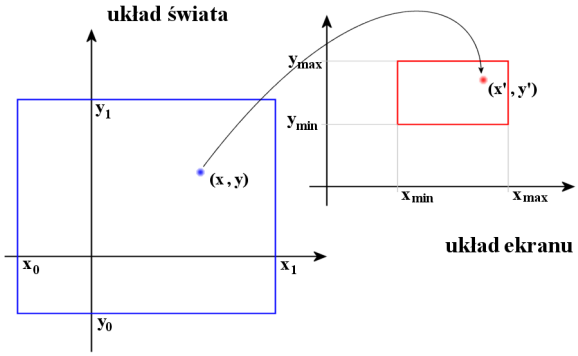
# Grafika 2D.

## Zadanie

Napisać program rysujący wykresy funkcji 2D. Do wyboru w programie dostępne są trzy funkcje:

1. $y = x + \sin(4 \cdot x)$
2. $y = x^2$
3. $y = 0.5 \cdot e^{4x - 3x^2}$

Użytkownik ma możliwość podania przedziału zmienności x. Program sam oblicza wartości minimalne i maksymalne przyjmowane przez funkcję ( **`y_min`** i **`y_max`** ) po każdej zmianie zakresu x-ów oraz wypisuje je na ekranie. Powinna istnieć możliwość ustawiania wartości stałych **`x0`** , **`y0`** , **`x1`** i **`y1`** odpowiadających za skalowanie i okienkowanie (patrz rysunek obok). Dodatkowo dostępny jest suwak, który umożliwia obracanie całości. Obrót może dokonywać się wokół środka układu jak i wokół środka ekranu.




Odpowiednie kontrolki pozwalają także przesunąć środek wykresu względem środka ekranu. Proszę zadbać o to, aby przy zmianie rozmiaru okna rysowany wykres był skalowany tak, jak skalowane jest okno. Na każdej z osi powinny się znaleźć co najmniej dwie **opisane** podziałki (po jednej na części dodatniej i ujemnej).

## Cel

Praktyczne zapoznanie się z podstawowymi transformacjami 2D, składaniem przekształceń w postaci macierzowej oraz sposobem użycia współrzędnych jednorodnych. Teoretyczne zapoznanie się z architekturą typu Model-Widok-Kontroler (Model View Controler – MVC)[1].

## Środki

Biblioteka wxWidgets.

## Opis istniejącego kodu

Program (dla celów dydaktycznych) oparto na koncepcji wzorców projektowych wykorzystując architekturę znaną pod nazwą MVC (Model – Widok – Kontroler).

Klasa **`GUIMyFrame1`** pełni rolę **Widoku** i odpowiedzialna jest za komunikację z użytkownikiem, wyświetlanie kontrolek i przygotowanie obszaru, na którym będzie rysowany wykres.

1 Koncepcja MVC będzie omówiona na wykładzie. Praktyczna znajomość na pracowni nie będzie wymagana.

Klasa **`ConfigClass`** pełni rolę **Kontrolera** i przechowuje wszystkie parametry rysowanego wykresu. Udostępnia również interfejsy pozwalające na ustawianie i odczytywanie poszczególnych parametrów wykresu. Znajdują się w niej również metody dzięki którym w łatwy sposób można zapisać do pliku i odczytać z pliku komplet parametrów wykresu.

Klasa **`ChartClass`** pełni rolę **Modelu** i zajmuje się przygotowaniem i odrysowaniem wykresu.

**Widok** po jakiejkolwiek zmianie parametru wykresu dokonanej przez użytkownika wywołuje metody **kontrolera** aktualizujące dane (**`Set_xxx()`**). Jeśli zachodzi potrzeba **widok** wywołuje również metodę modelu odrysowującą wykres (**`void Draw(wxDC *dc, int w, int h)`**).

**Kontroler** po wczytaniu parametrów wykresu z pliku wywołuje metodę **widoku** aktualizującą zawartość kontrolek w oknie głównym aplikacji (**`void UpdateControls()`**).

Zrozumienie powyższego opisu nie jest niezbędne do prawidłowego wykonania ćwiczenia. Jedynym brakującym fragmentem kodu jest funkcja **`void Draw(wxDC *dc, int w, int h)`**, którą należy uzupełnić. Można ją potraktować jako niezależną od reszty programu (z wyjątkiem odczytu parametrów wykresu zapisanych w obiekcie klasy **`ConfigClass`**).

