Interpolacja Grafiki Rastrowej

projekt numer 23 - dokumentacja

Marta Figurska
Szymon Kolanko
Bartosz Konopka

1 Tytuł projektu i autorzy projektu

W ramach wykonania projektu na przedmiot "Podstawy Grafiki Komputerowej" przygotowaliśmy projekt o tytule "Interpolacja Grafiki Rastrowej". Autorzy projektu i ich podział pracy prezentują się następująco:

- Marta Figurska - przygotowanie dokumentacji oraz testowanie projektu, pomoc oraz dopracowanie algorytmów interpolacji
- Szymon Kolanko - przygotowanie algorytmów lupy, obsługi aplikacji, załadowywania i zapisywania zdjęć oraz odpowiednie poprawki w GUI projektu
- Bartosz Konopka - przygotowanie GUI projektu, zaimplementowanie odpowiednich algorytmów interpoalcji, oraz pomoc w pisaniu dokumentacji

2 Opis projektu

Celem projektu jest stworzenie oprogramowania umożliwiającego porównanie i wybór jak najlepszej metody interpolacji dla konkretnego obrazu.

3 Założenia wstępne przyjęte w realizacji projektu

3.1 Wymagania podstawowe

Użytkownik ma możliwość załadowania obrazu w formacie BMP. Obraz pokazuje się na ekranie, a wraz z nim pięć okienek znajdujących się poniżej. Każde okienko odpowiada za inny rodzaj interpolacji fragmentu załadowanego obrazu. Owy urywek obrazka użytkownik wskazuje najeżdżając na niego kursorem, natomiast za pomocą suwaka znajdującego się po prawej stronie, określa wielkość kwadratu zaznaczającego pożądany fragment. Użytkownik ma możliwość zapisu wykonanej interpolacji w postaci pliku typu BMP.

3.2 Wymagania rozszerzone

W wersji poszerzonej program oprócz powiększania powinien umożliwiać pomniejszanie obrazu, obracanie wybranego obszaru o dowolny kąt oraz łączenia obracania ze zmianą rozmiarów.

4 Analiza projektu

4.1 Specyfikacja danych wejściowych

Program powinien umożliwić wczytywanie plików typu BMP, które poddawane będą w późniejszych krokach interpolacji różnymi sposobami.

4.2 Opis oczekiwanych danych wyjściowych

Aplikacja umożliwia zapisanie otrzymanych 4 wyników interpolacji, oraz oryginalnego obrazu w formacie pliku BMP. 5 powiększonych obrazków w postaci kwadracików znajduje się na zapisanym zdjęciu obok siebie.

4.3 Zdefiniowanie struktur danych

W projekcie zostało wykorzystane kilka kontenerów z biblioteki STD, w głównej mierze std::array. Został on wykorzystan min. do przechowywania obiektów wxImage będących kopiami aktualnie przybliżanych paneli.
Głowną klasą zdefiniowaną w celu wykonania projektu jest GUIMyFrame1, która dziedziczy po klasie MainFrame będącą klasą wygenerowaną przez program wxFormBuilder. Klasa napisana przez nasz zespół korzystając z metod wirtualnych oraz polimorfizmu rozszerza działanie klas bazowych, tworząc potrzebny do wykonania zadania interfejs programistyczny.

4.4 Specyfikacja interfejsu użytkownika

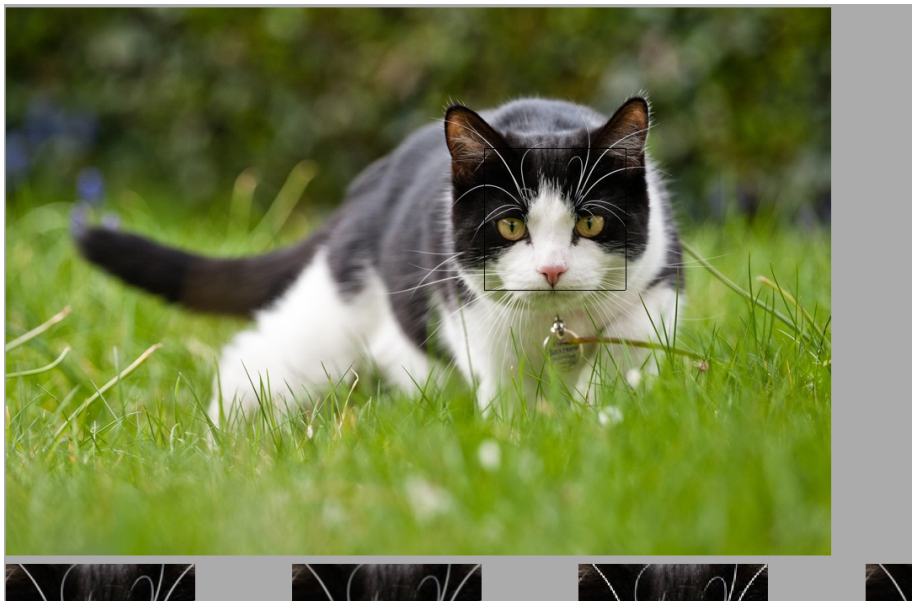
4.4.1 Okno aplikacji

Okno aplikacji dzieli się na panel główny, dolne panele interpolacji, oraz boczny panel umożliwiający wykonywanie odpowiednich operacji przez użytkownika.

