# Projekt nr 17-Porównywanie Obrazów

Mirosław Kołodziej, Paweł Sipko, Jakub Trześniewski (WFiIS AGH)

# 1 Opis projektu

Projekt ten ma na celu umożliwienie użytkownikowi porównać ze sobą dwa obrazy (formaty JPG, BMP oraz PNG). Aplikacja wyświetla porównywane obrazy obok siebie, pozwala na przybliżanie ich, stworzenie oraz zapisanie obrazu porównawczego, a także zapisanie zaznaczonego wycinka obu obrazów do pliku.

# 2 Założenia wstępne

## 2.1 Założenia podstawowe:

Program ma pozwolić użytkownikowi na :

- dodanie dwóch obrazów w formacie .bmp o tych samych wymiarach,
- wyświetlenie ich obok siebie w głównym oknie aplikacji,
- przesuwanie obrazów synchronicznie (jeśli nie mieszczą się w oknie),
- powiększanie symultanicznie obu obrazów o wartości: 50%, 100%, 200%, 400%,
- utworzenie zdjęcia różnicowego i wyświetlenie go w głównym oknie,
- zapisanie w formacie .bmp zdjęcia różnicowego.

## 2.2 Założenia rozszerzone:

W wersji rozszerzonej program powinien pozwolić użytkownikowi na:

- wykonywanie dowolnych powiększeń na obu obrazach jednocześnie (przedział wartości powiększeń 1%-400%)
- wczytywanie zdjęć mających różne rozmiary (w takiej sytuacji najpierw powinno się ustalać względne powiększenie jednego zdjęcia względem drugiego, a później obsługiwać je synchronicznie)
- zaznaczenie wybranego obszaru na pierwszym zdjęciu, wycięcie tego obszaru z obu zdjęć i po sklejeniu obok siebie wybranych fragmentów w postaci jednego zdjęcia zapisać je do pliku o formacie .bmp
- wczytanie zdjęć mających inne formaty (.png i .jpg).

# 3 Analiza projektu

## 3.1 Specyfikacja danych wejściowych

Do aplikacji przekazywane są dwa obrazy zapisane w formatach .jpg, .bmp lub .png (nie muszą być w tym samym formacie). Po przekazaniu ich, automatycznie wczytują się one w oknie aplikacji.

## 3.2 Opis oczekiwanych danych wyjściowych

Stosujemy podział na dane “materialne” (tj. pliki wyjściowe) oraz “niematerialne” (informacje wyświetlane na ekranie).

### 3.2.1 Pliki wyjściowe

Aplikacja pozwala na zapisanie w formacie .bmp zdjęcia różnicowego oraz wycinków obszaru obu zdjęć połączonych w jeden obraz (jeden obok drugiego).

### 3.2.2 Informacje wyjściowe

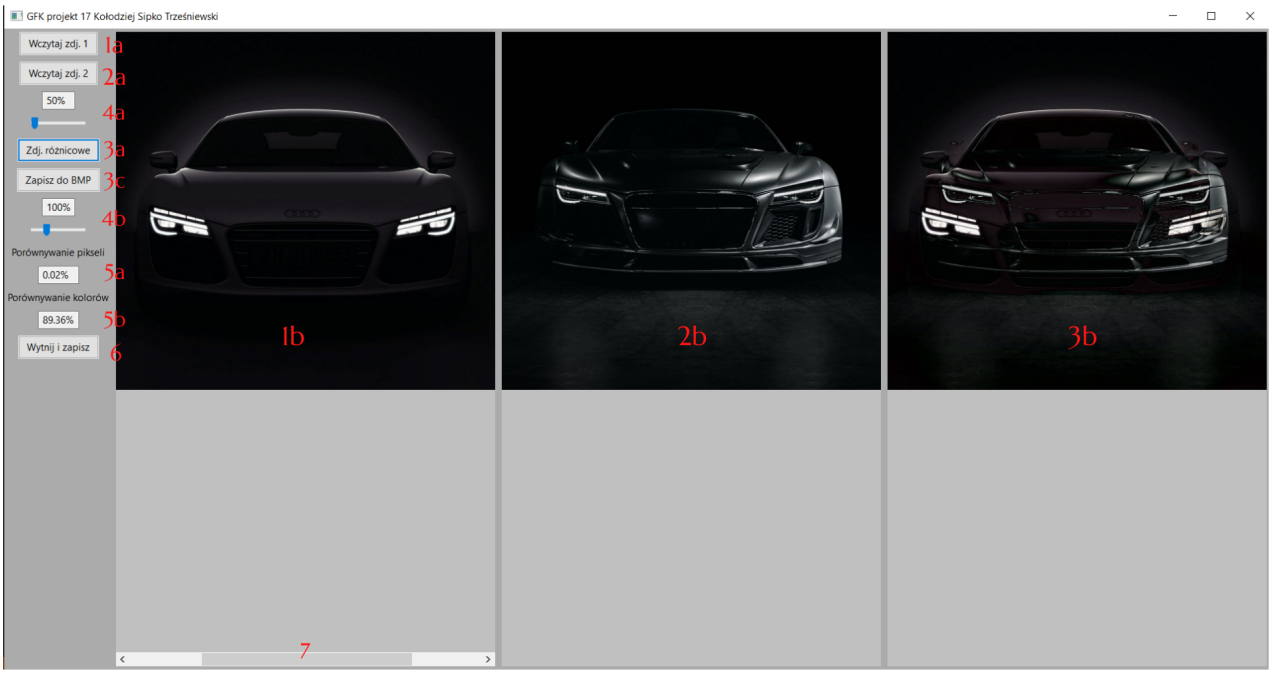
Na ekranie przede wszystkim wyświetlane są oba obrazy wejściowe. Przycisk “Zdj. różnicowe” wywołuje utworzenie i wyświetlanie na ekranie (obok obrazów początkowych) zdjęcia różnicowego. Dodatkowo wyświetlane są wtedy dwie wartości:

