Urządzenia peryferyjne
Skaner płaski

Autor: Julia Krok 272981

Gr. Czw. Np. 13:15 – 16:15

1. Zadania do wykonania

W ramach podstawowej wersji zadania należy napisać aplikację, która będzie umożliwiać:

- Uzyskanie obrazu na ekranie monitora po zeskanowaniu obiektu.
- Zapisanie obrazu do zbioru z możliwością wybrania jednego z kilku podanych formatów.
- Realizację takich opcji jak zmiana parametrów skanowania (rozdzielczość, tryb skanowania), obracanie obrazu o 90, 180, 270 stopni w obie strony, itd.)

Dodatkowo zostały dodana funkcjonalności:

- Wybranie zaznaczonego wycinka obrazu do zeskanowania za pomocą myszy

2. Wstęp teoretyczny

• Budowa skanera

Skaner to urządzenie służące do przekształcania dokumentów fizycznych na format cyfrowy. Wykorzystywane są do tego komponenty:

- **Czujnik Obrazu** (Matryca CCD/CMOS) – element odpowiedzialny za konwersję odbitego światła na sygnał elektryczny. CCD (Charge-Coupled Device) i CMOS (Complementary Metal-Oxide-Semiconductor) to dwa główne typy matryc używanych w skanerach.
- **Układ optyczny** – Zazwyczaj składa się z luster i soczewek. Zadaniem układu optycznego jest skierowanie odbitego światła na matrycę obrazową. Źródłem światła najczęściej są lampy LED, fluorescencyjne lub ksenonowe, które oświetlają dokument.
- **Silnik krokowy** – Umożliwia przesuwanie głowicy skanującej wzdłuż dokumentu, aby umożliwić precyzyjne skanowanie.
- **Elektronika sterująca** – układy odpowiedzialne za przetwarzanie sygnałów, sterowanie ruchem oraz komunikację z komputerem.
- **Elementy zewnętrzne** – obudowa chroniąca komponenty oraz szkło, pod którym umieszcza się dokumenty.

Zasada działania skanera:

- Przygotowanie dokumentu – umieszczanie dokumentu na szybie skanera lub w podajniku (dla skanerów z ADF - Automatic Document Feeder).
- Oświetlenie dokumentu i zarejestrowanie odbitego światła przez matrycę.
- Przetworzenie obrazu na sygnały elektryczne, które następnie są konwertowane na obraz cyfrowy.

Główne technologie skanowania to:

- **Skanowanie płaskie** – najpopularniejszy typ, gdzie dokument kładzie się na płaskiej szybie skanera. Stosowany w domowych i biurowych urządzeniach.
- **Skanowanie bębnowe** – w profesjonalnych zastosowaniach. Dokument jest owinięty wokół bębna, a skaner odczytuje dane za pomocą ruchomej głowicy.

- **Skanowanie przestrzenne** – wykorzystuje laser lub inne technologie do odwzorowania trójwymiarowych obiektów.

W zależności od ustawionych parametrów skanera możemy wpływać na otrzymany skan. Wyróżnia się w szczególności:

- **Rozdzielczość** (DPI - Dots Per Inch) – liczba punktów skanowanych na cal. Wyższa rozdzielczość oznacza lepszą jakość obrazu. Typowe wartości to 300, 600, 1200 DPI.
- **Głębia koloru** – liczba bitów używanych do reprezentacji koloru każdego piksela. Najczęściej 24-bitowa lub 48-bitowa.
- **Szybkość skanowania** – mierzona w tronach na minutę (PPM) lub w sekundach na stronę.
- **Obszar skanowania** – maksymalny rozmiar dokumentu, który może być zeskanowany. Standardowy format to A4.

• Kolory cyfrowe

Kolory cyfrowe są centralnym elementem w procesie przetwarzania obrazu, szczególnie w kontekście skanowania i fotografii. W celu uzyskania dokładnego odwzorowania kolorów, stosuje się różnorodne technologie i techniki, takie jak filtry, balans bieli, rozdzielczość skanowania, de-mozaikowanie i zoom.

