# Graficzna prezentacja danych.

## Zadanie

Zadanie polega na napisaniu programu rysującego funkcje typu $f(x,y)$. Funkcja podana jest w postaci zbioru wartości $w_i$ w różnych punktach $(x_i,y_i)$ przy czym wiadomo, że wartości funkcji w tych punktach zawsze mieszczą się w zakresie od -2.5 do 2.5. Punkty są rozrzucone przypadkowo. Funkcja będzie prezentowana w postaci pseudo trójwymiarowego „krajobrazu" wygenerowanego metodą „*voxel space*" w wersji kolorowej lub w odcieniach szarości. Idea tej metody będzie wyjaśniona na wykładzie.

## Cel

Praktyczne wykorzystanie metody Sheparda do interpolacji nieregularnie rozmieszczonych danych. Wizualizacja metodą „*voxel space*".

## Środki

Środowisko wxDev-C++.

## Zarys możliwego rozwiązania

Przygotowane fragmenty kodu obsługują sterowanie programem oraz odświeżanie okna z rysowanymi funkcjami. Okno to jest odświeżane na podstawie bitmapy znajdującej się w pamięci, która powinna zawierać prawidłową mapę. Po każdej zmianie parametru wołana jest metoda `draw(…)`, do której przekazywane są wszystkie dane potrzebne do narysowania „krajobrazu". Zadaniem programisty jest napisanie odpowiedniej metody `draw(…)`. Cały przygotowany kod znajduje się w plikach projektu o nazwach Lab_07.* , Lab_07App.* i Main_Frm.*. Plików tych nie wolno modyfikować. Plik o nazwie draw.cpp zawiera funkcję `draw(…)` i jest to jedyny plik, który wolno edytować. Wszystkie pomocnicze funkcje, dodatkowe zmienne i cokolwiek co będzie potrzebne powinno się znaleźć wyłącznie w tym pliku.

## Opis funkcji draw(…)

Metoda „draw" przyjmuje cztery argumenty: `int function, int alpha, int view_angle, int color`, **gdzie:**

- `int function` – określa, którą funkcję mamy wizualizować,
- `int alpha` – określa o jaki kąt wokół osi OZ obrócony jest „krajobraz" (w stopniach, względem środka),
- `int view_alngle` - określa jak wysoko nad płaszczyzną XY znajduje się obserwator,
- `int color` - pozwala na określenie czy „krajobraz" rysowany jest w kolorze (niebieski dla wartości najmniejszej, czerwony dla największej), czy też w odcieniach szarości (czarny dla minimum biały dla maksimum).

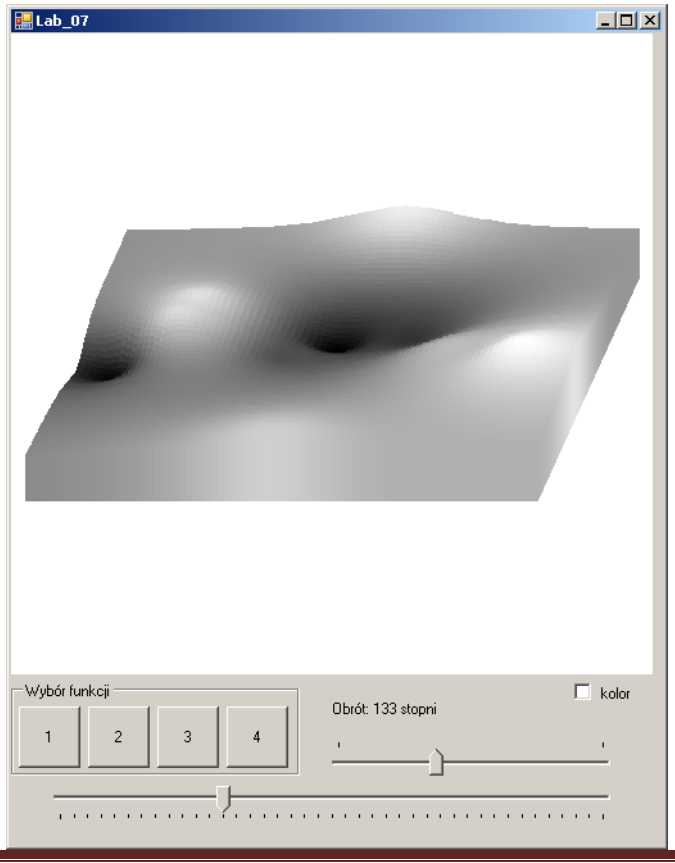
Dodatkowo dane są dwie zmienne globalne oraz jedna funkcja. Tablica „**`float FunctionData[100][3]`**” zawiera współrzędne X (drugi indeks równy 0), Y (drugi indeks równy 1) punktu oraz wartość funkcji w tym punkcie (drugi indeks równy 2). Zmienna „**`NoPoints`**” określa ile punktów znajduje się w tej tablicy (pierwszy indeks tablicy **`FunctionData`** numeruje punkty). Funkcja „**`PrepareData(int fun)`**” przyjmuje tylko jeden argument: określający dla jakiej funkcji mają zostać obliczone punkty i po jej wywołaniu następuje ustalenie wartości zmiennej **`NoPoints`** oraz tablicy **`FunctionData`**.

## Przykładowa zawartość funkcji draw(…).

Pierwsze linijki funkcji tworzą pamięciowy kontekst urządzenia rysującego i wiążą z nim bitmapę, która jest przechowywana w pamięci. Linie te należy pozostawić nie zmienione. Od tej pory korzystając z kontekstu **`memDC`** można rysować na bitmapie. **Uwaga: bitmapa ma rozmiary 500x500 pikseli i nie należy ich przekraczać.** Następnych kilka linijek służy wyłącznie pokazaniu, że to co jest rysowane w funkcji będzie wyświetlane w oknie (jest to wypisanie parametrów przekazanych do funkcji). Ten fragment kodu można wyrzucić. Resztę kodu trzeba uzupełnić. Dla przykładu, funkcja może wyglądać następująco:

1. Obracamy wszystkie punkty o zadany kąt alfa.
2. Interpolujemy funkcję do regularnej siatki metodą Sheparda
3. Wyznaczamy najmniejszą i największą wartość funkcji
4. Rysujemy krajobraz metodą „voxel space”

W dostarczonej paczce znajdują się: źródła szkicu aplikacji oraz skompilowana wersja kompletnego programu przykładowego. Szkic posiada zaimplementowane kompletne GUI wraz z reakcjami na zdarzenia. GUI przedstawia poniższy rysunek:



## Jak się przygotować przed zajęciami

Zadanie nie jest szczególnie trudne (choć w pierwszej chwili może na takie wyglądać) i powinno dać się w całości wykonać podczas zajęć. Podstawą jest jednak dobre zrozumienie problemu i wcześniejsze przemyślenie sposobu implementacji, zwłaszcza algorytmu „voxel space”. Dlatego sugeruję:

- Zapoznanie się z metodą Sheparda interpolacji nieregularnych danych.
- Zapoznanie się (na podstawie wykładu) z metodą „voxel space”.
- Zastanowienie się nad implementacją algorytmu w kontekście postawionego zadania.
- Opracowanie sposobu przeliczania wartości funkcji na składowe RGB rysowanego punktu. Sposób ten powinien zapewniać płynną zmianę barwy od np. niebieskiego dla minimum funkcji do czerwonego dla maksimum funkcji.