- “Porównywanie pikseli” - ile procent pikseli obu obrazów jest identyczne
- “Porównywanie kolorów” - w jakim stopniu zdjęcia są do siebie podobne (dokładne wytłumaczenie wyniku i działania algorytmu w podpunkcie 5.3.2).

## 3.3 Zdefiniowanie struktur danych

Obrazy przekazywane do aplikacji oraz zdjęcie różnicowe (oryginały i kopie) są przechowywane w obiektach klasy wxImage.
Wartości RGB każdego piksela tych obrazów pobrane metodą wxImage::GetData() zapisywane są do tablicy typu unsigned char.
Obszar zaznaczania wycinka obrazów widoczny dla użytkownika jest dzięki klasie wxRect (pochodna klasa wxCore).

## 3.4 Specyfikacja interfejsu użytkownika




*Rysunek 1. : Widok okna aplikacji*

1. Obsługa zdjęcia nr. 1
    a) Przycisk wczytywania zdjęcia (po wciśnięciu wyskakuje okno wyboru zdjęcia)
    b) Wyświetlone zdjęcie nr. 1

2. Obsługa zdjęcia nr. 2
    a) Przycisk wczytywania zdjęcia (po wciśnięciu wyskakuje okno wyboru zdjęcia)
    b) Wyświetlone zdjęcie nr. 2

3. Obsługa zdjęcia różnicowego
    a) Przycisk generowania zdjęcia różnicowego (po wciśnięciu wyświetla się w miejscu 3b)
    b) Wyświetlone zdjęcie różnicowe

c) Przycisk zapisu zdjęcia różnicowego (po wciśnięciu mamy możliwość zapisania go w wybranym miejscu w pamięci w formacie .bmp).

4. Powiększanie zdjęć
    a) Suwak z pozycjami 50%, 100%, 200%, 400%
    b) Suwak powiększający o dowolną wartość z przedziału 1%-400%

5. Wartości porównawcze obrazów
    a) Wartość procentowa ile odpowiadających pikseli z obrazów 1. i 2. jest takie same (równe wartości RGB)
    b) Wartość procentowa podobieństwa kolorystycznego obrazów 1. i 2.

6. Wycinanie wybranego obszaru z obu zdjęć
   Po wciśnięciu kursor myszki się zmienia, a po zaznaczeniu obszaru pojawia się okno zapisu wycinka.

7. Przesuwanie synchroniczne obrazów.
   Suwaki (poziomy/pionowy) pojawiają się, gdy obraz jest większy niż przewiduje okno wyświetlania (1b,2b,3b).