Rysunek 1: Widok całego okna aplikacji

4.4.2 Panel główny

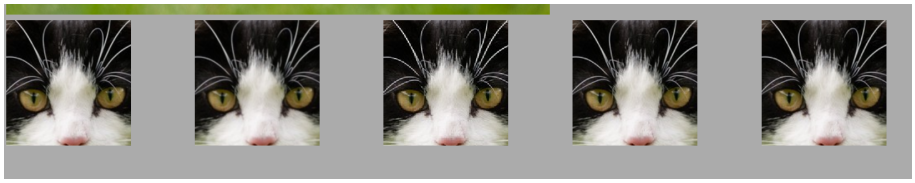
Panel główny prezentuje załadowane zdjęcie bez dokonania powiększeń. Na zdjęciu za pomocą myszki można sterować lupą, czyli kwadratem ukazującym ktory obszar zostanie powiększony i poddany interpolacji.



Rysunek 2: Widok panelu głównego

4.4.3 Panele interpolacji

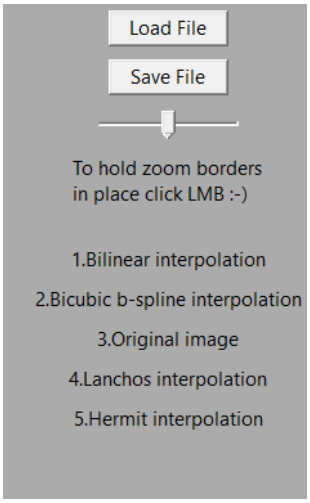
Pięć panelów interpolacji przedstawia od lewej: interpolacje dwuliniową, interpolację dwusześcienną b-spline, oryginalny obraz nie poddany interpolacji, interpolację Lanchos, oraz interpolację Hermita. W każdym panelu ukazywany jest odpowiednio interpolowana część obrazu z panelu głównego, objęta aktualnie zakresem lupy.



Rysunek 3: Widok panelów interpolacji

4.4.4 Panel umożliwiający odpowiednie operacje

Panel znajdujący się z prawej bocnej strony aplikacji umożliwia załadowanie pliku, oraz zapisanie wyników interpolacji. Pod przuciskami wczytania i zapisu, znajduje się suwak decydujący o wartości przybliżenia obrazu. Maksymalnie wychylenie na lewo nie przybliża obrazu, im większa wartość na prawo tym obraz zostanie bardziej powiększony. Panel informuje również użytkownika że podczas naciśnięcia lewego przycisku myszy lupa przestanie śledzić myszkę i zatrzyma się w jednym miejscu, do następngo naciśnięcia myszki. Informuje również o użytych algorytmach interpolacji w kolejności od lewej do prawej.



Rysunek 4: Widok panelu operacji

4.5 Wyodrębnienie i zdefiniowanie zadań

1. Analiza projektu, zdefiniowanie zadań, środowiska programistycznego oraz celu projektu.
2. Utworzenie repozytorium Github, konfiguracja CMake.
3. Zaprojektowanie interfejsu graficznego w wxFormBuilder.
4. Implementacja działania aplikacji, zaprojektowanie działania lupy, przygotowanie kodu pod implementację algorytmów interpolujących.
5. Opracowanie potrzebnych algorytmów interpolujących i ich implementacja w projekcie.
6. Testowanie aplikacji, naprawa bugów.
7. Sporządzenie dokumentacji.

4.6 Decyzja o wyborze narzędzi programistycznych

W celu przechowywania projektu i efektywnego rozwijania kodu zdecydowaliśmy się na użycie systemu kontroli wersji Git. Kod pisaliśmy w edytorze Visual Studio Code, zaś kompilację i linkowanie kodu postawnowiliśmy wydelegować do narzędzia CMake. Główną biblioteką do realizacji projektu stało się wxWigets.

