255 * 100, 0) + "%";
    suma = 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);
            avg = (pixel.R + pixel.G + pixel.B) / 3;
            suma += (avg - jasnoc) * (avg - jasnoc);
        }
    kontrast = Math.Sqrt(suma / (L * K));
    kontrast_value.Text = Math.Round(kontrast / (255 / 2) * 100, 0) + "%";
    SetBitmap(ref m_ekran);
}
```

Efekt

Mateusz Jasiński WCY20IJ1S1 03.11.2022

Obraz w pamięci L x K = 300 x 300

Obraz na ekranie

LOAD

RESET

Efekty

- ☒ efekt przewijania poziomego w lewo
- ☐ efekt zasłaniania pionowego w górę
- ☐ efekt przesuwania przekątnego w dół-prawo

Wykonywanie:

samoczynne

START STOP

po jednej klatce

Kolejna klatka

klatka = 0

Zmień obraz

RGB na HLS

RGB na HSV(B)

Średnia RGB

Jasność: 51%

Kontrast: 48%

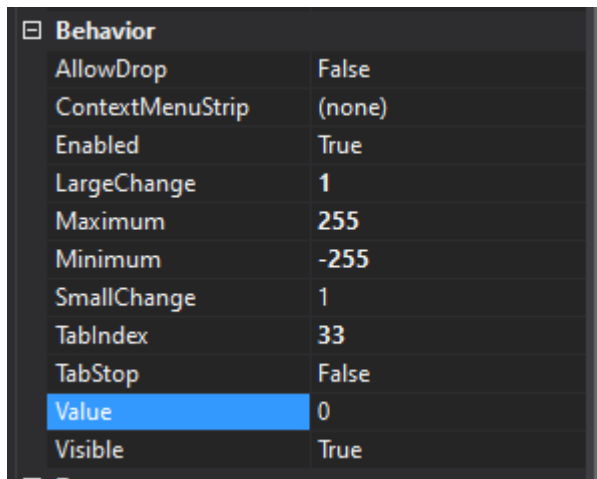
Jasność: < >

Kontrast: < >

Zmiana jasności

Zmiana jasności piksela polega na zwiększeniu lub zmniejszeniu wartości trzech składowych RGB.

W okienku graficznym programu dodajemy z narzędzia Toolbox element „hScrollBar” i ustawiamy jego minimalną wartość na -255 a maksymalną na 255.



W funkcji obsługującej pasek przewijania przechodzimy po każdym pikselu obrazu i dodajemy do jego składowych wartość obecnie ustawioną na pasku przewijania. Dodajemy również sprawdzenie czy wartości składowych nie są mniejsze 0 lub większe 255, jeżeli tak, to ustawiamy daną składową na przekroczoną skrajną wartość. Do funkcji również dodajemy wyliczenie jasności oraz kontrastu identyczną jak w poprzednich zadaniach.

Kod programu

```
public void Jasnosc_change()
{
    System.Drawing.Color pixel;
    int scroll_val = hScrollBar1.Value, r, g, b;
    double suma = 0, jasnosc, kontrast, avg;
    for (int j = 1; j <= L; j++)
        for (int i = 1; i <= K; i++)
        {
            pixel = m_obraz_w_pamieci.GetPixel(i - 1, j - 1);
            r = pixel.R + scroll_val;
            if (r > 255) r = 255;
            else if (r < 0) r = 0;
            g = pixel.G + scroll_val;
            if (g > 255) g = 255;
            else if (g < 0) g = 0;
            b = pixel.B + scroll_val;
            if (b > 255) b = 255;
            else if (b < 0) b = 0;
            pixel = System.Drawing.Color.FromArgb(r, g, b);
            avg = (r + g + b) / 3;
            suma += avg;
            m_ekran.SetPixel(i - 1, j - 1, pixel);
        }
}
```



```


    }
    jasnoc = suma / (L * K);
    jasnoc_value.Text = Math.Round(jasnoc / 255 * 100, 0) + "%";