## 3.5 Wyodrębnienie i zdefiniowanie zadań

Proces tworzenia projektu można podzielić na 4 główne moduły:

1) Tworzenie GUI przy pomocy aplikacji wxFormBuilder 3.9.0
    a) zaprojektowanie wyglądu okna aplikacji
    b) zapewnienie dobrej widoczności i łatwości w obsłudze aplikacji
    c) przekazania kodu wygenerowanego przez wxFormBuilder do projektu
2) Zaimplementowanie funkcji przyjmujących i tworzących pliki
    a) Zapewnienie poprawnego wyświetlania plików wprowadzanych do aplikacji (suwaki przesuwania, dopasowanie rozmiaru drugiego obrazu do pierwszego, funkcja Repaint)
    b) Możliwość zapisu zdjęcia różnicowego
    c) Tworzenie i zapisywanie wycinka obu zdjęć
3) Zaimplementowanie funkcji manipulujących danymi plików
    a) Funkcja tworząca zdjęcie różnicowe
    b) Funkcje przybliżenia obrazów
4) Zaimplementowanie wyświetlania informacji dodatkowych (połączone z tworzeniem zdjęcia różnicowego)
    a) Obliczenie i wyświetlenie procentowej wartości identycznych pikseli
    b) Obliczenie i wyświetlenie procentowej wartości “zbieżności kolorów”

## 3.6 Wybór narzędzi programistycznych

Do wykonania projektu wykorzystano bibliotekę wxWidgets w wersji 3.0.5, gdyż pozwala ona na tworzenie GUI w przystępny sposób. Korzystaliśmy z niej także w trakcie zajęć laboratoryjnych, więc mieliśmy już doświadczenie w korzystaniu z niej. Do kompilacji użyliśmy kompilatora MSVC w wersji 14.29. Projekt zrealizowaliśmy za pomocą 64-bitowej wersji programu Visual Studio 2019.

W realizacji projektu pomogło nam również repozytorium założone w serwisie GitHub. Służyło nam ono do łatwej wymiany plików między sobą oraz umożliwiło edycję wszystkim członkom zespołu projektowego.

# 4 Podział pracy i analiza czasowa

| | Mirosław Kołodziej | Paweł Sipko | Jakub Trześniewski |
|---|---|---|---|
| Repozytorium | X | X | 30 min |
| Planowanie | 2h | 2h | 2h |
| wxFormBuilder | 15 min | 45 min | 15min |
| Wymagania podstawowe | 4h | 9h | 2h |
| Pierwsze testy i poprawki | 30 min | 45 min | 30 min |
| Wymagania rozszerzone | 8h | 5h | 3h |
| Ostateczne testy i poprawki | 30 min | 30 min | 30 min |
| Dokumentacja | 2h | 1h | 8h |
| Kosmetyka kodu i przygotowanie do wysłania | 30 min | 30 min | 30 min |

Warto zauważyć, że podane wyżej wartości są przybliżone oraz większość prac wykonywaliśmy wspólnie.

# 5 Opracowanie i opis niezbędnych algorytmów

## 5.1 Pobieranie danych z obrazów i tworzenie zdjęcia różnicowego

Możliwość manipulowania zawartością obrazów została zapewniona przez funkcję wxImage::GetData(). Jej działanie polega na przechodzeniu przez obraz piksel po pikselu z góry na dół z lewej do prawej (tj. pierwszy piksel pierwszego rzędu, drugi piksel pierwszego rzędu itp.). Z każdego piksela pobierane są wartości R(red), G(green) i B(blue). Jako dane wyjściowe otrzymujemy wskaźnik na tablicę typu unsigned char wielkości 3*n (n-ilość pikseli danego obrazu) zapisaną w sposób $[R_1,G_1,B_1,R_2,G_2,B_2,\ldots,R_n,G_n,B_n]$.

Korzystając z powyżej opisanej tabeli danych byliśmy w stanie utworzyć zdjęcie różnicowe. Początkowo zdjęcie różnicowe jest kopią zdjęcia 1., następnie przechodząc przez jego tablicę imgData (nazwa własna w kodzie ad. pkt 6) nadpisujemy każdy element tablicy wartością bezwzględną z różnicy odpowiadających elementów tablic imgData dla obu obrazów wejściowych.

## 5.2 Przybliżanie obrazów

W celu skutecznego porównania obrazów, w momencie dodawania drugiego obrazu (niezależnie w jakiej kolejności) zdjęcie 2. jest skalowane do wymiarów zdjęcia 1.
Dzięki temu jesteśmy w stanie rzetelnie analizować fragmenty dwóch obrazów jednocześnie. Dodatkowo dwa suwaki powiększeń pozwalają na dokładne obejrzenie mniejszych fragmentów obrazu. Za zmiany w wielkości obserwowanych obrazów odpowiada funkcja wxImage::Scale skalująca obiekt wxImage do podanych w kodzie rozmiarów (podawane są dwa mnożniki: ilukrotnie ma być zwiększona/zmniejszona szerokość oraz ilukrotnie szerokość obrazu).

