1. Tytuł projektu i autorzy projektu.

Przekroje brył 3D.
Magdalena Gajek – projektowanie i odczytywanie szkieletów brył geometrycznych;
Justyna Gacek – tworzenie interfejsu użytkownika;
Wojciech Franczuk – rysowanie figur i ich przekrojów;

2. Opis projektu.

Celem projektu było napisanie programu, który tworzy przekroje trójwymiarowych brył szkieletowych. Do programu wczytywane są dowolne bryły, zakodowane zgodnie ze sposobem opisanym w podpunkcie „specyfikacja danych wejściowych". Przekrój bryły trójwymiarowej z płaszczyzną, rozumiemy jako kształt utworzony z punktów powstałych w miejscach przecięcia ścian tworzących bryłę z zadaną powierzchnią. Po wciśnięciu przycisku start, widoczna jest linia, która wyznacza aktualną pozycję przekroju. Użytkownik ma możliwość wyboru prędkości poruszania się płaszczyzny, jak również osi, wzdłuż której ona się porusza.

3. Założenia wstępne przyjęte w realizacji projektu.

Pierwszym zadaniem było stworzenie systemu kodowania figur, który umożliwi nam łatwe kodowanie i odczyt zawartych w nim kształtów. Zdecydowaliśmy się na przedstawienie figur stworzonych z trójkątów, zakodowanych za pomocą ich wierzchołków. W celu ułatwienia użytkownikowi wyobrażenia sobie figury, której przekrój obserwuje, stworzyliśmy dodatkowy panel, który przedstawia rzut figury na odpowiednią płaszczyznę. Osie lewego panelu dobrane są w taki sposób, aby oś wobec której ma poruszać się płaszczyzna znajdowała się równolegle do dolnej krawędzi ekranu, zwrócona w prawo. Według tej osi zawsze poruszać się będzie płaszczyzna. Na prawym panelu, po naciśnięciu przycisku "start", będą pojawiać się punkty (lub linie), w których płaszczyzna widoczna na lewym panelu przecina figurę.

4. Analiza projektu.

- **Specyfikacja danych wejściowych**

Skonstruowaliśmy trzy bryły szkieletowe na potrzeby tego projektu, są to:

- ❏ Sześcian
- ❏ Bryła składająca z trzech ostrosłupów trójkątnych
- ❏ Bryła składająca się z pięciu graniastosłupów, nazwana Przestrzenną Liczbą Szczęścia Studenta(PLSS).

W plikach tekstowych sześcian.txt, inna_figura.txt oraz plss.txt zawarte są dane potrzebne do narysowania brył. Aby narysować daną figurę przestrzenną, dzielimy jej ściany tak, aby wszystkie składały się z dowolnych trójkątów. Dane zawarte w plikach tekstowych mają postać współrzędnych wszystkich trójkątów składających się na dany obiekt (3 wiersze w pliku = jeden trójkąt). Dla każdego wierzchołka jest również przyporządkowana flaga rysowania, informująca nas czy rysujemy krawędź między punktem tym i następnym (flaga = 1) czy też nie (flaga = 0).
Przykład:

A:	1	1	0	1
B:	1	1	1	1
C:	1	0	1	0

W tym przypadku łączymy punkty A i B oraz B i C, nie rysujemy krawędzi między C i A

D: 1 1 0 0
E: 1 1 1 1
F: 1 0 1 1
W tym przypadku łączymy punkty E i F oraz F i D, nie rysujemy krawędzi między D i E
Wszystkie współrzędne w plikach są liczbami całkowitymi dodatnimi.

- **Opis oczekiwanych danych wyjściowych**

Nasz program nie generował żadnych danych wyjściowych. Efektem jego działania jest wyświetlenie na ekranie wczytanej bryły, oraz jej przekroju.

- **Zdefiniowanie struktur danych**

Każdy punkt jest reprezentowany przez obiekt klasy Wektor. Klasa ta zawiera tablice z danymi (typu double) do której zapisujemy współrzędne punktu oraz informację o rysowaniu krawędzi. Konstruktor domyślny wypełnia tablicę z danymi zerami, a konstruktor z zadanymi wartościami zapisuje je do odpowiednich miejsc w tablicy z danymi (0->x, 1->y, 2->z, 3->flaga). Klasa ta zawiera również metodę ustaw, która umożliwia zmianę współrzędnych zawartych w tablicy z danymi. Metoda wypisz wypisuje współrzędne wektora oraz flagę oddzielone spacjami. Gdy zapiszemy trzy kolejne wiersze z plików tekstowych do wektorów, to możemy stworzyć trójkąt. W naszym programie trójkąt jest zdefiniowany jako obiekt klasy Macierz. Każdy obiekt klasy Macierz zawiera trzyelementową tablice typu Wektor, która zawiera współrzędne wierzchołków danego trójkąta. Konstruktor klasy Macierz zapisuje zadane wektory do tablicy z danymi. Metoda wypisz służy do wypisywania tablicy z Wektorami. Aby narysować bryłę tworzymy listę trójkątów (czyli u nas obiektów klasy Macierz), które składają się na tę bryłę. Do tego zadania wykorzystujemy kontener tablicy std::vector.

