POLITECHNIKA POZNAŃSKA
WYDZIAŁ ELEKTRYCZNY

PODPOWIADARKA DO PUZZLI

PODSTAWY TELEINFORMATYKI - PROJEKT

Piotr Goździewski
Martyna Markiewicz
Łukasz Śmierzchalski

Prowadzący:
mgr inż. Przemysław Walkowiak

Poznań, 2018

Spis treści

1. Ogólna charakterystyka projektu

Celem projektu jest stworzenie aplikacji desktopowej, której zadaniem jest pomoc podczas układania puzzli. Aplikacja ma za zadanie podpowiedzieć, w którym miejscu umieścić puzzel.

Aplikacja ma ułatwić często czasochłonną sztukę układania puzzli, dzięki wykorzystaniu mocy obliczeniowej komputera (z systemem Windows). Zadanie użytkownika ogranicza się do wykonania dwóch zdjęć, pierwszego przedstawiającego obraz który ma powstać po ułożeniu puzzli oraz rozsypanych puzzli. W kolejnym kroku należy przesłać zdjęcia na komputer i wczytać w aplikacji. Możliwa jest także korekta ustawień zdjęcia puzzli w celu dokładniejszego wykrycia konturów. Następnie specjalne algorytmy, służące do wykrywania krawędzi, dzielą obraz puzzli na pojedyncze puzzle, które są wyświetlane na liście. Po wybraniu danego puzzla z listy, na obrazie wyświetlane jest miejsce, w którym powinien znaleźć się puzzel.

Interfejs aplikacji jest bardzo prosty i przyjazny dla użytkownika. Podpowiedzi w postaci tekstu oraz specjalnych grafik pozwalą każdemu komfortowo korzystać z aplikacji.

Temat podpowiadarki do puzzli zaciekawił nas, ponieważ nigdy wcześniej nie spotkaliśmy się z podobnym programem. Przedmiot obieralny “Przetwarzanie obrazów i systemy wizyjne” pozwolił nam zdobyć wiedzę, którą chcielibyśmy wykorzystać w realizacji tego projektu. Program ten pozwoli na szybsze ułożenie puzzli, dzięki podpowiedziom, w której części obrazu trzeba umieścić dany puzzel. Podpowiadarka przyda się zarówno dzieciom jak i osobom dorosłym.

2. Wymagania

W tym rozdziale przedstawiono wymagania funkcjonalne, niefunkcjonalne oraz szczegółowe funkcjonalności oferowane przez aplikację.

2.1. Wymagania funkcjonalne

- Wczytanie zdjęcia obrazka, który prezentuje wynik po ułożeniu wszystkich puzzli.
- Wczytanie zdjęcia rozłożonych puzzli.
- Zmiana ustawień wczytanego zdjęcia rozłożonych puzzli w celu wykrycia wszystkich konturów puzzli.
- Przeglądanie listy z puzzlami.
- Sprawdzenie pozycji danego puzzla na obrazku.

2.2. Wymagania niefunkcjonalne

- Aplikacja desktopowa przeznaczona na system Windows.
- Język programowania - C#.
- Interfejs aplikacji w języku polskim.
- Niewymagane połączenie z Internetem.
- Wymagany aparat cyfrowy, tablet, skaner albo smartfon w celu wykonania zdjęć.
- Wymagany kabel, karta SD lub standard łączności bezprzewodowej Bluetooth do przesyłania zdjęć na komputer.

2.3. Funkcjonalności oferowane przez aplikację

1) Znalezienie konturów puzzli
    a. wczytanie zdjęcia z puzzlami,
    b. konwersja obrazu na skalę szarości,
    c. wykonanie progowania adaptacyjnego na obrazie (parametry ustawiane przez użytkownika aplikacji),
    d. znalezienie konturów i dodanie ich do wektora punktów (*VectorOfPoints*),
    e. filtrowanie znalezionych konturów w celu odrzucenia błędnie znalezionych konturów na podstawie ich rozmiaru,
    f. dodanie poprawnych konturów do wektora punktów,
    g. rysowanie konturów,
    h. informowanie użytkownika o ilości znalezionych konturów.

2) Wyodrębnienie puzzli
    a. odczyt danych z wektora punktów i dodanie ich do listy,
    b. znalezienie wartości brzegowych danego konturu,
    c. wycięcie/wyodrębnienie puzzla,
    d. zapis wyodrębnionych puzzli do listy obrazów.

3) Wyszukanie cech charakterystycznych danego puzzla i porównienie go z cechami obrazu wynikowego
    a. znalezienie cech zdjęć za pomocą detektora SIFT i dodanie ich do wektora,
    b. porównanie cech pomiędzy dwoma obrazkami,
    c. rysowanie prostokąta na podstawie rozmiaru zdjęcia wyodrębnionego puzzla.

3. Wykorzystane narzędzia

W tym punkcie przedstawione zostały wykorzystane narzędzia, biblioteki i środowisko programistyczne podczas tworzenia projektu.

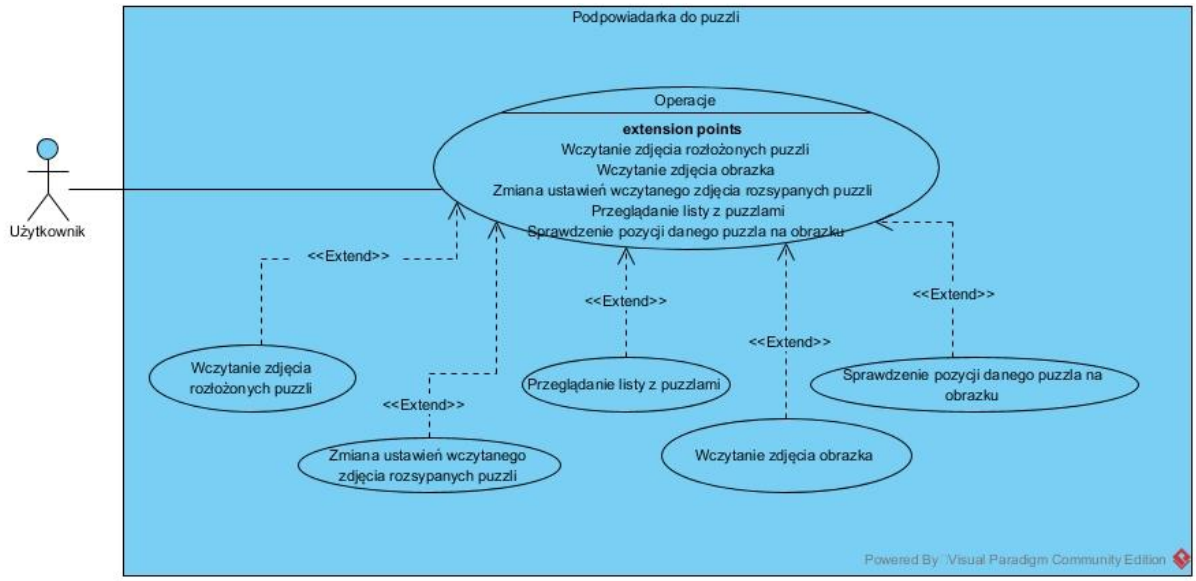
- Środowisko programistyczne Visual Studio 2015 Enterprise.
- Visual Paradigm Community Edition.
- Git.
- Inkscape.
- Gimp.
- EmguCV.