    suma = 0;
    for (int j = 1; j <= L; j++)
        for (int i = 1; i <= K; i++)
        {
            pixel = m_ekran.GetPixel(i - 1, j - 1);
            avg = (pixel.R + pixel.G + pixel.B) / 3;
            suma += (avg - jasnoc) * (avg - jasnoc);
        }
    kontrast = Math.Sqrt(suma / (L * K));
    kontrast_value.Text = Math.Round(kontrast / (255 / 2) * 100, 0) + "%";
    SetBitMap(ref m_ekran);
}

```

Efekt

Mateusz Jasiński WCY20IJ1S1 03.11.2022


Obraz w pamięci L x K = 300 x 300



LOAD

RESET

Obraz na ekranie



Efekty

☒ efekt przewijania poziomego w lewo

☐ efekt zasłaniania pionowego w górę

☐ efekt przesuwania przekątnego w dół-prawo

Wykonywanie:

samoczynne

START STOP

po jednej klatce

Kolejna klatka

klatka = 0

Zmień obraz

RGB na HLS

RGB na HSV(B)

Średnia RGB

Jasność: 8%


Kontrast: 19%

Jasność: < >

Kontrast: < >

Mateusz Jasiński WCY20IJ1S1 03.11.2022


Obraz w pamięci L x K = 300 x 300



LOAD

RESET

Obraz na ekranie



Efekty

☒ efekt przewijania poziomego w lewo

☐ efekt zasłaniania pionowego w górę

☐ efekt przesuwania przekątnego w dół-prawo

Wykonywanie:

samoczynne

START STOP

po jednej klatce

Kolejna klatka

klatka = 0

Zmień obraz

RGB na HLS

RGB na HSV(B)

Średnia RGB

Jasność: 92%

Kontrast: 26%

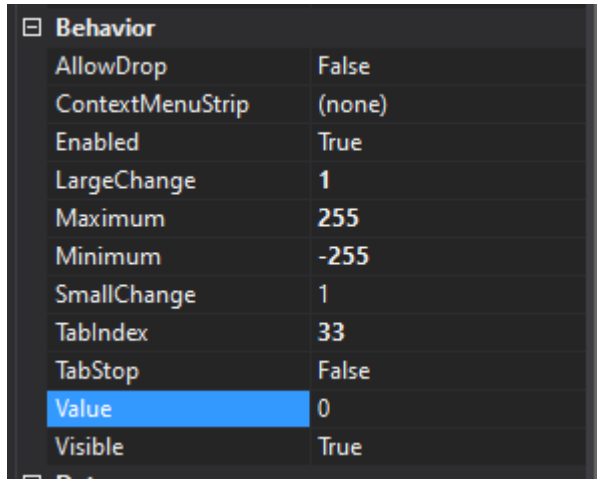
Jasność: < >

Kontrast: < >

Zmiana kontrastu

Zmiana kontrastu polega na zmianie różnicy jasności między pikselami.

W okienku graficznym programu dodajemy z narzędzia Toolbox element „hScrollBar” i ustawiamy jego minimalną wartość na -255 a maksymalną na 255.



Zmiana kontrastu sprowadza się do wykonania przekształcenia obrazu przy pomocy odpowiednio przygotowanej tablicy LUT. Współczynniki w takiej tablicy wyznaczone są według następującego wzoru:

$$LUT(i) = \begin{cases} 0 & \text{jeżeli } a \left(i - \frac{i_{\max}}{2} \right) + \frac{i_{\max}}{2} < 0 \\ a \left(i - \frac{i_{\max}}{2} \right) + \frac{i_{\max}}{2} & \text{jeżeli } 0 \leq a \left(i - \frac{i_{\max}}{2} \right) + \frac{i_{\max}}{2} \leq i_{\max} \\ i_{\max} & \text{jeżeli } a \left(i - \frac{i_{\max}}{2} \right) + \frac{i_{\max}}{2} > i_{\max} \end{cases}$$

Gdzie i_{\max} oznacza maksymalną dopuszczalną wartość składowej RGB piksela obrazu. Jeżeli wartość stałej a , czyli współczynnika kierunkowego prostej, jest większa od 1, to nastąpi zwiększenie kontrastu obrazu, a w przeciwnym wypadku zmniejszenie.

