Fraktale - **Zbiór Mandelbrota**

Szymon Myszor

Czym są fraktale?

W znaczeniu potocznym oznacza zwykle obiekt samopodobny (tzn. taki, którego części są podobne do całości). Ze względu na olbrzymią różnorodność przykładów matematycy obecnie unikają podawania ścisłej definicji fraktali.

Możemy określić zbiór cech jakie każdy fraktal powinien posiadać

- ma nietrywialną strukturę w każdej skali,

- struktura ta nie daje się łatwo opisać w języku tradycyjnej geometrii euklidesowej,

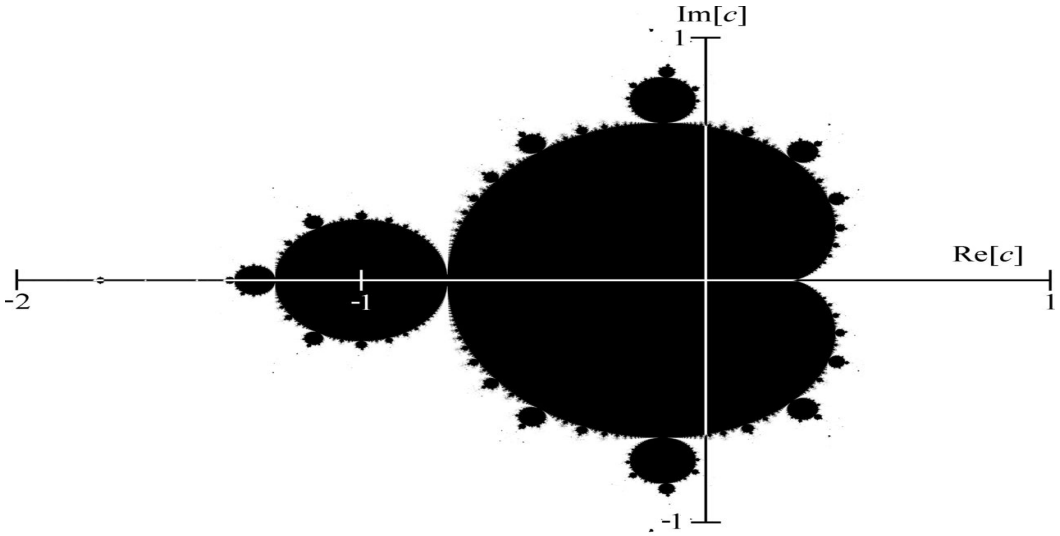
- jest samopodobny, jeśli nie w sensie dokładnym, to przybliżonym lub stochastycznym(losowym),

- ma względnie prostą definicję rekurencyjną,

- ma naturalny („poszarpany”, „kłębiasty” itp.) wygląd

Zbiór Mandelbrota

Podzbiór płaszczyzny zespolonej, którego brzeg jest jednym z najbardziej znanych fraktali „najsłynniejszym obiektem współczesnej matematyki". Nazwa tego obiektu została wprowadzona dla uhonorowania jego odkrywcy, matematyka Benoit Mandelbrota[2].



By zdefiniować zbiór Mandelbrota, zdefiniuje najpierw dla danego punktu *p* na płaszczyźnie zespolonej nieskończony ciąg liczb zespolonych z_0, z_1, z_2, ... o wartościach zdefiniowanych następująco:

$z_0=0$

$z_{n+1}= Z_n^2 + p$

Zbiór Mandelbrota (*ang. Mandelbrot Set*) definiujemy jako zbiór liczb zespolonych *p* takich, że zdefiniowany powyżej ciąg nie dąży do nieskończoności.

