Korekty Barwne

Bartosz Biesaga, Mateusz Wardawa

1 Opis projektu

Projekt polega na napisaniu programu umożliwiającego wykonywanie selektywnych korekt barwnych wczytywanych plików graficznych, poprzez wybór na wczytanym obrazie koloru do zamiany, a następnie na sześciokącie barw koloru docelowego, natomiast siła oraz zakres zmian regulowane są przy pomocy suwaków.

2 Założenia wstępne przyjęte w realizacji projektu

Program ma:

- umożliwiać wczytywanie i wyświetlanie obrazka,
- wyświetlać sześciokąt barw,
- umożliwiać wybór koloru do zamiany na obrazku oraz koloru docelowego na sześciokącie barw,
- wykonywać korektę barwną koloru wybranego do zamiany na kolor docelowy oraz wykonywać korektę składowych kolorów pozostałych punktów proporcjonalnie do odwrotności wartości bezwzględnej z różnicy składowej koloru wybranego do zamiany i składowej danego punktu,
- umożliwiać regulację zakresu oraz siły zmian przy pomocy suwaków,
- umożliwiać regulację jasności sześciokąta barw przy pomocy suwaka.
- informować użytkownika o kolorze wybranym na obrazku poprzez zaznaczenie go na sześciokącie barw oraz poprzez pomalowanie na ten kolor odpowiedniego panelu
- informować użytkownika o kolorze wybranym na sześciokącie poprzez pomalowanie na ten kolor odpowiedniego panelu

3 Analiza projektu

3.1 Specyfikacja danych wejściowych

Program umożliwia wczytywanie plików graficznych w formatach takich jak:

- PNG (`*.png`)
- JPEG (`*.jpg`)
- BMP (`*.bmp`)

3.2 Opis oczekiwanych danych wyjściowych

Oczekiwanymi danymi wyjściowymi są wczytane obrazy poddane korekcie barwnej wyświetlane w aplikacji.

3.3 Zdefiniowanie struktur danych

- `wxImage m_image` - oryginalny obraz wczytany przez użytkownika.
- `wxImage m_modifiedImage` - kopia oryginalnego obrazu poddana korekcie barwnej.
- `wxImage m_scaledModifiedImage` - kopia obrazka po korekcie barwnej przeskalowana do wyświetlania.
- `wxImage m_hexagonImage` - obraz sześciokąta używanego do wyboru kolorów.
- `wxImage m_hexagonImageCopy` - kopia obrazu sześciokąta o odpowiednim poziomie jasności.
- `std::vector<wxPoint> m_hexagonVertices` - wektor przechowujący wierzchołki sześciokąta.
- `wxColour m_selectedColor` - kolor wybrany z obrazu.
- `wxColour m_newColor` - nowy kolor wybrany z sześciokąta.
- `wxPoint m_closestPoint` - współrzędne punktu na obrazie sześciokąta, który najlepiej odpowiada swoim kolorem kolorowi wybranemu na obrazie.
- `double m_mixing_level` - poziom mieszania obrazu oryginalnego ze zmodyfikowaną kopią.
- `double m_strength` - poziom intensywność zmiany koloru.
- `int m_brightness` - jasność sześciokąta.
- `double m_change_coefficient` - współczynnik proporcjonalności korekty barwnej.

3.4 Specyfikacja interfejsu użytkownika

Interfejs użytkownika składa się z następujących elementów:

- `wxPanel* m_imagePanel` - Panel do wyświetlania wczytanego obrazu
- `wxPanel* m_hexImagePanel` - Panel do wyświetlania sześciokąta barw
- `wxPanel* m_imageColorPanel` - Panel do wyświetlania koloru wybranego z obrazu
- `wxPanel* m_hexColorPanel` - Panel do wyświetlania koloru wybranego z sześciokąta
- `wxSlider* m_sliderMixingLevel` - Suwak do regulowania poziomu mieszania obrazu oryginalnego ze zmodyfikowaną kopią.
- `wxSlider* m_sliderStrength` - Suwak do regulowania intensywności zmiany koloru.
- `wxSlider* m_sliderBrightness` - Suwak do regulowania jasności sześciokąta.
- `wxButton* m_loadImageButton` - Przycisk do wczytywania obrazu

Użytkownik może wczytać obraz przy pomocy przycisku *Load Image*, kliknąć na wyświetlony obraz, aby wybrać kolor do zamiany, a następnie wybrać nowy kolor z sześciokąta barw. Za pomocą suwaków, użytkownik może dostosować poziom mieszania obrazu oryginalnego ze zmodyfikowaną kopią, intensywność zmiany oraz jasność sześciokąta.