- **Filtry** – w układzie Bayera (RGGB) światło padające na każdy piksel jest filtrowane przez filtry: czerwony, zielony i niebieski. Filtrów zielonych jest dwa razy więcej niż pozostałych, ponieważ ludzkie oko jest bardziej czułe na kolor zielony. Istnieją również alternatywne technologie, jak filtry Foveon, które nie korzystają z układu Bayera.
- **Balans bieli** – pozwala na dostosowanie kolorów obrazu do temperatury barwowej światła oświetlającego scenę. Różne źródła światła (światło słoneczne, oświetlenie żarowe, neonowe, itp.) mają różne temperatury barwowe, co wpływa na odcienie kolorów na zdjęciu. Balans bieli koryguje te odchylenia, dzięki czemu białe obiekty w obrazie wyglądają naturalnie.
- **Rozdzielczość skanowania** – rozdzielczość skanu zależy od liczby pikseli na matrycy. Większa rozdzielczość pozwala na uzyskanie ostrzejszych i bardziej szczegółowych obrazów. Jednocześnie większa rozdzielczość wymaga większej mocy obliczeniowej do przetwarzania obrazów oraz większej pojemności pamięci do przechowywania plików.
- **De-mozaikowanie** – proces interpolacji kolorów, który pozwala uzyskać pełny obraz kolorowy z matrycy pokrytej filtrem Bayera. Ponieważ każdy piksel matrycy widzi tylko jeden kolor (czerwony, zielony lub niebieski), procesor oblicza kolor wypadkowy dla każdego piksela na podstawie pikseli sąsiednich, aby zrekonstruować pełny kolor obrazu.
- **Zoom** – Istnieje podział na zoom optyczny i cyfrowy. W przypadku skanerów zoom optyczny oznacza fizyczne przesunięcie układu optycznego, co pozwala na zbliżenie obrazu bez utraty jakości. Zoom cyfrowy polega na powiększaniu obrazu poprzez skalowanie danych cyfrowych. Może prowadzić do utraty jakości, ponieważ polega na interpolacji pikseli.

- ## Biblioteki do obsługi skanerów

Oprogramowanie do obsługi skanerów wykorzystuje różnorodne biblioteki i interfejsy API, które ułatwiają komunikację między sprzętem a aplikacjami. Wśród najpopularniejszych bibliotek i standardów wykorzystywanych do skanowania są:

- **TWAIN** (Technology Without An Interesting Name) – standardowy interfejs API opracowany z myślą o łatwej integracji skanerów z aplikacjami. Jest najczęściej stosowany w systemach Windows i macOS. Umożliwia bezpośrednie skanowanie do aplikacji takich jak edytory graficzne czy pakiety biurowe. Obsługuje podstawowe i zaawansowane funkcje skanowania, takie jak wybór rozdzielczości, formatu pliku czy trybu kolorów.
- **WIA** (Windows Image Acquisition) – natywny interfejs API firmy Microsoft dla systemu Windows, zaprojektowany do obsługi urządzeń obrazujących, takich jak skanery, aparaty cyfrowe i kamery. Umożliwia aplikacjom na systemie Windows bezpośrednie uzyskiwanie obrazów z urządzeń. Obsługuje również zarządzanie urządzeniami i konfigurację parametrów skanowania.
- **SANE** (Scanner Access Now Easy) – otwarta biblioteka API dla systemów operacyjnych Linux i Unix, która umożliwia dostęp do różnych skanerów. Zapewnia szerokie wsparcie dla różnorodnych modeli skanerów. Umożliwia zaawansowaną konfigurację ustawień skanowania i jest często wykorzystywana w połączeniu z oprogramowaniem typu open source.
- **ISIS** (Image and Scanner Interface Specification) – wydajny i skalowalny interfejs API stworzony przez firmę EMC (obecnie OpenText), zaprojektowany dla profesjonalnych zastosowań skanowania. Oferuje zaawansowane funkcje zarządzania przepływem skanowania, takie jak skanowanie dużych ilości dokumentów z wysoką prędkością i precyzją. Obsługuje również szeroką gamę formatów plików i opcji skanowania.
- **ICA** (Image Capture Architecture) – natywny interfejs API dla systemu macOS, służący do obsługi urządzeń obrazujących. Zapewnia prostą integrację skanerów i innych urządzeń obrazujących z aplikacjami w środowisku macOS. Umożliwia podstawowe funkcje skanowania i zarządzania obrazami.

