Zadania domowe

Ćwiczenie 2

Rysowanie obiektów 2-D przy pomocy tworów pierwotnych biblioteki graficznej OpenGL

<u>Zadanie 2.1</u>

Fraktal plazmowy (*Plasma fractal*)

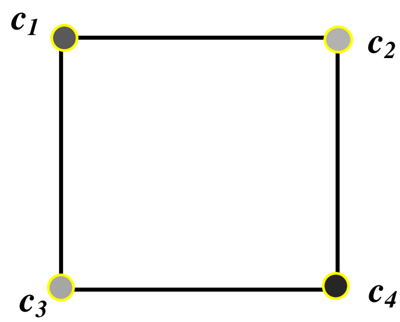
<u>Założenia:</u>

Kwadrat należy pokryć prostokątną siatką 2^n x 2^n punktów. Każdemu punktowi siatki należy w pewien sposób przypisać kolor lub stopień szarości.

<u>*Algorytm generacji fraktala plazmowego:*</u>

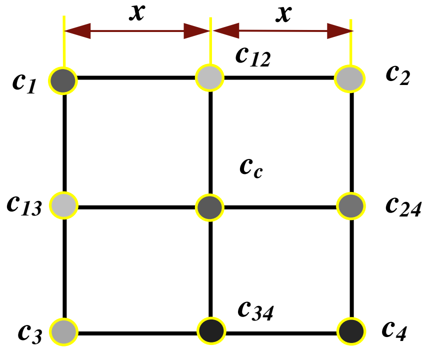
<u>Krok 1</u>

W narożnych punktach prostokąta siatki umieścić punkty o losowo wybranym (przy pomocy zadanego dyskretnego rozkładu prawdopodobieństwa) kolorze. W przypadku generacji obrazu monochromatycznego dla punktów wylosować stopień szarości.



<u>Krok 2</u>

Wyliczyć kolory dla pięciu nowych punktów siatki. Nowe punkty są generowane w środkach boków i w środku kwadratu.

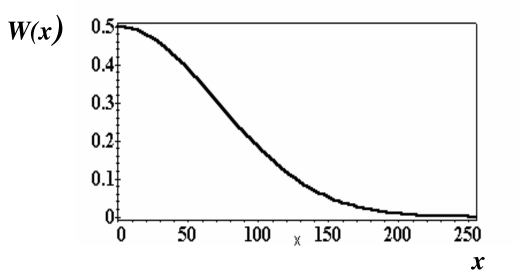


<u>Obliczenie koloru punktu w środku boku (np. c_{12})</u>

- Wylosować kolor c'_{12}
- Obliczyć kolor c_{12} ze wzoru

$$c_{12} = (1 - 2W(x)) \cdot c'_{12} + W(x) \cdot c_1 + W(x) \cdot c_2$$

gdzie funkcja *W(x)* ma na przykład przebieg pokazany na rysunku poniżej.

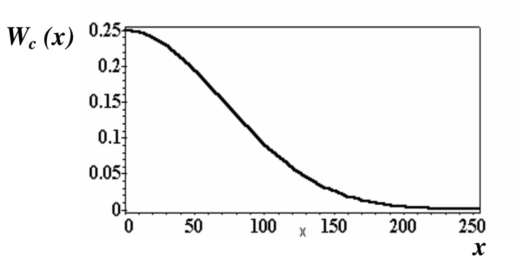


<u>Obliczenie koloru punktu c_c (w środku kwadratu)</u>

- Wylosować się kolor c'_c
- Obliczyć kolor c_c ze wzoru

$$c_c = (1 - 4W_c(x)) \cdot c'_c + \sum_{i=1}^{4} W_c(x) \cdot c_i$$

gdzie funkcja $W_c(x)$ wygląda tym razem na przykład tak:



Krok 3

Otrzymano w ten sposób cztery nowe kwadraty. Dla każdego z nich, powtarza się krok 2 a wynik dzieli się znów na mniejsze kwadraty, aż do osiągnięcia granicy rozdzielczości siatki. Efekt (w przypadku monochromatycznym) może wyglądać na przykład tak.