5 Podział pracy i analiza czasowa

1. dogłębne przeanalizowanie tematu projektu, przypomnienie nabytych wcześniej umiejętności pracy z wxWidgets, podział pracy, przygotowanie środowiska 6h
2. stworzenie interfejsu 8h
3. przygotowanie aplikacji, stworzenie narzędzia lupy 9h
4. implementacja odpowiednich algorytmów 17h
5. spotkania pozwalające ustalić dotychczasowe postępy oraz omówić ewentualne watpliwości 8h
6. realizacja testów, poprawki 10h
7. przygotowanie dokumentacji 7h

6 Opracowanie i opis niezbędnych algorytmów

W aplikacji skorzystaliśmy z następujących algorytmów interpolacji:

- Algorytm interpolacji dwuliniowej (Bilinear Interpolation) - Algorytm interpolacji bilinear korzysta z czterech najbliższych sąsiadów piksela, aby obliczyć jego wartość pośrednią. Są to dwa piksele znajdujące się po lewej i prawej stronie oraz dwa piksele znajdujące się powyżej i poniżej interpolowanego piksela.
  Proces interpolacji bilinearnej odbywa się w kilku krokach:
  - Określenie współrzędnych sąsiadujących pikseli: Niech (x, y) będą współrzędnymi interpolowanego piksela. Zaokrąglamy x i y do najbliższych całkowitych wartości, aby uzyskać współrzędne czterech najbliższych sąsiadów.
  - Obliczenie wag interpolacji: Obliczamy odległości pomiędzy współrzędnymi interpolowanego piksela (x, y) a czterema najbliższymi sąsiadami. Dla każdego sąsiada obliczamy wagę interpolacji, która jest odwrotnie proporcjonalna do odległości od tego sąsiada. Im bliżej sąsiada znajduje się interpolowany piksel, tym większa jest jego waga.
  - Obliczenie wartości piksela pośredniego: Dla każdego kanału koloru (czerwony, zielony, niebieski) obliczamy wartość pośrednią. Mnożymy wartość kanału sąsiada przez wagę interpolacji dla tego sąsiada, a następnie sumujemy te wartości dla wszystkich czterech sąsiadów.
  - Powtarzanie kroków 2 i 3 dla każdego kanału koloru, jeśli obraz jest w kolorze.
  - Zwrócenie obliczonych wartości kanałów koloru jako wynik interpolacji dla piksela (x, y).

  W programie wykorzystany został algorytm wbudowany w bilbliotekę wxWidgets.

- Algorytm interpolacji dwusześciennej b-spline(Bicubic b-spline Interpolation) - B-spline to gładka krzywa definiowana przez węzły kontrolne i funkcje bazowe B-spline. Algorytm interpolacji B-spline polega na znalezieniu odpowiednich wag dla węzłów kontrolnych w celu przybliżenia wartości pikseli na podstawie ich otoczenia.
  Algorytm bicubic b-spline wygląda następująco:

- Określenie węzłów kontrolnych: Na podstawie dostępnych pikseli określamy węzły kontrolne, które będą służyć do interpolacji. Węzły kontrolne mogą być równoodległe lub dostosowane do danych pikseli.
- Wybór funkcji bazowych: Dla algorytmu interpolacji B-spline wykorzystuje się funkcje bazowe, które będą decydować o wpływie każdego węzła kontrolnego na wartość interpolowanego piksela. Najczęściej stosowane są funkcje bazowe B-spline stopnia trzeciego, takie jak funkcje B-spline Kubelki (B-spline cubic).
- Obliczenie wag interpolacji: Dla każdego piksela interpolowanego obliczamy wagi interpolacji dla węzłów kontrolnych. Wagi te zależą od odległości piksela interpolowanego od węzłów kontrolnych i od wybranej funkcji bazowej. Często używa się funkcji wagowej, takiej jak funkcja wagowa B-spline, aby wprowadzić pewne wygładzenie i kontrolować wpływ węzłów kontrolnych na interpolowany piksel.
- Obliczenie wartości piksela pośredniego: Dla każdego kanału koloru (czerwony, zielony, niebieski) obliczamy wartość pośrednią. Mnożymy wartość kanału węzła kontrolnego przez odpowiadającą mu wagę interpolacji, a następnie sumujemy te wartości dla wszystkich węzłów kontrolnych.
- Powtarzanie kroku 4 dla każdego kanału koloru, jeśli obraz jest w kolorze.
- Zwrócenie obliczonych wartości kanałów koloru jako wynik interpolacji dla piksela interpolowanego.

W programie wykorzystany został algorytm wbudowany w bilbliotekę wxWidgets.

- Algorytm interpoalcji Lanczosa (Lanchos Interpolation) - jest techniką interpolacji stosowaną do przybliżania wartości funkcji na podstawie jej dyskretnych próbek. Algorytm ten korzysta z funkcji okna, zwanej funkcją Lanczosa, w celu wagowego wygładzenia próbek i generowania płynnej interpolacji.