- ## Formaty zapisu informacji graficznej

Podczas zapisu obrazu cyfrowego w pliku istnieje wiele formatów, z których każdy ma swoje unikalne cechy, zalety i zastosowania. Wśród najpopularniejszych formatów graficznych są:

- **JPG** (Joint Photographic Experts Group) – jeden z najczęściej używanych formatów do zapisywania zdjęć i obrazów o dużej liczbie kolorów. Charakteryzuje go kompresja stratna, która zmniejsza rozmiar pliku kosztem utraty pewnej ilości danych i jakości obrazu. Obsługuje 24-bitową głębię koloru (16,7 miliona kolorów).
- **PNG** (Portable Network Graphics) – format zaprojektowany jako bezstratna alternatywa dla GIF, idealny do grafiki z przezroczystością. Cechuje go kompresja bezstratna, co oznacza, że nie traci jakości obrazu przy zapisywaniu i otwieraniu. Obsługuje przezroczystość (kanał alfa) i 24-bitową głębię koloru, przez co pliki te mają większy rozmiar w porównaniu do JPG.
- **TIFF** (Tagged Image File Format) – uniwersalny format graficzny używany w profesjonalnych zastosowaniach, takich jak edycja obrazów i archiwizacja. Obsługuje

zarówno kompresję stratną, jak i bezstratną. Może przechowywać obrazy o bardzo wysokiej rozdzielczości z dużą głębią koloru (do 48 bitów na piksel). Pliki mogą być bardzo duże ze względu na wysoką jakość zapisywanych danych.

- **BMP** (Bitmap) – Prosty format graficzny używany głównie w środowiskach Windows. Nie wykorzystuje kompresji, przez co pliki mają duży rozmiar. Obsługuje szeroki zakres głębi kolorów, w tym monochromatyczne i 24-bitowe obrazy kolorowe. Jego struktura jest prosta, przez co umożliwia bezpośredni dostęp do pikseli.
- **RLE** (Run-Length Encoding) – technika kompresji danych graficznych stosowana głównie w formatach wektorowych i bitmapowych. Stosuje kompresję bezstratną, ale jest mniej efektywna dla obrazów o dużej liczbie szczegółów i różnorodności kolorów.
- **GIF** (Graphics Interchange Format) - Obsługuje animacje i 8-bitową paletę kolorów (256 kolorów). Kompresja bezstratna, ale ograniczona ilość kolorów czyni go mniej odpowiednim dla zdjęć.
- **HEIF** (High Efficiency Image Format) – nowoczesny format zaprojektowany do kompresji obrazów z wyższą efektywnością niż JPG, obsługuje HDR i animacje.
- **RAW** - Format bezstratny używany przez aparaty cyfrowe do zapisu surowych danych z matrycy, co umożliwia zaawansowaną edycję i korekcję.

3. Realizacja ćwiczenia

Realizacja ćwiczenia została rozpoczęta od zaprojektowania GUI aplikacji Windows Forms, do którego później zostaną zaimplementowane wymagane funkcjonalności. Jako ekran wykorzystaliśmy element „PictureBox”.

