16.06.2024, Kraków

WYKRESY 3D

PROJEKT NR 007 - dokumentacja

Anna Nowak
Bartosz Fryska
Aleksander Kopyto

1. Tytuł projektu i autorzy projektu

W ramach kursu „Podstawy Grafiki Komputerowej” mieliśmy za zadanie wykonać **projekt o numerze 007 – „Wykresy 3D”** w grupie trzyosobowej.

Członkowie zespołu i podział pracy:

a) Anna Nowak
   1. zaprojektowanie i przygotowanie GUI projektu
   2. funkcjonalności związane z interakcją pomiędzy użytkownikiem, a programem
   3. możliwość druku oraz zapisu wykresu w postaci zdjęcia w formacie .png
   4. algorytm mapy konturowej
   5. skalowalność aplikacji
   6. ankieta dla użytkowników testujących aplikację
   7. przygotowanie dokumentacji

b) Bartosz Fryska
   1. algorytm rzutu perspektywicznego
   2. algorytm mapy konturowej
   3. obrót wykresu w postaci rzutu perspektywicznego, jego przybliżanie i oddalanie
   4. generowanie wykresów w postaci rzutu perspektywicznego i mapy konturowej w oparciu o dane pobrane od użytkownika
   5. pomoc przy tworzeniu dokumentacji

c) Aleksander Kopyto
   1. testy programu jako całości.

2. Opis projektu

Celem projektu było stworzenie programu generującego wykresy funkcji $f(x, y)$ w postaci mapy konturowej lub rzutu perspektywicznego w zależności od wyboru użytkownika.

3. Założenia wstępne przyjęte w realizacji projektu

3.1. wymagania podstawowe

Użytkownik ma możliwość wprowadzenia funkcji w postaci tekstowej, ustalenia jej przedziałów względem zmiennej x, y oraz wartości minimalnej i maksymalnej funkcji (z_{min} i z_{max}). W zależności od wyboru użytkownika możliwe jest generowanie wykresu w postaci rzutu perspektywicznego oraz mapy konturowej. Istnieje możliwość druku wykresu oraz zapisania go w postaci pliku graficznego w formacie .png.

3.2. wymagania rozszerzone

Wykres w postaci rzutu perspektywicznego można obracać za pomocą ruchu myszy po panelu, gdzie generowany jest wykres, przy jednoczesnym przytrzymaniu lewego

klawisza myszy. Przy pomocy scroll'a w myszy możliwe jest przybliżanie i oddalanie wykresu.

Interfejs użytkownika jest intuicyjny i przejrzysty, a nad polem tekstowym służącym do wprowadzania funkcji został umieszczony przycisk, którego naciśnięcie powoduje wyświetlenie informacji dotyczącej wskazówek co do poprawności formatu wprowadzanej przez użytkownika funkcji.

Poprawność wszystkich danych wprowadzanych przez użytkownika przed wygenerowaniem wykresu jest sprawdzana i w przypadku wystąpienia błędów pola tekstowe, gdzie znajdują się błędne dane, są czyszczone.

Zdecydowaliśmy się dodać gradient kolorów do mapy konturowej, który obrazuje zmianę wysokości punktów na mapie (kolor niebieski – punkty najniżej położone, kolor czerwony – punkty najwyżej położone).

4. Analiza projektu

4.1. specyfikacja danych wejściowych

Program przyjmuje od użytkownika funkcję, obszar zmienności dla argumentów x i y oraz wartość minimalną i maksymalną funkcji widoczną na wykresie w formie tekstowej (std::string). Następnie funkcja analizowana jest przy pomocy biblioteki TinyExpr, dzięki czemu może zostać wykorzystana do wygenerowania tablicy wartości badanej funkcji. Zakres argumentów x i y oraz z_{min} i z_{max} konwertowany jest do wartości typu double. Oprócz tego program umożliwia ustalenie formy wyświetlanego wykresu przy pomocy naciśnięcia przez użytkownika odpowiedniego wxRadioButton. Wszystkie z powyższych ustawień są możliwe do zmiany przez użytkownika dowolną ilość razy przez cały czas działania programu.

Na panelu dostępne są przyciski „generate" – generuje wykres, „print" – otwiera okno dialogowe służące do drukowania, „save" – otwiera okno dialogowe umożliwiające zapis pliku w formacie .png.

Ponadto program pobiera dane na temat tego, czy mysz użytkownika znajduje się w polu gdzie generowany jest wykres i czy lewy przycisk myszy jest naciśnięty. Na tej podstawie możliwa jest manipulacja położeniem wykresu w postaci rzutu perspektywicznego. Oprócz tego pobierane są również dane dotyczące ruchu scroll'em myszy, co przekłada się na przybliżanie i oddalanie rzutu perspektywicznego o ile kursor znajduje się na panelu z wykresem.

4.2. opis oczekiwanych danych wyjściowych

Aplikacja generuje dowolny, wybrany przez użytkownika wykres w postaci rzutu perspektywicznego bądź mapy konturowej na panelu do tego przeznaczonym. Pozwala również na zapisanie wykresu w formacie .png oraz druk wykresu.

4.3. zdefiniowanie struktur danych

W celu przechowywania danych takich jak wartości funkcji uzyskane przy pomocy biblioteki TinyExpr (*GUIMyFrame1.h*), wykorzystano klasę C++ STL – vector. Jest to