Projekt został wykonany przy użyciu środowiska programistycznego Visual Studio 2015 Enterprise. Skorzystaliśmy z wieloplatformowego *wrappera EmguCV*, który jest kompatybilny z wybranych środowiskiem. Wybór wyżej wymienionej technologii był uzasadniony kilkoma powodami tj. znajomość środowiska Visual Studio oraz doświadczenie programowania w języku C#.

4. Diagramy UML

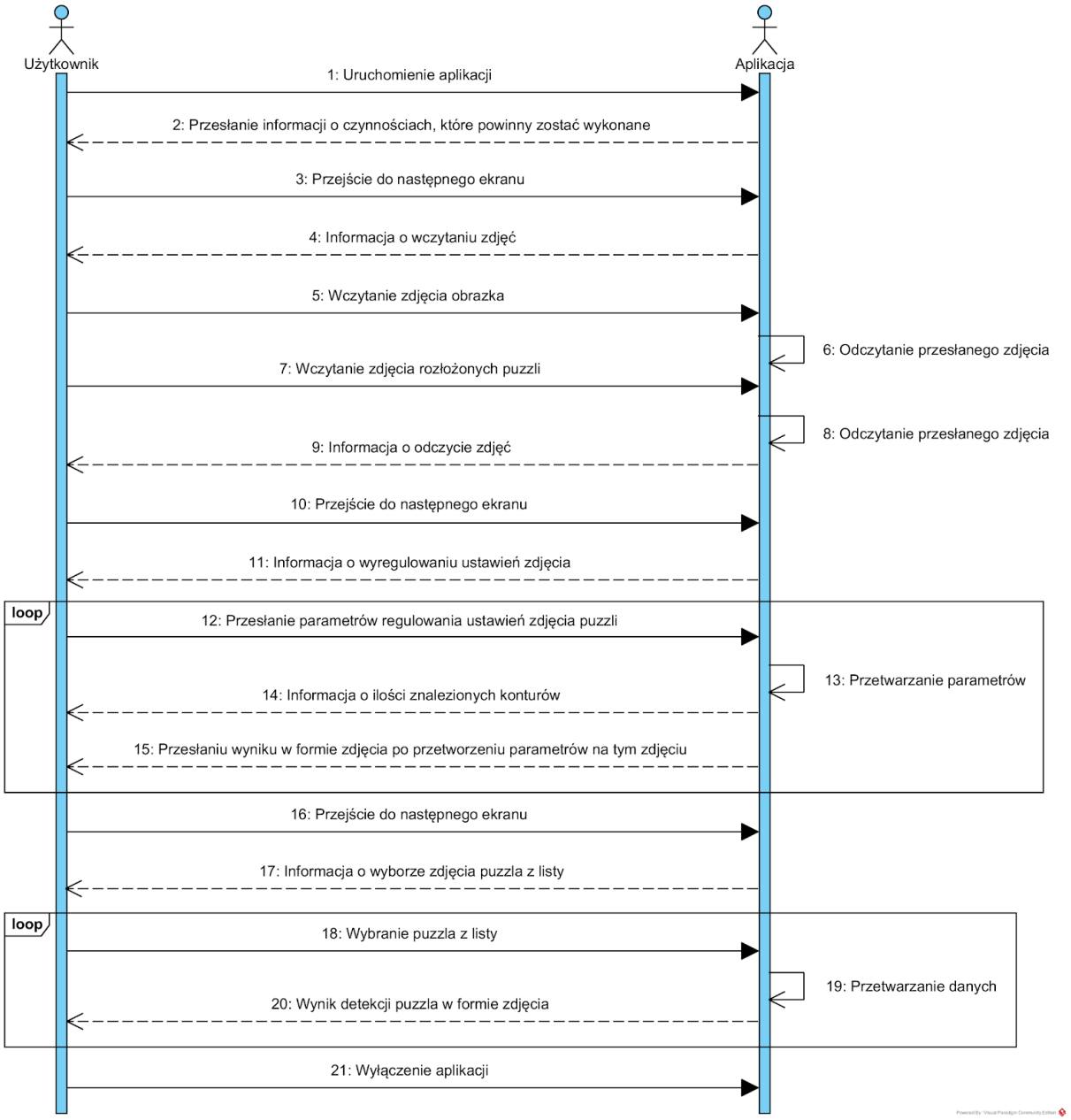
Na rysunku 4.1 przedstawiono diagram przypadków użycia. Zostały na nim przedstawione wszystkie operacje aplikacji, które może dokonać użytkownik podczas użytkowania aplikacji. Do tych operacji należą między innymi takie funkcjonalności jak: wczytanie zdjęcia obrazka, wczytanie zdjęcia rozłożonych puzzli, zmiana ustawień

wczytanego zdjęcia rozsypanych puzzli, przeglądanie listy z puzzlami, sprawdzenie pozycji danego puzzla na obrazku.



Rysunek 4.1 Diagram przypadków użycia aplikacji.

Na rysunku 4.2 przedstawiono diagram sekwencyjny, który prezentuje zasadę działania aplikacji. Użytkownikowi, po uruchomieniu aplikacji, zostaje przedstawiony ekran główny, na którym znajduje się opis czynności, które należy wykonać przed rozpoczęciem korzystania z programu. Zadaniem użytkownika jest przygotowanie zdjęcia puzzli, zdjęcia rozsypanych puzzli oraz zgranie zdjęć na komputer. W kolejnym kroku użytkownik w odpowiednie miejsca wczytuje wcześniej przygotowane zdjęcia. Następnie użytkownik ma możliwość ustawienia parametrów dla zdjęcia rozsypanych puzzli w celu poprawnego wykrycia wszystkich konturów. W ostatnim kroku użytkownik ma do dyspozycji listę wszystkich wyodrębnionych puzzli. Po wybraniu danego puzzla, jego pozycja zostanie zaznaczona na wczytanym wcześniej zdjęciu puzzli.

