Przebiegi czasowe [Projekt 18]

Szymon Gwóźdź, Maksymilian Walicki, Jan Kwiatkowski

Opis projektu

Projekt polega na stworzeniu narzędzia do wizualizacji przebiegu zmienności pojedynczego parametru w czasie. Narzędzie umożliwia również równoczesną wizualizację dwóch przebiegów, z czego jeden jest historyczny.

Założenia wstępne przyjęte w realizacji projektu

- Program pobiera w równych odstępach dane z pliku o zadanej nazwie i wyświetla je na ekran.
- Plik modyfikowany jest przez zewnętrzny program, będący “generatorem”.
- W pliku znajduje się maksymalnie 10s przebiegu. Następnie plik jest czyszczony i pojawia się w nim nowy przebieg.
- Dostępne są dwa tryby pracy:
    a. Rysowanie wykresu ostatnio wczytanych danych
    b. Rysowanie wykresów ostatnio i przedostatnio wczytanych danych różniących się kolorami

Analiza projektu

Specyfikacja danych wejściowych

- Dane wizualizowanego parametru
    - Postać: Współrzędne x i y zapisywane parami w rzędach
    - Format: Plik .dat
- Tryb pracy
    - Postać: Radio button w panelu aplikacji
    - Format: Enum
    - Wartości: Obecne dane, Obecne i poprzednie dane, Wszystkie dane
- Opcje miarki
    - Postać: Checkbox w panelu aplikacji
    - Format: Bool
    - Wartości: Włączona/Wyłączona

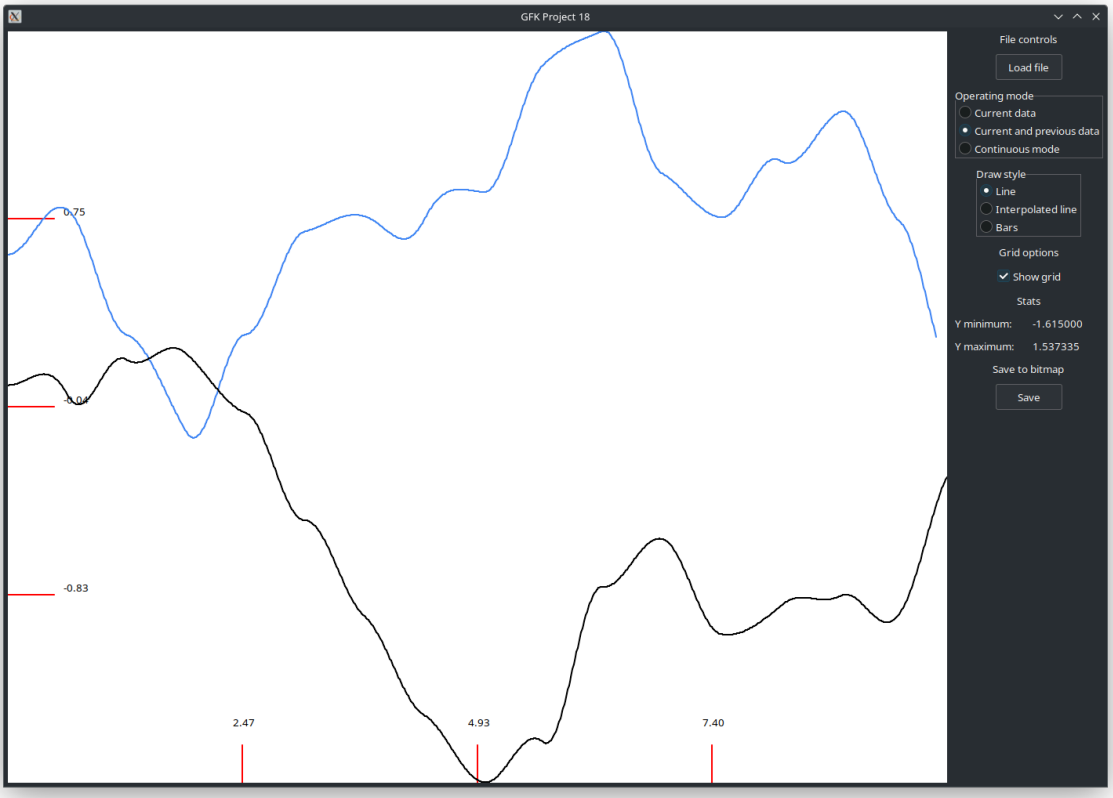
Specyfikacja danych wyjściowych

- Wykres
    - Postać: obraz w oknie aplikacji
- Y Minimum i Y Maximum
    - Postać: zmienne tymczasowe
    - Format: float
- Bitmapa
    - Postać: Eksport do pliku .bmp