3.5 Wyodrębnienie i zdefiniowanie zadań

Projekt został podzielony na następujące moduły:

- **Interfejs użytkownika:**
  - Przycisk do wczytywania plików,
  - Dwa panele do wyświetlania obrazu oraz sześciokąta,
  - Dwa panele obrazujące kolory wybrany z sześciokąta i z obrazu,
  - Trzy suwaki, jeden do regulowania poziomu mieszania obrazu oryginalnego ze zmodyfikowaną kopią, do regulowania intensywności zmiany koloru, trzeci do regulowania jasności sześciokąta,
  - Teksty opisujące suwaki oraz panele kolorów.
- **Obsługa sześciokąta:**
  - Generowanie obrazu sześciokąta barw,
  - Rysowanie sześciokąta barw ze wskazanym poziomem jasności,
  - Regulacja poziomu jasności sześciokąta,
  - Zaznaczenie koloru wybranego z obrazka na sześciokącie,
  - Zapisanie koloru wybranego na sześciokącie.

- **Obsługa obrazka:**
  - Wyświetlenie wczytanego obrazka,
  - Zapisanie koloru wybranego na obrazku,
  - Korekta barwna obrazka.

3.6 Decyzja o wyborze narzędzi programistycznych

Do realizacji projektu użyto:

- **Środowisko:** Visual Studio - edytor kodu źródłowego. Wybrany ze względu na łatwość załączania zewnętrznych bibliotek oraz ze względu na jego dobre poznanie podczas laboratoriów.
- **Biblioteki:** wxWidgets - biblioteka do tworzenia aplikacji graficznych. Wybrana ze względu na dobre poznanie jej na laboratoriach oraz łatwe tworzenie interfejsu użytkownika i obsługę grafiki.
- **Kompilator:** Microsoft Visual C++ (MSVC) - wybrany ze względu na to, że jest to domyślny kompilator Visual Studio.

4 Podział pracy i analiza czasowa

Bartosz Biesaga zajmuje się:

- generowaniem oraz rysowaniem sześciokąta barw,
- korektą barwną obrazka,
- tworzeniem dokumentacji dokumentacji.

Mateusz Wardawa zajmuje się:

- interfejsem użytkownika,
- wczytywaniem i wyświetlaniem obrazka,
- pobieraniem i zapisywaniem koloru pobranego z obrazka oraz z sześciokąta,
- tworzeniem dokumentacji dokumentacji.

Na wykonanie projektu przewiduje się dwa tygodnie – pierwszy tydzień na rozplanowanie zadań oraz implementację, natomiast drugi na przetestowanie programu, wprowadzenie koniecznych poprawek oraz przygotowanie dokumentacji.

5 Opracowanie i opis niezbędnych algorytmów

Generowanie sześciokąta barw (funkcja GenerateHexagonImage)

1. Inicjalizacja wymiarów kwadratu, w którym znajdować się będzie sześciokąt, wyznaczenie jego środka oraz promienia okręgu wpisanego w ten kwadrat. Stworzenie obrazka o wymiarach kwadratu.

2. Określenie i zapisanie położeń wierzchołków sześciokąta jako punktów na okręgu opisanym na sześciokącie, odległych od siebie o łuk o kącie środkowym równym $\frac{\pi}{3}$. Pierwszy wierzchołek znajduje się pionowo nad środkiem okręgu.

