KOREKTY BARWNE

Tomasz Rajchel
Jakub Salamon
Jędrzej Słomiński

Projekt nr 11
Czerwiec 2019

Spis Treści

1 Tytuł i autorzy projektu....2

2 Opis projektu....2

3 Założenia wstępne przyjęte w realizacji projektu....2

4 Analiza projektu....3

Specyfikacja danych wejściowych....3
Opis oczekiwanych danych wyjściowych....3
Zdefiniowanie struktur danych....3
Specyfikacja interfejsu użytkownika....3
Wyodrębnienie i zdefiniowanie zadań....4
Decyzja o wyborze narzędzi programistycznych....5

5 Podział pracy i analiza czasowa....5

6 Opracowanie i opis niezbędnych algorytmów....6

Budowanie sześciokąta barw....6
Algorytm korekcji barwnej obrazka....6
Algorytm korekty na kanałach jasności, nasycenia i kontrastu....7

7 Kodowanie....7

8 Testowanie....8

9 Wdrożenie, raport i wnioski....8

10 Szczegółowy opis klas, funkcji i zmiennych....8

1 Tytuł i autorzy projektu

Projekt nosi tytuł „Korekty Barwne” nr 11. Autorami projektu są 3 osoby:

Tomasz Rajchel, Jakub Salamon, Jędrzej Słomiński.

2 Opis projektu

Celem projektu jest napisanie programu umożliwiającego wykonywanie selektywnych korekt barwnych wczytywanych plików graficznych.
W wersji rozszerzonej istnieje możliwość wykonania korekty również w kanałach jasności i nasycenia i kontrastu. Korekty te są wspomagane rysunkami odpowiednich histogramów. Oprócz tego wyświetlany jest również sześciokąt zmodyfikowany ustawionymi korektami.

3 Założenia wstępne przyjęte w realizacji projektu

Program będzie posiadał poniższe opcje:

- Obsługiwane formaty
  Wejściowy i wyjściowy plik będzie mógł być w formacie jpg, jpeg, bmp lub png.
- Zapis/odczyt z pliku

- Wizualizacja modyfikowanego koloru
  Kolor wybrany na sześciokącie barw będzie na nim wskazywany. Także modyfikowany, wskazany na oryginalnym obrazku kolor będzie wyświetlany.
- Regulacja współczynnika proporcjonalności modyfikowanej barwy przy pomocy suwaka

Wymagania rozszerzone:

- korekta w kanałach jasności i nasycenia
- histogramy oryginalnego i zmodyfikowanego obrazka
- wyświetlanie sześciokąta zmodyfikowanego ustawionymi korektami

Dodatki:

- korekta w kanale kontrastu
- wyświetlanie dokonywanych operacji na opcjonalnym pasku statusu

4 Analiza projektu

Specyfikacja danych wejściowych

W naszym programie będziemy pracować na plikach graficznych. Będą one w formacie jpeg, jpg, bmp lub png. Obrazki mogą zostać wczytane poprzez przystosowane do tego okno.

Opis oczekiwanych danych wyjściowych

Danymi wyjściowymi są obrazy ze zmodyfikowanymi kolorami. Można je zapisać w formatach: *.bmp,*.jpeg,*.jpg,*.png

Obrazy wyjściowe mają takie same wymiary jak obrazy wejściowe.

Zdefiniowanie struktur danych

Podczas działania programu obrazy są przechowywane w:

- obiekcie klasy *wxImage*
- obiekcie klasy *wxBitmap*
- Zwykłej tablicy typu *unsigned char* – wartości pikseli 0-255.

Program w zależności od potrzeb dokonuje konwersji między tymi formatami.

Specyfikacja interfejsu użytkownika

Interfejs użytkownika składa się z głównego okna w którym wyświetlane są oba obrazy – oryginalny po lewej i zmodyfikowany po prawej. Jeżeli obraz jest zbyt duży by zmieścić się w głównym oknie to można skorzystać z pionowych i poziomych pasków przewijania.

Po prawej stronie głównego okna znajduje się panel z zakładkami wyświetlający sześciokąt i histogramy kolorów.

