Sprawozdanie końcowe -
przebieg projektu z przedmiotu:
Programowanie Komputerów

Temat: Prosty prezenter wykresów funkcji matematycznych

Rok akademicki: 2022/2023	Prowadzący zajęcia: dr inż. Łukasz Maliński

Wydział Inżynierii Materiałowej Politechniki Śląskiej

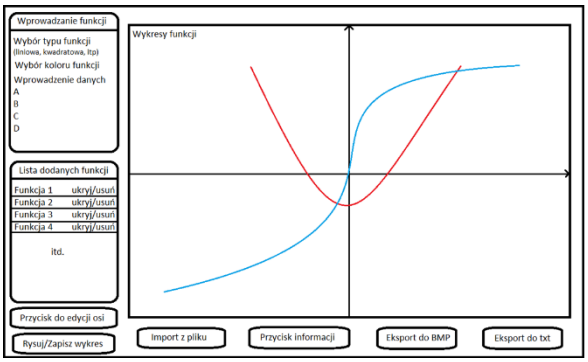
Autorzy:

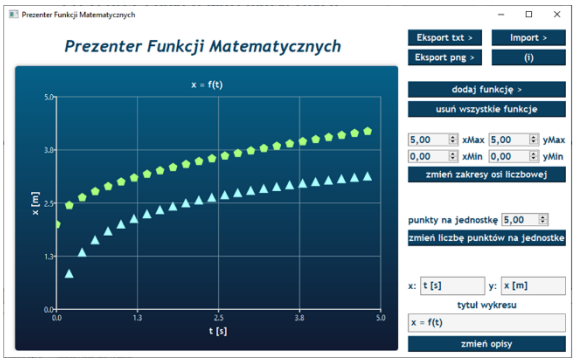
Informatyka Przemysłowa profil praktyczny – semestr 3, sekcja 2

Damian Barczak IPpp22
Wiktor Kobryń IPpp22

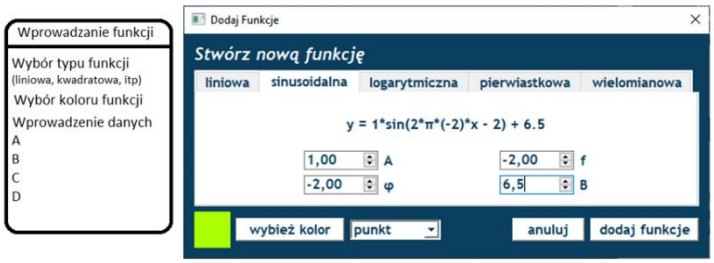
Zakres zmian jakie nastąpiły od pierwotnego projektu do wersji finalnej:

1) Przebudowa wyglądu GUI, oraz rozmieszczenia kontrolek i przycisków. Zmiana strony zagnieżdżenia layoutów z przyciskami ze względu na większy komfort korzystania z programu.

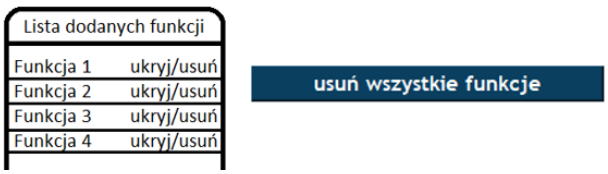




2) Przeniesienie fragmentu odpowiedzialnego za wybór typu funkcji i jej parametrów w oknie głównym do dodatkowego okna wyświetlającego się po kliknięciu przycisku *„dodaj funkcję >*”.



3) Rezygnacja z listy wyświetlającej dodane funkcje i możliwości ich pojedynczego dodawania i ukrywania. Ze względu na chaos jaki tworzy się na wykresie po dodawaniu znacznej ilości funkcji, które również zmieniają się z dostosowywaniem zakresu osi lub punktów na jednostkę, dla bardziej klarownego wyglądu GUI oraz szybszego użytkowania programu dodano jedynie możliwość usuwania zbiorczo wszystkich funkcji na wykresie poprzez przycisk „*usuń wszystkie funkcje*”.

