 POLITECHNIKA WROCŁAWSKA

WYDZIAŁ INFORMATYKI I TELEKOMUNIKACJI

Urządzenia peryferyjne

**Laboratorium 2**

Obsługa kamery USB

Termin zajęć: Środa TP, 17:30

Autorzy:                               Prowadzący zajęcia:

Daria Jeżowska, 252731                         dr inż. Jarosław Mierzwa

**Cel ćwiczenia**

Celem ćwiczenia było stworzenie programu, który wykorzystał kamerę USB z następującymi założeniami:

1. Wylistuj urządzenia typu cap (kamery) i stwórz interfejs umożliwiający wybór po nazwie urządzenia (drivera) z którym chcesz się połączyć
2. Połącz się z wybranym urządzeniem i za pomocą odpowiednich komunikatów łączących się z driverami kamery - skonfiguruj ją.
3. Za pomocą programu powinno dać się zmieniać opcje kamery (rozdzielczość obrazu, nasycenie, kontrast, ew. zoom, sterowanie kamera etc.)
4. Zapisz obraz z kamery w dowolnym formacie (wskazany JPG)
5. Zapisz obraz z kamery w postaci filmu AVI
6. Rozbuduj program o:

* stwórz prosty detektor ruchu - poprzez analizę obrazu z kamery w czasie rzeczywistym (wystarczy sprawdzać zmiany koloru kilku punktów (pikseli), ćwiczenie można rozwinąć o najprostsze algorytmy wykrywające krawędzie etc.)

**Wstęp**

Kamera cyfrowa to urządzenie rejestrujące obraz i dźwięk oraz zapisujące sygnał audiowizualny w postaci cyfrowej. Najczęściej komunikuje się z komputerem za pomocą portu USB, a w laptopach jest to część wbudowana. W ostatnim czasie kamerki stały się podstawowym wyposażeniem każdego domu z komputerem i okazały się być niezbędne w naszej codzienności. Za ich pomocą prawie codziennie każdy się komunikuje z innymi ludźmi w pracy, na uczelni czy po prostu jako forma kontaktu z bliskimi. Historia kamerek internetowych sięga początku lat 90 – w 1993 roku pierwszy raz obraz z takiej kamery został umieszczony w internecie.

**Przebieg ćwiczenia**

Program został zrealizowany w języku *C#* z pomocą *Windows Forms* oraz biblioteki *AVICAP32.dll.* Ta biblioteka jest używana w celu przechwytywania obrazu wideo z rozszerzeniem *.avi* z kamerek internetowych oraz innych tego typu urządzeń. Jest to dosyć stara biblioteka i wiele nowych urządzeń nie jest przez nią obsługiwana. Abyśmy mogli jej użyć w efektywny sposób skorzystaliśmy z *wrappera* dla avicap32.dll. Poszczególne funkcjonalności uzyskuje się za pomocą funkcji SendMessage, która za argumenty przyjmuje *uchwyt* (HWND) z którego korzysta WIN32 API oraz odpowiednich wartości od których zależy co się może stać. Np. zmienna *WM\_CAP\_DRIVER\_CONNECT* przyjmująca wartość *0x40a* odpowiada za wychwycenie kamerki.

**Wykrycie kamery i wyświetlanie obrazu**

Funkcja odpowiedzialna za wykrycie kamery i wyświetlanie obrazu jest już w naszym wrapperze i wygląda następująco:

Obraz zawierający tekst

Opis wygenerowany automatycznie

Na początku tworzymy nasz *uchwyt* hHwnd, który stworzy nam nowe okienko w którym będziemy mogli wybrać urządzenie z którego chcemy skorzystać. *SendMessage* zwraca wartości 0 lub 1 i na podstawie tego możemy uzyskać różne funkcjonalności. W pierwszy ifie sprawdzamy czy istnieje urządzenie z którym możemy się połączyć, jeśli tak jest to wyświetlamy podgląd obrazu z kamerki. W przeciwnym wypadku niszczymy nasze powstałe wcześniej okienko. Dodatkowo w SetWindowPos możemy ustawić rozdzielczość obrazu.

**Robienie i zapisywanie zdjęć**

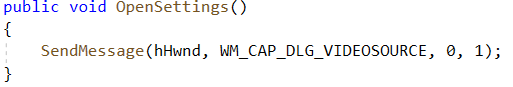
Funkcja odpowiedzialna za zapisanie obrazu wygląda następująco:

Obraz zawierający tekst

Opis wygenerowany automatycznie

Tworzymy obiekt *saveFileDialog*, co pozwala nam na otworzenie okna w którym wybieramy lokalizację dla pliku z rozszerzeniem *.jpg*, a następnie za pomocą SendMessage kopiujemy aktualny obraz z kamerki, a następnie przechowujemy go w zmiennej *data* i przerabiamy na bitmapę, a następnie zapisujemy.

**Otwieranie ustawień**



Za pomocą WM\_CAP\_DLG\_VIDEOSOURCE uruchamiamy okienko ustawień.