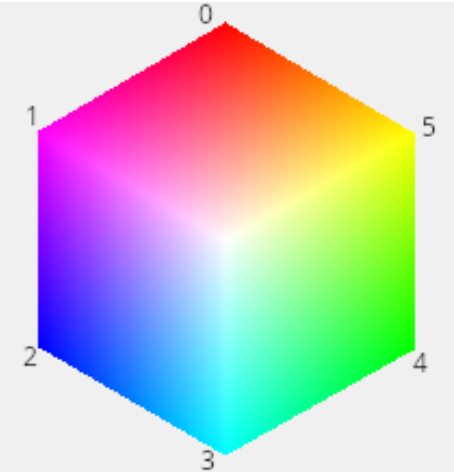
3. Następnie dla każdego punktu obrazka sprawdza się czy znajduje się on wewnątrz sześciokąta (algorytm sprawdzania opisany jest później).

   (a) Jeśli znajduje się poza, to jest malowany na biało.

   (b) Jeśli znajduje się wewnątrz, to określa się, w którym rombie składającym się na sześciokąt się on znajduje oraz jaka jest jego pozycja względem jego rzutów równoległych wzdłuż danego boku na drugi z boków (algorytm sprawdzania opisany jest później), na czego podstawie maluje się punkt na odpowiedni kolor. Jeśli punkt znajduje się w rombie wyznaczonym przez punkty 5,1,0, to maluje się go na kolor RGB(255, $\beta \cdot 255$, $\alpha \cdot 255$), jeśli w rombie (1,2,3), to na RGB($\beta \cdot 255$, $\alpha \cdot 255$, 255), a jeśli w rombie (3,4,5), to na RGB($\alpha \cdot 255$, 255, $\beta \cdot 255$), gdzie α, β, uzyskuje się podczas określania usytuowania punktu w sześciokącie (funkcja pointInWhichRhombus).

4. Na koniec kolor maski ustawiany jest na biały, co pozwala na utworzenie przeźroczystego tła dla sześciokąta. Tworzona jest kopia obrazka, która będzie służyła do modyfikacji i wyświetlania bez niszczenia oryginału.

Sprawdzanie czy punkt znajduje się w sześciokącie (funkcja pointInHexagon)



Rysunek 1: Rysunek sześciokąta z numeracją wierzchołków.

Jeśli którykolwiek z poniższych warunków jest spełniony, to punkt $Q(x_q, y_q)$ znajduje się poza sześciokątem i zwracana jest wartość false.

- Punkt znajduje się na lewo od krawędzi (1,2),
- Punkt znajduje się na prawo od krawędzi (4,5),
- Punkt znajduje się ponad wierzchołkiem 0,
- Punkt znajduje się poniżej wierzchołka 3,
- Punkt znajduje się ponad krawędzią (0,1),
- Punkt znajduje się ponad krawędzią (5,0),
- Punkt znajduje się poniżej krawędzi (2,3),
- Punkt znajduje się poniżej krawędzi (3,4),

Jeśli żaden z powyższych warunków nie zachodzi, to punkt znajduje się wewnątrz sześciokąta i zwracana jest wartość true.

W celu sprawdzenia pozycji danego punktu względem krawędzi (0,1), (5,0), (2,3) i (3,4) wyznacza się ich równania w następujący sposób. Współczynnik kierunkowy prostej jako stosunek różnic odpowiednich współrzędnych punktów $a = \frac{\Delta y}{\Delta x}$, natomiast wyraz wolny jako $b = y_0 - a \cdot x_0$, gdzie $P(x_0, y_0)$ to jeden z wierzchołków wchodzących w skład krawędzi. W celu sprawdzenia pozycji punktu Q względem wyznaczonej prostej, wykonuje się odpowiednie dla danej prostej porównanie ($>$ dla (0,1) i (5,0), $<$ dla (2,3) i (3,4)) wartości $a \cdot x_q + b$ oraz y_q.

Sprawdzanie usytuowania punktu w sześciokącie (funkcja pointInWhichRhombus)

1. Określenie pod jakim kątem względem półprostej zaczynającej się w środku sześciokąta i prostopadłej do krawędzi (4,5) znajduje się dany punkt $Q(x_q, y_q)$.

2. Określenie na podstawie uzyskanego kąta α, w którym rombie znajduje się punkt. Jeśli $\alpha \in [\frac{\pi}{6}, \frac{5\pi}{6}]$, to do rombu wyznaczonego przez wierzchołki 5,0,1, jeśli $\alpha \in (\frac{5\pi}{6}, \frac{3\pi}{2}]$ to do rombu (1,2,3), jeśli $\alpha \in (\frac{3\pi}{2}, 2\pi] \cup (0, \frac{\pi}{6}))$, to do rombu (3,4,5).