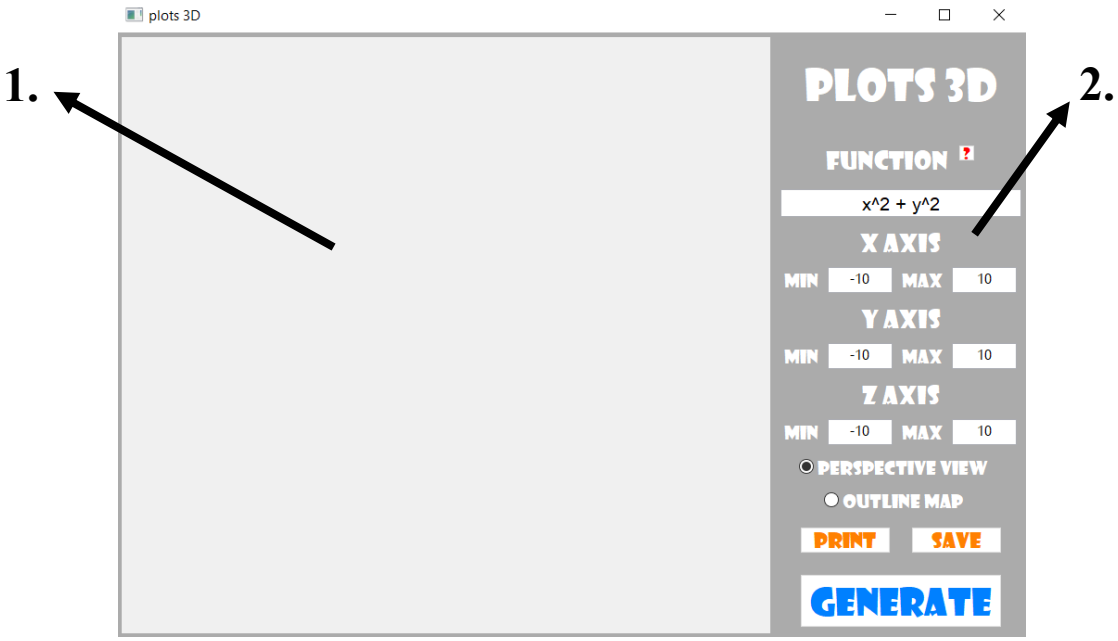
rodzaj kontenera, który ma zdolność do automatycznego powiększania swojej pojemności co czyni go bardzo elastycznym. Klasy vector użyto również do przechowywania odcinków służących do konstrukcji rzutu perspektywicznego (*perspectivic.h*) czy wartości punktów niezbędnych do generowania mapy (*map.h*).

W sytuacjach kiedy przez cały okres działania programu ilość przechowywanych danych była niezmienna w celu oszczędzenia używanej pamięci i optymalizacji działania programu wykorzystano tablice jedno- bądź dwuwymiarowe o stałej zdefiniowanej wielkości. Miało to miejsce przykładowo przy przechowywaniu punktów (x, y, z) dla rzutu perspektywicznego i mapy (*perspectivic.h*, *map.h*).

Do przechowywania danych odnośnie odcinków budujących rzut perspektywiczny wykorzystano struktury zawierające parametry punktów tworzących dane odcinki (*perspectivic.h – struct Segment*). Natomiast chcąc uzyskać obrót generowanego wykresu skorzystano z techniki rzutu perspektywicznego, co wiązało się z użyciem macierzy transformacji zbudowanych w oparciu o tablice dwuwymiarowe.

4.4. specyfikacja interfejsu użytkownika

Interfejs użytkownika został zaprojektowany tak żeby był jak najbardziej intuicyjny i poruszanie się po nim nie sprawiało trudności użytkownikowi.

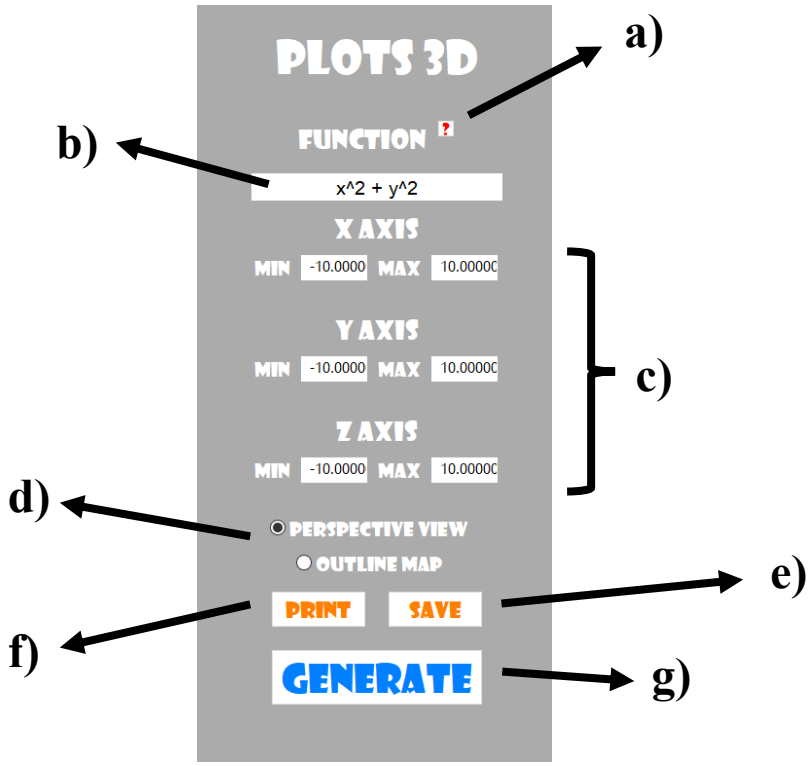


Zdjęcie 1. Okno aplikacji - start.

Istnieje możliwość zmiany wymiarów okna – wyświetlane wykresy regenerują się przy każdej zmianie wielkości panelu rysowania. Lewy panel (1.) dopasowuje się do każdego ustawionego przez użytkownika rozmiaru okna natomiast prawy panel (2.) ma ustaloną minimalną wysokość oraz szerokość co zabezpiecza przed złym rozmieszczeniem i zachodzeniem na siebie poszczególnych jego komponentów

w przypadku zbytniego zmniejszenia okna przez użytkownika. Komponenty na panelu 2. mają stałą wielkość w celu zachowania spójnego wyglądu aplikacji.

Panel 1. służy do wyświetlania wygenerowanych funkcji. Pobierane są dane na temat tego czy mysz znajduje się w obrębie panelu, współrzędne jej położenia oraz czy lewy przycisk myszy jest naciśnięty. Dane te pozwalają na implementację funkcji obracającej wykres w postaci rzutu perspektywicznego kiedy użytkownik przyciska lewy klawisz myszy i przesuwa kursor w obrębie panelu. Kolejną funkcjonalnością jest możliwość powiększania i pomniejszania rzutu perspektywicznego w oparciu o dane na temat użycia przez użytkownika scroll'a myszy o ile kursor znajduje się w obrębie panelu 1.