Aby otrzymać fraktal będę obliczał kolejne przybliżenia. I tak kolejne przybliżenia zdefiniujemy jako zbiór liczb zespolonych *p* takich, że:

- ***1 przybliżenie: wszystkie punkty***
- **2 przybliżenie: $|z_1| < 2$**
- **3 przybliżenie: $|z_1| < 2$ oraz $|z_2| < 2$**
- **4 przybliżenie: $|z_1| < 2$ oraz $|z_2| < 2$ oraz $|z_3| < 2$**
- **...**
- **n-te przybliżenie: $|z_1| < 2$ oraz $|z_2| < 2$, ... $|z_{n-1}| < 2$**

Diagram klas

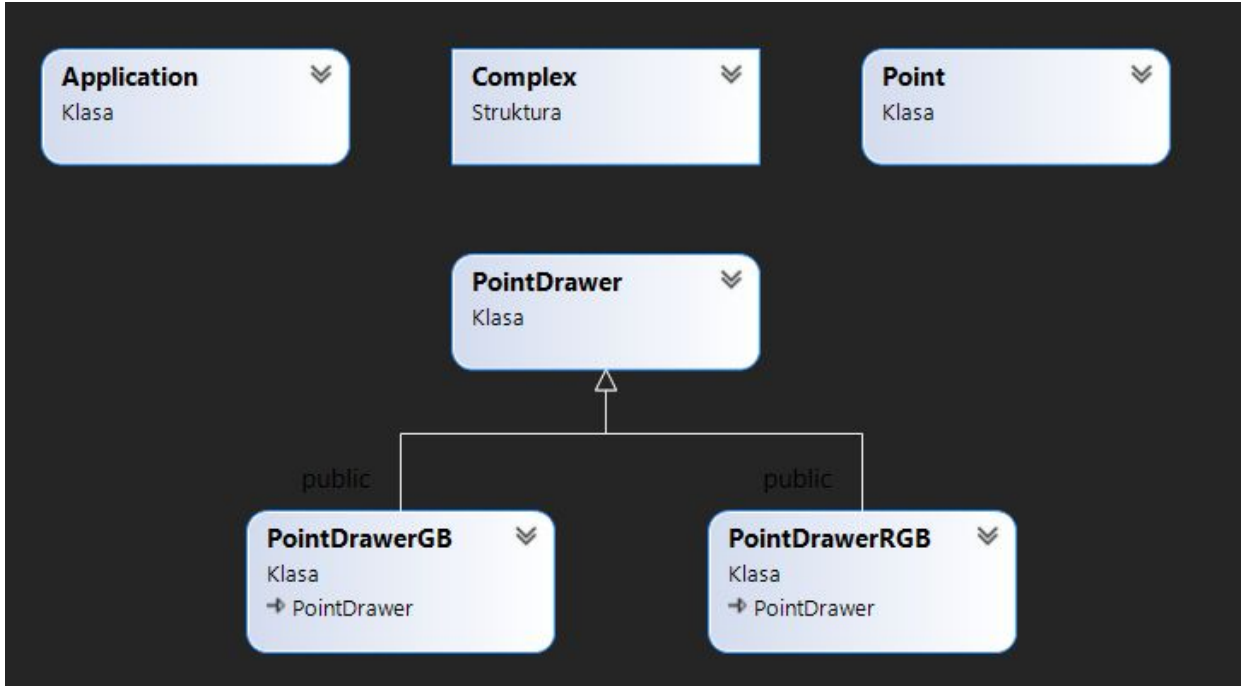


Diagram klas przedstawiający strukturę programu. Klasa PointDrawer jest klasą abstrakcyjną po której dziedziczą klasy PointDrawerGB i PointDrawerRGB. W ten sposób zainplementowałem polimorfizm.

Krótka instrukcja obsługi programu:

Uruchamianie programu:

Program uruchamiany debugując rozwiązanie Mandelbrot final.sln lub poprzez uruchomienie aplikacji .exe. (Mandelbrot final -> x64 -> Release -> Mandelbrot final.exe)

Działanie programu:

Po uruchomieniu programu zgodnie z powyższą instrukcją generuję ustalony wcześniej zbiór Mandelbrota.

Wyjście z programu:

Aby zakończyć pracę programu należy nacisnąć na klawiaturzę ustalony przycisk w przypadku mojego programu jest to “q” .