Sześciokąt – korekty barwne

Aby wybrać kolor do korekty barwnej wystarczy kliknąć dwa razy na dowolny piksel obrazu oryginalnego. Następnie na sześciokącie można zaznaczyć nowy kolor na który zostaną zamienione wszystkie punkty w starym kolorze znajdujące się na obrazku. Suwak pod sześciokątem reguluje siłę zmian 0-100%. Korekty dokonujemy klikając przycisk poniżej.

Histogramy

W drugiej zakładce znajdują się histogramy kolorów. Pierwszy z nich to histogram kumulacyjny, kolejne odpowiadają składowym R, G, B obrazu. Histogramy są skalowane względem wartości występującej najczęściej. Po wygenerowaniu histogramów, uaktualniają się one automatycznie wraz ze wprowadzanymi zmianami.

Pasek menu

Pasek menu w lewym górnym rogu zawiera zakładki ‘file’ i ‘view’. Korzystając z pierwszej z nich można wczytywać i zapisywać pliki. W zakładce ‘view’ można włączyć/wyłączyć okno dialogowe z rozszerzonymi korektami i pasek statusu.

Pasek statusu

Na samym dole głównego okna znajduje się pasek statusu, którym program komunikuje swoje akcje.

Wyodrębnienie i zdefiniowanie zadań

Opisane w punkcie 5 Podział pracy i analiza czasowa

Decyzja o wyborze narzędzi programistycznych

Będziemy korzystać z biblioteki wxWidgets. Do stworzenia GUI posłużymy się programem wxFormBuilder. Zapewni to nam szybsze generowanie kodu i mniejszą ilość błędów niż gdybyśmy pisali GUI ręcznię.

IDE - ze względu na dużo funkcjonalności i dobrą znajomość zdecydowaliśmy się na Microsoft Visual Studio 2017. Skorzystamy z kompilatora Microsoft C++ compiler.

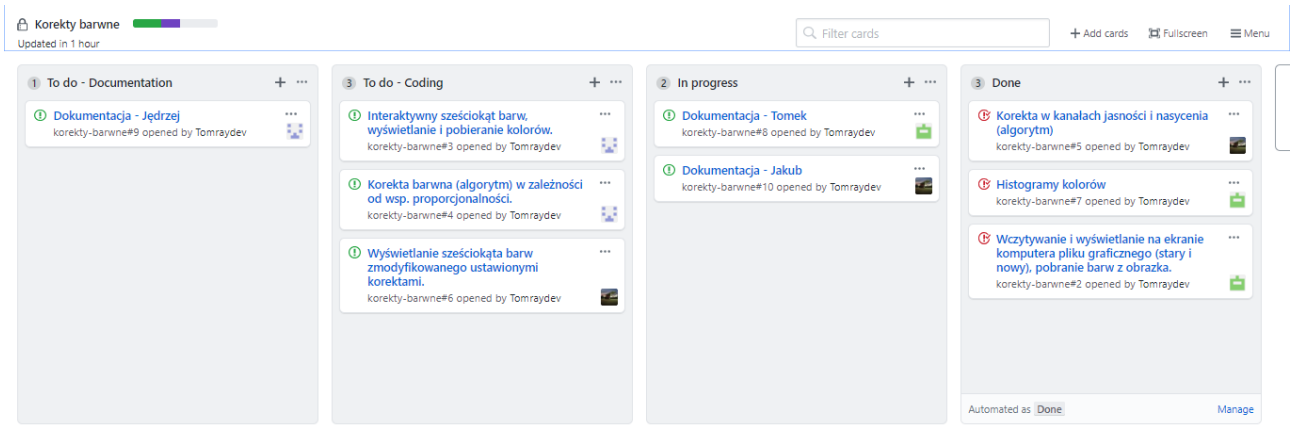
Cały projekt wykonujemy na systemie operacyjnym Microsoft Windows 10.

5 Podział pracy i analiza czasowa

Projekt podzieliliśmy na mniejsze zadania. Na pierwszym spotkaniu porozdzielaliśmy je między siebie według punktacji z opisu projektu, własnej estymacji czasu zadań i preferencji.

