Wykresy 3D, funkcji trzech zmiennych we współrzędnych sferycznych - nr 33.

Jakub Jucha, Jan Pazdan, Patryk Hołubowicz

18 Czerwca 2024

1 Opis projektu

Projekt ma na celu stworzenie programu do rysowania wykresów różnych funkcji $f(r, \theta, \phi)$ we współrzędnych sferycznych. Program posiada dwa główne tryby rysowania oraz opcje zapisu wyników do pliku graficznego.

2 Założenia wstępne przyjęte w realizacji projektu

2.1 Wymagania podstawowe

W wersji podstawowej program pozwala na rysowanie funkcji w dwóch postaciach. W pierwszej rysowana jest sfera składająca się z punktów. Kolor punktów zależy od tego jaką wartość posiada funkcja w danym punkcie (minimum – kolor niebieski, maksimum – kolor czerwony). Użytkownik ma możliwość wyboru parametru r oraz liczby punktów podziału zakresu zmienności parametrów θ i ϕ. W drugim trybie program rysuje również punkty, lecz tym razem punkty są oddalone od środka układu współrzędnych tym dalej im większa wartość funkcji w danym punkcie (dla ustalonego r) - dla minimum funkcji punkt pokrywa się z początkiem układu współrzędnych. Oczywiście można dowolnie zmieniać kąt, pod którym patrzymy na wykres. W każdej chwili wynik można zapisać na dysk do pliku graficznego.

2.2 Wymagania rozszerzone

Program pozwala na wpisanie wzoru rysowanej funkcji w trakcie jego działania. Suwak pozwalający przybliżać oraz oddalać rysowaną geometrię. Zaimplementowany został system obsługi obracania obiektu za pomocą przeciągnięcia myszką po głównym panelu rysowania. Użycie kółka myszki na głównym panelu rysowniczym powoduje obsługę suwaka przybliżająco-oddalającego.

3 Analiza projektu

3.1 Specyfikacja danych wejściowych

- parametr promienia r
- liczba punktów podziału zakresu zmienności - θ i ϕ (kąty)
- obrót po osiach X,Y,Z (według kątów)
- wzór funkcji
- przybliżenie kształtu

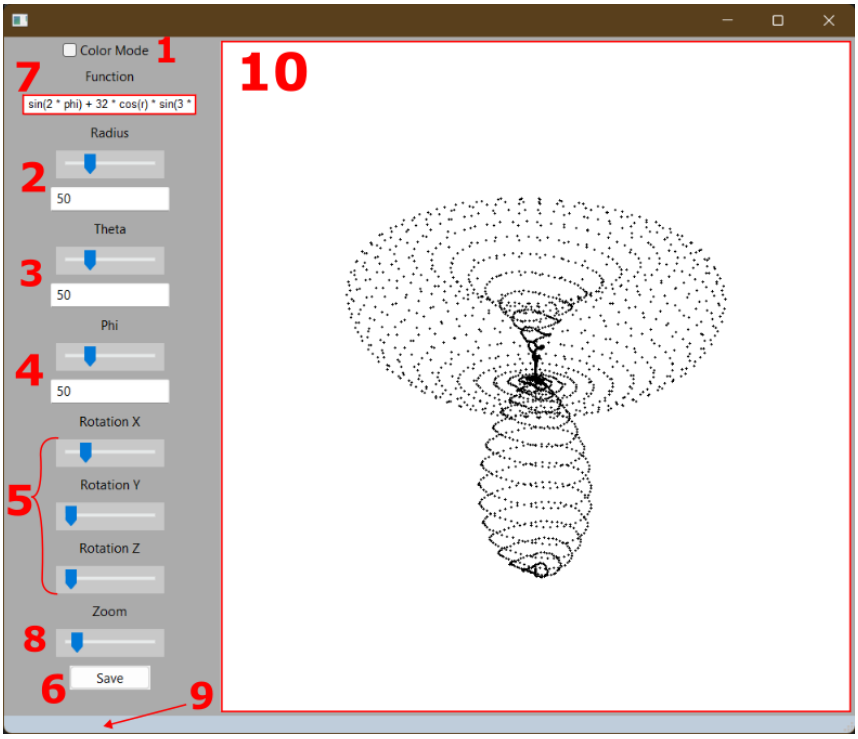
3.2 Opis oczekiwanych danych wyjściowych

- wykresy 3D w dwóch postaciach (różnokolorowe punkty i punkty oddalone od środka)
- plik graficzny z zapisem wykresu

3.3 Zdefiniowane struktur danych

- *Matrix* - struktura przechowująca macierz. Składa się z dwóch wskaźników na tablice dynamiczną jednowymiarową, reprezentujących wiersze i kolumny macierzy.
- *Vector* - struktura przechowująca wektor. Składa się z wskaźnika na tablicę dynamiczną jednowymiarową, reprezentującą elementy wektora.
- Ponadto, w pliku *GUIMyFrame1.h* zdefiniowana jest struktura *MyFrame1*, która jest klasą okna aplikacji.