Zdjęcie 2. Panel 2 programu.

Na panelu 2. w celu interakcji z użytkownikiem znajdują się kolejno:

a) **wxButton** *buttonInfo* odpowiadający za wyświetlenie informacji ze wskazówkami dotyczącymi poprawnego wprowadzenia funkcji przez użytkownika
b) **wxTextCtrl** *textCtrlFunkcja* umożliwiający wprowadzenie funkcji przez użytkownika przy użyciu klawiatury, w przypadku jeśli wprowadzone dane będą niepoprawne pole zostanie wyczyszczone i wykres nie zostanie wygenerowany
c) **wxTextCtrl** *textCtrlXMin, textCtrlXMax, textCtrlYMin, textCtrlYMax, textCtrlZMin, textCtrlZMax* służące do ustalenia zakresu argumentów x, y oraz wartości mininalnej i maksymalnej funkcji, w przypadku kiedy podany przedział będzie niepoprawny (np. $x_{min} >= x_{max}$) pola tekstowe z błędnym parametrami zostaną wyczyszczone

d) **wxRadioButton** *radioRzut, radioMapa* pozwalają na wybranie rodzaju wykresu, który ma zostać wyświetlony, jednocześnie tylko 1 z opcji może zostać wybrana
e) **wxButton** *buttonSave* jego kliknięcie powoduje przekierowanie do okna dialogowego zapisu wykresu w formie obrazu w formacie .png
f) **wxButton** *buttonPrint* jego kliknięcie powoduje przekierowanie do okna dialogowego druku wykresu
g) **wxButton** *buttonGenerate* umożliwia wygenerowanie wykresu o określonych przez użytkownika parametrach, jeśli którykolwiek z parametrów jest błędnym wykres nie zostanie wygenerowany, a pola tekstowe z błędnymi danymi zostaną wyczyszczone.

4.5. wyodrębnienie i zdefiniowanie zadań

Kolejne działania podjęte w celu realizacji projektu:

1. Analiza tematu projektu, dyskusja nad możliwymi rozwiązaniami w oparciu o wiedzę zdobytą podczas kursu „Podstawy Grafiki Komputerowej".
2. Konsultacja z prowadzącym przedmiot w celu ustalenia najlepszych środków do realizacji zadania, doprecyzowania funkcjonalności związanej z drukowaniem wykresów oraz możliwości użycia bibliotek pozwalających na konwertowanie funkcji wprowadzanej przez użytkownika z typu std::string do formy umożliwiającej wykonywanie operacji matematycznych z wykorzystaniem pozyskanej funkcji.
3. Rozdział pracy i ustalenie terminów do których poszczególne części projektu mają zostać zrealizowane.
4. Utworzenie repozytorium na GitHub razem z odpowiednim plikiem *.gitignore*.
5. Zaprojektowanie interfejsu graficznego w wxFormBuilder.
6. Praca nad klasami obsługującymi generowanie rzutu perspektywicznego (*perspectivic.h*), mapy konturowej (*map.h*) oraz wprowadzenie wszystkich niezbędnych funkcjonalności interfejsu graficznego (*GUIMyFrame1.h*) takich jak pobieranie danych wprowadzonych przez użytkownika, drukowanie i zapisywanie wykresów w formacie .png.
7. Uzależnienie generowanych wykresów od parametrów pobranych od użytkownika – połączenie funkcjonalne między *perspectivic.h*, *map.h* oraz *GUIMyFrame1.h*.
8. Analiza dotychczas uzyskanych efektów, dyskusja nad możliwymi poprawkami i dodaniem dodatkowych funkcjonalności. Przeprowadzenie testów poprawności działania programu.
9. Praca nad obrotem i przybliżaniem rzutu perspektywicznego przy użyciu ruchów myszki, wprowadzenie poprawek do interfejsu użytkownika, praca nad optymalizacją generowania mapy konturowej i dodanie gradientu kolorów.
10. Testowanie poszczególnych funkcjonalności aplikacji w oparciu o scenariusze testowe, wprowadzenie ostatecznych poprawek w celu optymalizacji działania aplikacji.
11. Testy aplikacji na użytkownikach i zbieranie opinii z wykorzystaniem formularza.
12. Sporządzenie dokumentacji.

4.6. decyzja o wyborze narzędzi programistycznych

W celu umożliwienia efektywnej i płynnej pracy grupowej zdecydowaliśmy się na użycie serwisu GitHub wykorzystującego system kontroli wersji Git. Umożliwiło nam to pracę nad różnymi częściami projektu jednocześnie, co zapewniło największą efektywność osiągania kolejnych postawionych sobie celów i łatwość wymieniania się nawzajem zaimplementowanymi zmianami.

Najważniejszą biblioteką z jakiej korzystaliśmy w naszym projekcie jest wxWidgets. Umożliwia ona tworzenie graficznych interfejsów użytkownika. Potrzebowaliśmy zatem kompilatora, który będzie wspierał tą biblioteką i w tym celu wybraliśmy darmowym środowisku programistycznym Microsoft Visual Studio, które jest polecane przez twórców wxWidgets jako szybkie i efektywne IDE do pracy z tą biblioteką.

W celu konwersji funkcji pobieranej od użytkownika jako ciąg znaków wykorzystaliśmy TinyExpr – prosty w użyciu i szybki parser wyrażeń matematycznych. TinyExpr pozwoliło na o wiele większą dowolność w generowaniu funkcji przez użytkownika, co równocześnie zwiększyło użyteczność aplikacji.

5. Podział pracy i analiza czasowa

Wykonanie programu podzieliliśmy na 3 główne etapy, w którym każdy z członków zespołu miał swoje zadania do wykonania.