  Oto podstawowy opis algorytmu interpolacji Lanczosa:

  - Na początku należy zdefiniować funkcję Lanczosa. Funkcja ta jest zdefiniowana na przedziale [-N, N], gdzie N jest parametrem określającym stopień wygładzenia. Funkcję Lanczosa można zdefiniować przy użyciu wzoru:
    $w(x) = sinc(x) * sinc(x/N)$,
  - Następnie dla każdej próbki x w zbiorze dyskretnych próbek obliczamy wagę interpolacyjną W(x) przy użyciu funkcji Lanczosa. Waga interpolacyjna jest obliczana jako iloczyn wartości próbki z funkcją Lanczosa, czyli $W(x) = w(x) * f(x)$, gdzie $f(x)$ to wartość próbki w punkcie x .
  - Aby obliczyć wartość interpolowanej funkcji w dowolnym punkcie x, sumujemy wagi interpolacyjne dla wszystkich próbek, przeskalowujemy je i normalizujemy.
  - Interpolacja Lanczosa może być również stosowana do interpolacji funkcji wielowymiarowych. W takim przypadku obliczenia są wykonywane w każdym wymiarze niezależnie.

  Algorytm interpolacji Lanczosa ma wiele zastosowań w różnych dziedzinach, nie tylko w grafice komputerowej ale również jest przydatny przy przetwarzaniu sygnałów. Jest często wykorzystywany do generowania płynnych krzywych i przybliżania sygnałów o wysokiej częstotliwości próbkowania.

- Algortym interpolacji Hermita (Hermit Interpolation) - jest techniką interpolacji stosowaną do przybliżania wartości funkcji na podstawie zarówno jej dyskretnych próbek, jak i jej pochodnych w tych punktach. Algorytm ten pozwala na generowanie bardziej dokładnych interpolacji, uwzględniając informacje o nachyleniu funkcji w poszczególnych punktach. Algorytm interpolacji Hermita można podzielić na kilka kroków:
  - Zebranie danych wejściowych w postaci punktów (x, y) oraz odpowiadających im wartości pochodnych (dy/dx) w tych punktach.
  - Na podstawie zebranych danych obliczamy różnice dzielone dla punktów i ich pochodnych. Możemy użyć metody różnic dzielonych Newtona, ale uwzględniamy zarówno wartości y, jak i dy/dx.
  - Na podstawie obliczonych różnic dzielonych tworzymy funkcję interpolującą. Możemy użyć wzoru interpolacyjnego Newtona lub innego odpowiedniego wzoru interpolacyjnego.
  - Używając funkcji interpolującej, możemy obliczyć wartość funkcji w dowolnym punkcie, który nie był częścią danych wejściowych. Możemy również obliczyć wartość pochodnej w tym punkcie.

7 Kodowanie

Zdefiniowane zostały następujące klasy:

1. Klasa MainFrame - reprezentuje główne okno aplikacji. Klasa ta zawiera różne elementy interfejsu użytkownika, takie jak panele, przyciski, suwaki, pola tekstowe, które są wykorzystywane do tworzenia interfejsu graficznego.

2. Klasa GUIMyFrame1 - określa rozmiar obrazu i inne parametrów związane z rozmiarem. Przechowuje wymiary paneli, dane obszaru powiększenia, metody odpowiedzialne za przycinanie obrazu. Odpowiada za śledzenie ruchów myszy, scrolla, możliwość załadowania i zapisania obrazu.

8 Testowanie

1. Testy bloków - odbywały się w czasie trwania prac nad kodem. Błędy były na bieżąco identyfikowane oraz, w miarę możliwości, eliminowane.

2. Testy powiązań bloków - zostaly zrealizowane wraz z całościowymi testami.

3. Testy całościowe - miały miejsce po zrealizowaniu projektu na poziomie podstawowym. Przetestowane zostaly wszystkie funkcjonalności programu w zróżnicowanych konfiguracjach, na wszelakich systemach operacyjnych. Testy były przeprowadzane zarówno przez osoby realizujące projekt, jak i, już na etapie końcowym, przez osoby niezainteresowane i niewtajemniczone, aby uzyskać informację zwrotną o przejrzystości projektu.

9 Wdrożenie, raport i wnioski

Projekt został zrealizowany na poziomie podstawowym. Jego poprawność oceniamy pozytywnie - wszelkie funkcjonalności i założenia zostały wykonane. Udało się zaimplementować kod we wskazanym przez prowadzącego czasie. Pojawiające się błędy oraz wątpliwości były sukcesywnie eliminowane, dzięki umiejętności współpracy. Projekt pozwolił nam zapoznać się z wieloma użytecznymi narzędziami w praktyce. Realizacja powyższego projektu zweryfikowała również umiejętości planowania czasu oraz pracy w grupie - wyciągnięte zostały istotne wnioski zarówno pozytywne, jak i negatywne, które z pewnością przyczynią się do rozwoju wielu użytecznych mechanizmów oraz doprowadzą do wprowadzenia pomocnych założeń przy kolejnych, równie złożonych, projektach.