Dokumentacja:

Została utworzona w programie Doxygen, pełną dokumentację można odnaleźć w pliku Dokumentacja_html

Funkcjonalność programu

Kolor generowanego fraktala:

Przed uruchomieniem programu możemy ustalić kolor w jakim fraktal będzie się generował. Możemy wybrać generację w kolorzę lub czarnobiałą. Funkcjonalność tą zaimplementowałem w funkcjach odpowiednio PointDrawerRGB oraz PointDrawerGB. Aby ustalić odpowiedni kolor musimy Application.cpp podmienić odpowiednio

```
Application::Application() :
    drawer(new PointDrawerRGB()) // PointDrawerGB - czarno biale, PointDrawerRGB - kolorowe
{
```

Rozmiar okna programu:

Aby zmienić rozmiar okna programu należy przed uruchomieniem ostalić interesujące nas wartości w odpowiednim Application.h

```
const int width{ 800 };
const int high{ 700 };
```

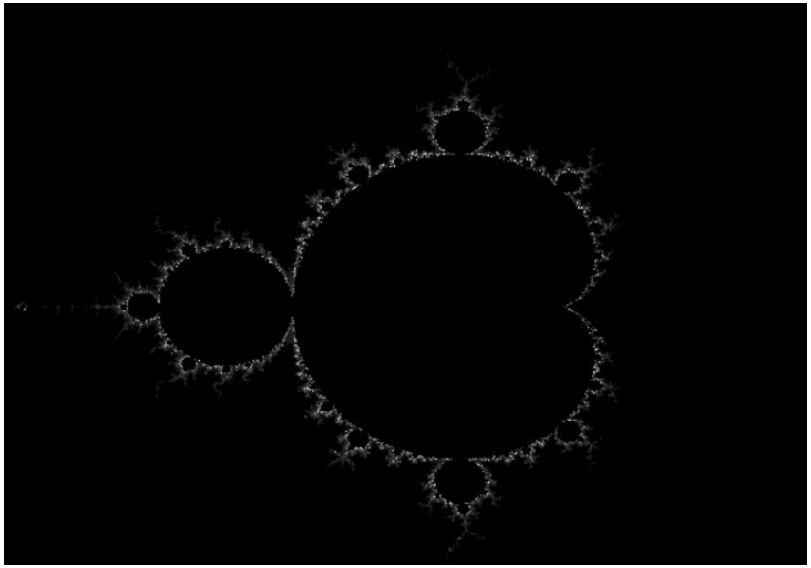
Funkcjonalność programu

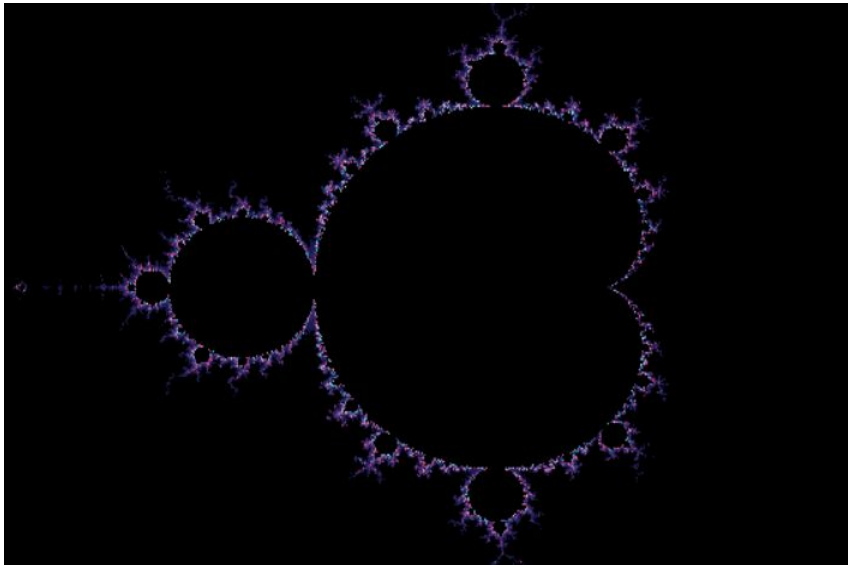
Parametry generowania fraktala:

Poprzez zmianę odpowiednich wartości możemy zminić parametry generowania naszego fraktala.
Zmian tych dokonujemy w Application.h
min,max - zakres punktów w jakich będzie generował się fraktal
maxIterrations - liczba wykonywanych iteracji
factory - zmiana szybkości przybliżania

```
double min{ -2.84 };
double max{ 2.0 };
int maxIterations{ 100 };
double factory{ 1.0};
```

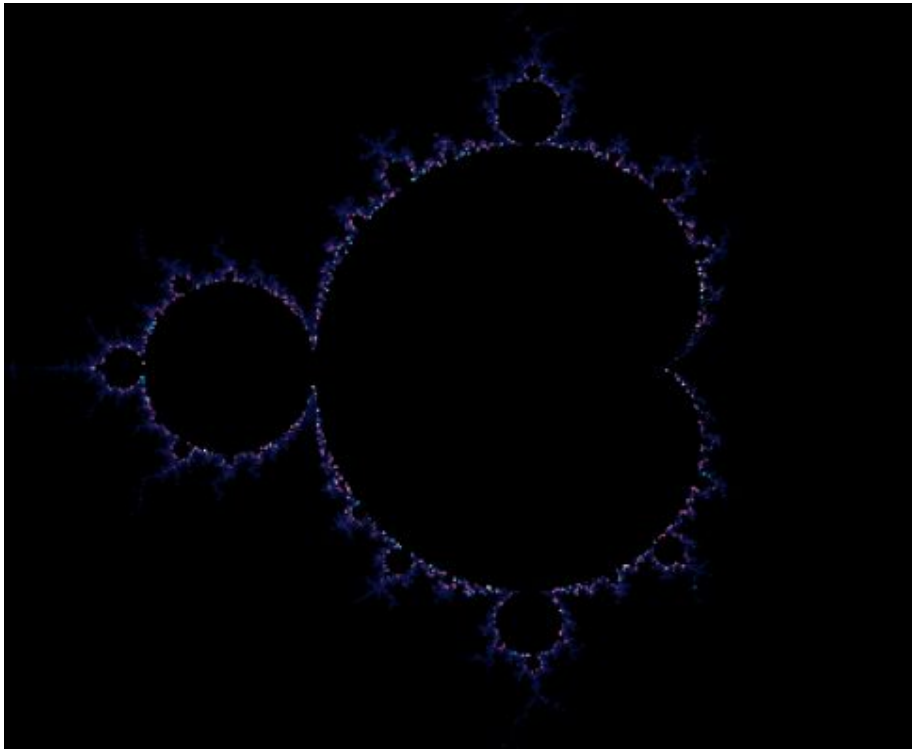
Testy





Generacja fraktala przy użyciu PointDrawerGB oraz PointDrawerRGB

Testy



```
double min{ -3.84 };
double max{ 2.0 };
int maxIterations{ 300 };
double factory{ 0.0};
const int width{ 850 };
const int high{ 850 };
```

Fraktal wygenerowany dla powyższych parametrów. Po ustawieniu factory =0 możemy zaobserwować tworzące się kolejne punkty naszego zbioru. Po zwiększeniu ilości wykonywanych iteracji zaobserwowałem spadek wydajności komputera.

Wnioski

Projekt został zrealizowany zgodnie z założeniami programowania obiektowego. Do przedstawienia generowanego fraktalu wykorzystałem bibliotekę SDL.
Zmieniając poszczególne parametry programu sprawdziłem jaki wpływ mają one na generacje fraktala. Możemy zaobserwować, że funkcję zmiany koloru fraktala działają w odpowiedni sposób co pokazałem odpowiednimi screenami. Po zamianie odpowiednich parametrów programu zaobserwowałem zmiany jakie mają one na działanie programu. Po zwiększeniu ilości iteracji obraz staje się bardziej zagęszczony, odczułem również spadek wydajności komputera, spowodowane jest większą ilością obliczeń jakie musimy przeprowadzić.