Rysunek 1 Interfejs do obsługi skanera

- ## Przygotowanie aplikacji

Do obsługi skanera wykorzystaliśmy bibliotekę WIA. Wartości dla tej biblioteki standardowo podawane są w calach dlatego zastosowane zostały stosowne przeliczniki:

```
const double millimetersToInches = 25.4, inchesToMillimeters = 1/25.4;
static double a4_height = (297 / millimetersToInches);
static double a4_width = (210 / millimetersToInches);
```

- ## Wybór skanera

Wybór skanera odbywał się automatycznie po przyciśnięciu przycisku połącz. Domyślnie wybierany był pierwszy skaner z listy dostępnych urządzeń.

```
private void polacz_Click(object sender, EventArgs e)
{
      if(deviceManager.DeviceInfos.Count > 0)
      {
            device = deviceManager.DeviceInfos[1].Connect();
            connected = true;
            MessageBox.Show("Polaczono ze skanerem!", "",
MessageBoxButtons.OK);
      }
      else
      {
            MessageBox.Show("Nie wykryto zadnych podlaczonych urzadzen!",
"", MessageBoxButtons.OK);
      }
}
```

- ## Skanowanie obrazu

Skanowanie obrazu odbywało po sprawdzeniu, czy skaner jest podłączony (connected). Po pomyślnym sprawdzeniu połączenia, kod uzyskuje element skanujący z urządzenia *item*, który będzie używany do ustawiania parametrów i wykonania skanowania. Domyślne parametry skanowania ustawione są przez odpowiednie ustawienie *item.setProperties()*. Odpowiednie wartości ustawiane są pod odpowiednimi wartościami *item.Properties.get_Item(wartość)*, gdzie wykorzystywane są:

- **Tryb skanowania** (6146): Określa tryb, w jakim będzie skanowany obraz (np. kolorowy, czarno-biały).
- **Rozdzielczość DPI** (6147, 6148): Ustawia rozdzielczość skanowania w poziomie i pionie.
- **Początkowe współrzędne skanowania** (6149, 6150): Określają początkowe współrzędne (X, Y) skanowania. Wartości te są obliczane na podstawie wejściowych wartości z pól tekstowych, a stała 46.75 została dobrana eksperymentalnie.
- **Szerokość i wysokość skanowania** (6151, 6152): Wymiary obszaru skanowania, również obliczane na podstawie wartości wejściowych i rozdzielczości DPI.

Otrzymany obraz rysowany jest na elemencie *pictureBox1*. Następnie obraz jest skanowany i transferowany do obiektu ImageFile w zadanym formacie *(format*). Ostatecznie obraz zapisujemy do pliku.

```
 private void skanuj_Click(object sender, EventArgs e)
 {
       if (connected)
       {
              Item item = device.Items[1];
              // ustawienie parametrów skanowania
              item.Properties.get_Item("6146").set_Value(trybSkanowania);
              // tryb skanera
              item.Properties.get_Item("6147").set_Value(dpi);
              // dpi poziome
              item.Properties.get_Item("6148").set_Value(dpi);
              // dpi pionowe
       item.Properties.get_Item("6149").set_Value(Double.Parse(textBoxX.Text)
* dpi / 46.75);
       // x_poczatek - pobierane z pola tekstowego
       item.Properties.get_Item("6150").set_Value(Double.Parse(textBoxY.Text)
* dpi / 46.75);
       // y_poczatek - pobierane z pola tekstowego
       item.Properties.get_Item("6151").set_Value(dpi * a4_width *
(Double.Parse(textBoxWidth.Text) / 396.0));
       // szerokosc skanowania - pobierana z pola tekstowego
       item.Properties.get_Item("6152").set_Value(dpi * a4_height *
(Double.Parse(textBoxHeight.Text) / 546.0));
       // wysokosc(dlugosc) skanowania

              pictureBox1.Paint += PictureBox_Paint;
              // dodanie zdarzenia polegającego na narysowaniu prostokata

              pictureBox1.Invalidate(); // odswiezenie picturebox

              // przekazanie wyniku skanowania do pliku
              ImageFile imageFile = (ImageFile)item.Transfer(format);
              pictureBox1.Paint -= PictureBox_Paint; // usuniecie zdarzenia
(wiec rowniez prostokata)
              pictureBox1.Invalidate(); // odswiezenie picturebox
              string path = "PlikSkan" + extension;
              if (File.Exists(path))
              {
                     File.Delete(path);
              }
              imageFile.SaveFile(path);
              // wyswietlenie w pictureBox1 zapisanego pliku
              pictureBox1.ImageLocation = path;
       }
       else
       {
              MessageBox.Show("Brak podlaczonego urzadzenia!", "",
MessageBoxButtons.OK);
       }
 }
```