Osoba	Zadanie
Tomasz Rajchel	Podstawowy interfejs użytkownika. Wczytywanie i wyświetlanie na ekranie komputera pliku graficznego (stary i nowy).
	Histogramy kolorów
	Dokumentacja projektu
Jakub Salamon	Korekta w kanałach jasności i nasycenia (algorytm)
	Wyświetlanie sześciokąta barw zmodyfikowanego ustawionymi korektami.
	Dokumentacja – założenia wstępne, specyfikacja danych wejściowych, kodowanie, opis swoich algorytmów
Jędrzej Słomiński	Interaktywny sześciokąt barw, wyświetlanie i pobieranie kolorów.
	Korekta barwna (algorytm) w zależności od wsp. proporcjonalności.
	Dokumentacja – opis swoich algorytmów

Do śledzenia postępu skorzystaliśmy z Github Project Boards.



6 Opracowanie i opis niezbędnych algorytmów

Budowanie sześciokąta barw

Sześciokąt barw składa się z odpowiednio przekształconych kwadratów, które są definiowane w sposób następujący:

kwadratu mającego stałą składową R = 255 i odpowiednio zmienne składowe G i B od wartości 0 do wartości 255.

kwadratu mającego stałą składową G = 255 i odpowiednio zmienne składowe R i B od 0 od wartości 0 do wartości 255.

kwadratu mającego stałą składową B = 255 i odpowiednio zmienne składowe R i G od 0 od wartości 0 do wartości 255.

Na koniec należy kwadraty odpowiednio przeskalować i dopasować w taki sposób, aby tworzyły sześciokąt, korzystając m.in. z możliwości biblioteki graficznej (metody Scale(), a także Rotate()).

Algorytm korekcji barwnej obrazka

Zakładamy, że pobieramy z obrazka pewien kolor X o składowych X_R, X_G, X_B. Każda składowa pobranego koloru ma wartości z przedziału od 0 do 255.

Następnie wybieramy na sześciokącie barw pewien kolor Y o składowych Y_R, Y_G, Y_B. Każda składowa pobranego koloru ma również wartości z przedziału od 0 do 255.

Algorytm przechodzi po wszystkich pikselach załadowanego obrazu. Zakładamy, że aktualnie przetwarzany piksel ma kolor Z o składowych Z_R, Z_G, Z_B (również o wartościach od 0 do 255). Rozważamy dwa przypadki:

- **Z == X (przetwarzany piksel jest równy pikselowi uprzednio pobranemu z obrazka)**
  Wtedy następuje podmiana aktualnego piksela Z na ten, który został zaznaczony na sześciokącie barw (czyli Z = Y).
- **każda inna możliwość niż powyższa**
  Wtedy każda nowa składowa aktualnego piksela stanowi (100-P) procent ze składowej Z (P to parametr odpowiadający za siłę korekcji), a pozostałą część koloru stanowi P procent wartości odpowiedniej składowej koloru na który zamieniamy :
  $Z_R = Z_R * (100-P) + Y_R * P$
  $Z_G = Z_G * (100-P) + Y_G * P$
  $Z_B = Z_B * (100-P) + Y_B * P$

Algorytm korekty na kanałach jasności, nasycenia i kontrastu

Przy pomocy specjalnego okna ustawiamy wartość jasności, nasycenia i kontrastu.

Dla kanału jasności zmieniamy wartość R, G oraz B. Dla każdego z pikseli dodajemy wartość odczytaną z nastaw. Przedział nastaw to [−255, 255]. Jeżeli wartość dla któregoś z kanałów R, G, B przekroczy wartości graniczne, tj. 0 lub 255 ustawiamy dany kanał na wartość graniczną.

Dla kanału nasycenia wartością nastaw jest przedział [−1, 1]. Algorytm polega na zamianie RGB na HSV i dokonaniu korekty w kanale saturation (nasycenia). Po tej operacji zmieniamy HSV na RGB.