3. Wprowadzenie oznaczeń punkt lewy = pierwszy z wierzchołków wyznaczających romb, punkt środkowy = drugi z nich, punkt prawy = trzeci z nich.

4. Wyznaczenie równań prostych wyznaczonych przez punkt lewy ze środkowym (prosta l) oraz punkt prawy ze środkowym (prosta r) w następujący sposób. Współczynnik kierunkowy prostej jako stosunek różnic odpowiednich współrzędnych punktów $a_{l/r} = \frac{\Delta y}{\Delta x + 10^{-9}}$ (10^{-9} w mianowniku zapobiega dzieleniu przez 0), natomiast wyraz wolny jako $b_{l/r} = y_0 - a_{l/r} \cdot x_0$, gdzie $P(x_0, y_0)$ to punkt środkowy.

5. Wyznaczenie równań prostych równoległych do jednego z boków rombu (prosta równoległa do prostej l to prosta β, a druga to α) i przechodzących przez punkt Q w następujący sposób. Współczynnik kierunkowy $a_{\alpha/\beta}$ jest znany i jest to współczynnik kierunkowy prostej do której wyznaczana prosta jest równoległa, natomiast wyraz wolny wyznacza się jako $b_{\alpha/\beta} = y_q - a_{\alpha/\beta} * x_q$.

6. Wyznaczenie położenia punktów P_β, P_α będących przecięciami odpowiednio prostych β z r oraz α z l korzystając z równań

$$\begin{aligned} x_\beta &= \frac{b_\beta - b_r}{a_r - a_l} \\ y_\beta &= a_r \cdot x_\beta + b_r \\ x_\alpha &= \frac{b_l - b_\alpha}{a_r - a_l} \\ y_\alpha &= a_l \cdot x_\alpha + b_l \end{aligned}$$

7. Następnie wyznacza się współczynniki α, β (ich znaczenie jest opisane w opisie algorytmu generowania sześciokąta barw) w następujący sposób

$$\begin{aligned} \alpha &= \frac{\sqrt{(y_\alpha - y_q)^2 + (x_\alpha - x_q)^2}}{r} \\ \beta &= \frac{\sqrt{(y_\beta - y_q)^2 + (x_\beta - x_q)^2}}{r}, \end{aligned}$$

gdzie r to długość boku sześciokąta (a tym samym promienia okręgu na nim opisanego).

Aktualizowanie jasności sześciokąta (funkcja UpdateHexagonBrightness)

1. Stworzenie kopii obrazka sześciokąta.
2. Dla każdego punktu z kopii, sprawdzenie czy suma jego składowych RGB jest równa $3 \cdot 255 = 765$.
   (a) Jeśli tak, to punkt ten jest pomijany bo jest to tło, bądź sam środek sześciokąta przy maksymalnej jasności, który jest rozważany osobno.
   (b) Jeśli nie, to składowe tego punktu są przemnażane przez czynnik równy ilorazowi $\frac{\text{jasność}}{255}$.
3. Na koniec modyfikowana jest jasność punktu będącego środkiem sześciokąta, poprzez przemnożenie jego składowych przez czynnik równy ilorazowi $\frac{\text{jasność}}{255}$.

Zaznaczanie koloru na sześciokącie (funkcja MarkColorOnHexagon)

1. inicjalizacja zmiennej `smallestDistance` określającą najmniejszą odległość od koloru szukanego bardzo dużą wartością.
2. Dla każdego punktu kopii obrazka sześciokąta znajdującego się w sześciokącie, oblicza się odległość jego koloru (RGB(r,g,b)) od koloru szukanego (RGB(r_0, g_0, b_0)) `distance`, zgodnie ze wzorem

$$\texttt{distance} = \sqrt{(r_0 - r)^2 + (g_0 - g)^2 + (b_0 - b)^2}.$$

   Jeśli zachodzi `distance` $<$ `smallestDistance`, to za nową wartość `smallestDistance` przyjmuje się `distance`, a za nowy punkt, którego kolor jest najbliższy poszukiwanemu, przyjmuje się aktualny punkt.