### 5.2.1 Przybliżenia 50%, 100%, 200%, 400%

Wartość mnożników podawanych do metody Scale wyliczana jest ze wzoru

$$w_{nowe} = 50 * 2^{tmp}/100 * w_{stare}$$

$$h_{nowe} = 50 * 2^{tmp}/100 * h_{stare}$$

gdzie *w*-szerokość obrazu, *h*-wysokość obrazu, *tmp*-pozycja suwaka w danej chwili (suwak ma 4 pozycje : 0,1,2,3; 1-wartość domyślna).

### 5.2.2 Przybliżenia o wartość z przedziału [1%-400%]

Wartość mnożników podawanych do metody Scale wyliczana jest ze wzoru

$$w_{nowe} = tmp/100 * w_{stare}$$

$$h_{nowe} = tmp/100 * h_{stare}$$

gdzie *w*-szerokość obrazu, *h*-wysokość obrazu, *tmp*-pozycja suwaka w danej chwili (suwak ma 400 pozycji : 1,2,3,...,399,400; 100-wartość domyślna).

## 5.3 Wartości porównawcze obrazów

Zdjęcie różnicowe jest ciekawym sposobem na podkreślenie różnic pozornie podobnych do siebie obrazów. Postanowiliśmy jednak aby do graficznej interpretacji tych różnic dodać podejście stricte matematyczne. Przy pomocy prostych zaimplementowanych przez nas algorytmów użytkownik otrzymuje do przeanalizowania dwie dodatkowe wartości.

### 5.3.1 Porównywanie pikseli

Pierwszą z nich jest procentowa zgodność pikseli obu obrazów. Implementacja i zrozumienie tej wartości jest niezmiernie prosta. Przechodząc przez zdjęcie różnicowe piksel po pikselu sprawdzamy, ile pikseli jest czarnych (tj R,G,B=*0*,*0*,*0*). Wystąpienie takiego piksela oznacza iż odpowiadające sobie piksele w porównywanych zdjęciach są identyczne. Dzieląc ilość "czarnych" pikseli przez ilość wszystkich pikseli, a następnie mnożąc przez 100 otrzymujemy wartość procentową identycznych pikseli.

### 5.3.2 Porównywanie kolorów

Wartość zbieżności kolorów jest nieznacznie cięższa w implementacji, jednak dodaje pierwiastek człowieczeństwa do porównania pozornie podobnych obrazów.
Znów analizujemy piksele zdjęcia różnicowego. Sumujemy wartości "podobieństwa" (nazwa własna w kodzie) każdego piksela tego zdjęcia.

$$podobienstwo = \sum_{i=0}^{n} R_{\%} \cdot G_{\%} \cdot B_{\%}$$

gdzie:
*n*-liczba pikseli zdjęcia różnicowego
$R_{\%}$,$G_{\%}$,$B_{\%}$- wartości podobieństwa kolorów
[255 =0% (piksele maksymalnie odległe); 0=100%(piksele identyczne)]

# 6 Kodowanie

*void load_button1OnButtonClick(wxCommandEvent& event);*
Funkcja wczytująca pierwsze zdjęcie.
*void load_button2OnButtonClick(wxCommandEvent& event);*
Funkcja wczytująca drugie zdjęcie.
*void m_slider3OnScroll(wxScrollEvent& event);*
Funkcja nadająca wybrane powiększenie.
*void m_button8OnButtonClick(wxCommandEvent& event);*
Funkcja tworząca zdjęcie różnicowe.

*void save_button5OnButtonClick(wxCommandEvent& event);*

Funkcja zapisująca zdjęcie różnicowe.

*void m_scrolledWindow1OnUpdateUI(wxUpdateUIEvent& event);*
*void m_scrolledWindow2OnUpdateUI(wxUpdateUIEvent& event);*
*void m_scrolledWindow21OnUpdateUI(wxUpdateUIEvent& event);*

Funkcje obsługujące synchroniczne przewijanie obrazów.

*void Repaint(wxScrolledWindow* sw, wxImage& img);*

Funkcja odświeżająca obraz.

*void m_slider4OnScroll(wxScrollEvent& event);*

Funkcja nadająca dowolne powiększenie.