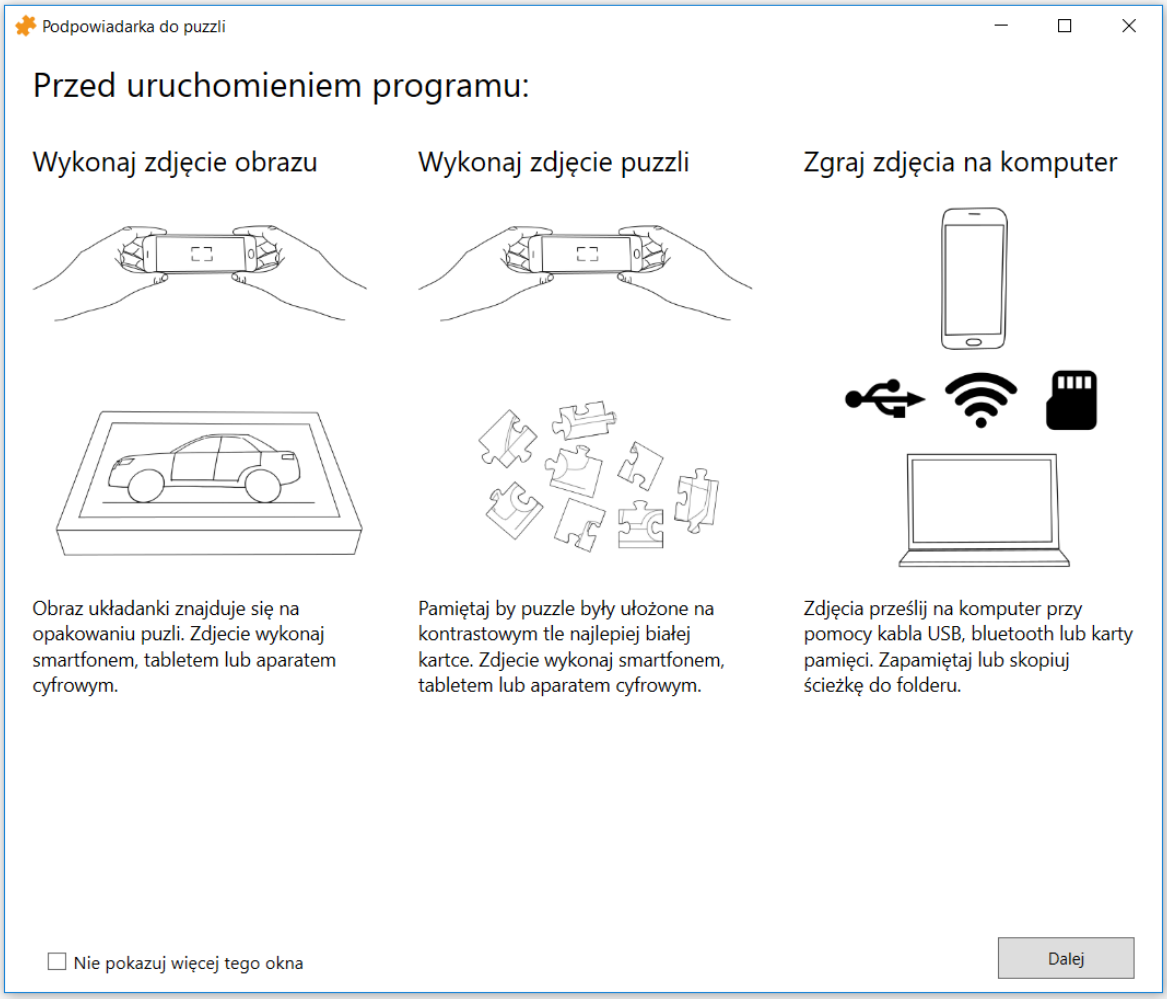


Rysunek 4.2 Diagram sekwencyjny aplikacji.

5. Projekt interfejsu graficznego aplikacji

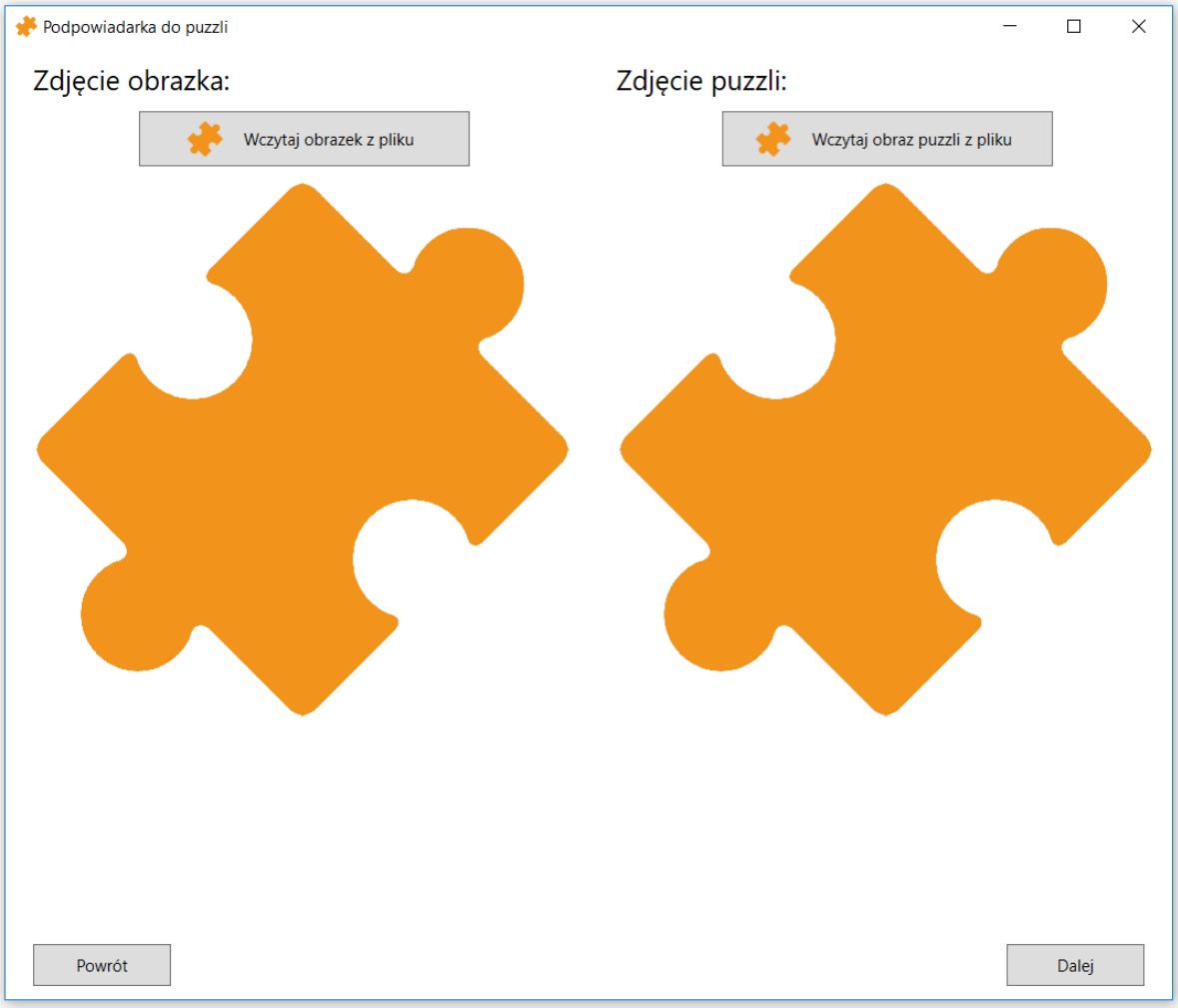
W poniższym rozdziale przedstawiono interfejs graficzny aplikacji. Zrzuty ekranu przedstawiają: ekran główny po uruchomieniu aplikacji, ekran wczytywania zdjęcia obrazka oraz rozłożonych puzzli, ekran regulowania ustawień zdjęcia oraz ekran detekcji puzzli.

Na rysunku 5.1 przedstawiony został ekran główny wyświetlający się po uruchomieniu aplikacji. Przedstawione zostały na nim informacje o czynnościach, takich jak: wykonanie zdjęcia obrazka puzzli i zdjęcia rozłożonych puzzli, a następnie przesłanie tych zdjęć na komputer, które powinien wykonać użytkownik przed przystąpieniem do kolejnego kroku.