Korekta barwna (funkcja ApplyColorChange)

1. Stworzenie kopii oryginalnego obrazka przeskalowanej do aktualnego rozmiaru panelu, na którym zmodyfikowany obrazek będzie wyświetlany.
2. Inicjalizacja zmiennych: składowe koloru wybrany do zamiany RGB(r_s, g_s, b_s), składowe nowego koloru docelowego RGB(r_n, g_n, b_n).
3. Dla każdego punktu obrazka:
   - inicjalizuje się jego zmienne będące składowymi jego koloru RGB(r, g, b),
   - oblicza się odległość jego koloru od koloru wybranego:

$$\texttt{distance} = \sqrt{(r - r_s + \epsilon)^2 + (g - g_s + \epsilon)^2 + (b - b_s + \epsilon)^2},$$

   gdzie $\epsilon = 10^{-7}$, co zapobiega dzieleniu przez zero.

- Oblicza się zmianę dla danej składowej

$$\Delta r = \Delta g = \Delta b = \frac{\alpha}{\texttt{distance}} - 50,$$

gdzie $\alpha = \frac{\texttt{siła zmiany} \cdot \texttt{współczynnik zmiany}}{100}$, siła zmiany jest ustalana przy pomocy suwaka, a współczynnik zmiany wynosi $20 \cdot 255$.

- Oblicza się maksymalną możliwą zmianę każdej ze składowych

$$\Delta r_{max} = r_n - r$$
$$\Delta g_{max} = g_n - g$$
$$\Delta b_{max} = b_n - b$$

- Jeśli maksymalna możliwa zmiana danej składowej jest ujemna, to zmianę dla danej składowej również zmienia się na ujemną.
- Jeśli moduł maksymalnej możliwej zmiany jest większy od modułu zmiany dla danej składowej, to za zmianę podstawia się maksymalną możliwą zmianę.
- Do każdej składowej danego punktu dodaje się składnik będący iloczynem zmiany danej składowej oraz czynnika

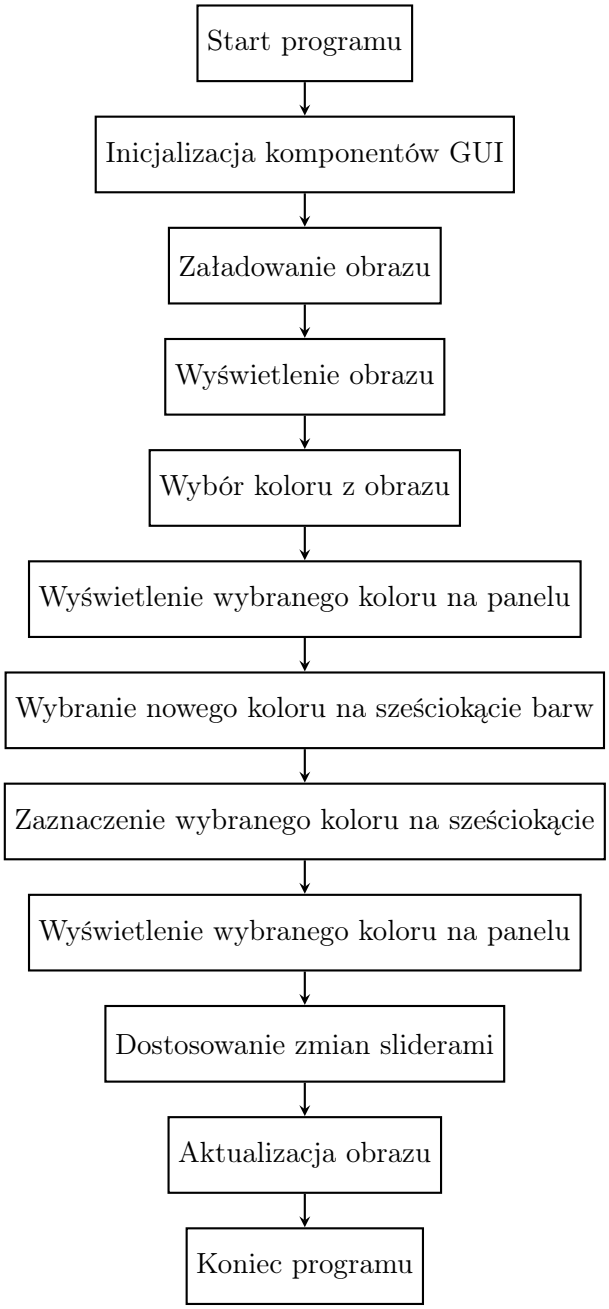
$$\texttt{mixing\_level} = \frac{\texttt{poziom mieszania}}{100},$$

gdzie poziom mieszania jest ustalany przy pomocy suwaka.