*void m_scrolledWindow1OnMouseEvents(wxMouseEvent& event);*

Funkcja obsługująca wycinanie.

*void save_button6OnButtonClick(wxCommandEvent& event);*

Funkcja umożliwiająca wycinanie (przy pomocy powyższej funkcji) i zapisywanie uzyskanego obrazu.

*MyProject1Frame(wxWindow* parent);*

Konstruktor.

*wxImage _image1;*
*wxImage _image2;*
*wxImage _image3;*

Zmienne przechowujące oryginały obrazów.

*wxImage _cpy1;*
*wxImage _cpy2;*
*wxImage _cpy3;*

Zmienne przechowujące kopie obrazów.

*wxImage _image2s;*

Zmienna przechowująca przeskalowany drugi obraz.

*wxCursor _kursor;*

Zmienna przechowująca aktualny styl kursora.

*unsigned _pow;*
*int _poz;*

Zmienne przechowujące aktualne pozycje suwaków.

*double _w = 0, _h = 0;*

Zmienne przechowujące wymiary pierwszego obrazu.

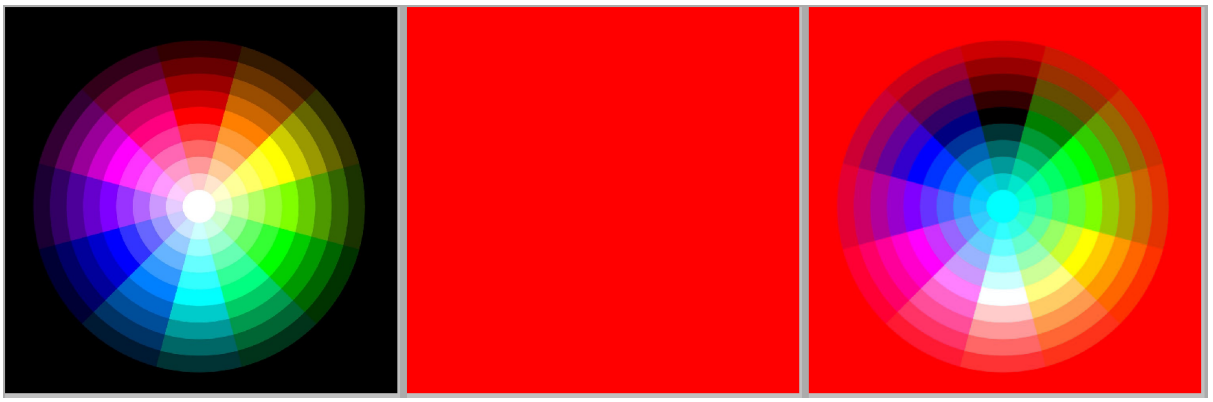
*int px1, py1, px2, py2;*

Zmienne przechowujące współrzędne prostokąta

*bool _klik = false;*

Zmienna tryb kursora.