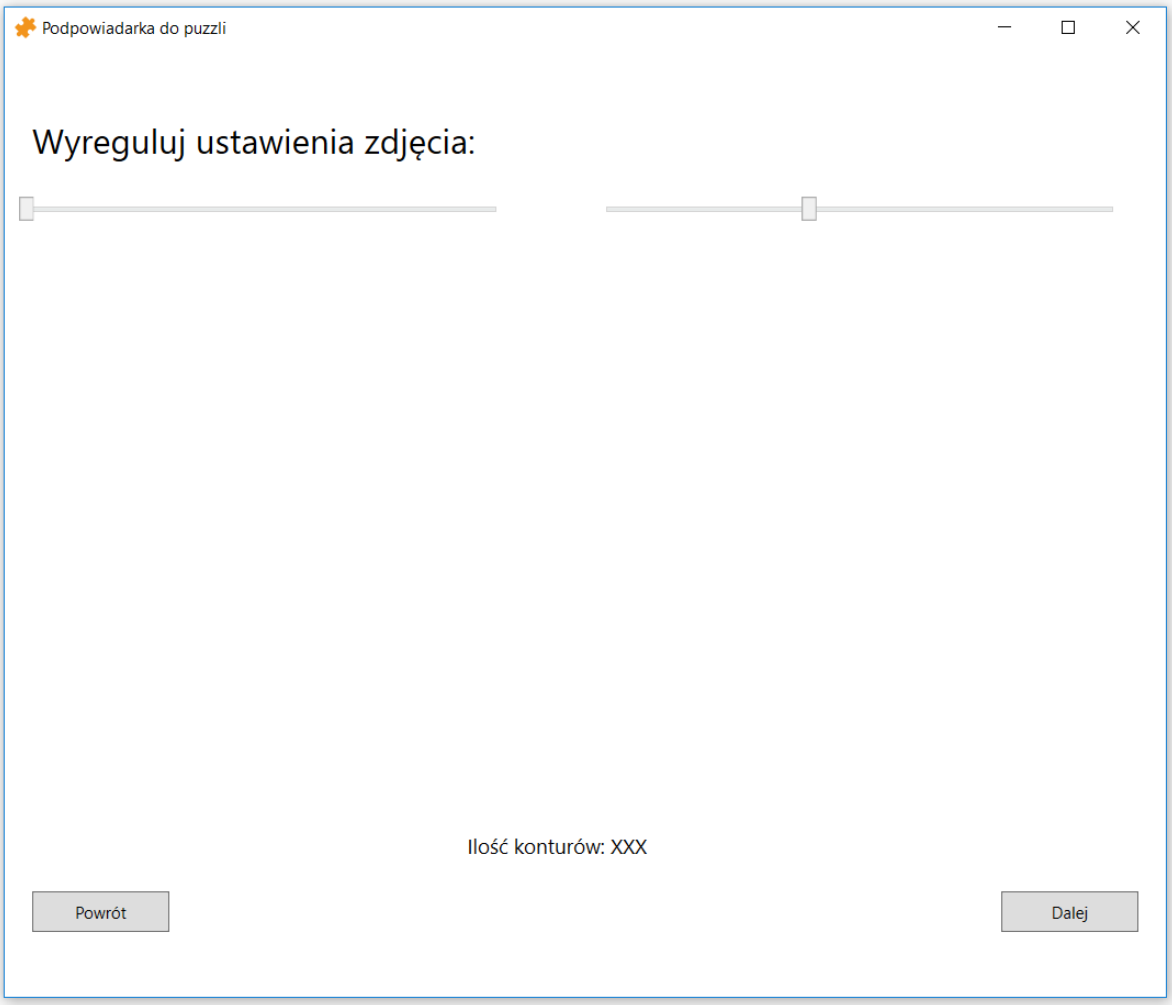
Rysunek 5.1 Ekran główny.

Na rysunku 5.2 przedstawiony został ekran wczytania zdjęć, takich jak: zdjęcie prezentujące obrazek wynikowy po ułożeniu wszystkich puzzli oraz zdjęcie prezentujące rozłożone do góry obrazkiem puzzle. Z poziomu tego ekranu użytkownik może powrócić do ekranu głównego lub po poprawnym wczytaniu zdjęć przejść do następnego kroku.



Rysunek 5.2 Ekran wczytywania zdjęć.

Na rysunku 5.3 przedstawiony został ekran, w którym użytkownik ma możliwość ustawienia parametrów dla zdjęcia rozsypanych puzzli w celu poprawnego wykrycia wszystkich konturów. Powodem, dla których użytkownik musi odpowiednio zmienić ustawienia jest to, że zdjęcia nie zawsze wykonane zostaną w tym samym świetle, co ma bardzo duży wpływ na działanie algorytmu. Użytkownik na bieżąco jest informowany o ilości wykrytych konturów puzzli. Z poziomu tego ekranu użytkownik może powrócić do poprzedniego ekranu, który został przedstawiony na rysunku 5.2 lub przejść do następnego kroku.



Rysunek 5.3 Ekran regulowania ustawień zdjęcia.

Na rysunku 5.4 przedstawiony został ekran detekcji puzzli. Użytkownik po wybraniu danego puzzla z listy, będzie informowany, w którym miejscu na obrazie wynikowym powinien znajdować się wybrany puzzel. Na podstawie cechy wyznaczane jest prawdopodobne miejsce umieszczenia puzzla w układance, zaprezentowane kolorowym prostokątem. Użytkownik będzie mógł powrócić do poprzedniego ekranu lub zakończyć działanie programu.

Rysunek 5.4 Ekran detekcji puzzli.

6. Najważniejsze fragmenty kodu aplikacji

W tym rozdziale przedstawione zostały najważniejsze fragmenty programu.

- Rysunek 6.1 przedstawia fragment kodu dotyczący wykrywania poprawnych konturów. Wyodrębnianie konturów odbywa się poprzez filtrację ich rozmiaru.

```
for (int i = 0; i < contours.Size; i++)
{
    double a = CvInvoke.ContourArea(contours[i], false);
    if (a > 600 && a < ((mask.Cols * mask.Rows) / 2))
        good_contours.Push(contours[i]);
}
```

Rysunek 6.1 Fragment kodu - wykrywanie konturów.

- Rysunek 6.2 przedstawia wyodrębnianie puzzla z obrazka. W tym celu należało wykryć kontur danego puzzla, który został zapisany do tablicy. Następnie należało odczytać dane i znaleźć wartość brzegową danego konturu.

```
var array = good_contours[i].ToArray();
p.AddRange(array);

var min_x = array.Min(a => a.X);
var min_y = array.Min(a => a.Y);
var max_x = array.Max(a => a.X);
var max_y = array.Max(a => a.Y);
```