Dla kanału kontrastu przedział nastaw to [−255, 255]. Następnie wartość nastawy jest przekształcana według wzoru:

$$kontrast = \frac{value + 256}{257 - value}$$

Wartość każdego z kanałów RGB jest obliczana następująco:

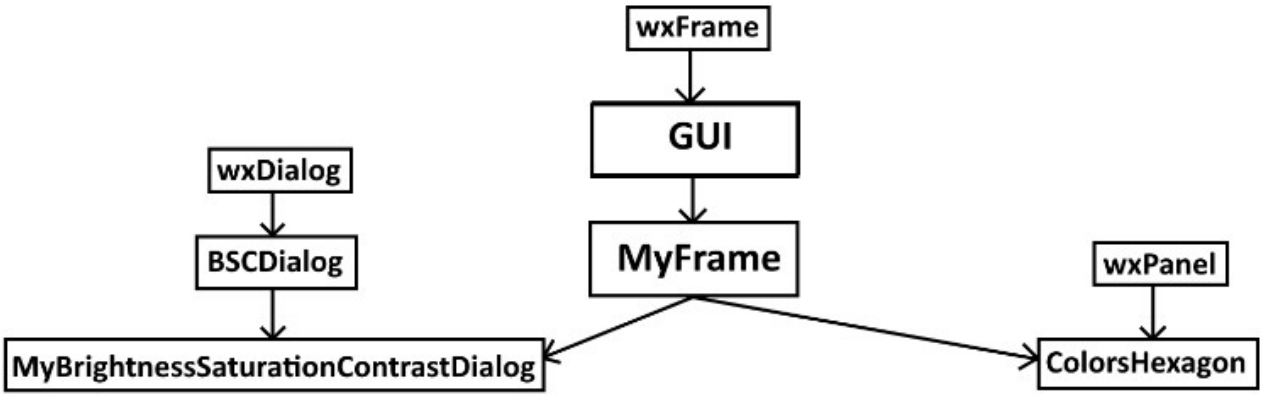
$$wartość = (kanał - 122.5) \cdot kontrast + 122.5$$

Jeżeli wartość ta przekracza wartości graniczne 0 lub 255 jest ustawiana na wartość graniczną oraz przypisywana do danego kanału.

Operacje te są wykonywane dla każdego z trzech kanałów każdego z pikseli.

Sześciokąt zmodyfikowany ustawionymi korektami powstaje tak samo jako sześciokąt służący do zmiany barw. Są jednak na nim wykonywane operacje zmieniające każdy z jego pikseli za pomocą wcześniej przedstawionych algorytmów.

7 Kodowanie



Zależności między klasami używanymi w programie

Szczegółowy opis klas, funkcji i zmiennych znajduje się na końcu tego dokumentu w rozdziale 10.

8 Testowanie

Program testowaliśmy manualnie, sprawdzając prawidłowe działanie każdej funkcjonalności bezpośrednio po jej wprowadzeniu. Skorzystaliśmy z zestawu plików testowych z dużą różnorodności kolorów.

9 Wdrożenie, raport i wnioski

Co się udało:

Program prawidłowo wczytuje i zapisuje dane.

Sześciokąt zmodyfikowany ustawieniami kolorów działa poprawnie. Wszystkie możliwe do wprowadzenia korekcje mają wpływ na jego wygląd. Dodatkową opcją jest możliwość włączenia lub wyłączenia jego wyświetlania. Opcja ta może być przydatna przy użyciu programu na słabych komputerach.
Wydaje się, że algorytm rysowania sześciokąta barw działa poprawnie. Na sześciokącie zostaje zaznaczony dowolny kolor wybrany przez użytkownika z obrazka.
Algorytm korekcji barwnej działa przyzwoicie. Pozwala on na odpowiednią zmianę koloru konkretnego piksela na inny. Odpowiednio zostają też zmodyfikowane inne piksele obrazu.

Ustawienia korekcji kolorów na kanałach jasności, nasycenia i kontrastu zostały wprowadzone w pełni poprawnie. Wszystkie opcje możemy ustawiać niezależnie od siebie. Ustawienia te nie wpływają negatywnie na pozostałe funkcjonalności. Wprowadzone zmiany możemy w każdej chwili zresetować, co jest bardzo przydatną opcją.

Co nie:

Niestety nie udało się stworzyć poprawnego sześciokąta. Jest on wydłużony. Wynikło to z konieczności dopasowania górnej części sześciokąta do bocznych, która została wykonana z użyciem funkcji z biblioteki graficznej (a nie po prostu narysowana tak jak boczne części).