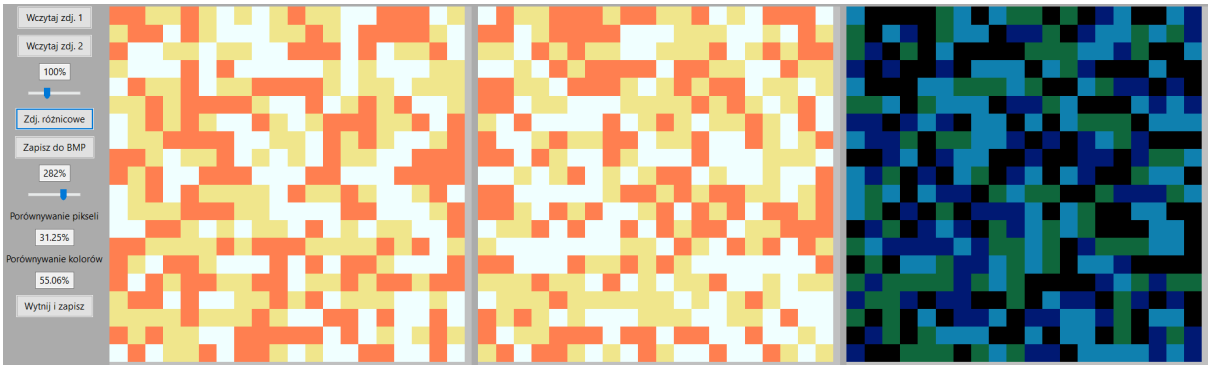
# 7 Testowanie



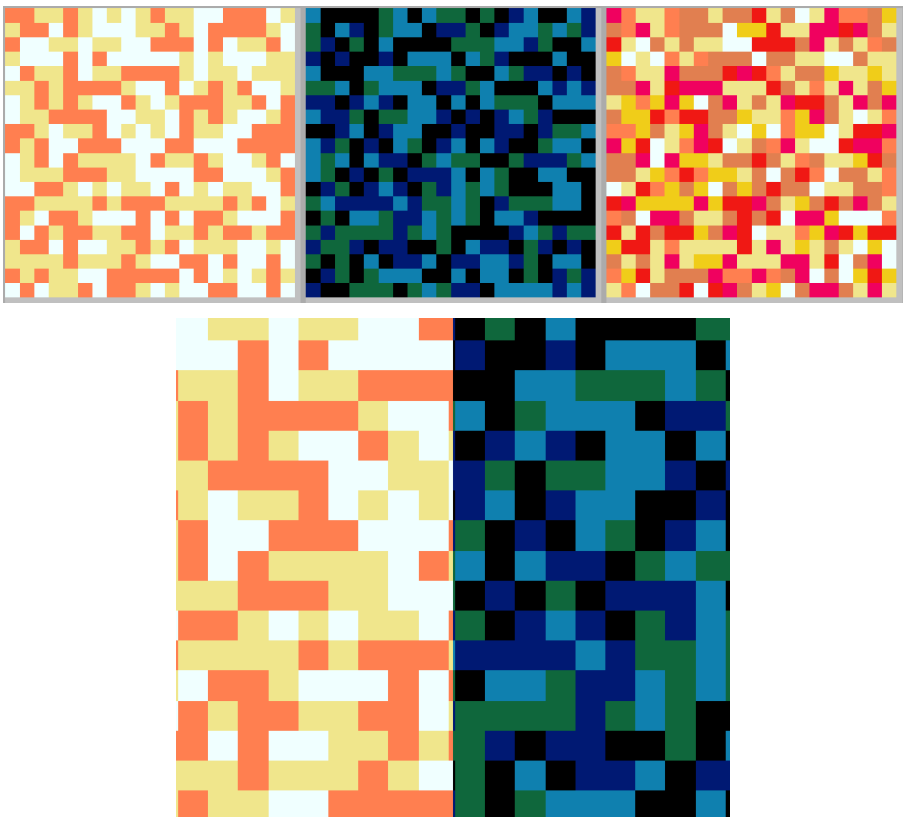
*Rysunek 2. : Test algorytmu odejmowania kolorów*


*Rysunek 3. : Test porównania zdjęcia oryginalnego z zaszumionym*

Jak widać na rysunku 2. nałożenie “szumu” na zdjęcie znacznie zmienia ilość identycznych pikseli na co nie zwrócą uwagi ludzkie zmysły.




*Rysunek 4. : Test porównania obrazów podobnych kolorystycznie*



*Rysunki 5-6. : Test wycinania fragmentów z obrazków*

# 8 Wdrożenie, raport i wnioski

Udało nam się zrealizować wszystkie zagadnienia podstawowe oraz zaproponowane zagadnienia rozszerzone. Dodatkowo poszerzyliśmy funkcjonalność aplikacji o dwa autorskie parametry porównawcze.

Jedyną zauważoną przez nas niedoskonałością działania programu jest “migający” prostokąt zaznaczający obszar wycinania (podpunkt 3.4.6). Prawdopodobnym rozwiązaniem problemu może być zastosowanie podwójnego buforowania. Zagadnienie to nie było jednak przećwiczone na zajęciach laboratoryjnych. Po uprzednim przedyskutowaniu z prowadzącym, nie podjęliśmy się implementacji tej metody

Głównym wnioskiem wyciągniętym z testów przeprowadzonych przez nas w czasie tworzenia projektu jest to jak małe lub nawet niezauważalne dla użytkownika zmiany w obrazie mogą być niezmiernie znaczące dla komputera porównującego piksel po pikselu.

# 9 Źródła

https://docs.wxwidgets.org/3.0/

https://www.vectorstock.com/royalty-free-vector/rgb-color-wheel-from-dark-to-light-on-black-vector-35310540

https://pinetools.com/random-bitmap-generator

https://onlinebitmaptools.com/draw-random-bmp

Piękne auta i jeszcze piękniejszy Wydział

http://www.fis.agh.edu.pl/~malinowski/GFK/files/17.pdf