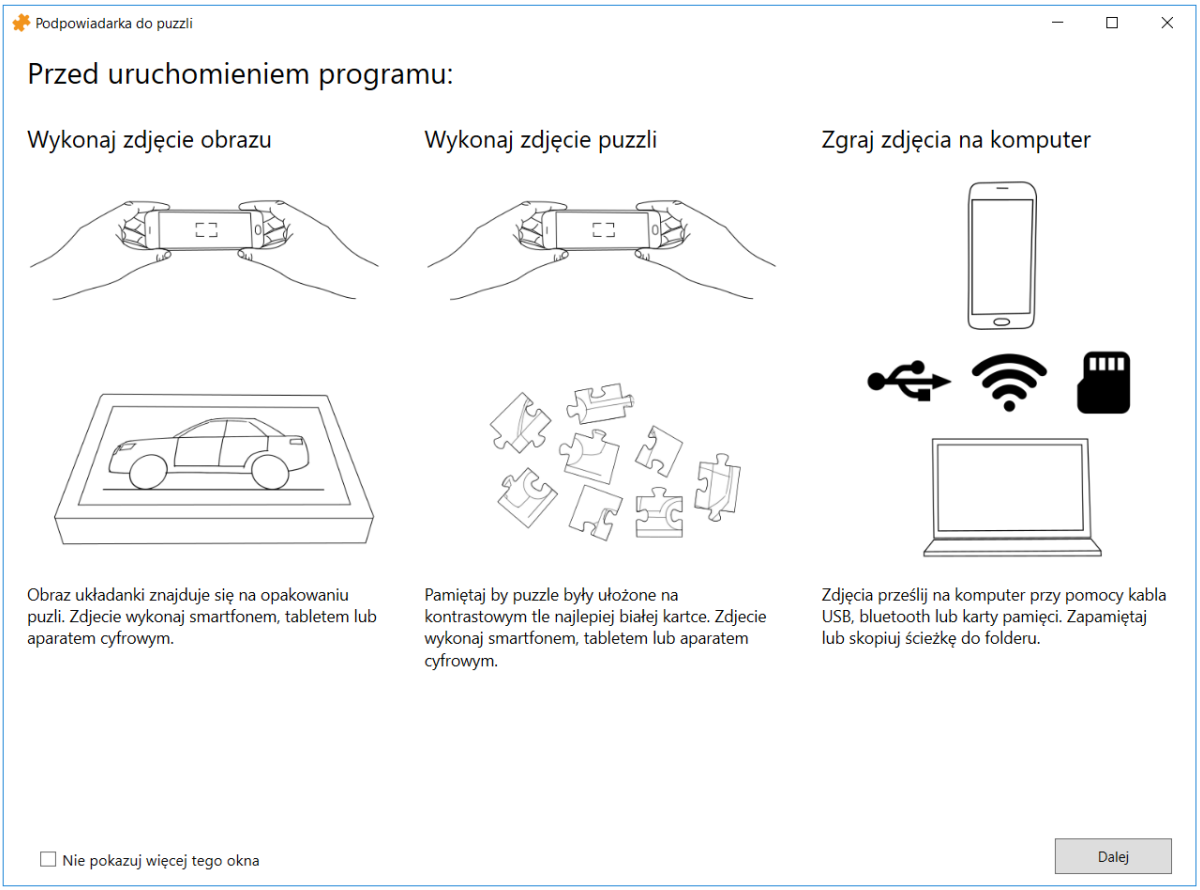
Rysunek 6.2 Fragment kodu - wyodrębnianie puzzla z obrazka.

7. Instrukcja użytkowania aplikacji

W tym punkcie przedstawiono instrukcję użytkowania aplikacji.

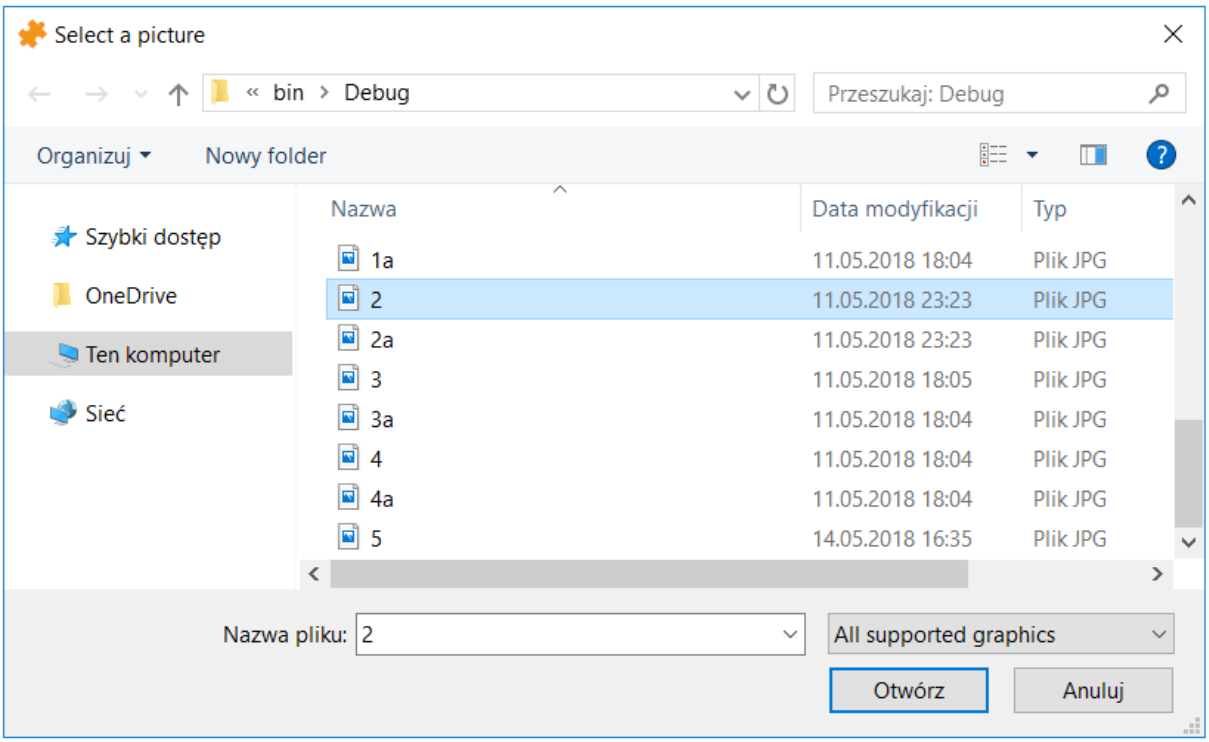
Na rysunku 7.1 przedstawiono ekran główny programu, na którym znajduje się opis następujących czynności:

1. Wykonanie zdjęcia obrazka puzzli na opakowaniu.
2. Wykonanie zdjęcia rozsypanych puzzli.
3. Przesłanie zdjęcia na komputer, w tym celu można wykorzystać kabel USB, bluetooth lub kartę pamięci.
4. Zapamiętanie lub skopiowanie ścieżki do folderu.
5. Wczytanie przygotowanych zdjęć z pliku.