Struktury danych

- Bufor wyświetlanego sygnału
    - Dane zapisywane są jako zbiór współrzędnych w formacie std::vector<std::pair<double, double>> jako składnik zdefiniowanej przez nas klasy Data.

Specyfikacja interfejsu użytkownika



1. “Load file” – Przycisk umożliwiający ustawienie pliku, z którego mają być pobierane dane

2. “Operating mode” – Wybór trybu pracy, w zależności od wybranej opcji rysowany jest obecny strumień danych z pliku, obecny strumień wraz z poprzednim, lub wszystkie archiwalne łącznie z obecnym
3. “Show grid” – Przełączanie opcji wyświetlania siatki
4. “Stats” – Minimalna i maksymalna wartość Y obecnie wyświetlanego wykresu
5. “Save” – Przycisk realizujący zapis wyświetlanego obecnie wykresu do pliku w formacie .bmp

Wyodrębnienie i zdefiniowanie zadań

Cykliczne pobieranie danych z pliku

Stworzenie metody pobierającej cyklicznie dane z plików. Utworzenie struktury danych dla bufora wartości gotowych do wyświetlenia.

Wyświetlanie danych z pliku na wykresie

Wykorzystanie wxPanel do wyświetlania wykresu z naszej struktury danych.

Wyświetlanie archiwalnych danych

Zapis archiwalnych sygnałów 10-sekundowych i wyświetlanie ostatniego lub wszystkich w zależności od wybranego trybu pracy.

Opcja podziałki

Implementacja metody rysującej podziałkę warunkowo w zależności od wybranej przez użytkownika opcji.

Opcja zapisu wykresu do bitmapy

Konwersja rysowanego wykresu do formatu .bmp oraz uruchomienie okna dialogowego umożliwiającego zapis do wybranej destynacji.

Decyzja o wyborze narzędzi programistycznych

IDE

Visual Studio – to zintegrowane środowisko programistyczne (IDE) firmy Microsoft, które jest szczególnie dobrze przystosowane do pracy z C++ i MSVC. Zalety używania Visual Studio:

- **Zaawansowane narzędzia do debugowania**: Visual Studio oferuje potężne narzędzia do debugowania, co jest nieocenione w trakcie rozwoju aplikacji,

szczególnie takich, które wymagają ciągłego monitorowania i aktualizacji danych w czasie rzeczywistym.
- **Intuicyjny interfejs użytkownika**: Ułatwia zarządzanie projektami i plikami, co jest kluczowe w projektach edukacyjnych, gdzie uczestnicy często uczą się zarówno programowania, jak i zarządzania projektami.
- **Integracja z różnymi narzędziami i bibliotekami**: Visual Studio umożliwia łatwą integrację z różnymi bibliotekami zewnętrznymi jak wxWidgets oraz innymi narzędziami, co sprawia, że jest idealnym wyborem dla projektów wieloplatformowych.

Biblioteki

wxWidgets – jest biblioteką umożliwiającą tworzenie interfejsów graficznych użytkownika (GUI). Wykorzystanie tej biblioteki w projekcie ma kilka zalet:

- **Przenośność**: wxWidgets wspiera różne systemy operacyjne, co jest istotne, gdyż aplikacja może być używana na różnych platformach.
- **Dojrzałość i stabilność**: Jako że wxWidgets jest rozwijana od wielu lat, oferuje stabilne i przetestowane komponenty do tworzenia zaawansowanych interfejsów graficznych.

