Grafika rastrowa.

Zadanie

Do przygotowanego programu należy dopisać funkcje realizujące szereg typowych operacji związanych z edycją grafiki rastrowej. Będą to: konwersja do skali szarości, rozmycie (*ang. blur*), odbicie lustrzane, zamiana kolorów, przeskalowanie, obrót, przesunięcie odcienia (*ang. hue*), nałożenie maski na obraz, zmiana jasności, zmiana kontrastu, filtracja metodą maski Prewitta oraz próg (*ang. treshold*). Cztery ostatnie operacje trzeba będzie zaimplementować od podstaw. W przypadku pozostałych można (i warto) wykorzystać funkcje wbudowane biblioteki wxWidgets. Wczytywanie obrazu, maski oraz sterowanie programem zostały już zaimplementowane. Miejsca do uzupełnienia wyraźnie zaznaczono w kodzie (metody w pliku *GUIMyFrame1.cpp*).

Cel

Pogłębienie umiejętności pracy z obiektami typu wxImage. Praktyczna implementacja podstawowych algorytmów grafiki rastrowej.

Środki

Biblioteka wxWidgets.

Opis istniejącego kodu

W programie oprócz standardowych deklaracji dodano kilka linijek. Zadeklarowano tam trzy nowe zmienne prywatne typu **wxImage: Img_Org, Img_Cpy, Img_Mask**. Pierwsza z nich będzie przechowywać oryginalny obrazek. Nigdy nie należy modyfikować jej zawartości. Druga zmienna będzie przechowywać obrazek zmodyfikowany, który będzie wyświetlany na ekranie. W trzeciej zaś, zostanie umieszczona maska obrazu. W pliku *main.cpp* najpierw zadeklarowano jakich formatów graficznych będziemy używać (linie 19-20) a następnie wczytano główny obrazek (w formacie JPG) oraz maskę (w formacie PNG). W pliku *GUIMyFrame1.cpp* w linii 10 ustawiono suwaki okna typu **wxScrolledWindow**. Jest to bardzo ważna czynność! Metoda **GUIMyFrame1::Repaint()** jest wywoływana zawsze, gdy system chce odświeżyć zawartość okna z suwakami. We wnętrzu tej metody tworzymy tymczasową bitmapę na podstawie kopii oryginalnego obrazka, na której pracujemy, następnie pobieramy kontekst okna i wywołujemy funkcję **DoPrepareDC**(…) . Jest to bardzo ważne, gdyż umożliwia prawidłowe rysowanie w oknie z suwakami niezależnie od aktualnego położenia suwaków. Na końcu rysujemy bitmapę na aktualnym kontekście.

Kod do uzupełnienia

W kodzie występuje szereg metod, których wnętrza pozostają puste i należy je zaimplementować. Przy implementacji należy zwrócić uwagę na to, które metody klasy **wxImage** modyfikują obrazek sam w sobie (działają na **this**), a które pozostawiają go nie naruszony i jedynie zwracają zmodyfikowaną kopię obrazka. Pamiętajmy, że w programie obrazek oryginalny ma pozostać nie zmieniony, a wciśnięcie dowolnego przycisku

ma spowodować przetworzenie oryginalnego obrazka i wyświetlenie kopii. Oto szczegóły realizacji poszczególnych funkcji:

- konwersja do skali szarości – należy wykorzystać metodę klasy **wxImage**
- rozmycie (*ang. blur*) – rozmycie o wielkości 5 pikseli, należy wykorzystać metodę klasy **wxImage**
- odbicie lustrzane – odbicie w poziomie, należy wykorzystać metodę klasy **wxImage**
- zamiana kolorów – należy zamienić kolor (254,0,0) na (0,0,255), należy wykorzystać metodę klasy **wxImage**
- przeskalowanie – należy przeskalować obrazek do rozmiarów 320x240, należy wykorzystać metodę klasy **wxImage** (trzeba zwrócić uwagę na oczyszczenie tła oraz prawidłową pozycję obrazu po przeskalowaniu)
- obrót – obrót o 30^o, należy wykorzystać metodę klasy **wxImage**
- przesunięcie odcienia (*ang. hue*) – należy przesunąć odcień o 180^o, należy wykorzystać metodę klasy **wxImage**
- nałożenie maski na obraz – na obraz ma zostać nałożona maska zapisana w zmiennej **Img_Mask** (maska ma kolor czarny), należy wykorzystać metodę klasy **wxImage**
- zmiana jasności – tą funkcję należy zaimplementować od podstaw
- zmiana kontrastu – tą funkcję należy zaimplementować od podstaw
- filtracja metodą maski Prewitta – tą funkcję należy zaimplementować od podstaw, maska powinna mieć postać: $\begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$. Należy pamiętać, że wynikowe wartości w każdym kanale barwnym nie mogą być ujemne.
- próg (*ang. treshold*) - tą funkcję należy zaimplementować od podstaw, przy czym próg powinien odcinać wartości powyżej i poniżej 128 w każdym kanale niezależnie

Aby zapewnić dużą szybkość wykonania w ostatnich czterech funkcjach należy wykorzystać metodę **wxImage::GetData()**.

Jak się przygotować przed zajęciami

Zadanie jest w gruncie rzeczy proste. Wszystkie potrzebne funkcje albo są zaimplementowane w bibliotece wxWidgets albo były omawiane na wykładzie i są zaimplementowane w kodach przykładowych programów. Niemniej, żeby sobie sprawnie poradzić z rozwiązaniem zadania, sugeruję:

- Przeczytać opisy klas **wxImage** i **wxImageHandler** w publicznie dostępnej dokumentacji,
- Przypomnieć sobie zagadnienia dotyczące grafiki rastrowej omawiane na wykładzie.