Co można poprawić:

Przydałaby się możliwość Skalowania obrazów.

Algorytm korekty barwnej można przyśpieszyć stosując obliczenia równoległe.

W kwestii pracy zespołowej, w celu poprawy jakości kodu można by wykonywać 'code review' wzajemnie przez poszczególnych członków zespołu.

10 Szczegółowy opis klas, funkcji i zmiennych

Klasy:

- **GUI** - klasa bazowa okna aplikacji

  Elementy:

  - **m_menuBarTop** - pasek menu
  - **m_menuFile** - opcje paska menu pozwalająca na wczytanie i zapis plików

- **m_menuView** - podmenu paska menu, odpowiadająca za wyświetlanie dodatkowych opcjonalnych okien
- **m_scrolledWindow1** - miejsce wyświetlana oryginalnego obrazka
- **m_scrolledWindow2** - miejsce wyświetlania nowego obrazka
- **m_notebook1** - okno pozwalające na wybór wyświetlanego panelu sześciokątów
- **m_panel _hexagon** - panel, w którym są wyświetlane panele sześciokątów kolorów
- **m_notebook3** -
- **m_panel_hexagon_color** - panel, w którym wyświetlany jest sześciokąt kolorów, służący do ich korekcji
- **m_panel_hexagon_mod** - panel, w którym wyświetlany jest sześciokąt zmodyfikowany ustawieniami kolorów
- **m_panel_hexagon_sizer**
- **m_panel_histograms** - panel, zawierający panele histogramów przed i po korekcjach
- **m_notebook4** - okno pozwalające na wybór wyświetlanych histogramów
- **M_panel_hist_old** - panel, w którym wyświetlane są histogramy przed korekcjami
- **m_panel_hist_new** - panel, w którym wyświetlane są histogramy po korekcjach
- **m_buttonHistogram** - przycisk generujący histogramy
- **m_statusBar1** - pasek wyświetlający status aplikacji

MyFrame - klasa pochodna klasy GUI dostosowująca ją do naszych potrzeb
Metody:

- **m_fileOpenOnMenuSelection(wxCommandEvent&)** - metoda pozwalająca na wczytanie pliku graficznego do programu
- **m_fileSaveAsOnMenuSelection(wxCommandEvent&)** - metoda pozwalająca na zapis powstałego obrazu do pliku
- **GUIOnUpdateUI(wxUpdateUIEvent&)** - metoda aktualizująca GUI
- **m_ViewStatusBarOnMenuSelection(wxCommandEvent&)** - metoda wywołująca okno dialogowe ustawień jasności, nasycenia i kontrastu
- **m_scrolledWindow1OnLeftDClick(wxMouseEvent&)** - metoda wybierająca kolor bazowy do sześciokąta
- **m_clickHexagonButton(wxCommandEvent&)**
- **changePixelsAlgo(bool)** - metoda zmieniająca nowy obraz na podstawie ustawień sześciokąta kolorów
- **changePropSlider(wxScrollEvent&)** - metoda odpowiadająca za właściwe wyświetlanie opisu paska ustawień sześciokąta kolorów
- **m_clickModHexagonButton(wxCommandEvent&)** - metoda wywołująca metodę changePixelAlgo(bool) po kliknięciu na przycisk pod sześciokątem kolorów
- **m_clickResetAllButton(wxCommandEvent&)** - metoda resetująca wszystkie ustawiania. Wywołuje metodę ReselAll z klasy MyBrightnessSaturationContrastDialog
- **generateModHexagon()** - metoda generująca sześciokąt zmodyfikowany ustawieniami kolorów