Zadanie do wykonania

Napisać program rysujący fraktal plazmowy w wersji monochromatycznej i barwnej z możliwością zmiany parametrów rysowania (wielkości obrazu, sposobu wyliczania barwy, parametrów probabilistycznych, funkcji ważących).

Zadanie 2.2

Układ odwzorowań iterowanych (*Iterated function system*)

Założenia:

Dane jest odwzorowanie (zwane odwzorowaniem afinicznym) w postaci:

$$\varphi: \begin{array}{l} x' = ax + by + c \\ y' = dx + ey + f \end{array}$$

Niech będą dane cztery odwzorowania afiniczne *φ1, φ2, φ2, φ4* o współczynnikach zapisanych w tabeli

φ	a	b	c	d	e	f
φ_1	-0.67	-0.02	0.00	-0.18	0.81	10.00
φ_2	0.40	0.40	0.00	-0.10	0.40	0.00
φ_3	-0.40	-0.40	0.00	-0.10	0.40	0.00
φ_4	-0.10	0.00	0.00	0.44	0.44	-2.00

Algorytm generacji zbioru punktów (fraktala):

Krok 1

Za punkt startowy procesu generacji zbioru wybrać dowolny punkt płaszczyzny R^2 i narysować jego obraz na ekranie.

Krok 2

Ze zbioru czterech odwzorowań φ_1, φ_2, φ_2, φ_4_ wylosować jedno, posługując się generatorem dyskretnej zmiennej losowej np. przyjąć rozkład równomierny: $p_i = 1/4$; $i=1, 2, 3, 4$

Krok 3

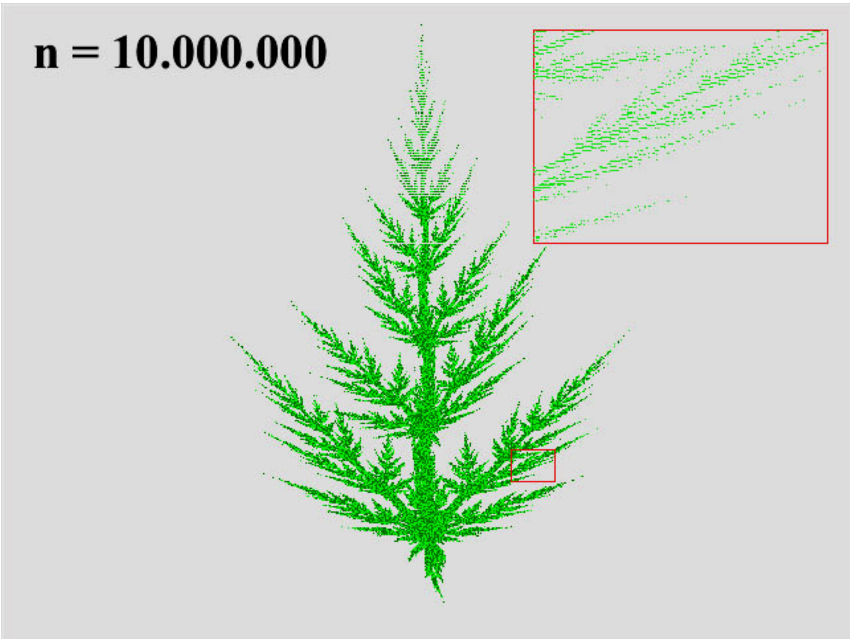
Używając wylosowanego odwzorowania wyliczyć współrzędne nowego punktu płaszczyzny R^2 i narysować obraz punktu.

Krok 4

Przyjąć wyliczony punkt, jako nowy punkt startowy i powtórzyć kroki 2 i 3.

Zadanie do wykonania

Napisać program rysujący określony prze podany powyżej algorytm zbiór z możliwością dokładniejszej wizualizacji jego fragmentu (rysunek poniżej). Zbadać wpływ liczby iteracji i losowości na proces generacji zbioru.