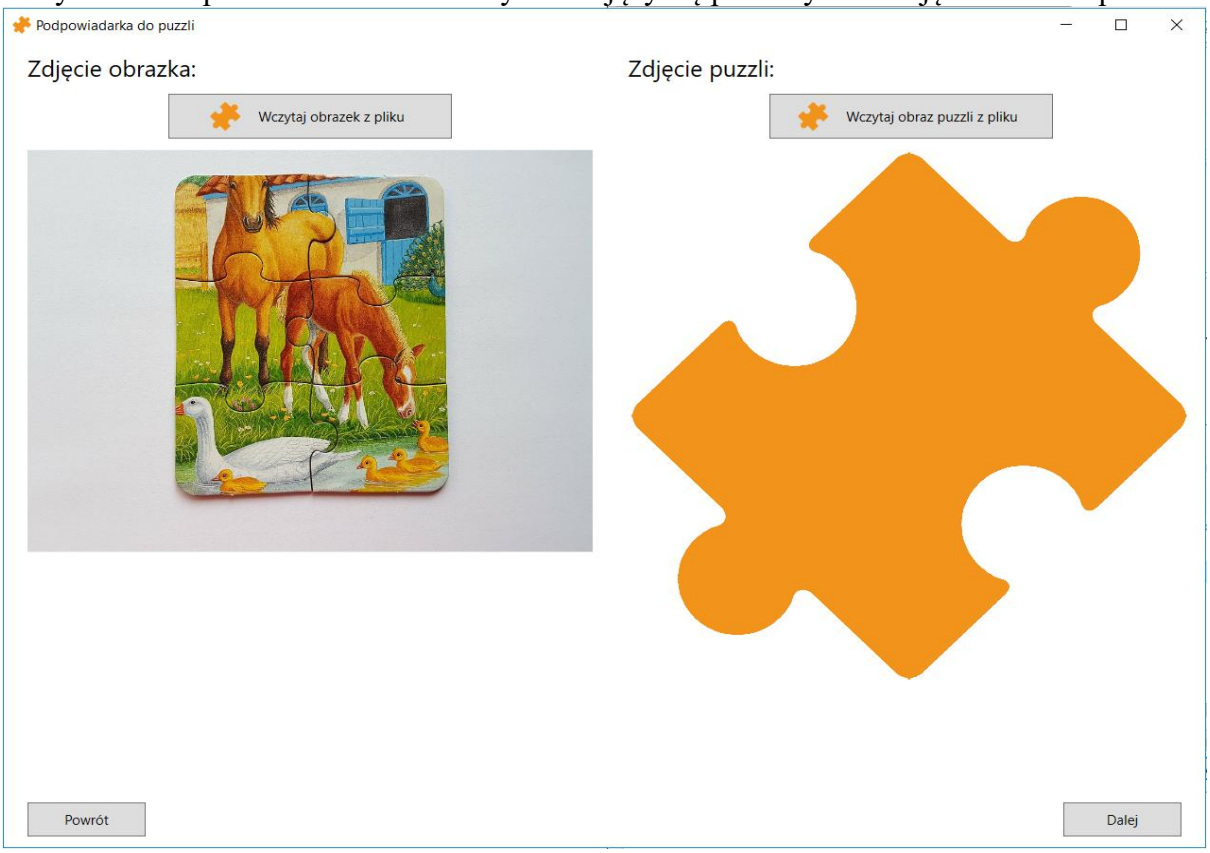
Rysunek 7.1 Ekran główny programu.

Na rysunku 7.2 przedstawiono okno wybierania zdjęcia do wczytania, na którym użytkownik wybiera odpowiednie zdjęcie puzzli, które chce użyć w programie.



Rysunek 7.2 Okno wczytywania zdjęcia.

Na rysunku 7.3 przedstawiono ekran wyświetlający się po wczytaniu zdjęcia obrazka puzzli.

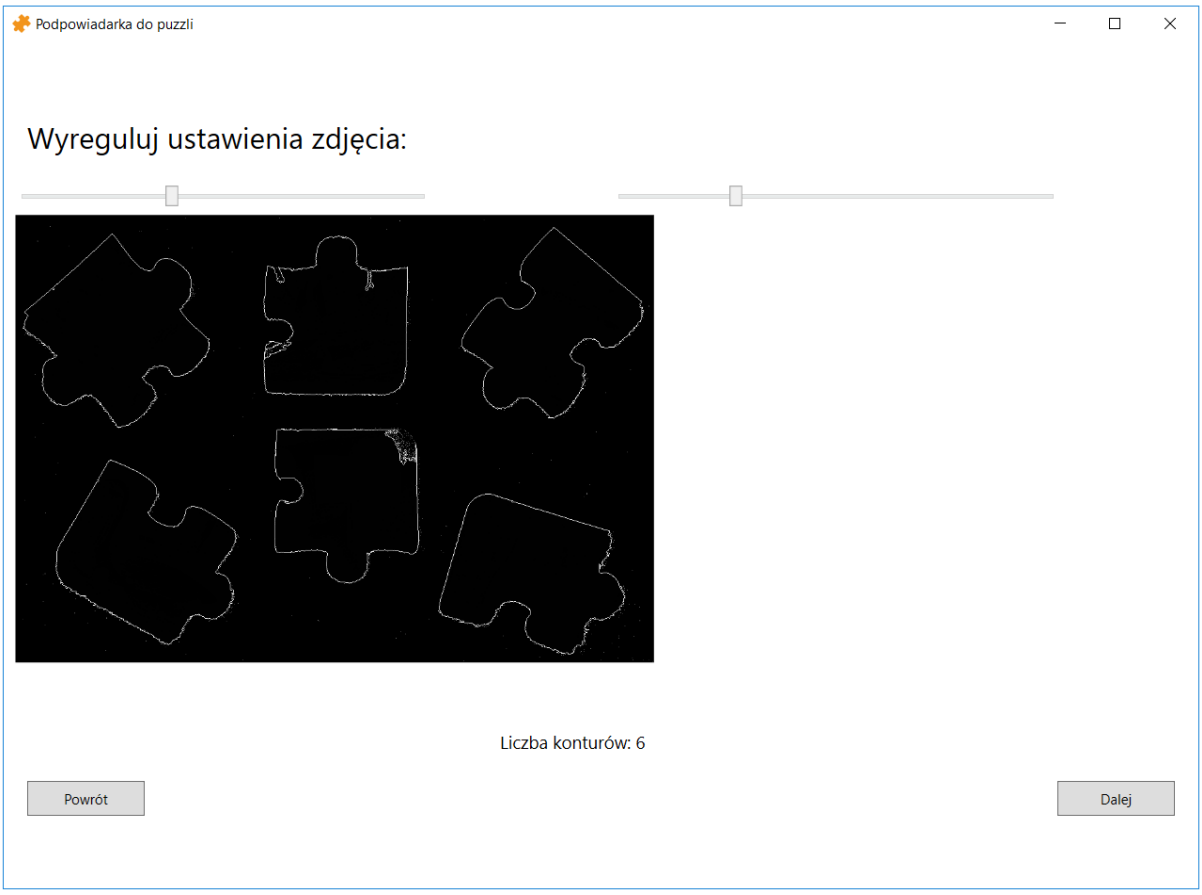


Rysunek 7.3 Ekran po wczytaniu zdjęcia obrazka puzzli.

Na rysunku 7.4 przedstawiono ekran z wczytanymi zdjęciami puzzli. Po lewej stronie znajduje się zdjęcie obrazka puzzli, po prawej stronie jest widoczne zdjęcie rozsypanych puzzli.

Rysunek 7.4 Ekran z wczytanymi zdjęciami puzzli.

Na rysunku 7.5 przedstawiono ekran zmiany ustawień zdjęcia. Użytkownik ma możliwość zmiany ustawień zdjęcia rozsypanych puzzli, ustawiając odpowiednie wartości trackbarów, tak aby każdy kontur puzzla był wyraźnie zaznaczony oraz aby liczba konturów zgadzała się z ilością puzzli na zdjęciu.



Rysunek 7.5 Ekran zmiany ustawień zdjęcia.

Na rysunku 7.6 oraz rysunku 7.7 przedstawiono ekrany listy wyboru puzzli. Użytkownik, w celu wykrycia wybranego puzzla na głównym zdjęciu puzzli, wybiera zdjęcie puzzla z dostępnej listy.

Rysunek 7.6 Ekran listy wyboru puzzli.

Rysunek 7.6 Ekran listy wyboru puzzli.

