

POLITECHNIKA WROCŁAWSKA
KATEDRA INFORMATYKI TECHNICZNEJ

URZĄDZENIA PERYFERYJNE

Kamera USB

Magdalena Biernat

Michał Bojzan

Prowadzący
dr inż. Jan Nikodem

12 listopada 2017

1 Wprowadzenie

Sprawozdanie dotyczy drugich zajęć. Na tych laboratoriach mieliśmy napisać aplikację, która za pomocą kamery USB będzie wykonywać zdjęcia oraz nakładać je na siebie.

2 Cel laboratorium

- Poznanie działania kamery USB.
- Umiejętność wykonania zdjęcia i przerobienia go

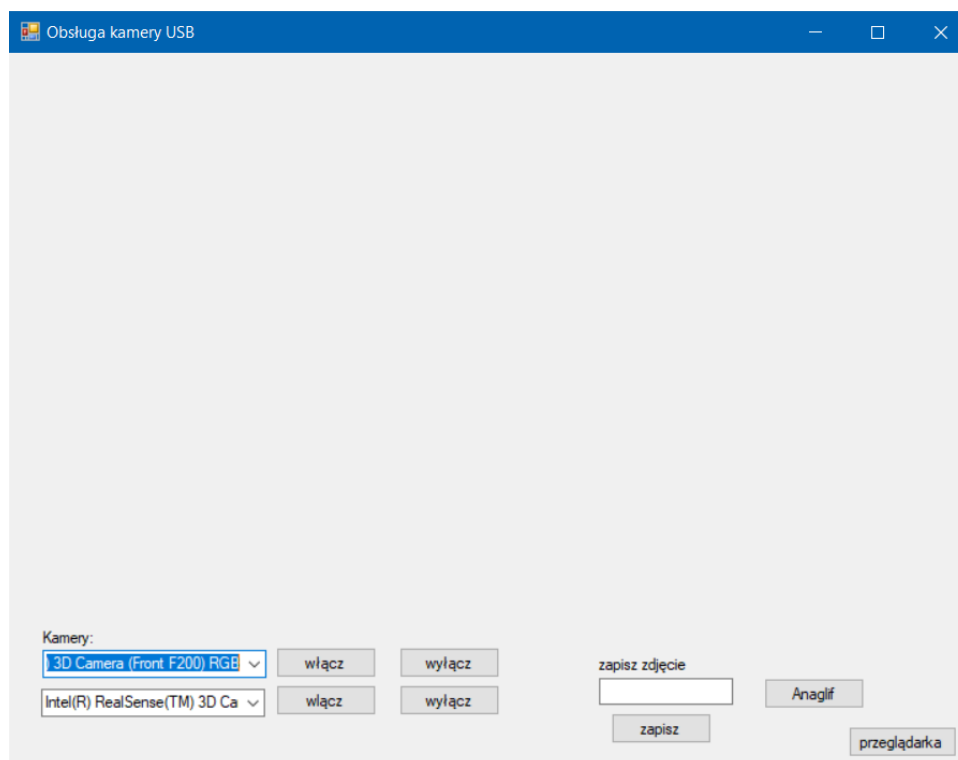
3 Laboratorium

3.1 Kamera USB

Kamera cyfrowa jest podłączana bezpośrednio do komputera, najczęściej za pomocą portu USB. Kamera ta może transmitować obrazy statyczne (co jakiś czas je odświeżając) lub w sposób ciągły. Kamera ta robi zdjęcia kiepskiej jakości, więc nadaje się głównie do komunikatorów internetowych. Szybkość działania też nie jest najlepsza.