- ## Zapis w odpowiednim formacie

Aby ustalić format pliku wykorzystaliśmy funkcję *setFormat()*, która ustawia odpowiedne zmienne globalne.

```
private void setFormat(string extension, string format)
{
      this.extension = extension;
      this.format = format;
}
```

Extension jest używane do generowania pełnej ścieżki pliku (path), w której zostanie zapisany zeskanowany obraz. Określa typ pliku wynikowego na podstawie wybranego formatu. *Format* wykorzystywany jest do odpowiedniego przekonwertowania pliku za pomocą funkcji *Transfer(format)*. Dzięki temu system WIA wie, w jakim formacie należy zwrócić wynik skanowania (np. JPG, PNG, TIFF). Format to GUID, czyli unikalny identyfikator przypisany do każdego formatu obrazu obsługiwanego przez Windows Image Acquisition (WIA).

```
setFormat(".jpg", "{B96B3CAE-0728-11D3-9D7B-0000F81EF32E}");
```

- ## Zmiana parametrów: tryby skanowania

Zmiana trybów skanowania wykonana jest za pomocą elementu *radioButton*, który ustawia odpowiednią wartość zmiennej globalnej *trybSkanowania*. Zaimplementowane są trzy tryby skanowania: kolorowy, skala szarości oraz czarno-biały. Odpowiedni tryb skanowania ustawiany jest później przy ustawieniach parametrów skanowania.

```
private void radioButton1_CheckedChanged(object sender, EventArgs e)
{
      trybSkanowania = 1;
      label1.Text = "Tryb skanowania:\nKolorowy";
}

private void radioButton2_CheckedChanged(object sender, EventArgs e)
{
      trybSkanowania = 2;
      label1.Text = "Tryb skanowania:\nSkala Szarosci";";
}

private void radioButton3_CheckedChanged(object sender, EventArgs e)
{
      trybSkanowania = 3;
      label1.Text = "Tryb skanowania:\nCzern i biel";
}
```

- **Zmiana parametrów: rozdzielczość**

Zmiana paramterów rozdzielczości odbywa się za pomocą elementu *trackBar*. Wartość jest odpowiednio mnożona aby uzyskać autentyczne wartości DPI.

```
private void trackBar1_Scroll(object sender, EventArgs e)
{
      dpi = trackBar1.Value * 50;
      label2.Text = "DPI: " + dpi;
}
```

- **Zmiana parametrów: obrót**

Obrót obrazu odbywa się za pomocą funkcji *obrocObraz()*. Wykorzystany jest typ *RotateFlipType,* który jest typem wyliczeniowym (enum) określającym sposób obrotu i ewentualnego odbicia obrazu. Następnie wykonywana jest dekompozycja krotek do zamiany wartości *Width* i *Height* kontrolki *PictureBox*.

```
private void obrocObraz(RotateFlipType rotation)
{
      Image image = pictureBox1.Image;
      image.RotateFlip(rotation);
      // zamiana wartosci atrybutow pictureBox przy uzyciu krotki
      (pictureBox1.Width, pictureBox1.Height) = (pictureBox1.Height,
pictureBox1.Width);
}
```