Zadanie 2.3

Obraz zbioru Mandelbrota (*Mandelbrota set*)

Założenia:

Niech będzie dany następujący ciąg liczb zespolonych

$$z_{j+1} = z_j^2 + c; \qquad z_0 = 0 + i\,0, \quad c - zespolone \qquad (*)$$

Definicja

Zbiór tych wartości parametru *c*, dla których ciąg liczb z_0, z_1, z_2, ... jest ograniczony, tworzy na płaszczyźnie zespolonej zbiór Mandelbrota (1980).

Algorytm generacji obrazu zbioru Mandelbrota:

Krok 1

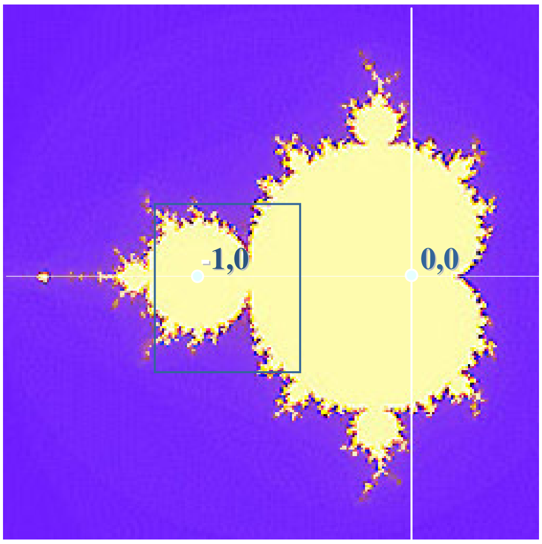
Współrzędne punktu kolejnego punktu płaszczyzny *(x, y)*, przyjąć jako cześć rzeczywistą i urojoną liczby *c*.

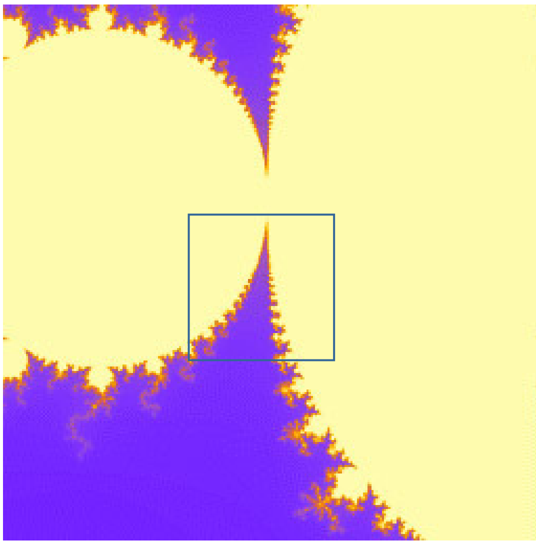
Krok 2

Sprawdzić czy dla liczby *c*, ciąg liczb zespolonych zadany wzorem (*) jest ograniczony. Jeśli tak, to narysować obraz punktu płaszczyzny *(x, y)* kolorem *A*, a w przypadku przeciwnym kolorem *B*.

Zadanie do wykonania

Napisać program rysujący obraz zbioru Mandelbrota z możliwością dokładniejszej wizualizacji jego fragmentu (rysunek poniżej). Dla uatrakcyjnienia rysunku można uzależnić kolor punktu zbioru od stopnia zbieżności ciągu z_j.





Uwaga:
Można znaleźć w literaturze inne (lepsze) algorytmy generacji zbioru Mandelbrota niż podany powyżej.