3.2 Program

3.2.1 Obsługa komponentów aplikacji



Rysunek 1: Stworzony diagram przypadków użycia

```
//utworzenie dwóch pictureBoxów do wyświetlania zdjęć z dwóch kamer
```

```
private void cam_NewFrame(object sender, NewFrameEventArgs eventArgs)
{
    Bitmap pictureBoxBitmap = (Bitmap)eventArgs.Frame.Clone();
    Bitmap movieFileBitmap = (Bitmap)eventArgs.Frame.Clone();
    obrazekLewy.Image = pictureBoxBitmap;
}
```

```
private void cam_NewFrame2(object sender, NewFrameEventArgs eventArgs)
{
    Bitmap pictureBoxBitmap2 = (Bitmap)eventArgs.Frame.Clone();
    Bitmap movieFileBitmap = (Bitmap)eventArgs.Frame.Clone();
    obrazekPrawy.Image = pictureBoxBitmap2;
}
```

```
//wczytanie podłączonych do komputera kamer dla obu textBoxów
```

```
private void Form1_Load(object sender, EventArgs e)
{
    usbCamera = new FilterInfoCollection(FilterCategory.VideoInputDevice);
    usbCamera2 = new FilterInfoCollection(FilterCategory.VideoInputDevice);
    foreach (FilterInfo VideoCaptureDevice in usbCamera)
    {
        listaKamerGórna.Items.Add(VideoCaptureDevice.Name);
    }
    foreach (FilterInfo VideoCaptureDevice in usbCamera2)
    {
        listaKamerDolna.Items.Add(VideoCaptureDevice.Name);
    }
    listaKamerGórna.SelectedIndex = 1;
    listaKamerDolna.SelectedIndex = 2;
    camera1 = new VideoCaptureDevice();
    camera2 = new VideoCaptureDevice();
}
```

```
//obsługa włączania kamer
```

```
private void włączGórne_Click(object sender, EventArgs e)
{
    if (camera2.IsRunning)
        camera2.Stop();
    camera2 = new VideoCaptureDevice(usbCamera2[listaKamerDolna.SelectedIndex].MonikerString);
    camera2.NewFrame += new NewFrameEventHandler(cam_NewFrame2);
    camera2.Start();
    writer2 = new VideoFileWriter();
    writer2.Open(@"C:\Users\magda\Desktop\Madzia\PWr\UP\cw12\film1.avi", 1280, 720, 30, VideoCodec.MPEG4);
}
```

```

}

private void włączDolne_Click(object sender, EventArgs e)
{
    if (camera1.IsRunning)
        camera1.Stop();
    camera1 = new VideoCaptureDevice(usbCamera[listaKamerGórna.SelectedIndex].MonikerString);
    camera1.NewFrame += new NewFrameEventHandler(cam_NewFrame);
    camera1.Start();
    writer1 = new VideoFileWriter();
    writer1.Open(@"C:\Users\magda\Desktop\Madzia\Pwr\UP\cw12\film.avi", 1280, 720, 30, VideoCodec.MPEG4)
}

//obsługa wyłączenia kamer
private void wyłączGórne_Click(object sender, EventArgs e)
{
    if (camera2.IsRunning)
    {
        camera2.Stop();
        writer2.Close();
        obrazekPrawy.Image = null;
    }
}

private void wyłączDolne_Click(object sender, EventArgs e)
{
    if (camera1.IsRunning)
    {
        camera1.Stop();
        writer1.Close();
        obrazekLewy.Image = null;
    }
}

//obsługa tworzenia anaglif
private void buttonAnaglif_Click(object sender, EventArgs e)
{
    Image im = new Bitmap(obrazekLewy.Image);
    Image im2 = new Bitmap(obrazekPrawy.Image);
    buttonZapisz_Click(sender, e);
    obraz = new Transtp1((Bitmap)im);
    obraz2 = new Transtp2((Bitmap)im2);

    obraz3 = new Transtp3(obraz.Transformacja(), obraz2.Transformacja1());
    string sciezka = @"C:\Users\lab\Videos\" + nazwaDoZapisuZdjecia.Text + ".jpg";
}

```

```

obraz3.Transformacja().Save(sciezka, ImageFormat.Jpeg);
}

//obsługa zapisu obrazka
private void buttonZapisz_Click(object sender, EventArgs e)
{
    if (camera1.IsRunning && camera2.IsRunning & nazwaDoZapisuZdjecia.Text != "")
    {
        if (camera1.IsRunning)
        {
            string sciezka = @"C:\Users\lab\Videos\" + nazwaDoZapisuZdjecia.Text + ".jpg";
            obrazekLewy.Image.Save(sciezka, ImageFormat.Jpeg);

        }
        if (camera2.IsRunning)
        {
            string sciezka = @"C:\Users\lab\Videos\" + nazwaDoZapisuZdjecia.Text + "1.jpg";
            obrazekPrawy.Image.Save(sciezka, ImageFormat.Jpeg);
        }
    }
}