**Wszelkie zmiany mogą być wykonywane jedynie w plikach** *ChartClass.cpp* **i** *ChartClass.h* **.**

Metoda **`void ChartClass::Draw(wxDC *dc, int w, int h)`** powinna wyrysować wykres na kontekście rysunkowym przekazanym do metody jako parametr **`*dc`**, przy czym szerokość wykresu powinna wynosić **`w`** pikseli, a wysokość **`h`** pikseli. Dane do wykresu znajdują się w obiekcie klasy **`ConfigClass`**, na który wskazuje **`cfg`** będący prywatną zmienną klasy **`ChartClass`**.

Dla ułatwienia operacji wektorowych i macierzowych przygotowano klasy **`Vector`** oraz **`Matrix`** reprezentujące odpowiednio wektor o trzech składowych oraz macierz o rozmiarach 3 x 3. Zarówno definicje klas jak i deklaracje ich metod umieszczono w pojedynczym pliku *vecmat.h* .

Klasa **`Vector`** zawiera metodę pozwalającą ustawić współrzędne wektora **`Set(x,y)`** oraz pobrać współrzędne **`GetX()`** oraz **`GetY()`**. Konstruktor klasy ustawia automatycznie składową (3,3) na wartość 1.0.

Klasa **`Matrix`** przeciąża operator „**`*`**” (mnożenie) dla operacji **`Matrix*Matrix`** oraz **`Matrix*Vector`**.

Użycie tych klas jest bardzo proste. Na przykład pomnożenie wektora przez macierz wygląda następująco:

```
Vector x,y;
Matrix m;
x.Set(2.3,1.2) ;
m.data[0][0]=2.0 ;
m.data[1][1]=3.0 ;
y=M*x;
```

## Zarys możliwego rozwiązania

Ponieważ wykres rysowany będzie jako zbiór odcinków wygodnie będzie przygotować sobie funkcję **`line2d(…)`**, która zamieni współrzędne punktów początku i końca odcinka wyrażone w układzie świata na odpowiadające im współrzędne wyrażone w układzie ekranu. Przejście z jednych współrzędnych na drugie powinno być zapisane w odpowiedniej macierzy transformacji. Wówczas symbolicznie można by zapisać:

$P'_{poczatku}$=transformacje*$P_{poczatku}$;

$P'_{konca}$=transformacje*$P_{konca}$;

Jedyne co jest nam potrzebne to macierz „transformacje". Wygodnie jest przygotować sobie funkcje, które zwracają macierz odpowiedzialną za poszczególne przekształcenia. Na przykład:

```
Matrix Set_Rotate(double alpha);
```

niech zwraca macierz odpowiadającą za obrót itd. Teraz wywołując po kolei odpowiednie funkcje i mnożąc (początkowo jednostkową) macierz „transformacje" przez zwracane macierze otrzymamy końcową macierz. Teraz możemy już przystąpić do rysowania. Kolejność przekształceń jest następująca:

- skalowanie i okienkowanie
- obrót wokół wybranego punktu (proszę pamiętać, że użytkownik ma możliwość obracania wokół środka wykresu lub środka obrazu – zmiana punktu wokół którego obracamy to nic innego jak dokonanie odpowiednich translacji przed i po obrocie, w tych translacjach należy uwzględnić translację, której zażądał użytkownik).

Z funkcji rysujących **wxWidgets** powinniśmy wykorzystywać w zasadzie tylko dwie: **`DrawLine(…)`** oraz **`DrawRotatedText(…)`**. Oprócz tego przydadzą się funkcje obsługi pióra (**`wxPen`**) oraz funkcja **`SetClippingRegion()`**. Oczywiście wykorzystanie innych funkcji jest jak najbardziej dopuszczalne.

## Jak się przygotować przed zajęciami

W celu sprawnego napisania programu proponuję zapoznać się z następującymi zagadnieniami:

- Rachunek wektorowy i macierzowy.
- Współrzędne jednorodne.
- Transformacje 2D i ich reprezentacja macierzowa.
- Metody okienkowania i obcinania.
- Zapoznać się z metodą **`wxDC::SetClippingRegion`**.