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:

1. 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
2. Bartosz Fryska
   1. algorytm rzutu perspektywicznego
   2. algorytm mapy konturowej
   3. obrót wykresu w postaci rzutu perspektywicznego, jego powiększanie  
      i pomniejszanie
   4. generowanie wykresów w postaci rzutu perspektywicznego i mapy konturowej w oparciu o dane pobrane od użytkownika
   5. pomoc przy tworzeniu dokumentacji
3. Aleksander Kopyto
   1. parser funkcji podanej przez użytkownika (końcowo niewykorzystany przez to, że powodował znaczne spowolnienie działania programu)
   2. testy poszczególnych funkcjonalności i programu jako całości
   3. podsumowanie danych pochodzących z testów na użytkownikach.
4. **Opis projektu**

Celem projektu było stworzenie programu generującego wykresy funkcji   
w postaci mapy konturowej lub rzutu perspektywicznego w zależności od wyboru użytkownika.

1. **Założenia wstępne przyjęte w realizacji projektu**
   1. **wymagania podstawowe**

Użytkownik ma możliwość wprowadzenia funkcji w postaci tekstowej, ustalenia jej przedziałów względem zmiennej x, y oraz wartości minimalnej i maksymalnej funkcji (zmin i zmax). 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.

* 1. **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 powiększanie  
i pomniejszanie 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).

1. **Analiza projektu**
   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 zmin i zmax 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 powiększanie i pomniejszanie rzutu perspektywicznego  
o ile kursor znajduje się na panelu z wykresem.

* 1. **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.

* 1. **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.

* 1. **specyfikacja interfejsu użytkownika**

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

A screenshot of a computer

Description automatically generated

**2.**

**1.**

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.

A screenshot of a phone

Description automatically generated

**e)**

**c)**

**g)**

**b)**

**f)**

**d)**

**a)**

Zdjęcie 2. Panel 2 programu.

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

1. wxButton *buttonInfo* odpowiadający za wyświetlenie informacji ze wskazówkami dotyczącymi poprawnego wprowadzenia funkcji przez użytkownika
2. 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
3. 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. xmin >= xmax) pola tekstowe z błędnym parametrami zostaną wyczyszczone
4. wxRadioButton *radioRzut, radioMapa* pozwalają na wybranie rodzaju wykresu, który ma zostać wyświetlony, jednocześnie tylko 1 z opcji może zostać wybrana
5. wxButton *buttonSave* jego kliknięcie powoduje przekierowanie do okna dialogowego zapisu wykresu w formie obrazu w formacie .png
6. wxButton *buttonPrint* jego kliknięcie powoduje przekierowanie do okna dialogowego druku wykresu
7. 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.
   1. **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 powiększeniem 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.
    1. **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.

1. **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:
   1. Anna Nowak
   2. Bartosz Fryska
   3. Aleksander Kopyto
2. Stworzenie aplikacji spełniającej założenia podstawowe:
   1. Anna Nowak
   2. Bartosz Fryska
   3. Aleksander Kopyto
3. Wprowadzenie poprawek, dodanie dodatkowych funkcjonalności i testowanie:
   1. Anna Nowak
   2. Bartosz Fryska
   3. Aleksander Kopyto
4. **Opracowanie i opis niezbędnych algorytmów**
5. **Kodowanie**
   1. **rzut perspektywiczny**

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.

* 1. **mapa konturowa**

Pracując nad wypełnieniem tła 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.

1. **Testowanie**
   1. **testy niezależnych bloków**
   2. **testy powiązań bloków**
   3. **testy całościowe**
   4. **określenie niezmienników**
2. **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 10 użytkowników. Odbiór był pozytywny. Wyniki ankiety są następujące:

1. .

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 swoją wiedzę dotyczącą korzystania z biblioteki wxWidgets. Poszerzyliśmy także kompetencje miękkie takie jak planowanie, rozdysponowywanie obowiązków i praca zespołowa.