```

3.2.2 Obsługa tworzenia anaglifu

```

//zmiana obrazka na czerwony
public class Transtp1
{
    private Bitmap obrazek;
    private Bitmap obrazekKopia;

    public Transtp1(Bitmap img)
    {
        this.obrazekKopia = this.obrazek = img;
    }

    public Bitmap Transformacja()
    {
        if (obrazekKopia.PixelFormat != PixelFormat.Format8bppIndexed && obrazekKopia.PixelFormat != PixelFo
        {
            Bitmap bmp = new Bitmap(obrazekKopia.Width, obrazekKopia.Height, PixelFormat.Format24bppRgb);
            Graphics g = Graphics.FromImage(bmp);
            g.DrawImage(obrazekKopia, 0, 0, obrazekKopia.Width, obrazekKopia.Height);
            g.Dispose();
            obrazekKopia = bmp;
            obrazek = bmp;
        }
    }
}

```

```

    }
    PixelFormat formatObrazka = (obrazek.PixelFormat == PixelFormat.Format8bppIndexed) ? PixelFormat.Format8bppIndexed : PixelFormat.Format24bppRgb;
    BitmapData daneWyjsciowe = obrazekKopia.LockBits(new Rectangle(0, 0, obrazek.Width, obrazek.Height), ImageLockMode.ReadOnly, formatObrazka);

    BitmapData daneWejsciowe = obrazek.LockBits(new Rectangle(0, 0, obrazek.Width, obrazek.Height), ImageLockMode.ReadOnly, formatObrazka);

    unsafe
    {
        byte* wskWyjsciowy = (byte*)daneWyjsciowe.Scan0;
        byte* wskWejsciowy = (byte*)daneWejsciowe.Scan0;

        int nOffset = daneWejsciowe.Stride - obrazek.Width * 3;
        for (int y = 0; y < obrazek.Height; y++)
        {
            for (int x = 0; x < obrazek.Width * 3; x++)
            {
                wskWyjsciowy[0] = (byte)(0.299 * wskWejsciowy[0] + 0.587 * wskWejsciowy[1] + 0.114 * wskWejsciowy[2]);
                wskWyjsciowy[1] = 0;
                wskWyjsciowy[2] = 0;

                wskWejsciowy += 3; wskWyjsciowy += 3;
            }
            wskWejsciowy += nOffset; wskWyjsciowy += nOffset;
        }
        wskWejsciowy += nOffset; wskWyjsciowy += nOffset;
    }
    obrazek.UnlockBits(daneWejsciowe);
    obrazekKopia.UnlockBits(daneWyjsciowe);

    return obrazekKopia;
}

//zmiana obrazka na niebieski
public class Transtp2
{
    private Bitmap obrazek;
    private Bitmap obrazekKopia;

    public Transtp2(Bitmap img)
    {
        this.obrazekKopia = this.obrazek = img;
    }

    public Bitmap Transformacja1()
    {
        if (obrazekKopia.PixelFormat != PixelFormat.Format8bppIndexed && obrazekKopia.PixelFormat != PixelFormat.Format24bppRgb)
    
```

```

{
Bitmap bmp = new Bitmap(obrazekKopia.Width, obrazekKopia.Height, PixelFormat.Format24bppRgb);
Graphics g = Graphics.FromImage(bmp);
g.DrawImage(obrazekKopia, 0, 0, obrazekKopia.Width, obrazekKopia.Height);
g.Dispose();
obrazekKopia = bmp;
obrazek = bmp;
}

PixelFormat formatObrazka = (obrazek.PixelFormat == PixelFormat.Format8bppIndexed) ? PixelFormat.Format8bppIndexed : PixelFormat.Format24bppRgb;
BitmapData daneWyjsciowe = obrazekKopia.LockBits(new Rectangle(0, 0, obrazek.Width, obrazek.Height), ImageLockMode.ReadOnly, formatObrazka);

BitmapData daneWejsciowe = obrazek.LockBits(new Rectangle(0, 0, obrazek.Width, obrazek.Height), ImageLockMode.ReadOnly, formatObrazka);

unsafe
{
byte* wskWyjsciowy = (byte*)daneWyjsciowe.Scan0;
byte* wskWejsciowy = (byte*)daneWejsciowe.Scan0;

int nOffset = daneWejsciowe.Stride - obrazek.Width * 3;
for (int y = 0; y < obrazek.Height; y++)
{
for (int x = 0; x < obrazek.Width * 3; x++)
{
wskWyjsciowy[0] = 0;
wskWyjsciowy[1] = (byte)(0.299 * wskWejsciowy[0] + 0.587 * wskWejsciowy[1] + 0.114 * wskWejsciowy[2]);
wskWyjsciowy[2] = (byte)(0.299 * wskWejsciowy[0] + 0.587 * wskWejsciowy[1] + 0.114 * wskWejsciowy[2]);

wskWejsciowy += 3; wskWyjsciowy += 3;
}
wskWejsciowy += nOffset; wskWyjsciowy += nOffset;
}
wskWejsciowy += nOffset; wskWyjsciowy += nOffset;
}

obrazek.UnlockBits(daneWejsciowe);
obrazekKopia.UnlockBits(daneWyjsciowe);

return obrazekKopia;
}

//połączenie obrazka czerwonego i niebieskiego
public class Transtp3
{
private Bitmap obrazek;
private Bitmap obrazekKopia;
}