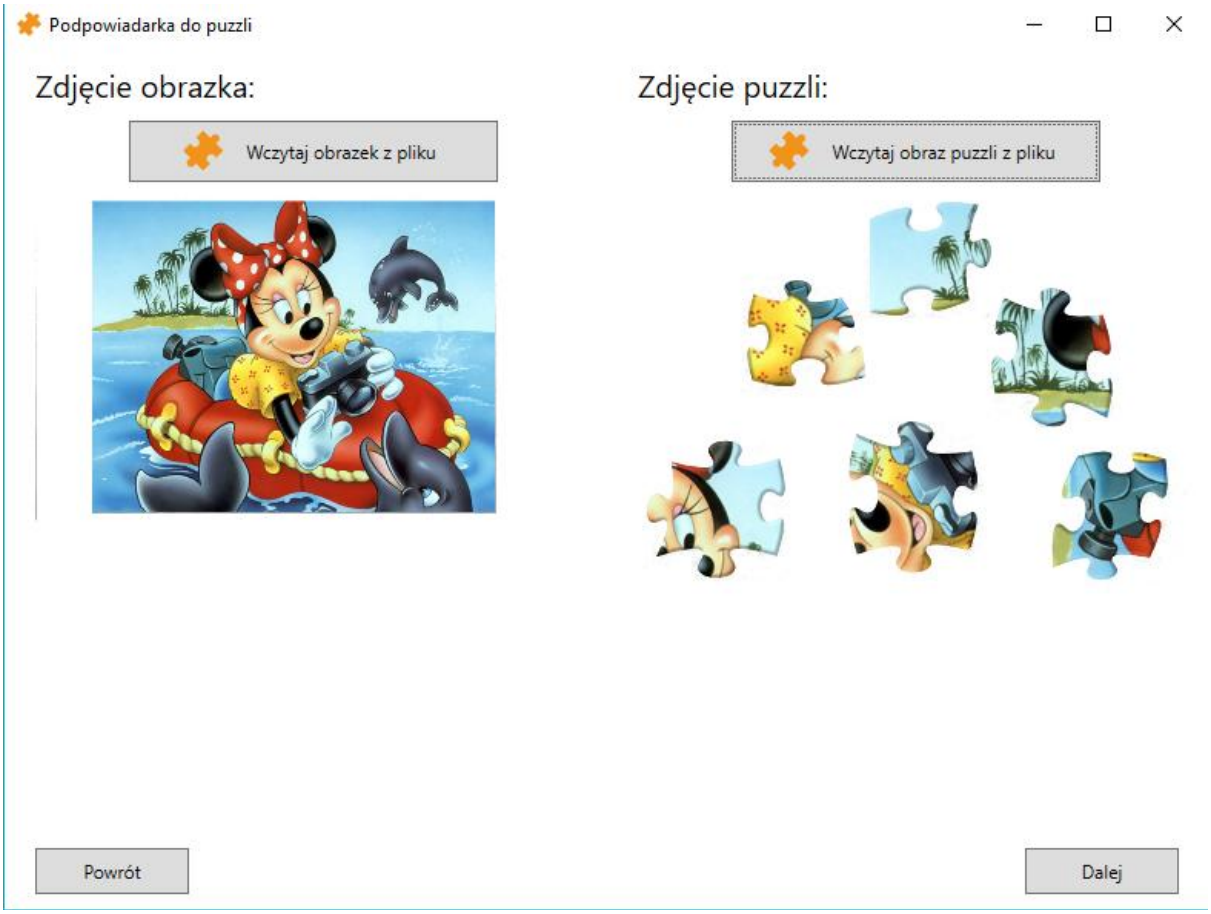
Po wybraniu puzzla z listy, na ekranie pojawi się zdjęcie główne puzzli wraz z zaznaczonym obszarem, w którym znajduje się wybrany puzzel, co zostało przedstawione na rysunku 7.8 oraz rysunku 7.9. Z poziomu tego ekranu możliwe jest również zamknięcie programu.

Rysunek 7.8 Ekran dopasowania puzzli.

Rysunek 7.9 Ekran dopasowania puzzli.

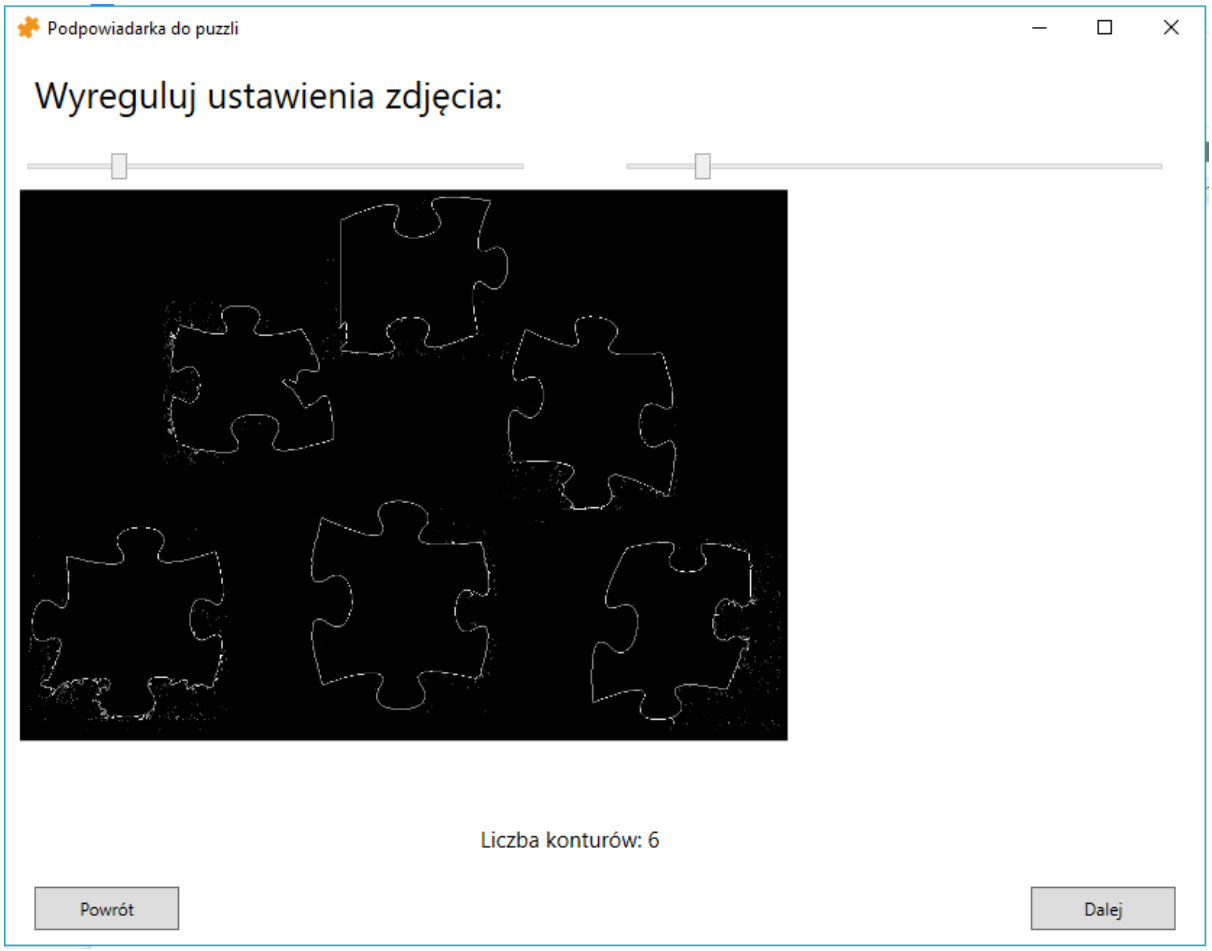
8. Testy aplikacji

W tym punkcie przedstawiono testy aplikacji. Testy zostały przeprowadzone na puzzlach widocznych na rysunku 8.1.