- **calculateModHexagon(int*)** - metoda obliczająca wartości pikseli w sześciokącie zmodyfikowanym ustawieniami kolorów
- **m_ViewBrightnessSaturationContrastWindowOnMenuSelection(wxCommandEvent&)** - metoda wyświetlająca okno dialogowe ustawień jasności, nasycenia i kontrastu
- **setBrightness(int, int, int, bool, bool)** - metoda ustawiająca jasności obrazu
- **setSaturation(int, int, int, bool, bool)** - metoda ustawiająca nasycenia obrazu
- **setContrast(int, int, int, bool, bool)** - metoda ustawiająca kontrast obrazu
- **m_buttonHistogramOnButtonClick(wxCommandEvent&)** - metoda generująca histogramy po naciśnięciu przycisku
- **generate_hist_img(wxImage &,wxBitmap &, int, int, int, int)** - metoda generująca histogram dla danego obrazu
- **paintHistograms()** - metoda rysująca histogramy
- **calculateHistograms(wxImage &, int, int, int, int)** - metoda wyliczająca histogramy kolorów
- **updateHistogram()** - metoda aktualizująca histogramy po wprowadzeniu korekt na obrazie
- **Repaint()** - metoda odświeżająca elementy aplikacji

Elementy:

- **imgOld** - oryginalny obraz
- **imgToBSC** - pomocniczy obraz
- **imgNew** - nowy obraz
- **bitMapOld** - bitmapa oryginalnego obrazu
- **bitMapNew** - bitmapa nowego obrazu
-
- **imgHistogramRGB** - obraz histogramu wszystkich kolorów z oryginalnego obrazu
- **imgHistogramR** - obraz histogramu czerwonego koloru z oryginalnego obrazu
- **imgHistogramG** - obraz histogramu zielonego koloru z oryginalnego obrazu
- **imgHistogramB** - obraz histogramu niebieskiego koloru z oryginalnego obrazu
- **bitMapHistogramRGB** - bitmapa histogramu wszystkich kolorów z oryginalnego obrazu
- **bitMapHistogramR** - bitmapa histogramu czerwonego koloru z oryginalnego obrazu
- **bitMapHistogramG** - bitmapa histogramu zielonego koloru z oryginalnego obrazu
- **bitMapHistogramB** - bitmapa histogramu niebieskiego koloru z oryginalnego obrazu
- **imgHistogramRGB_n** - obraz histogramu wszystkich kolorów z nowego obrazu
- **imgHistogramR_n** - obraz histogramu czerwonego koloru z nowego obrazu
- **imgHistogramG_n** - obraz histogramu zielonego koloru z nowego obrazu
- **imgHistogramB_n** - obraz histogramu niebieskiego koloru z nowego obrazu
- **bitMapHistogramRGB_n** - bitmapa histogramu wszystkich kolorów z nowego obrazu
- **bitMapHistogramR_n** - bitmapa histogramu czerwonego koloru z nowego obrazu
- **bitMapHistogramG_n** - bitmapa histogramu zielonego koloru z nowego obrazu
- **bitMapHistogramB_n** - bitmapa histogramu niebieskiego koloru z nowego obrazu
- **resetAllButton** - przycisk resetujący wszystkie ustawienia

- **hexagon** - obiekt klasy ColorsHexagon odpowiadający za działanie sześciokąta kolorów
- **brightnessDialog** - obiet klasy MyBrightnessSaturationDialog zawierający okno dialogowe służące do zmiany ustawień jasności, nasycenia i kontrastu
- **colorFromImageTxt** - tekst wyświetlany jako wybrany na oryginalnym obrazie kolor do edycji poprzez sześciokąt kolorów
- **colorFromHexagonTxt** - tekst wyświetlany jako wybrany na sześciokącie kolorów
- **hexagonButton** - przycisk służący do wywołania korekcji kolorów przy pomocy sześciokąta kolorów
- **m_propText** - opis suwaka ustawiającego stopień korekcji kolorów przy pomocy sześciokąta kolorów
- **m_propSlider** - suwak ustawiający stopień korekcji kolorów przy pomocy sześciokąta kolorów
- **modHexagonCheckBox** - włącznik wyświetlania sześciokąta zmodyfikowanego ustawieniami kolorów
- **redPartHex** - obraz czerwonej części sześciokąta zmodyfikowanego ustawieniami kolorów
- **bluePartHex** - obraz niebieskiej części sześciokąta zmodyfikowanego ustawieniami kolorów
- **greenPartHex** - obraz zielonej części sześciokąta zmodyfikowanego ustawieniami kolorów
- **bitmapRedPartHex** - bitmapa czerwonej części sześciokąta zmodyfikowanego ustawieniami kolorów
- **bitmapBluePartHex** - bitmapa niebieskiej części sześciokąta zmodyfikowanego ustawieniami kolorów
- **bitmapGreenPartHex** - bitmapa zielonej części sześciokąta zmodyfikowanego ustawieniami kolorów