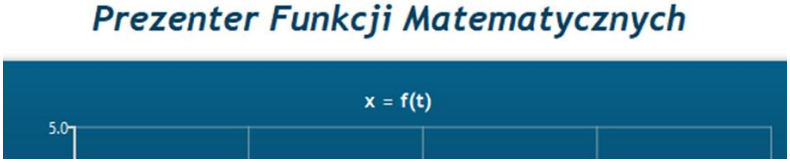


4) Zmieniono format zapisu pliku – do graficznego eksportu wykresu – z *.bmp do *.png.

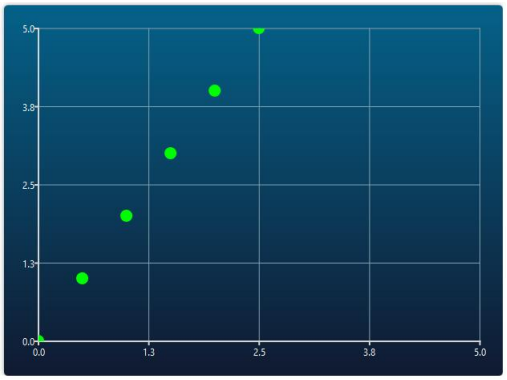
5) Zrezygnowano z możliwości ukrywania osi na wykresie – bez nich jego zawartość traci sens, nie jest to wtedy wykres w sensie matematycznym.

6) Zmieniono projekt kolorów GUI z jasnych na wariacje bieli, szarości oraz granatu również ze względu na dopasowanie reszty kontrolek do jednego z dostępnych w Qt styli QChart – *Blue Cerulean.*

7) Dodano możliwość zmiany opisów osi x i y, jak i tytułu samego wykresu.



8) Zrezygnowano z interpolacji pokazywanych wykresów – punktowy charakter funkcji oraz narastanie nagromadzenia punktów wraz z jej pochodną, subiektywnie znacznie bardziej pasuje do przyjętego stylu i wyglądu programu. Jednakże dodana została możliwość zmiany rodzaju znacznika punktu dla danej funkcji.



9) Rezygnacja z własnych tekstur dla kilku przycisków (min. „*usuń*”, „*pokaż*”)– dokonano testu działania co widoczne jest w kilku prezentacjach z zajęć, jednak nie zwiększało to czytelności GUI, mając wręcz przeciwne skutki.

10) Dziedziczenie klas zostało rozbudowane – klasa abstrakcyjna Funkcja dziedziczy po QScatterChart co ułatwia korzystanie z pozostałych funkcji, i podpinanie ich do dziedziczącej po QChart klasie OsLiczbowa.

11) Dokonano refaktoryzacji i ujednolicenia schematu nazewnictwa dla klas w programie co było sporym wyzwaniem już nawet przy dwuosobowym zespole.

12) Dodano komunikaty błędu o nieprawidłowych danych przy wprowadzaniu maksymalnych/minimalnych wartości osi x i y, oraz blokadę przeciwko wpisywaniu zabronionych wartości jak np. problem logarytmu z 0.

13) Rezygnacja z przycisku „Import” jako, że zabrakło czasu na sprawienie by algorytm odczytywania działał w pełni poprawnie.

Zakres prac wykonanych przez każdego członka sekcji:

Damian Barczak:
- opracowanie działania zapisu wykresów w pliku graficznym
- opracowanie działania zapisu punktów i wzorów funkcji w pliku tekstowym
- klasy funkcji – np. class Fliniowa, Fpierwiastkowa, itd.
- konstruktory, destruktory tych klas
- niektóre metody powyższych klas – np. tablica punktów i jej wypełnianie addToVector(), obliczWartFun() – obliczająca punkty ze wzorów
- przycisk „(i)” – QmessageBox pokazujący informację o autorach i programie
- naprawa drobnych błędów
- testowanie i szukanie błędów

Wiktor Kobryń:

- projekt GUI, dostosowanie kolorów i rozmieszczenia przycisków
- przebudowa programu do architektury 2-warstwowej (MainWindow – prezentacja, Projekt - sterowanie)
- opracowanie działania okna dodawania nowych funkcji
- zmiana opisów osi i wykresu
- responsywność GUI i reakcje na zmiany użytkownika
- skalowanie osi, zmiana liczby punktów na jednostkę
- usuwanie funkcji
- naprawa błędów