1. **Zbieranie pomysłów i projektowanie:**
    a. Anna Nowak – zaprojektowanie GUI aplikacji i zebranie informacji dotyczących możliwości drukowania wykresu przy pomocy biblioteki wxWidgets
    b. Bartosz Fryska - zebranie informacji na temat możliwych rozwiązań dotyczących generowania wykresów w postaci rzutu perspektywicznego oraz mapy konturowej.
2. **Stworzenie aplikacji spełniającej założenia podstawowe:**
    a. Anna Nowak – stworzenie interfejsu użytkownika (*GUI.h*), zaimplementowanie interakcji program – użytkownik (*GUIMyFrame1.h*) oraz napisanie algorytmu generującego mapę konturową (*map.h*)
    b. Bartosz Fryska – stworzenie algorytmu generującego rzut perspektywiczny (*perspectivic.h*) oraz połączenie funkcjonalne pomiędzy danymi pobieranymi od użytkownika, a generowanymi wykresami (*map.h, perspectivic.h*).
3. **Wprowadzenie poprawek, dodanie dodatkowych funkcjonalności i testowanie:**
    a. Anna Nowak – pobieranie danych dotyczących ruchu, kliknięć i scroll'a myszy w celu zaimplementowania możliwości poruszania oraz przybliżania wykresu w postaci rzutu perspektywicznego, wprowadzenie poprawek dotyczących skalowalności okna aplikacji oraz wychwytywania błędów w danych wprowadzanych przez użytkownika

b. Bartosz Fryska – implementacja możliwości poruszania, przybliżania i oddalania wykresu w postaci rzutu perspektywicznego, dodanie gradientu kolorów podczas generowania mapy konturowej
c. Aleksander Kopyto – wykonanie testów całościowych.

4. **Wykonanie dokumentacji:**
   a. Anna Nowak – napisanie i zredagowanie całości dokumentu
   b. Bartosz Fryska – pomoc przy opracowaniu punktów dokumentacji dotyczących użytych algorytmów
   c. Aleksander Kopyto – zamieszczenie informacji dotyczących wyników przeprowadzonych testów.

6. Kodowanie

6.1. *GUI.h* – interfejs użytkownika

Zawiera klasę MyFrame1 zawierającą wszystkie komponenty składające się na interfejs użytkownika. Do pobierania danych dotyczących funkcji, zakresu jej parametrów i wartości wykorzystano obiekty klasy wxTextCtrl pobierające dane w postaci ciągu znaków typu std::string. Obiekt wxPanel został wykorzystany jako pole generowania wykresów i zostały do niego dołączone funkcje pobierające informacje na temat tego czy kursor myszy znajduje się w obrębie panelu, jakie jest położenie kursora myszy, czy lewy klawisz myszy jest naciśnięty oraz czy scroll myszy został użyty.

6.2. *GUIMyFrame1.h* – interakcja użytkownik - program

Zawiera klasę GUIMyFrame1 dziedziczącą publicznie po klasie MyFrame1. Wewnątrz tej klasy zostały opisane wszystkie interakcje użytkownika z interfejsem aplikacji utworzone w klasie bazowej. Zawiera się w tym pobieranie danych dotyczących generowania wykresu dostarczonych przez użytkownika oraz sprawdzanie ich poprawności, skalowanie wykresów przy zmianie wielkości okna aplikacji, generowanie wykresów w oparciu o wybrany przez użytkownika typ wykresu, umożliwienie drukowania oraz zapisu wykresu do pliku w formacie .png. Co więcej klasa zawiera również funkcje pobierające dane na temat położenia, ruchu, kliknięć i użycia scroll'a myszy. Te parametry są przechowywane i pozwalają na wykonanie operacji obrotu, powiększania i pomniejszania wykresu w postaci rzutu perspektywicznego. W GUIMyFrame1 wykorzystywana jest również biblioteka TinyExpr służąca do analizy funkcji wprowadzonej przez użytkownika, co umożliwia uzyskanie wektora wartości niezbędnych do wygenerowania obrazu funkcji.

6.3. *MyPrinter.h* – obsługa drukowania wykresu

Zawiera klasę ImagePrintout, która obsługuje funkcję drukowania wykresu. Klasa ta dziedziczy publicznie po klasie wxPrintOut należącej do biblioteki wxWidgets. Kiedy przycisk druku wykresu zostaje naciśnięty przez użytkownika polecenie to odbierane jest przez odpowiednią funkcję w klasie GUIMyFrame1. Tworzony jest obiekt klasy ImagePrintOut, a następnie dochodzi do sprawdzenia czy nie występują żadne błędy

związane z operacją druku wykresu. Jeśli wszystko jest w porządku wykonywana jest operacja drukowania rysunku wykresu przy użyciu drukarki lub na przykład do pliku w formacie .pdf.

6.4.*perspectivic.h* - rzut perspektywiczny

Zawiera klasę Perspectivic, wewnątrz której zostały zawarte wszystkie operacje potrzebne do transformacji funkcji oraz rysowania jej w postaci rzutu perspektywicznego w przedziałach, które zostały uprzednio podane przez użytkownika i poddane sprawdzeniu poprawności w klasie GUIMyFrame1. W celu poprawnego wygenerowania wykresu klasa potrzebuje:

a) pointer'a do obiektu klasy wxPanel, na którym wykres ma zostać rysowany
b) aktualnych wymiarów panelu, na którym na zostać narysowany wykres
c) wektora wartości funkcji w punktach (x, y), które są wyliczane z odpowiednim próbkowaniem
d) ilości przesunięć scroll'a (umożliwia przybliżanie i oddalanie wykresu funkcji)
e) kątów rotacji wykresu w osiach OX i OZ (kąty są wyliczane w GUIMyFrame1 dzięki danym na temat położenia myszy w obrębie panelu)
f) przedziałów rysowania funkcji we wszystkich 3 wymiarach: x_{min}, x_{max}, y_{min}, y_{max} oraz z_{min}, z_{max}.