- **Specyfikacja interfejsu użytkownika**

Po uruchomieniu programu należy użyć przycisku „wczytaj dane” (wxButton), wówczas pojawi się okno, w którym użytkownik powinien wybrać jeden z plików tekstowych reprezentujących bryłę – po dokonaniu wyboru na lewym panelu pojawi się rzut bryły szkieletowej na płaszczyznę OXY, ponieważ domyślnie wybrany jest kierunek poruszania płaszczyzny wzdłuż osi X. Następnie za pomocą jednego z trzech przycisków (wxRadioButton) można dokonać jego zmiany, wybierając kierunek wzdłuż osi X, Y lub Z. Po wciśnięciu przycisku „start” (wxButton) płaszczyzna zacznie poruszać się z domyślną prędkością 1, a na prawym panelu będzie widoczny jej przekrój. Prędkość tę można modyfikować używając paska (wxScrollBar), na którym użytkownik wybiera wartość z przedziału [1, 11] . Pod paskiem wyświetlana jest wytypowana przez użytkownika prędkość.

- **Wyodrębnienie i zdefiniowanie zadań**

✓ Stworzenie graficznego interfejsu użytkownika,
✓ Stworzenie plików z danymi reprezentującymi trójwymiarową bryłę szkieletową,
✓ Rysowanie rzutu bryły korzystając z stworzonego wcześniej kodowania,
✓ Stworzenie ruchomej płaszczyzny, która wyznacza w którym miejscu figura jest przecinana przez płaszczyznę,
✓ Rysowanie przekroju figury przecinanej przez płaszczyznę,

- ✓ Stworzenie precyzyjnej dokumentacji.

- **Decyzja o wyborze narzędzi programistycznych**

Projekt został w pełni wykonany przy użyciu języka C++ oraz biblioteki graficznej wxWidgets. Użyte środowisko to wxDev-C++. Dokonaliśmy takiego wyboru, ponieważ biblioteka ta pozwala tworzyć programy, które posiadają naturalny wygląd i obsługę przyjazną dla użytkowników każdego środowiska. Kolejnym argumentem był fakt, że środowisko to jest nam dobrze znane z zajęć.

5. Podział pracy i analiza czasowa.

Zadania realizowaliśmy w kolenności wypisywania:

Justyna Gacek - Używając odpowiednich narzędzi biblioteki stworzyłam intuicyjny i przyjazny dla użytkownika interfejs graficzny. Wykorzystuje on zróżnicowane opcje, tak aby użytkownik miał możliwość w jak najprostszy sposób sterować funkcjami programu oraz obserwować efekt jego działania. Menu zostało podzielone na trzy osobne obszary, o odpowiednich nazwach, które ułatwiają obsługę programu. Następnie zostały stworzone dwa duże panele, na których widoczny jest efekt działania programu.

Magdalena Gajek- Aby wyznaczyć punkty potrzebne do narysowania bryły szkieletowej, ręcznie naszkicowałam wszystkie figury przestrzenne, a następnie z rysunków odczytałam ich współrzędne. Otrzymane w ten sposób dane zapisałam do plików tekstowych pamiętając o dopisaniu odpowiednich flag, potrzebnych do rysowania krawędzi. Korzystając z biblioteki fstream wczytałam dane z plików i zapisałam je do wcześniej stworzonych struktur: Macierz i Wektor.

Wojciech Franczuk - Korzystając z narzędzi dostarczonych przez bibliotekę WxWidgets stworzyłem program, który jest w stanie odczytać i narysować rzuty na płaszczyznę brył odpowiednio zakodowanych. Następnie korzystając ze współrzędnych punktów przekazanych do programu, wyznaczyłem równania prostych, które w bardzo łatwy sposób umożliwiają wyznaczenie punktów, w jakich płaszczyzna przecina zakodowaną bryłę.

