Dokumentacja projektu

Compare Images – Narzędzie do porównywania obrazów

Autorzy projektu:
Gracjan Adamus
Dominik Godek
Kacper Wojtowicz

30 stycznia 2026

Spis treści

1 Tytuł projektu i autorzy projektu **3**

2 Opis projektu **3**

3 Założenia wstępne przyjęte w realizacji projektu **3**

4 Analiza projektu **3**
4.1 Specyfikacja danych wejściowych 3
4.2 Opis oczekiwanych danych wyjściowych 4
4.3 Zdefiniowanie struktur danych 4
4.4 Specyfikacja interfejsu użytkownika 4
4.5 Wyodrębnienie i zdefiniowanie zadań 4
4.6 Decyzja o wyborze narzędzi programistycznych 5

5 Podział pracy i analiza czasowa **5**

6 Opracowanie i opis niezbędnych algorytmów **5**

7 Kodowanie **5**

8 Testowanie **6**
8.1 Testy niezależnych bloków 6
8.2 Testy powiązań bloków 6
8.3 Testy całościowe 6
8.4 Niezmienniki 6

9 Wdrożenie, raport i wnioski **6**

1 Tytuł projektu i autorzy projektu

Tytuł projektu: Compare Images – Narzędzie do porównywania obrazów cyfrowych
Autorzy projektu: Gracjan Adamus, Dominik Godek, Kacper Wojtowicz
Zakres odpowiedzialności:

- analiza problemu i zaprojektowanie architektury aplikacji,
- implementacja logiki przetwarzania obrazów,
- stworzenie interfejsu graficznego użytkownika,
- testowanie oraz przygotowanie dokumentacji.

2 Opis projektu

Projekt *Compare Images* jest aplikacją desktopową umożliwiającą wizualne i numeryczne porównywanie dwóch obrazów cyfrowych. Program pozwala na jednoczesne wyświetlanie dwóch obrazów, synchroniczne powiększanie i przesuwanie, zaznaczanie dowolnych obszarów oraz generowanie obrazu różnicowego na podstawie składowych RGB.

Aplikacja została napisana w języku **C++** z wykorzystaniem bibliotek **SFML** oraz **ImGui**, co zapewnia wysoką wydajność oraz intuicyjny interfejs użytkownika.

3 Założenia wstępne przyjęte w realizacji projektu

Przy realizacji projektu przyjęto następujące założenia:

- aplikacja działa jako program okienkowy,
- użytkownik porównuje dokładnie dwa obrazy jednocześnie,
- obsługiwane są popularne formaty plików graficznych (BMP, PNG, JPG, TGA),
- obrazy mogą mieć różne rozdzielczości,
- porównanie różnic odbywa się w przestrzeni kolorów RGB,
- interfejs użytkownika musi umożliwiać intuicyjną obsługę bez znajomości szczegółów technicznych.

Dodatkowo założono brak konieczności połączenia z siecią oraz brak zależności od zewnętrznych usług.

4 Analiza projektu

4.1 Specyfikacja danych wejściowych

Danymi wejściowymi są:

- pliki graficzne zapisane na dysku lokalnym użytkownika,

- ścieżki do plików podawane w postaci tekstowej,
- zdarzenia wejściowe użytkownika (mysz, klawiatura).

Formaty danych wejściowych: BMP, PNG, JPG/JPEG, GIF (tylko odczyt), TGA.

4.2 Opis oczekiwanych danych wyjściowych

Danymi wyjściowymi są:

- obraz różnicowy zapisany w pliku graficznym,
- obraz zawierający wybrane obszary obu obrazów zestawione obok siebie,
- komunikaty statusowe wyświetlane w interfejsie aplikacji.

Formaty danych wyjściowych: BMP, PNG, JPG.

4.3 Zdefiniowanie struktur danych

Główną strukturą danych w projekcie jest struktura `AppState`, która przechowuje:

- obrazy (`sf::Image`),
- tekstury (`sf::Texture`),
- flagi stanu aplikacji,
- parametry powiększenia i przesunięcia,
- informacje o zaznaczonym obszarze.

Dodatkowo wykorzystywane są struktury wektorowe `sf::Vector2f` oraz `sf::Vector2u`.

4.4 Specyfikacja interfejsu użytkownika

Interfejs użytkownika składa się z:

- panelu sterowania (ładowanie obrazów, zoom, zapisywanie),
- dwóch paneli wyświetlania obrazów,
- okien modalnych do prezentacji obrazu różnicowego i zaznaczeń.

Obsługa odbywa się za pomocą myszy (kliknięcia, przeciąganie, kółko myszy) oraz pól tekstowych.

4.5 Wyodrębnienie i zdefiniowanie zadań

Projekt został podzielony na następujące moduły:

- wczytywanie i zapisywanie obrazów,
- renderowanie obrazów,
- obsługa interfejsu użytkownika,
- generowanie obrazu różnicowego,
- obsługa zaznaczania obszarów.

4.6 Decyzja o wyborze narzędzi programistycznych

- **Język:** C++20 – wysoka wydajność i kontrola pamięci,
- **Biblioteka graficzna:** SFML – prosta obsługa grafiki 2D,
- **GUI:** ImGui – szybkie tworzenie interfejsów,
- **System budowania:** Meson + Ninja – szybka kompilacja.

5 Podział pracy i analiza czasowa

- Analiza wymagań – 1 tydzień,
- Projekt architektury – 1 tydzień,
- Implementacja – 3 tygodnie,
- Testowanie – 1 tydzień,
- Dokumentacja – 1 tydzień.

6 Opracowanie i opis niezbędnych algorytmów

Najważniejszym algorytmem jest algorytm generowania obrazu różnicowego:

$$R = |R_1 - R_2|, \quad G = |G_1 - G_2|, \quad B = |B_1 - B_2|$$

Algorytm zaznaczania obszaru polega na normalizacji współrzędnych punktów początkowych i końcowych oraz ich mapowaniu na drugi obraz.

7 Kodowanie

Kod programu został napisany w jednym pliku źródłowym `main.cpp`. Zawiera on:

- definicję struktury stanu aplikacji,
- funkcje pomocnicze do przetwarzania obrazów,
- główną pętlę aplikacji,
- obsługę zdarzeń i renderowania.

Schemat blokowy programu obejmuje:

1. inicjalizację okna,
2. pętlę obsługi zdarzeń,
3. aktualizację interfejsu,
4. renderowanie obrazów.

8 Testowanie

8.1 Testy niezależnych bloków

- test wczytywania poprawnych i błędnych plików,
- test zapisu obrazów.

8.2 Testy powiązań bloków

- wczytywanie + renderowanie,
- zaznaczanie + zapis.

8.3 Testy całościowe

Testy przeprowadzono na różnych rozdzielczościach i formatach obrazów.

8.4 Niezmienniki

- zoom nie wychodzi poza zakres,
- zaznaczenie zawsze mieści się w granicach obrazu.

9 Wdrożenie, raport i wnioski

Aplikacja została uruchomiona i przetestowana na niezależnych danych wejściowych. Projekt spełnia założone wymagania funkcjonalne.

Możliwe usprawnienia:

- porównywanie obrazów w innych przestrzeniach barw,
- obsługa więcej niż dwóch obrazów,
- eksport raportu różnic.