Klasa korzysta również z obiektów klas Vector4 oraz Matrix4 pochodzących z *vecmat.h*. Opisują one 4-argumentowy wektor oraz macierz o wymiarach 4 x 4, które wykorzystane są do transformacji każdego punktu (x, y, z). Transformacje na punktach wykonywane są z pomocą macierzy przekształceń: skalowania, przesunięcia początkowego, obrotu końcowego wokół wszystkich osi, przesunięcia końcowego (na ten moment tylko przybliżenia i oddalenia) oraz operacji niezbędnych do wykonania rzutu perspektywicznego. Dla wartości funkcji wychodzących poza zakres $< z_{min}, z_{max} >$ wykres rysowany jest w kolorze przezroczystym.

Osie rysowane razem z wykresem przedstawiają jedynie kierunki, w których rozchodzi się wykres – oś czerwona jako OX, oś zielona jako OY oraz oś niebieska jako OZ.

6.5.*vecmat.h* – macierze przekształceń

Pliki *vecmat.h* oraz *vecmat.cpp* opisują dwie klasy: Matrix4 oraz Vecrtor4, których autorem jest dr inż. Janusz Malinowski (prowadzący kurs). Klasa Matrix4 reprezentuje macierz 4 x 4 z pomocą dwuwymiarowej tablicy i pozwala wykonywać na niej operacje mnożenia przez inny obiekt Matrix4 lub Vector4. Klasa Vector4 reprezentuje 4-argumentowy wektor za pomocą jednowymiarowej tablicy formatu [x_cordinate, y_cordinate, z_cordinate, 1.0] dla punktu (x, y, z). Wektor ten służy do przekształceń punktów (x, y, z) i pozwala wykonywać na nim wykonywać operacje odejmowania innego obiektu Vector4.

Obiekt klasy Vector4 przechowujący punkt (x, y, z) pomnożony przez macierz przekształcenia 't' klasy Matrix4 zwraca punkt po przekształceniu 't'.

6.6.*map.h* - mapa konturowa

Zawiera klasę Map, wewnątrz której zostały umieszczone wszystkie operacje potrzebne do rysowania mapy konturowej funkcji oraz dopełnienia jej gradientem koloru w zależności od wartości funkcji w przedziale $< z_{min}, z_{max} >$. W celu poprawnego wygenerowania mapy konturowej niezbędne są:

a) pointer do obiektu klasy wxPanel, na którym mapa ma zostać narysowana
b) aktualne wymiary panelu
c) obiekt typu std::string zawierający funkcję do wygenerowania (z wykorzystaniem biblioteki TinyExpr przeprowadzana jest konwersja funkcji $f(x, y)$ w celu wyliczenia jej wartości dla każdego piksela panelu wxPanel)
d) przedziały rysowania funkcji we wszystkich 3 wymiarach: x_{min}, x_{max}, y_{min}, y_{max} oraz z_{min}, z_{max}.

Kolejne linie wysokości rysowane są dla równych różnic wysokości. Gradient koloru rysowany jest w kolorystyce przechodzącej z niebieskiego (przy wartości równej z_{min}) do czerwonego (przy wartości równej z_{max}). Kolor w danym punkcie ustalany jest wyłącznie na podstawie jego wartości względem z_{min} i z_{max}, które określone zostały przez użytkownika. Dla wartości wychodzących poza zakres $< z_{min}, z_{max} >$ mapa rysowana jest na biało.

Pracując nad gradientem dla mapy konturowej pojawił się problem dotyczący znacznego spowolnienia pracy programu podczas generowania wykresu. Przyczyną tego okazało się być wykorzystanie funkcji DrawPoint pochodzącej z klasy wxBufferedDC, która wypełniała cały panel punkt po punkcie. Rozwiązaliśmy ten problem zamieniając funkcję DrawPoint na DrawRectangle pochodzącą z tej samej klasy. Rysując kwadraty o wymiarach 2 x 2 pixele zmniejszyliśmy ilość wykonywanych iteracji czterokrotnie co dało zadawalający efekt.

7. Testowanie

7.1.testy niezależnych bloków

Testy niezależnych bloków odbywały się systematycznie w czasie trwania pracy nad projektem i dodawania nowych funkcjonalności. Każda kolejna funkcjonalność była poddawana testom, a błędy były na bieżąco identyfikowane i eliminowane.

7.2.testy powiązań bloków

Po połączeniu funkcjonalnym pomiędzy interfejsem użytkownika (*GUIMyFrame1.h*), a klasami rzutu perspektywicznego i mapy konturowej (*perspectivic.h, map.h*) rozpoczęto testy powiązań bloków takie jak na przykład:

a) tworzenie wykresu w postaci rzutu perspektywicznego oraz mapy konturowej w zależności od wyboru użytkownika (ustawiony odpowiedni wxRadioButton)
b) poprawność pobierania przez klasy rzutu perspektywicznego i mapy konturowej aktualnych wymiarów panelu, na którym generowany jest wykres
c) akceptowanie i odrzucanie parametrów zadanych przez użytkownika w zależności od ich poprawności

d) widok wykresu w postaci rzutu perspektywicznego w zależności od ustawienia parametrów macierzy przekształceń.

Zauważone błędu były odnotowywane i następnie w miarę możliwości eliminowane.

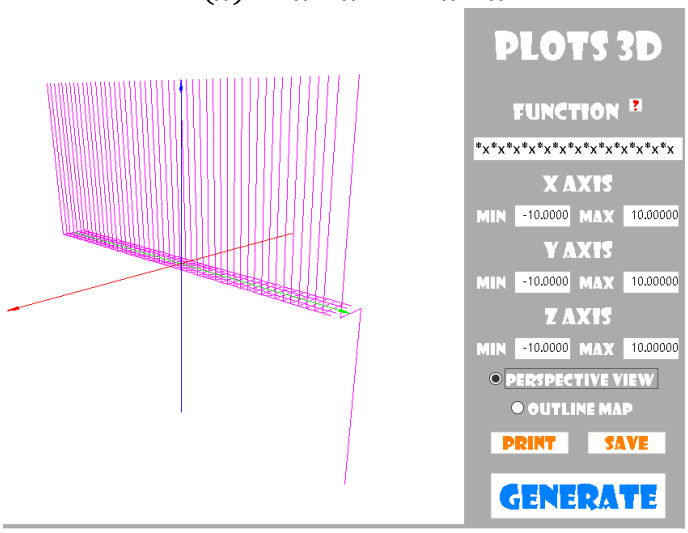
7.3.testy całościowe

Po skończeniu pracy nad projektem wykonanym na poziomie podstawowym rozpoczęto testowanie całościowe. Testy wykonano ręcznie, sprawdzając pracę programu dla różnych zestawów danych wejściowych.