6. Opracowanie i opis niezbędnych algorytmów

- Wyznaczanie prostych

W celu wyznaczenia prostych przechodzących przez punkty użyłem parametrycznego zapisu prostej. W momencie pobierania parametrów, tworzę osobną tablice przechowującą położenie końcowe i początkowe wszystkich prostych w trzech wymiarach. Na przykład, dla wybranej osi OX, najpierw wyznaczamy wartość parametru t ze wzoru:

$$t = \frac{x - x1}{x2 - x1}$$

Gdzie: x - aktualna pozycja płaszczyzny na osi, x1 - początek prostej, x2 - koniec prostej
Znając wartość parametru t, mogę w łatwy sposób wyznaczyć pozostałe wymagane współrzędne, ponownie korzystając z postaci parametrycznej prostej:

$$y = t(y2 - y1) + y1$$
$$z = t(z2 - z1) + z1$$

Ostatecznie wystarczy narysować punkt o współrzędnych y oraz z. Proces oczywiście powtarzamy dla każdej prostej, zmieniając odpowiednio parametry przy zmianie wyboru osi.

Osobno obsługiwane jest rysowanie miejsc, w których przecięciem bryły przez płaszczyznę jest prosta. Jeśli koniec i początek prostej ma taką samą wartość jak aktualnie sprawdzana zmienna, to zamiast punktu rysowana jest prosta między tymi dwoma punktami. Jest to dobrze widoczne podczas obserwowania przekrojów sześcianu.

7. Kodowanie

Punkty wczytane z pliku umieszczone są w obiekcie std::vector o nazwie Lista, będącej prywatnym elementem klasy Projekt1Frm. Wektor ten przechowuje obiekty klasy Macierz, stworzone przez nas na potrzeby zadania. Udostępnia to swobodny dostęp do danych z dowolnej metody tej klasy. Elementy te wykorzystywane są do rysowania figury na prawym panelu, oraz do wyznaczania równań prostych. Wszystkie proste przechowywane są również w obiekcie prywatnym klasy Projekt1Frm o nazwie Proste. Ponownie wykorzystaliśmy stworzoną przez nas klasę Macierz do przechowywania wymaganych informacji. Ten drugi wektor wykorzystywany był do wyznaczania i rysowania punktów przecięcia płaszczyzny i brył. Dokładne kodowanie obu bloków jest następujące:

Lista: przechowuje n obiektów typu macierz, o indeksach [0 ; n-1], kolejne indeksy reprezentują kolejne trójkąty. Każda macierz posiada tablicę wektorów o nazwie dane, o indeksach [0; 2], oznaczającą kolejne wierzchołki naszych trójkątów. Każdy wektor posiada czterowymiarową tablicę typu double, o indeksach [0; 3], które są opisane w podpunkcie “specyfikacja danych wejściowych”.

Proste: przechowuje m macierzy o indeksach [0; m-1], reprezentujących kolejne proste. Każda macierz ma tablicę dane, typu wektor, o indeksach [0; 2], reprezentujących kolejne wymiary tej prostej. Każdy wektor posiada tablice typu double o indeksach [0; 2], w polu 1 przechowywana jest współrzędna początku prostej, w polu 2 współrzędna jej końca, a w polu 0 różnica między tymi dwoma wartościami, wykorzystywana do wyliczenia parametru t.

Współrzędne początku i końca są potrzebne, żeby wyznaczyć w jakich granicach należy obliczać punkty przecięcia prostej i płaszczyzny.

8. Testowanie

Podczas sprawdzania działania naszego programu, zauważyliśmy, że punkty, w których nasza bryła przecinała płaszczyznę są mało widoczne. Dlatego zdecydowaliśmy się na reprezentacje każdego punktu jako koła o środku w punkcie przecięcia prostej i płaszczyzny, i promieniu 2 pikseli.

Dla trzech opisanych wyżej figur, sprawdziliśmy, że program działa, rysując zarówno przekrój jak i rzut figur poprawnie.

Kontrola prędkości i wybór płaszczyzny także działa zgodnie z założeniami i bez zarzutów.

9. Wdrożenie, raport i wnioski

Osiągnęliśmy wszystkie założone cele w wymaganiach podstawowych, dodatkowo implementując oglądanie rzutu figury na płaszczyznę. Dane wprowadzone w opisany przez nas sposób są rysowane poprawnie na obu panelach. Zadanie wykonaliśmy i oddaliśmy w terminie. Dzięki wykonaniu projektu osiągnęliśmy lepsze zrozumienie biblioteki WxWidgets.