Kompilator

MSVC – jest kompilatorem C++ rozwijanym przez Microsoft, często używanym razem z Visual Studio. Zalety użycia MSVC to:

- **Optymalizacja pod Windows**: Kompilator jest optymalizowany pod kątem aplikacji Windows, co może przyczynić się do lepszej wydajności programu.
- **Wsparcie dla nowoczesnego C++**: MSVC regularnie aktualizuje wsparcie dla najnowszych standardów C++, co jest ważne w przypadku akademickich projektów programistycznych, gdzie aktualna wiedza i praktyki są kluczowe.
- **Integracja z Visual Studio**: MSVC jest doskonale zintegrowany z Visual Studio, co upraszcza proces budowania i debugowania aplikacji.

Podział pracy i analiza czasowa

W ramach realizacji projektu zadania rozdzielane były tak, by każdy uczestniczył w równej części w każdym z procesów.

Pobieranie danych z pliku – 10 h
Wyświetlanie danych z pliku na wykresie – 4 h
Wyświetlanie archiwalnych danych – 4 h
Opcja siatki – 1 h
Opcja zapisu do bitmapy – 2 h

Opracowanie i opis niezbędnych algorytmów

1. Algorytm Rysowania Wykresów

Algorytm rysowania wykresów w funkcji `Plotter::draw` składa się z kilku kroków:

- **Czyszczenie tła**: Ustawienie białego tła i czyszczenie obszaru rysowania.
- **Obliczanie parametrów transformacji**: Na podstawie zakresów danych (`x0`, `x1`, `y0`, `y1`), obliczana jest skala (`Sx`, `Sy`) i przesunięcie (translacja), aby dostosować dane do obszaru rysowania.
- **Tworzenie macierzy transformacji**: Kombinacja operacji skalowania i translacji w jedną macierz transformacji, dodatkowo aplikowane są operacje takie jak odwracanie osi Y (dla poprawnego wyświetlania w układzie współrzędnych komputera, gdzie górny lewy róg to (0,0)).
- **Rysowanie danych**: Iteracja przez dane i rysowanie linii między punktami, z zastosowaniem wcześniej obliczonej transformacji. W zależności od trybu pracy (`CURRENT`, `CURRENT_AND_PREVIOUS`, `CONTINUOUS`), dane są wizualizowane w różnych kolorach i stylach.

2. Operacje na Macierzach i Wektorach

- **Mnożenie macierzy**: Wykorzystuje standardowy algorytm mnożenia macierzy, gdzie każdy element nowej macierzy jest sumą iloczynów odpowiednich elementów macierzy wejściowych.
- **Transformacje geometryczne**: Algorytmy transformacji, takie jak translacja, skalowanie, rotacja i odwracanie, są realizowane przez modyfikację odpowiednich elementów macierzy transformacji, co pozwala na łatwe ich łączenie i aplikowanie na punkty danych.
- **Aplikacja transformacji na wektory**: Punkt reprezentowany przez wektor jest przekształcany przez macierz, co pozwala na zmianę jego położenia w przestrzeni dwuwymiarowej, zgodnie z zadaną transformacją.

3. Zarządzanie Danymi

- **Wczytywanie danych**: Dane są wczytywane z pliku, a następnie przypisywane do odpowiednich struktur (`current`, `previous`, `historic`), zależnie od ich wielkości i historii wczytywania.
- **Przesunięcia czasowe w danych historycznych**: Jeśli nowe dane są dodawane do danych historycznych, to są one przesuwane czasowo, aby odzwierciedlały ciągłość zebranych danych.
- **Aktualizacja buforów danych**: Dane są aktualizowane w taki sposób, że najnowszy zestaw danych zawsze staje się bieżącym, a poprzedni zestaw

staje się poprzednim, co umożliwia porównywanie dwóch najnowszych zestawów danych.

Kodowanie

Przegląd

Ten program w języku C++ jest zaprojektowany do symulacji aplikacji podobnej do oscyloskopu przy użyciu biblioteki wxWidgets. Obsługuje wizualizację danych graficznych opartą na ustawieniach zdefiniowanych przez użytkownika oraz danych wczytanych z plików. Główne komponenty programu to klasy `Config`, `CppOscilloscopeMainFrame`, `Data`, `DataLoader` i `Plotter`, wraz z klasami pomocniczymi do operacji na wektorach i macierzach (`Vector` i `Matrix`).