Rodzaje wykonywanych testów wraz z przykładowymi danymi wejściowymi i wyciągnięte na podstawie testów wnioski:

a) próba przeciążenia programu długą funkcją, dającą ogromne wartości

$$F(x) = x * x * \ldots * x * x$$



Program bez problemu radzi sobie z wyrażeniami o długości nawet 255 znaków.

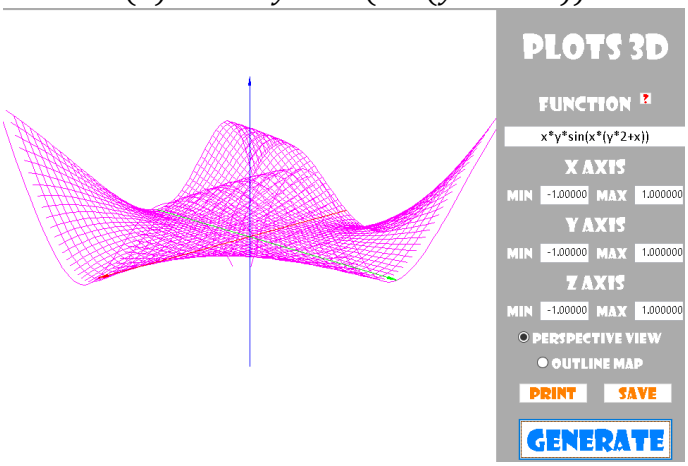
b) wejście puste lub zawierający napis nie będący wyrażeniem matematycznym

$$F(x) = zwykly\ napis$$

Przy wejściu pustym lub nie stanowiącym wyrażenia matematycznego, program nic nie rysuje.

c) wejście zawierające skomplikowane wyrażenie matematyczne, zmiana rozmiaru obszaru rysowanego

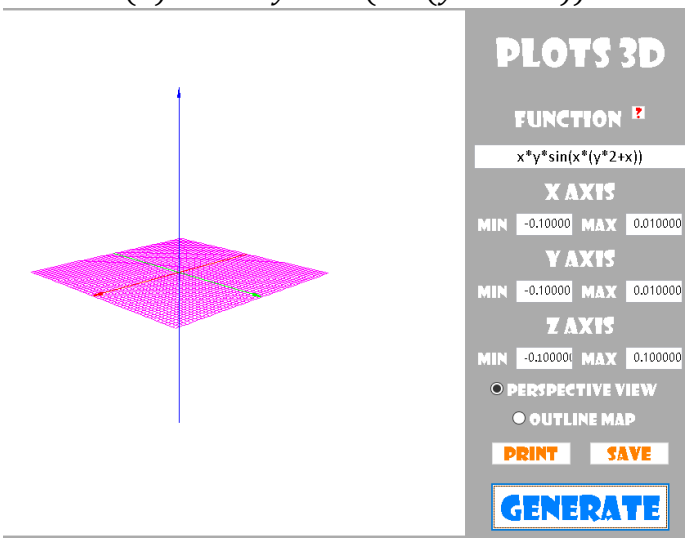
$$F(x) = x * y * sin(x * (y * 2 + x))$$



Program radzi sobie doskonale ze skomplikowanymi funkcjami. Przy zmniejszeniu rozmiarów wycinka płaszczyzny dalej wyświetlana jest ta sama liczba punktów siatki, program zapewnia czytelną rozdzielczość, niezależnie od zakresu rysowania.

d) wąski zakres wykresu

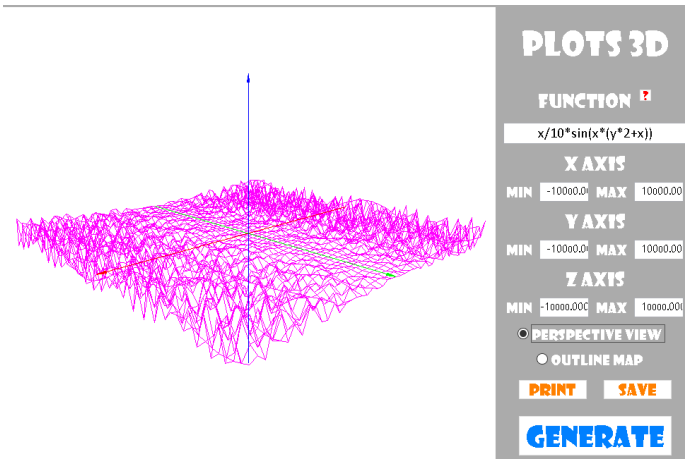
$$F(x) = x * y * sin(x * (y * 2 + x))$$



Program działa sprawnie nawet, przy niewielkim zakresie rysowania.

e) szeroki zakres wykresu

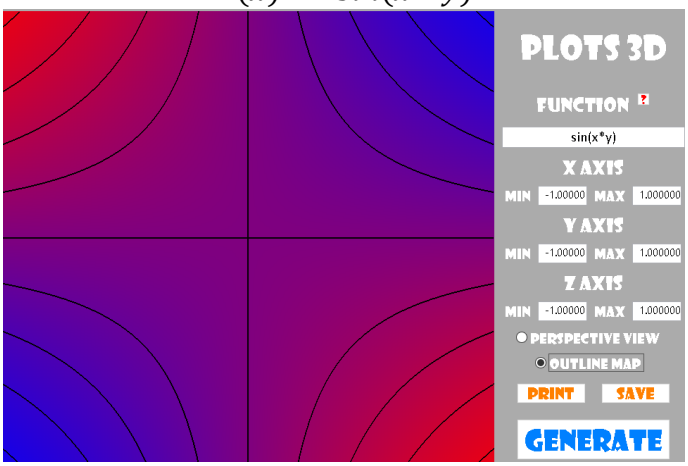
$$F(x) = x/10 * sin(x * (y * 2 + x))$$



Program działa sprawnie przy bardzo szerokim zakresie rysowania.