Ocena zdobytych doświadczeń przy realizacji projektu:

Damian Barczak:

- Praca w grupie jest dużo prostsza, zwłaszcza przy wykorzystaniu komunikatorów głosowych (np. Discord) oraz możliwości udostępnienia ekranu. W sytuacji gdy brakowało nam pomysłu lub trzeba było wyszukać błędu, spotkanie się razem na czacie głosowym, wraz z udostępnionym ekranem, może dużo pomóc.
- Praca z dokumentacją. Prawie przez cały czas trzeba było mieć włączoną przeglądarkę z dokumentacją (w moim przypadku to było głównie Qt Documentation albo cplusplus), gdyż znacznie ułatwiało to pracę.
- Biblioteka STL. Głównie używanie list lub vector. Jako przykład, znacznie łatwiej było zrobić vector<QString> tablicaPkt, gdzie można było po prostu użyć push_back(), niż mój pierwotny pomysł zrobienia dynamicznej tablicy tych punktów.
- Architektura 2-warstowowa jest przyjaźniejsza dla pracy. O wiele łatwiej było móc wprowadzić zmiany w którymś algorytmie bez jednoczesnego naruszania GUI. Na przykład, w pamiętniku z zajęć w pewnym momencie trudno było mi wprowadzić zmiany bez uszkodzenia działania programu.
- Dowiedziałem się, że bardziej lubię programować razem z GUI, jest to o wiele bardziej satysfakcjonujące jeśli rozwiąże się problem i poźniej można graficznie zobaczyć działanie programu, rozwiązanego problemu.
- Pogląd na to jak wygląda praca w zespole, praca przy użyciu github'a. Programowanie gier czy aplikacji nie wydaje się już tak bardzo trudne. Chociaż często zdarzało się, że musieliśmy się pytać poprzez komunikator (tj. Discord) czy druga osoba aktualnie pracuje nad programem. Było tak z racji tego, że jeśli zostały wprowadzone jakieś istotne zmiany (ale jeszcze nie „spushowane") to lepiej było zapytać i potem pracować na aktualnej wersji programu niż na starszej.

Wiktor Kobryń:

- Przeglądanie dokumentacji i znajdywanie najbardziej opłacalnego rozwiązania problemu biorąc pod uwagę ramy czasowe i możliwości / własne umiejętności – szczególnie w początkowych fazach projektu, gdy wizja funkcjonowania samych wykresów i osi współrzędnych kilkukrotnie się zmieniała np. przy próbach wdrożenia QCustomPlot.
- Używanie większej ilości widgetów w Qt jak np. QTabWidget, edytowanie ich (przez dziedziczenie np. w przypadku problemu klasy abstrakcyjnej Funkcja) na własne potrzeby tak aby mogły wciąż współpracować z resztą gotowych narzędzi Qt lub biblioteki stl.
- Dostosowywanie GUI w taki sposób aby było przede wszystkim przyjemne w użyciu i funkcjonalne, co jednak nie udało się w projekcie w takim stopniu jak chciałem.
- Warto w pierwszej kolejności pisać program w zamierzonej architekturze, zamiast przenoszenia i edytowania całej jego zawartości do architektury 2-warstwowej co pochłonęło znacznie więcej czasu i pracy niż się spodziewałem, nawet przy niewielkiej ilości funkcjonalności.
- Szukanie błędów w kodzie jest znacznie prostsze pracując w zespole, szczególnie z chat-em głosowym co uczy również pisania kodu w zrozumiały i czytelny sposób aby właśnie ułatwić pracę w tym przypadku drugiej osobie i znacznie ułatwia późniejsze rozbudowywanie programu i dodawanie nowych funkcjonalności. Jednakże pracowanie w repozytorium wyłącznie na main branch jest mało komfortowe nawet przy zaledwie 2 osobach.
- Również z samego matematycznego punktu widzenia (problemy z „niewidzialnym" sinusem) dowiedziałem się sporo o samej budowie i działaniu programów pokroju Photomath lub GeoGebra, które często stosuje podczas prowadzenia korepetycji, mając teraz inną alternatywę przy prezentacji wykresów funkcji.