Zmienne:

- **hexagonColor** - zmienna typu wxColor przechowująca wybrany na sześciokącie kolorów kolor
- **pickedColor** - zmienna typu wxColor przechowująca wybrany na oryginalnym obrazie kolor
- **lastImageColor** - zmienna typu wxColor przechowująca ostatni wybrany na obrazie kolor
- **lastPickedColor** - zmienna typu wxColor przechowująca ostatni wybrany na sześciokącie kolorów kolor
- **showBrightnessSaturationContrastDialog** - zmienna typu bool informująca czy okno ustawień jasności, nasycenia oraz kontrastu jest otwarte
- **hexagonSliderValue** - zmienna typu int przechowująca wartość suwaka korekt wprowadzanych przy pomocy sześciokąta kolorów
- **hexagonChanged** - zmienna typu bool informująca czy ustawienia sześciokąta kolorów zostały zmienione

- **lastHexagonValue** - zmienna typu int przechowująca ostatnią wartość suwaka sześciokąta kolorów
- **modHexagonGenerated** - zmienna typu bool informująca czy sześciokąt zmodyfikowany ustawieniami kolorów mają być generowane
- **histogramsGenerated** - zmienna typu bool informująca czy histogramy mają być generowane

BrightnessSaturationContrastDialog - klasa bazowa okna ustawień jasności, nasycenia i kontrastu.

Elementy:

- **m_BrightnessText** - tekst podpisujący elementy ustawiające jasność
- **m_SaturationText** - tekst podpisujący elementy ustawiające nasycenia
- **m_ContrastText** - tekst podpisujący elementy ustawiające kontrast
- **m_BrightnessScrollBar** - suwak pozwalający na zmianę ustawień jasności
- **m_SaturationScrollBar** - suwak pozwalający na zmianę ustawień nasycenia
- **m_ContrastScrollBar** - suwak pozwalający na zmianę ustawień kontrastu
- **m_BrightnessSetOnEnter** - okno pozwalające na wprowadzanie za pomocą klawiatury lub przycisków wartości jasności
- **m_SaturationSetOnEnter** - okno pozwalające na wprowadzanie za pomocą klawiatury lub przycisków wartości nasycenia
- **m_ContrastSetOnEnter** - okno pozwalające na wprowadzanie za pomocą klawiatury lub przycisków wartości kontrastu
- **m_BrightnessResetButton** - przycisk resetujący wprowadzone zmiany ustawień jasności
- **m_SaturationResetButton** - przycisk resetujący wprowadzone zmiany ustawień nasycenia
- **m_ContrastResetButton** - przycisk resetujący wprowadzone zmiany ustawień kontrastu

MyBrightnessSaturationContrastDialog - pochodna klasy BSCDialog służąca do wyświetlania okna ustawień jasności, nasycenia i kontrastu. Ustawienia te dostępne są po włączeniu okna przy pomocy paska narzędzi view/Brightness, Saturation, Contrast.

Metody:

- **OnScrollingBrightness(wxScrollEvent&)** - metoda modyfikująca ustawienia jasności obrazu. Wywołuje metody setBrightness, setSaturation oraz setContrast z klasy MyFrame. Jest wywoływana przy przesuwaniu suwakiem, oknem wprowadzania z klawiatury i przyciskami obok tego okna
- **OnScrollingSaturation(wxScrollEvent&)** - metoda modyfikująca ustawienia nasycenia obrazu. Wywołuje metody setBrightness, setSaturation oraz setContrast z klasy MyFrame. Jest wywoływana przy przesuwaniu suwakiem, oknem wprowadzania z klawiatury i przyciskami obok tego okna
- **OnScrollingContrast(wxScrollEvent&)** - metoda modyfikująca ustawienia kontrastu obrazu. Wywołuje metody setBrightness, setSaturation oraz setContrast z klasy MyFrame.