f) test mapy konturowej

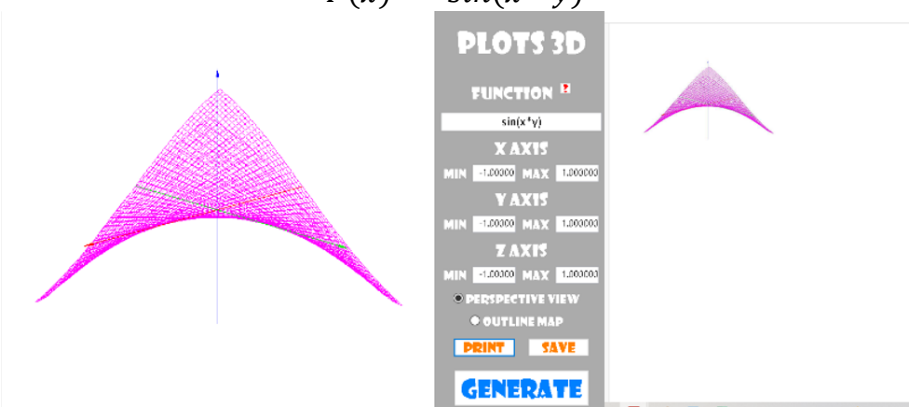
$$F(x) = sin(x * y)$$



Opcja mapy konturowej działa sprawnie, dokładnie reprezentując daną funkcję, kolory i linie izometryczne są dobrane do rozpiętości wykresu.

g) test wydruku (na zdjęciu: po lewej stronie okno programu, a po prawej wydruk otwarty w zewnętrznym programie)

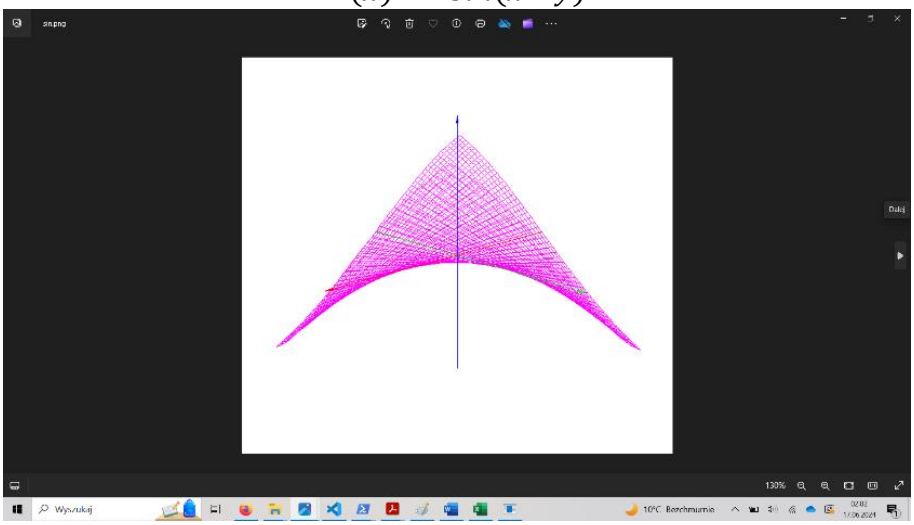
$$F(x) = sin(x * y)$$



Program zapewnia dokładny wydruk wyświetlanego na ekranie wykresu funkcji.

h) zapis do pliku

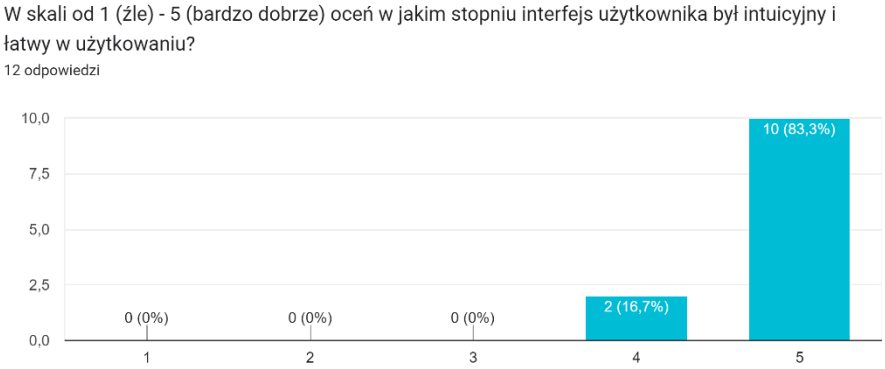
$$F(x) = sin(x * y)$$



Program bezbłędnie zapisuje przedstawioną funkcję do pliku.

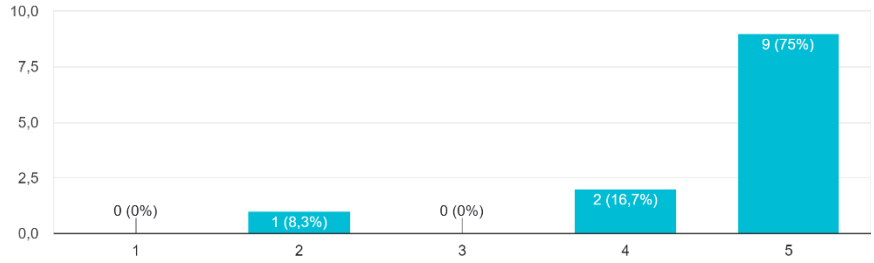
8. Wdrożenie, raport i wnioski

Finalnym etapem projektu było przeprowadzenie testu na użytkownikach i wypełnienie przez nich ankiety dotyczącej doświadczeń związanych z korzystaniem z aplikacji. Program został przetestowany przez grupę kilkunastu użytkowników. Odbiór był pozytywny. Wyniki ankiety są następujące:



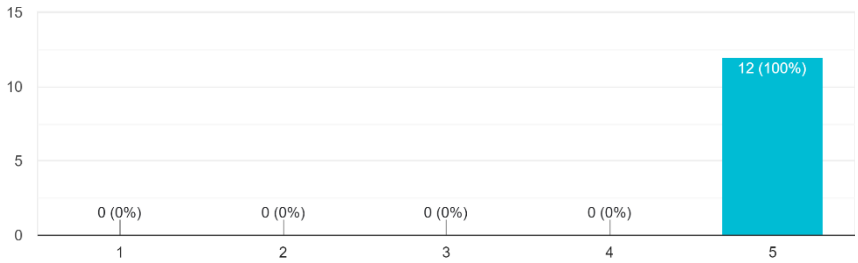
W skali od 1 (źle) - 5 (bardzo dobrze) oceń w jakim stopniu generowane wykresy były dla ciebie czytelne?