4. Tworzy się kopię uzyskanego zmodyfikowanego obrazka, która będzie służyć od wyświetlania go na ekranie.

6 Kodowanie

6.1 Schemat blokowy



6.2 szczegółowy opis klas, funkcji i zmiennych

- **MyFrame(const wxString& title)**: Konstruktor inicjalizujący główne okno aplikacji.
- **void OnLoadImage(wxCommandEvent& event)**: Obsługa ładowania obrazu z pliku.
- **void OnImageClick(wxMouseEvent& event)**: Obsługa kliknięcia na obraz, wybieranie koloru.
- **void OnHexagonClick(wxMouseEvent& event)**: Obsługa kliknięcia na sześciokąt, wybieranie koloru.
- **void OnSliderUpdate(wxCommandEvent& event)**: Aktualizacja poziomu mieszania i siły zmiany koloru.
- **void OnBrightnessSliderUpdate(wxCommandEvent& event)**: Zmiana jasności sześciokąta.
- **void UpdateImageColorPanel(const wxColour& color)**: Aktualizacja panelu z wybranym kolorem z obrazu.
- **void UpdateHexColorPanel(const wxColour& color)**: Aktualizacja panelu z wybranym kolorem z sześciokąta.
- **void GenerateHexagonImage()**: Generowanie obrazu sześciokąta.
- **void MarkColorOnHexagon(const wxColour& color)**: Zaznaczanie koloru na sześciokącie.
- **void DisplayImage()**: Skalowanie i wyświetlanie obrazu.
- **void ApplyColorChange()**: Zastosowanie zmiany koloru na obrazie.
- **void UpdateHexagonBrightness()**: Zmiana jasności sześciokąta.
- **void OnResize(wxSizeEvent& event)**: Obsługa zmiany rozmiaru okna.
- **void DrawHexagon()**: Rysowanie sześciokąta.

7 Testowanie

7.1 Przygotowanie testowych zestawów danych

Przygotowane zostały obrazy w różnych formatach (PNG, JPG, BMP), rozmiarach, kolorach i intensywnościach. Między innymi sześciokąt barw, gdzie najlepiej widać czy korekta barwna działa prawidłowo.

7.2 Testy niezależnych bloków

7.2.1 Testowanie wczytywania obrazów

Sprawdzono czy program wczytuje poprawnie pliki (PNG, JPG, BMP) oraz czy program obsługuje błędne ścieżki plików i wyświetla odpowiednie komunikaty o błędach.

7.2.2 Testowanie interakcji z obrazem

Sprawdzono czy kliknięcie na obraz powoduje zapisanie poprawnych składowych RGB oraz czy prawidłowo wyświetlany jest ten kolor na panelu.

7.2.3 Testowanie sześciokąta

Sprawdzono czy kolory sześciokąta są poprawnie generowane i czy zmieniają się zgodnie z zmianą jasności.