Opisy klas

1. **Config**
    - **Cel**: Zarządza ustawieniami konfiguracyjnymi aplikacji oscyloskopowej.
    - **Główne atrybuty**:
        - `_operatingMode`: Enum definiujący tryb pracy (np. bieżący, historyczny).
        - `_drawStyle`: Enum określający styl rysowania (linie, punkty itp.).
        - `_showGrid`: Boolean do przełączania widoczności siatki.
        - `_filepath`: Łańcuch znaków przechowujący ścieżkę do pliku z danymi.
    - **Metody**:
        - Konstruktory do inicjalizacji ustawień bezpośrednio lub za pomocą konwersji typu.
        - Gettery i settery dla każdego atrybutu, umożliwiające dostęp i modyfikację ustawień konfiguracyjnych.
2. **CppOscilloscopeMainFrame**
    - **Cel**: Służy jako główna klasa okna, obsługuje zdarzenia GUI i renderowanie.
    - **Inicjalizacja**:
        - Konfiguracja głównego okna, ustawienia domyślne i inicjalizacja obsługi zdarzeń.
    - **Obsługa zdarzeń**:
        - Obsługa malowania, aktualizacji UI, kliknięć przycisków i zdarzeń timera do odświeżania danych, wizualizacji danych i zapisywania wyników.
    - **Wizualizacja danych**:

        - Rysowanie wizualnych reprezentacji danych na panelu w oparciu o bieżącą konfigurację i załadowane dane.
3. **Data**
    - **Cel**: Przechowuje i zarządza punktami danych do wizualizacji.
    - **Główne atrybuty**:
        - `x_min`, `x_max`, `y_min`, `y_max`: Śledzą minimalne i maksymalne wartości współrzędnych x i y.
        - `data_points`: Wektor par liczby double reprezentujących punkty danych.
    - **Metody**:
        - `addDataPoint()`: Dodaje punkt danych i aktualizuje wartości min/max.
        - `getDataPoints()`, `getXMin()` itp.: Akcesory do właściwości danych.
        - Przeciążone operatory porównania do porównywania obiektów danych.
4. **DataLoader**
    - **Cel**: Obsługuje wczytywanie danych z plików do obiektów `Data`.
    - **Metoda**:
        - `loadDataFromFile()`: Otwiera określony plik, czyta dane i wypełnia obiekty `Data`. Obsługuje przełączanie między bieżącymi, poprzednimi i historycznymi danymi w zależności od wielkości nowych danych.
5. **Plotter**
    - **Cel**: Obsługuje graficzne renderowanie danych na kontekście urządzenia.
    - **Główne atrybuty**:
        - `_config`: Współdzielony wskaźnik do bieżących ustawień konfiguracji.
    - **Metody**:
        - `draw()`: Rysuje dane na płótnie w oparciu o skonfigurowane style i tryby.
        - `line2d()`, `text2d()`: Funkcje pomocnicze do rysowania linii i tekstu, transformowane przez operacje macierzowe.
6. **Vector** i **Matrix**
    - **Cel**: Wspierają transformacje geometryczne potrzebne do wizualizacji danych.
    - **Metody Vector**:
        - `GetX()`, `SetX()`, `GetY()`, `SetY()`: Dostęp i modyfikacja komponentów wektora.
    - **Operacje Matrix**:
        - Obsługuje mnożenie macierzy oraz transformacje takie jak translacja, skalowanie, rotacja i ścinanie.

Dodatkowe klasy i metody pomocnicze

- **Klasy wxWidgets**: Używane do komponentów GUI takich jak okna, panele i okna dialogowe.
- **Obsługa zdarzeń**: Aplikacja obsługuje zdarzenia takie jak kliknięcia przycisków, malowanie i timery, aby wchodzić w interakcję z użytkownikiem i odświeżać wyświetlacz.