3.4 Specyfikacja interfejsu użytkownika



Rysunek 3.1: Enter Caption

1. Zmiana trybu rysowania z odległościowego na sferę z kolorami reprezentującymi wartość.
2. Suwak oraz pole do zmiany promienia r.
3. Suwak oraz pole do zmiany kąta θ.
4. Suwak oraz pole do zmiany kąta ϕ.
5. Suwaki odpowiadające za obrót obiektu.
6. Przycisk zapisu rysowanego obiektu.
7. Pole do wpisania wzoru funkcji.
8. Suwak do przybliżania i oddalania obiektu.
9. Pasek statusu informujący o błędzie we wpisywanej funkcji.
10. Panel rysujący obiekt o zadanych parametrach, obsługujący przeciąganie myszą oraz kółko.

3.5 Wyodrębnienie i zdefiniowane zadań

- Projekt i realizacja wyglądu interfejsu
- Implementacja zmiany współrzędnych sferycznych na kartezjańskie
- Implementacja rysowania przykładowego obiektu.
- Implementacja podstawowych funkcjonalności interfejsu.
- Optymalizacja rysowania geometrii.
- Implementacja rozszerzonych funkcjonalności.

3.6 Decyzja o wyborze narzędzi programistycznych

W celu zrealizowania projektu zostało wybrane środowisko *Visual Studio*. Program był pisany w języku *C++*, ze względu na jego dobrą znajomość. Główna wykorzystywana biblioteka to *wxWidgets*, która pozwala na tworzenie aplikacji okienkowych oraz interpretacji graficznej.

4 Podział pracy i analiza czasowa

4.1 Podział pracy

- Jakub Jucha: Wymagania podstawowe, optymalizacja projektu
- Jan Pazdan: Wymagania rozszerzone, interfejs użytkownika
- Patryk Hołubowicz: Dokumentacja

4.2 Analiza czasowa

- Projekt i realizacja wyglądu interfejsu - 2h
- Implementacja zmiany współrzędnych sferycznych na kartezjańskie - 1h
- Implementacja rysowania przykładowego obiektu - 6h
- Implementacja podstawowych funkcjonalności interfejsu - 6h
- Optymalizacja rysowania geometrii - 8h
- Implementacja rozszerzonych funkcjonalności - 5h
- Dokumentacja - 2h

5 Opracowanie i opis niezbędnych algorytmów

W projekcie wykorzystano następujące algorytmy:

- Przejście ze współrzędnych sferycznych na kartezjańskie.
- Zdefiniowanie operacji na macierzach i ich podstawowych transformacji tj. translacja, skala oraz obrót.
- Zastosowanie algorytmu malarza.
- Optymalizacja struktur przechowujących punkty.
- Wykorzystanie bezpośredniego dostępu do bitmapy w celu optymalizacji rysowania punktów.
- Wykorzystanie zewnętrznej biblioteki *tinyexpr* do interpretacji wpisanego wyrażenia na funkcję matematyczną.

6 Kodowanie

Opis kodu wygenerowany za pomocą Doxygen'a. Katalog *doxygen/index.html* .

7 Testowanie

7.1 testowanie niezależnych bloków

Sprawdzenie wyglądu wygenerowanego GUI. Testowanie poprawności obrotu wokół każdej osi osobno. Sprawdzenie prawidłowości działania opcji zapisu wyniku do pliku.

7.2 testy powiązań bloków

Sprawdzenie jednoczesnego obrotu kształtu różnych osi i zapisu wyniku do pliku.

7.3 testy całościowe

Zmiany promienia, kątów θ i ϕ , jednoczesny obrót kształtu wokół osi X,Y,Z za pomocą myszy oraz przybliżenie kształtu i zapis wyniku do pliku.

7.4 Wdrożenie, raport, wnioski

Program uruchomiony z domyślnymi danymi działa poprawnie. Udało się zrealizować wszystkie z założonych zadań podstawowych jak i zadań rozszerzonych. Nie udało się wdrożyć obrotu wokół osi tak, aby osie były niezależne od siebie ze względu na ograniczenia matematyczno-logiczne powiązane z brakiem wiedzy o kwaternionach. Skorzystanie z biblioteki wxWidgets przyspieszyło pracę nad interfejsem użytkownika, lecz również ten wybór wymusił dodatkową pracę związaną z optymalizacją działania programu w czasie rzeczywistym.