Kod programu

```
public void Kontrast_change()
{
    System.Drawing.Color pixel;
    int scroll_val = hScrollBar2.Value, r, g, b;
    double suma = 0, jasnoc, kontrast, avg, kontrast_new;
    if (scroll_val <= 0)
        kontrast_new = 1.0 + (scroll_val / 255.0);
    else
        kontrast_new = 255.0 / Math.Pow(2, Math.Log(256 - scroll_val, 2));
    for (int j = 1; j <= L; j++)
        for (int i = 1; i <= K; i++)
        {
            pixel = m_obraz_w_pamieci.GetPixel(i - 1, j - 1);
            r = (int)((pixel.R - 255 / 2) * kontrast_new) + 255 / 2;
            if (r > 255) r = 255;
            else if (r < 0) r = 0;
            g = (int)((pixel.G - 255 / 2) * kontrast_new) + 255 / 2;
            if (g > 255) g = 255;
            else if (g < 0) g = 0;
            b = (int)((pixel.B - 255 / 2) * kontrast_new) + 255 / 2;
            if (b > 255) b = 255;
            else if (b < 0) b = 0;
            pixel = System.Drawing.Color.FromArgb(r, g, b);
            m_ekran.SetPixel(i - 1, j - 1, pixel);
        }
    for (int j = 1; j <= L; j++)
        for (int i = 1; i <= K; i++)
        {
            pixel = m_ekran.GetPixel(i - 1, j - 1);
            avg = (pixel.R + pixel.G + pixel.B) / 3;
            suma += avg;
        }
    jasnoc = suma / (L * K);
    jasnoc_value.Text = Math.Round(jasnoc / 255 * 100, 0) + "%";
    suma = 0;
    for (int j = 1; j <= L; j++)
        for (int i = 1; i <= K; i++)
        {
            pixel = m_ekran.GetPixel(i - 1, j - 1);
            avg = (pixel.R + pixel.G + pixel.B) / 3;
            suma += (avg - jasnoc) * (avg - jasnoc);
        }
    kontrast = Math.Sqrt(suma / (L * K));
    kontrast_value.Text = Math.Round(kontrast / (255 / 2) * 100, 0) + "%";
    SetBitmap(ref m_ekran);
}
```

Efekt

Mateusz Jasiński WCY20IJ1S1 03.11.2022

Obraz w pamięci L x K = 300 x 300



LOAD

RESET

Obraz na ekranie



Efekty

- ☒ efekt przewijania poziomego w lewo
- ☐ efekt zasłaniania pionowego w górę
- ☐ efekt przesuwania przekątnego w dół-prawo

Wykonywanie:

samoczynne

START

STOP

po jednej klatce

Kolejna klatka

klatka = 0

Zmień obraz

RGB na HLS

RGB na HSV(B)

Średnia RGB

Jasność: 57%
Kontrast: 81%

Jasność: < >

Kontrast: < >

Mateusz Jasiński WCY20IJ1S1 03.11.2022

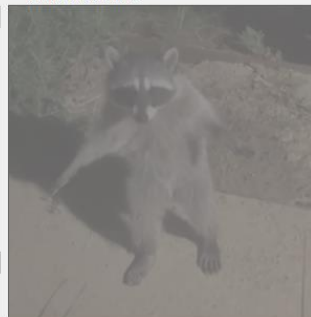
Obraz w pamięci L x K = 300 x 300



LOAD

RESET

Obraz na ekranie



Efekty

- ☒ efekt przewijania poziomego w lewo
- ☐ efekt zasłaniania pionowego w górę
- ☐ efekt przesuwania przekątnego w dół-prawo

Wykonywanie:

samoczynne

START

STOP

po jednej klatce

Kolejna klatka

klatka = 0

Zmień obraz

RGB na HLS

RGB na HSV(B)

Średnia RGB

Jasność: 50%
Kontrast: 10%

Jasność: < >

Kontrast: < >

4. Podsumowanie

Wszystkie postawione przede mną zadania zostały pomyślnie zrealizowane.

W czasie wykonywania poleceń z pierwszego laboratorium należało zwrócić szczególną uwagę na liczbę generowanych pikseli w każdej klatce animacji. W przypadku rastrowego generowania grafiki każda klatka bowiem składa się z takiej samej liczby pikseli ($L \times K$).

Najmniejsze odchylenie od tej reguły od razu powoduje błędną generację całego obrazu i powstawanie zupełnie niespodziewanych wyników działania. Należy jednak zauważyć, że sterowanie generatorem adresu odczytu grafiki źródłowej pozwala na uzyskanie wielu bardzo ciekawych efektów (m.in. przewijanie, przesuwanie, zastanianie, obracanie obrazu).

Przy poleceniach z drugiego laboratorium przydatna była wiedza o różnych modelach opisu przestrzeni barw. Najprzydatniejszy był model RGB, który był głównie używany i jest bardzo prosty do zrozumienia. Model ten składa się z trzech składowych reprezentujących trzy kolory: czerwony, zielony i niebieski. Dzięki manipulacji wartościami składowych można uzyskać 2^{24} kolorów w tym 2^8 odcieni szarości. Różne odcienie szarości powstają, gdy wszystkie składowe mają tę samą wartość, im jest ona większa tym odcień będzie jaśniejszy, a czym mniejsza tym ciemniejszy. Zmiana jasności obrazu działa tak samo jak przy szarości, trzeba zwiększać lub zmniejszać wartości trzech składowych w pikselu. Problemem jest to, że każda składowa może mieć inną wartość, dlatego należało pamiętać o sprawdzaniu i ewentualnej korekcji, gdy jedna z nich przekraczała zakres $[0, 255]$. Zmiana kontrastu polega na zmianie różnic w jasności między pikselami. Do edycji kontrastu obrazu potrzebne było użycie krzywej tonalnej, którą można uzyskać przy pomocy tablicy LUT.

Dzięki wiedzy oraz umiejętnościom projektowania i implementacji algorytmów przerobionych na zajęciach, można wytwarzać np. animowane elementy stron WWW czy też wzbogacać funkcjonalność programów do edycji grafiki rastrowej.