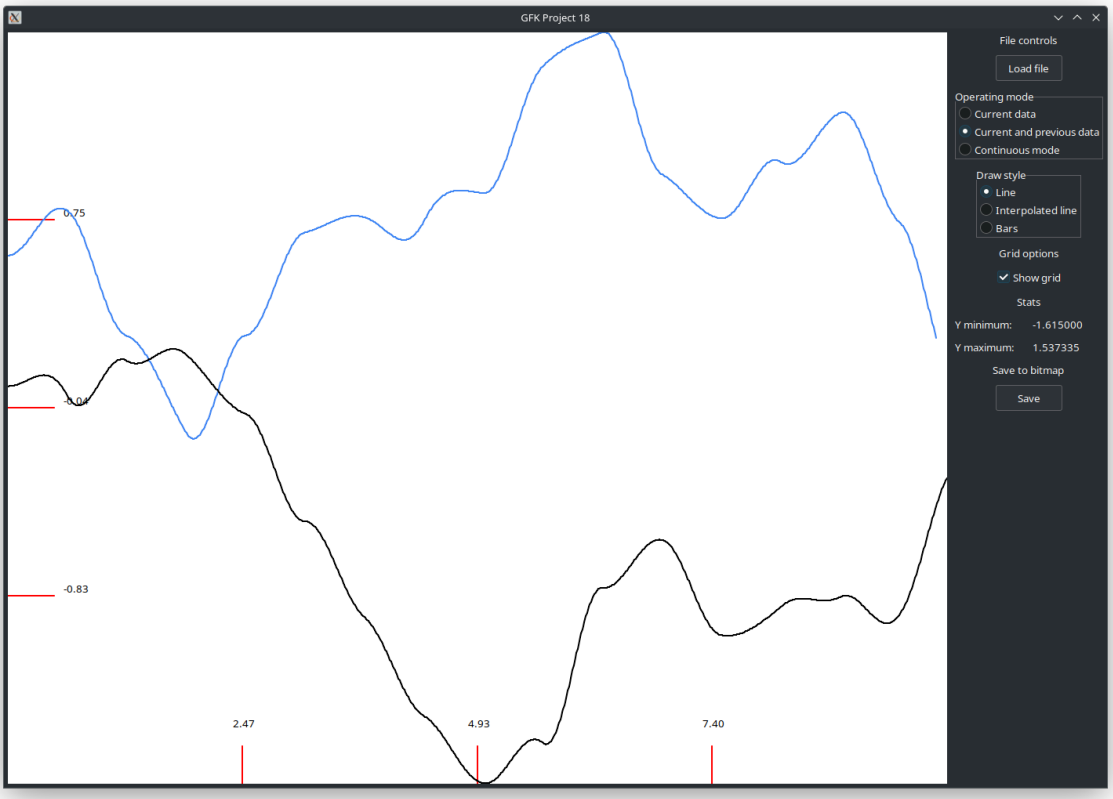
Testowanie

Testowanie odbywa się na podstawie wygenerowanych danych. Dane są pseudolosowe, co sprawia, że przy wystarczająco wysokiej ilości iteracji sprawdzony zostaje cały szereg różnych wartości.

Przed próbą na pliku generowanym automatycznie zasilono program poniższymi danymi:

```
0.000000 -0.034742
0.010000 -0.034777
0.020000 -0.034881
0.030000 -0.035055
0.040000 -0.035296
0.050000 -0.035603
0.060000 -0.035976
0.070000 -0.036411
0.080000 -0.036906
0.090000 -0.037459
0.100000 -0.038067
0.110000 -0.038725
0.120000 -0.039431
0.130000 -0.040179
0.140000 -0.040966
0.150000 -0.041786
...
```

Otrzymując poniższy wykres:



Wdrożenie, raport i wnioski

Udało nam się zaimplementować wszystkie podstawowe założenia projektu. Program sprawnie odczytuje dane z generowanego pliku oraz poprawnie wyświetla zarówno sam sygnał bieżący jak i w towarzystwie archiwalnych. Nasza implementacja bufora ewoluowała w trakcie rozwoju projektu, znacznie zwiększając jego efektywność oraz niezawodność.

Co więcej udało nam się zaimplementować również rozszerzone założenia. Jest to możliwość przechowywania oraz wyświetlania pełnej historii przebiegów. Dodaliśmy również możliwość zapisu obrazu do pliku .bmp.