**Nagrywanie filmiku**

Podobnie jak w przypadku zapisywania zdjęcia tutaj też musimy stworzyć obiekt *saveFileDialog*, aby móc zapisać plik o danym rozszerzeniu. *saveFileDiaglog.ShowDialog()* otwiera nam okienko wyboru lokalizacji naszego przyszłego filmiku. Następnie kolejno dzięki *WM\_CAP\_EDIT\_COPY* kopiujemy bufor z poszczególnymi ramkami filmu, *WM\_CAP\_FILE\_SET\_CAPTURE\_FILE* nadaje filmowi nazwę wcześniej nadaną przy wybieraniu lokalizacji, *WM\_CAP\_FILE\_SAVEAS* kopiuje nam nasz przechwycony obraz, a *WM\_CAP\_SEQUENCE* zapisuje nam film do pliku*.* Natomiast funkcja *StopRecording()* wykorzystuje funkcję *CloseConnection()* do zakończenia przechwytywania obrazu.

Obraz zawierający tekst

Opis wygenerowany automatycznie

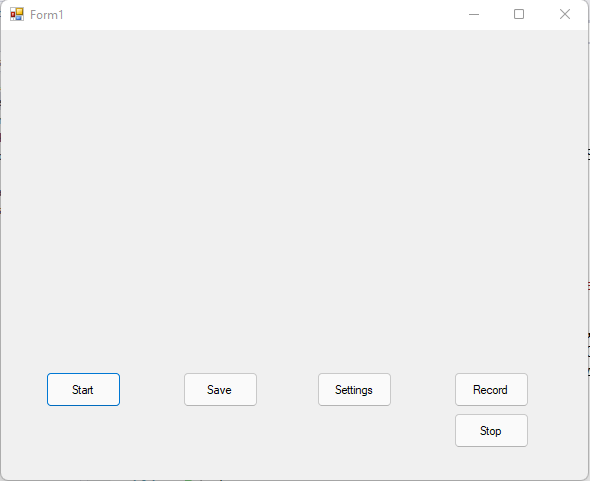
**Aplikacja okienkowa**

W pliku *Form1.cs* są funkcje odpowiadające m.in. za wywoływanie funkcji obiektu webCam przy kliknięciu odpowiedniego guzika, a które zostały opisane wyżej.

Obraz zawierający tekst

Opis wygenerowany automatycznie

**Wygląd aplikacji i działanie**



Po naciśnięciu przycisku *Start* wyświetla się nam okno wyboru urządzenia wideo.

Obraz zawierający tekst, zrzut ekranu, monitor, wewnątrz

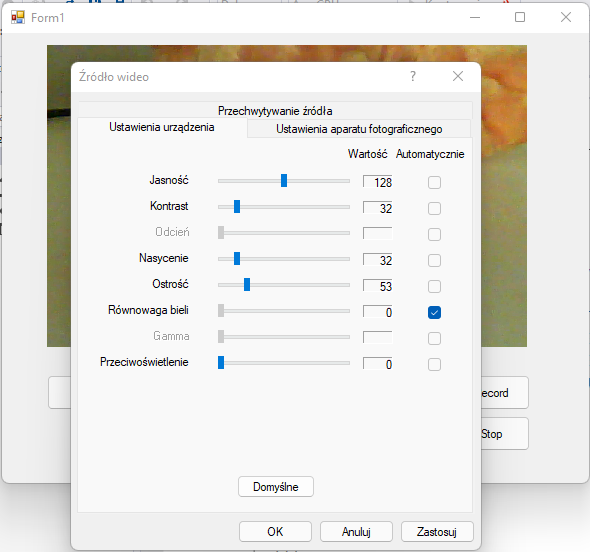
Opis wygenerowany automatycznie

Po wyborze urządzenia i zatwierdzeniu przyciskiem *OK* wyświetla się obraz z kamerki.

Obraz zawierający tekst, wewnątrz, zrzut ekranu

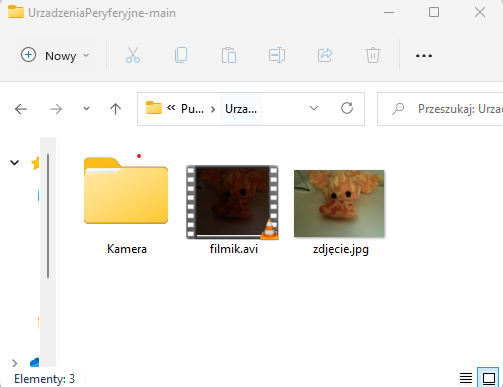
Opis wygenerowany automatycznie

Po kliknięciu przycisku *Settings* otwiera się okno ustawień i możemy zmienić jasność, kontrast, nasycenie, itd.



Po kliknięciu guzika *Save* otwiera nam się okno wyboru lokalizacji i nazwy pliku jpg, natomiast po kliknięciu *Record* otwiera się okno wyboru lokalizacji i nazwy pliku wideo. Po naciśnięciu przycisku *Stop* obraz się zapisuje i program przerywa przechwytywanie obrazu.

|  |  |
| --- | --- |
| Obraz zawierający tekst  Opis wygenerowany automatycznie | Obraz zawierający tekst  Opis wygenerowany automatycznie |



**Wnioski**

Wykonaliśmy podstawowy wariant aplikacji niezawierający detekcji ruchu. Zadanie nie było najłatwiejsze z początku, ponieważ w porównaniu do nowoczesnych narzędzi nie było zbyt dużo informacji czy dokumentacji i wiele czasu poświęciliśmy na reaserch. Ostatecznie program działa poprawnie i spełnia swoje zadania.