Wartość *rotation* zmieniana jest po nacięsnięciu odpowiedniego przycisku.

```
private void right90_Click(object sender, EventArgs e)
{
      // obrot o 90 w prawo
      obrocObraz(RotateFlipType.Rotate90FlipNone);
}

private void left90_Click(object sender, EventArgs e)
{
      // obrot o 90 w lewo (270 w prawo)
      obrocObraz(RotateFlipType.Rotate270FlipNone);
}
```

- **Skanowanie wycinka obrazu**

Wybór wycinka obrazu do zeskanowania wykonywany jest przez odpowiednie obsłużenie wydarzeń myszy na elemencie *pictureBox1*. Początkowe współrzędne myszy rejestrowane są w zmiennych *x_start* oraz *y_start* a końcowe współrzędne w zmiennych *x_end* oraz *y_end*. Różnica między współrzędnymi początkowymi a końcowymi pozwala obliczyć szerokość i wysokość wybranego obszaru. Wartości te są następnie wyświetlane w polach tekstowych (textBoxWidth, textBoxHeight), dla celów weryfikacyjnych.

```
private void PictureBox1_MouseUp(object sender, MouseEventArgs e)
{
      x_end = e.X;
      y_end = e.Y;
      width = x_end - x_start;
      height = y_end - y_start;
      textBoxWidth.Text = width.ToString();
      textBoxHeight.Text = height.ToString();
}


private void PictureBox1_MouseDown(object sender, MouseEventArgs e)
{
      x_start = e.X;
      y_start = e.Y;
      textBoxX.Text = x_start.ToString();
      textBoxY.Text = y_start.ToString();
}
```

W wybranym miejscu metoda *PictureBox_Paint* rysuje prostokąt w *PictureBox,* który wizualizuje wybrany przez użytkownika obszar.

```
private void PictureBox_Paint(object sender, PaintEventArgs e)
{
      Rectangle rectToDraw = new Rectangle(Int32.Parse(textBoxX.Text),
Int32.Parse(textBoxY.Text), Int32.Parse(textBoxWidth.Text),
Int32.Parse(textBoxHeight.Text));
      using (Pen pen = new Pen(Color.Black, 2))
      {
            e.Graphics.DrawRectangle(pen, rectToDraw);
      }
}
```

Podczas skanowania wartości zostają odpowiednio zmodyfikowane i przekształcone w celu dopasowania do rozmiarów skanera.

4. Podsumowanie i wnioski

Obsługa skanera na początku nie była dla nas intuicyjna z powodu wykorzystania amerykańskiego standardu miar, opartego na calach, przez co ciężko było oszacować poprawność zmiennych dla wycinka ekranu. Przejrzystość wprowadziło dodanie odpowiedniego przelicznika, przez co mogliśmy operować na wygodnych dla nas milimetrach.

W czasie zajęć nie udało nam się uzyskać odpowiednich proporcji podczas wyboru fragmentu obrazu do skanowania, ale pozostałe funkcjonalności działały poprawnie. Skanowanie wycinka obrazu zostało przez nas poprawione, a poprawiona wersja została zamieszczona w sprawozdaniu.

5. Źródła

- https://learn.microsoft.com/en-us/windows/win32/wia/-wia-startpage
- https://hanzo.com.pl/Skanery-definicja-budowa-parametry
- https://miroslawzelent.pl/informatyka/skanery-ccd-cis-lide-ocr-twain/
- https://robimylogo.pl/formaty-plikow-graficznych
- https://en.wikipedia.org/wiki/TWAIN
- http://www.sane-project.org/intro.html
- https://support.alarisworld.com/-/media/static-picturepark-assets/impublicwebsite/manuals/desktop-scanners/s2000_s2000w-series-scanners_-isis-scanning-setup-guide.pdf
- https://www.apple.com/au/main/device_includes/scanners.html