Rysunek 8.1 Ekran wczytanych zdjęć puzzli.

Po wczytaniu puzzli dokonano regulacji ustawień zdjęcia tak, aby kontury były dobrze widoczne oraz aby ich liczba zgadzała się z liczbą puzzli na wczytanym zdjęciu, co przedstawiono na rys. 8.2.



Rysunek 8.2 Ekran zmieniania ustawień zdjęcia.

Na rysunkach od rys. 8.3 do rys. 8.8 przedstawiono dopasowanie poszczególnych sześciu puzzli do miejsca ich występowania na zdjęciu głównym puzzli.

Rysunek 8.3 Dopasowanie puzzla pierwszego.

Rysunek 8.4 Dopasowanie puzzla drugiego.

Rysunek 8.5 Dopasowanie puzzla trzeciego.

Rysunek 8.6 Dopasowanie puzzla czwartego.

Rysunek 8.7 Dopasowanie puzzla piątego.

Rysunek 8.8 Niepoprawne dopasowanie puzzla szóstego.

Wnioski

Na powyższych sześciu obrazkach zostało przedstawione wykrywanie wyodrębnionych puzzli na obrazie początkowym. Pięć z sześciu wyodrębnionych puzzli zostało poprawnie wykrytych i zaznaczonych na obrazie. Jeden z puzzli nie został poprawnie wykryty i zamiast poprawnie zaznaczonego obszaru widać tylko przechodzącą przez fragment obrazu linię. Niepoprawne dopasowanie może być spowodowane faktem, że zostało wykryte zbyt mało cech, aby dopasować wyodrębniony puzzel do wejściowego obrazka puzzli. Aby program mógł poprawnie wykryć cechy obrazu trzeba pamiętać, aby zdjęcia były zrobione w dobrej jakości i wykonane były przy dobrym oświetleniu.

9. Podsumowanie

W tym rozdziale przedstawiono podsumowanie wykonanego projektu.

9.1. Podział prac

W tym podpunkcie przedstawiony został podział prac wykonany przez daną osobę w zespole. Podział wykonanych prac przedstawia tabela 9.1.

Tabela 9.1 Podział prac na członków zespołu.

Osoba	Wykonane prace
Piotr Goździewski	1. Implementacja funkcjonalności odrzucania błędnie znalezionych konturów. 2. Implementacja algorytmu rozpoznającego obraz danego puzzla wraz z jego zaznaczeniem na obrazie wynikowym. 3. Przeprowadzanie testów działania aplikacji. 4. Praca nad dokumentacją.
Martyna Markiewicz	1. Implementacja algorytmu wykrywającego i zaznaczającego kontur danego puzzla. 2. Implementacja metody wyodrębniania puzzli. 3. Przeprowadzanie testów działania aplikacji. 4. Praca nad dokumentacją.
Łukasz Śmierzchalski	1. Wykonanie grafik oraz opracowanie tekstów z podpowiedziami dla użytkownika. 2. Implementacja responsywnego interfejsu graficznego. 3. Testowanie aplikacji dla kłopotliwych danych wejściowych. 4. Praca nad dokumentacją.

9.2. Cele zrealizowane i niezrealizowane

Podczas realizacji projektu udało się zrealizować wszystkie początkowe założenia względem aplikacji:

- Stworzono interfejs użytkownika.
- Zaprojektowano i wykonano grafiki z podpowiedziami.
- Zaimplementowano funkcjonalne wgrywanie zdjęć do programu.
- Zaimplementowano funkcjonalny oraz prosty w obsłudze algorytm dzielący zdjęcie puzzli na osobne puzzle.
- Zaimplementowano algorytm detekcji cech.

9.3. Napotkane problemy

Początkowo projektowano interfejs aplikacji jako jedno okno mieszczące wszystkie funkcje w sobie bez konieczności otwieranie dodatkowych okien. Niestety okazało się, że interfejs wymusza postępowanie użytkownika w określonej kolejności i niemożliwym jest umieszczenie wszystkiego w jednym widoku bez obsługiwania dużej liczby wyjątków, do której może doprowadzić ciekawość użytkownika chcącego np.: przed wgraniem zdjęcia uruchomić algorytm odpowiadający pozycję puzzla. Dodatkowo, gdyby określone elementy miały znikać i pojawiać się, oprogramowanie algorytmu odpowiadającego za ich widoczność w oknie programu zajęłaby ogromną liczbę linijek kodu w programie.

9.4. Rozwiązania napotkanych problemów

Problem interfejsu użytkownika rozwiązaliśmy implementując system “stron”, który jest oferowany programistom aplikacji typu Windows Presentation Foundation. “Strony” pozwalają z jednej strony na zaprojektowanie interfejsu w języku XAML, który pozwala

bardzo wygodnie projektować nawet bardzo skomplikowane interfejsy, a dzięki przeglądowi aktualnego widoku okna programu możemy łatwo i szybko debugować ewentualne błędy w tej części programu, oraz testować wiele różnych ustawień i układów bez konieczności komplikowania za każdym razem programu.

9.5. Perspektywa rozwoju

Aplikację można rozwijać, dodając nowe funkcjonalności oraz poprawiając istniejące. Rozwój programu podzielono na dwie kategorie: bliższą w czasie, czyli możliwą do zaimplementowania w stosunkowo krótkim czasie oraz perspektywę bardziej odległą w czasie, której implementacja zajęłaby zdecydowanie więcej czasu.

1) W bliskiej perspektywie
    a. możliwość dodawania wielu zdjęć puzzli,
    b. specjalny algorytm proponujący parametry na ekranie regulacji ustawień zdjęcia,
    c. zaimplementowanie możliwości dostosowywania wyglądu aplikacji np.: zmiana typu, wielkości lub koloru czcionki, zmiana koloru tła lub możliwość ustawienia nocnych styli całej aplikacji.

2) W odległej perspektywie
    a. dodanie narzędzia pozwalającego łatwiej przesyłać zdjęcia z urządzenia posiadającego aparat na komputer tak, aby aplikacja desktopowa wiedziała jakiego zdjęcia szuka i w jakim miejscu, bez konieczność wyszukiwania go ręcznie w eksploratorze plików,
    b. stworzenie wersji aplikacji przystosowanej do wielu platform sprzętowych np.: Android, IOS, Linux,
    c. zaimplementowanie algorytmu korzystającego z kształtu puzzla dopasowując jego kształt do pozostałych puzzli i na tej podstawie ułożenie całej układanki numerując puzzle wierszami i kolumnami. Algorytm wymagałby, nie tylko dopasowania krawędzi dwóch puzzli do siebie, ale na podstawie wielu takich dopasowań stworzyć obraz puzzli. W kolejnych krokach należałoby zaprezentować wynik użytkownikowi, co byłoby związane z zaprojektowaniem dodatkowego algorytmu prezentującego te informacje użytkownikowi. Korzystając z tego algorytmu, program byłby w stanie układać puzzle bez znajomości obrazu całości, ponieważ wystarczyłyby mu same kształty puzzli.