```

```

}
public class Transtp3
{
private Bitmap obrazek;
private Bitmap obrazekKopia;
private Bitmap obraz3D;

public Transtp3(Bitmap img, Bitmap img1)
{
this.obrazekKopia = img;
this.obrazek = img1;
}
public Bitmap Transformacja()
{
if (obrazekKopia.PixelFormat != PixelFormat.Format8bppIndexed && obrazekKopia.PixelFormat != PixelFo
{
Bitmap bmp = new Bitmap(obrazekKopia.Width, obrazekKopia.Height, PixelFormat.Format24bppRgb);
Graphics g = Graphics.FromImage(bmp);
g.DrawImage(obrazekKopia, 0, 0, obrazekKopia.Width, obrazekKopia.Height);
g.Dispose();
obrazekKopia = bmp;
Bitmap bmp1 = new Bitmap(obrazek.Width, obrazek.Height, PixelFormat.Format24bppRgb);
Graphics g1 = Graphics.FromImage(bmp);
g1.DrawImage(obrazek, 0, 0, obrazek.Width, obrazek.Height);
g1.Dispose();
obrazek = bmp1;
obraz3D = bmp1;
}
PixelFormat formatObrazka = (obrazek.PixelFormat == PixelFormat.Format8bppIndexed) ? PixelFormat.For
PixelFormat formatObrazka1 = (obrazekKopia.PixelFormat == PixelFormat.Format8bppIndexed) ? PixelForm
BitmapData daneWyjsciowe = obrazekKopia.LockBits(new Rectangle(0, 0, obrazek.Width, obrazek.Height),
BitmapData daneWyjsciowe1 = obrazek.LockBits(new Rectangle(0, 0, obrazek.Width, obrazek.Height), Image
BitmapData daneWejsciowe = obrazek.LockBits(new Rectangle(0, 0, obrazek.Width, obrazek.Height), Image
BitmapData daneWejsciowe1 = obrazekKopia.LockBits(new Rectangle(0, 0, obrazek.Width, obrazek.Height)
unsafe
{
byte* wskWyjsciowy = (byte*)daneWyjsciowe.Scan0;
byte* wskWejsciowy = (byte*)daneWejsciowe.Scan0;
byte* wskWyjsciowy1 = (byte*)daneWyjsciowe1.Scan0;
byte* wskWejsciowy1 = (byte*)daneWejsciowe1.Scan0;

int nOffset = daneWejsciowe.Stride - obrazek.Width * 3;
for (int y = 0; y < obrazek.Height; y++)
{
for (int x = 0; x < obrazek.Width * 3; x++)

```



```

{
wskWyjsciowy[0] = (byte)(0.299 * wskWyjsciowy[0] + 0.587 * wskWyjsciowy[1] + 0.114 * wskWyjsciowy[2]
wskWyjsciowy[1] = (byte)(0.299 * wskWyjsciowy1[0] + 0.587 * wskWyjsciowy1[1] + 0.114 * wskWyjsciowy1
wskWyjsciowy[2] = (byte)(0.299 * wskWyjsciowy1[0] + 0.587 * wskWyjsciowy1[1] + 0.114 * wskWyjsciowy1

wskWejsciowy += 3; wskWyjsciowy += 3;
}
wskWejsciowy += nOffset; wskWyjsciowy += nOffset;
}
wskWejsciowy += nOffset; wskWyjsciowy += nOffset;
}
obrazek.UnlockBits(daneWejsciowe);
obrazekKopia.UnlockBits(daneWyjsciowe);

return obrazekKopia;
}
}

```

4 Wnioski

- Obrazek wczytuje się w formie bitmapy
- Aby połączyć zdjęcia, trzeba je przekształcić w bitmapę
- Do stworzenia anaglifa potrzeba dwóch obrazków - czerwonego i niebieskiego
- Niestety z powodu braku dwóch kamer jednocześnie nie byliśmy w stanie sprawdzić, czy anaglif dobrze nam się tworzy