7.3 Testy powiązanych bloków

7.3.1 Testowanie interakcji panelu obrazów i sześciokąta RGB

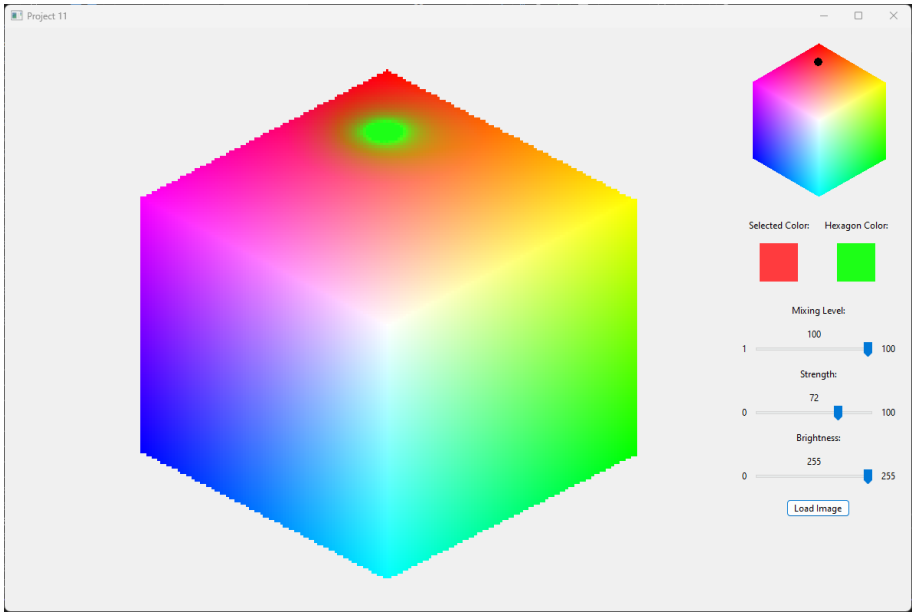
Sprawdzono, czy wybrany punkt na obrazie poprawnie zaznacza odpowiedni kolor na sześciokącie oraz czy nowo wybrany kolor na sześciokącie zamienia wcześniej wybrany kolor na nowy.

7.3.2 Testowanie interakcji suwaków z panelem obrazu i sześciokąta

Sprawdzono czy suwak jasności poprawnie reguluje kolory sześciokąta oraz czy suwaki poziomu mieszania i siły poprawnie wpływają na zmianę koloru na obrazie.

7.4 Testy całościowe

Wczytano obraz i sprawdzono, czy wszystkie elementy interfejsu działają zgodnie z oczekiwaniami. Wybrany został punkt na obrazie i sprawdzono, czy zmienia się on zgodnie z ustawieniami suwaków oraz czy zmiana koloru w sześciokącie jest poprawnie aplikowana na obraz. Jednym z testów było przeprowadzenie korekty barwnej na sześciokącie barw, czego efekty zamieszczono poniżej.



Rysunek 2: Wynik testu na sześciokącie barw.

8 Wdrożenie, raport i wnioski

8.1 Wdrożenie

Program uruchomiono z nieużywanym w czasie testów obrazem i potwierdzono, że program zachowuje się poprawnie.

8.2 Raport

Poprawnie zostały zrealizowane poniższe funkcjonalności:

- Wczytanie i wyświetlenie na ekranie komputera pliku graficznego oraz sześciokąta barw
- Możliwość pobrania koloru w dowolnym miejscu obrazka
- Zaznaczenie pobranego koloru na sześciokącie barw
- Zaznaczenie na sześciokącie barw nowego koloru, na który zostaną zamienione wszystkie punkty w starym kolorze znajdujące się na obrazku
- Składowe kolorów pozostałych ulegają modyfikacji proporcjonalnie do odwrotności wartości bezwzględnej z różnicy składowej koloru referencyjnego (wybranego do zamiany) i składowej modyfikowanej barwy.
- regulacja suwakami:

- Współczynnika proporcjonalności
- Siły zmian poprzez mieszanie obrazu oryginalnego ze skorygowanym
- Jasności sześciokąta

Nie udało się zrealizować wymagań rozszerzonych projektu z powodu nie podjęcia się ich realizacji

Poprawić możnaby algorytm zaznaczania wybranego koloru na sześciokącie, ponieważ w aktualnej formie przeszukuje on cały sześciokąt w poszukiwaniu punktu o kolorze najbardziej zbliżonym do wybranego, co jest nieefektywne, ale ponieważ jest to rzadko wykonywana operacja, to nie wpływa to znacznie na szybkość wykonywania programu. Zamiast tego możnaby przynajmniej obszar poszukiwań ograniczyć do odpowiedniego rombu składającego się na sześciokąt, a najlepszym rozwiązaniem byłoby obliczanie pozycji danego koloru na sześciokącie.

8.3 Wnioski

Program spełnia wymagania podstawowe, jego działanie możnaby zoptymalizować, ale najbardziej obciążające operacje wykonują się z zadowalającą szybkością.