12 odpowiedzi



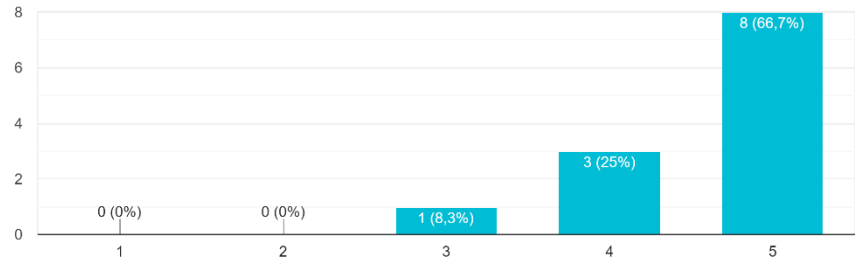
W skali od 1 (źle) - 5 (bardzo dobrze) oceń w jakim stopniu manipulacja wykresem w postaci rzutu perspektywicznego zachodziła bez zakłóceń?

12 odpowiedzi



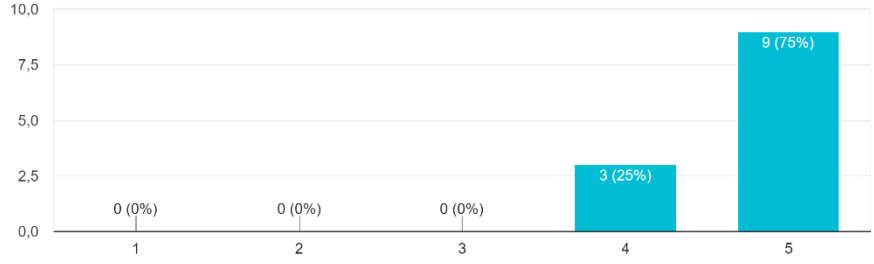
W skali od 1 (źle) - 5 (bardzo dobrze) oceń w jakim stopniu czas generowania wykresów był dla ciebie zadowalający?

12 odpowiedzi



W skali od 1 (źle) - 5 (bardzo dobrze) oceń w jakim stopniu aplikacja spełniła twoje oczekiwania?

12 odpowiedzi



Podsumowując wyniki ankiety zdecydowana większość użytkowników była usatysfakcjonowana korzystaniem z naszego programu. Dużym sukcesem okazała się być płynność działania obrotów oraz powiększania i pomniejszania funkcji w postaci rzutu perspektywicznego. Funkcjonalności, nad którymi należałoby jeszcze popracować są na pewno szybkość generowania wykresów, a w szczególności mapy konturowej oraz dodanie skali i opisu osi do wykresów. Niemniej jednak w ogólnym odbiorze użytkowników program spisał się dobrze i spełnił ich oczekiwania.

Realizację projektu uważamy za sukces. W trakcie pracy napotkaliśmy wiele przeszkód, szczególnie dotyczących generowania mapy konturowej, jednak ostatecznie program spełnia wszystkie podstawowe założenia projektu jak i zawiera kilka dodatkowych funkcjonalności. Warto zwrócić też uwagę, że interfejs jest intuicyjny i przyjazny użytkownikowi, aplikacja jest skalowalna, a wszystkie zastosowane rozwiązania są satysfakcjonująco efektywne dzięki czemu program działa dość płynnie.

Oczywiście w dalszym ciągu aplikacja nie jest idealna. Pierwszą funkcjonalnością, która wymaga poprawy jest generowanie mapy konturowej. Po naciśnięciu przycisku odpowiadającego za jej wyświetlenie bądź zmianie rozmiaru okna trzeba niestety poczekać około 2 sekund zanim wykres się wygeneruje. Oprócz tego warto byłoby dopracować osie rysowane na wykresie w postaci rzutu perspektywicznego – można byłoby dodać do nich skalę i je podpisać. Inną użyteczną funkcjonalnością byłoby dodanie możliwości przesuwania całego wykresu w obrębie panelu 1 (zdjęcie 1) przykładowo przy pomocy strzałek. Tak samo dla mapy konturowej przydatnym rozwiązaniem byłoby umieszczenie osi x, y wzdłuż lewego i dolnego boku panelu. Chcąc umożliwić użytkownikowi większy zakres personalizacji generowanych wykresów można też pokusić się o dodanie możliwości wyboru koloru wykresu, bądź koloru początkowego i końcowego w przypadku gradientu mapu konturowej.

Projekt o numerze 007 – „Wykresy 3D” pozwolił nam wykorzystać wiedzę zdobytą podczas kursu „Podstawy Grafiki Komputerowej”, a także zapoznać się z wieloma użytecznymi narzędziami które mogą okazać się dla nas przydatne w przyszłych projektach. Pracując nad programem nauczyliśmy się korzystać z systemu kontroli wersji Git, zapoznaliśmy się z biblioteką TinyExpr, zdobyliśmy wiedzę z zakresu rzutowania obiektów trójwymiarowych na płaszczyznę, a także poszerzyliśmy umiejętności korzystania z biblioteki wxWidgets. Rozwinęliśmy także kompetencje miękkie takie jak planowanie, rozdysponowywanie obowiązków i praca zespołowa.