Jest wywoływana przy przesuwaniu suwakiem, oknem wprowadzania z klawiatury i przyciskami obok tego okna

- **BrightnessReset(wxCommandEvent&)** - metoda resetująca jasność do wartości domyślnych. Wywołuje metodę OnScrollingBrightness
- **SaturationReset(wxCommandEvent&)** - metoda resetująca nasycenia do wartości domyślnych. Wywołuje metodę OnScrollingSaturation
- **ContrastReset(wxCommandEvent&)** - metoda resetująca kontrast do wartości domyślnych. Wywołuje metodę OnScrollingContrast
- **OnSpinBrightness(wxSpinEvent&)** - metoda obsługująca okno wprowadzania wartości z klawiatury lub przy pomocy przycisków obok tego okna. Wywołuje metodę OnScrollingBrightness
- **OnSpinSaturation(wxSpinEvent&)** - metoda obsługująca okno wprowadzania wartości z klawiatury lub przy pomocy przycisków obok tego okna. Wywołuje metodę OnScrollingSaturation
- **OnSpinContrast(wxSpinEvent&)** - metoda obsługująca okno wprowadzania wartości z klawiatury lub przy pomocy przycisków obok tego okna. Wywołuje metodę OnScrollingContrast
- **getBrightness()** - metoda zwracająca aktualną wartość jasności ustawioną na suwaku
- **getMinBrightness()** - metoda zwracająca minimalną możliwą do ustawienia wartość jasności na suwaku
- **getMaxBrightness()** - metoda zwracająca maksymalną możliwą do ustawienia wartość jasności na suwaku
- **getSaturation()** - metoda zwracająca aktualną wartość nasycenia ustawioną na suwaku
- **getMinSaturation()** - metoda zwracająca minimalną możliwą do ustawienia wartość nasycenia na suwaku
- **getMaxSaturation()** - metoda zwracająca maksymalną możliwą do ustawienia wartość nasycenia na suwaku
- **getContrast()** - metoda zwracająca aktualną wartość kontrastu ustawioną na suwaku
- **getMinContrast()** - metoda zwracająca minimalną możliwą do ustawienia wartość kontrastu na suwaku
- **getMaxContrast()** - metoda zwracająca maksymalną możliwą do ustawienia wartość kontrastu na suwaku
- **ResetAll()** - metoda resetująca wszystkie ustawienia

Zmienne:

- **brightnessValue, saturationValue, contrastValue** - zmienne typu int przechowujące aktualna wartość ustawień kolejno jasności, nasycenia i kontrastu
- **resetBrightness, resetSaturation, resetContrast** - zmienne typu bool zawierające informacje czy został naciśnięty przycisk reset odpowiednio dla resetu jasności, nasycenia i kontrastu.
- **wasLastKeyResetBrightness, wasLastKeyResetSaturation, wasLastKeyResetContrast** - zmienne typu bool przechowujące informacje czy przycisk reset został naciśnięty pierwszy raz odpowiednio dla resetu jasności, nasycenia i

kontrastu. Wartość true, jeżeli nie wykonano innej akcji dla danego ustawienia niż wciśnięcie przycisku reset

ColorsHexagon - klasa pomocnicza służąca do wyświetlania i operacji na sześciokącie kolorów

Metody:

- **paintEvent(wxPaintEvent&)** - odpowiada za rysowanie sześciokąta
- **leftClick(wxMouseEvent&)** - odpowiada za pobranie koloru po kliknięciu na sześciokącie
- **getSelectedColor()** - zwraca z sześciokąta kolor, z pozycji na której znajduje się wskaźnik
- **setSelectedColor(const wxColor&)** - ustawia znacznik na kolorze o danej pozycji
- **setPointerPosition(int, int)** - ustawia pozycję znacznika na sześciokącie

Zmienne:

- **hexagonWidth** - szerokość pola na którym rysowany będzie sześciokąt
- **hexagonHeight** - wysokość pola na którym rysowany będzie sześciokąt
- **selectedColor** - wybrany kolor
- **Parent** - okno rodzica
- **windowDC** - referencja do nadrzędnego okna
- **areaImage** - obraz zawierający sześciokąt
- **areaMap** - bitmapa zawierająca sześciokąt
- **ptrPos_x** - pozycja znacznika x na sześciokącie
- **ptrPos_y** - pozycja znacznika y na sześciokącie