Zadanie 2.4

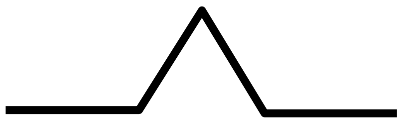
Płatek śniegu Kocha (*Koch snowflake*)

Założenia:

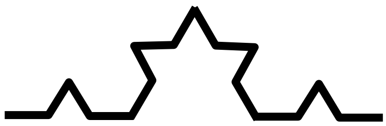
Algorytm tworzenia zbioru zwanego krzywą von Kocha jest następujący:

- dany jest odcinek $[0,1]$

- odcinek podzielić na trzy równe części i zbudować łamaną,



- odcinki łamanej podzielić na trzy równe części i na każdym z nich zbudować znów łamaną,

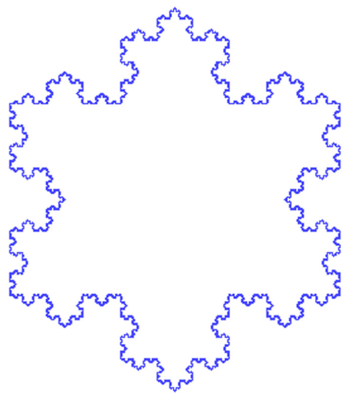


- i tak dalej

W granicy powstanie zbiór zwany krzywą von Kocha.

Zadanie do wykonania

Napisać program rysujący obraz trzech krzywych von Kocha, powstających z boków trójkąta równobocznego. Obszar ograniczony przez krzywą spróbować wypełnić w jakiś sposób, korzystając z algorytmu interpolacji koloru rysowanego wieloboku .



Jacek Jarnicki
Instytut Informatyki, Automatyki i Robotyki Politechniki Wrocławskiej

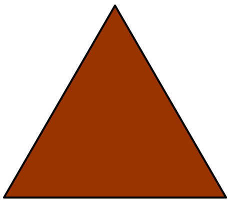
Zadanie 2.5

Trójkąt Sierpińskiego (*Sierpinski triangle*)

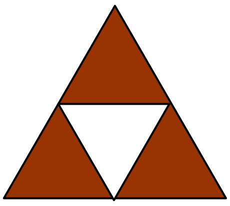
Założenia:

Trójkąt Sierpińskiego jest zbiorem, który powstaje podobnie jak opisany w instrukcji ćwiczenia dywan Sierpińskiego. Algorytm konstrukcji może być następujący:

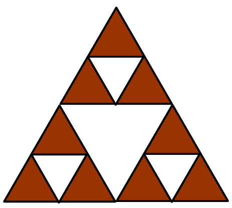
- dany jest trójkąt równoboczny,



- boki trójkąta dzielone są na dwie równe części i po połączeniu punktów podziału powstają cztery mniejsze trójkąty, z których jeden jest usuwany,



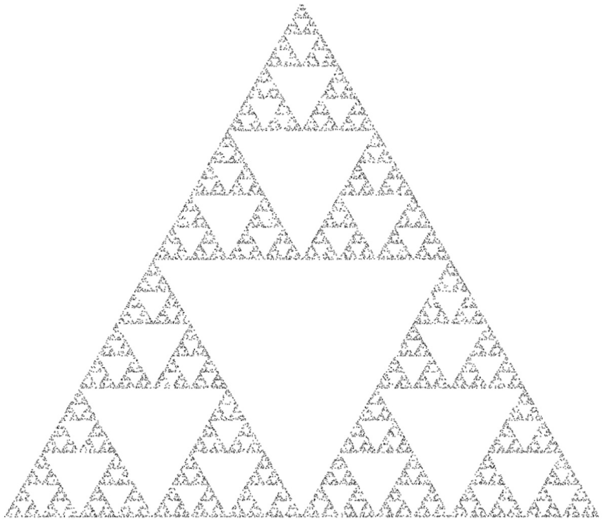
- dla czterech mniejszych trójkątów powtarzane są poprzednio wykonane czynności



- i tak dalej

W granicy powstanie zbiór zwany trójkątem Sierpińskiego.

Trójkąt można także uzyskać także inną metodą, przez wyliczanie kolejnych punktów. Obraz trójkąta po wykonaniu pewnej liczby iteracji wygląda na przykład tak.



Zadanie do wykonania

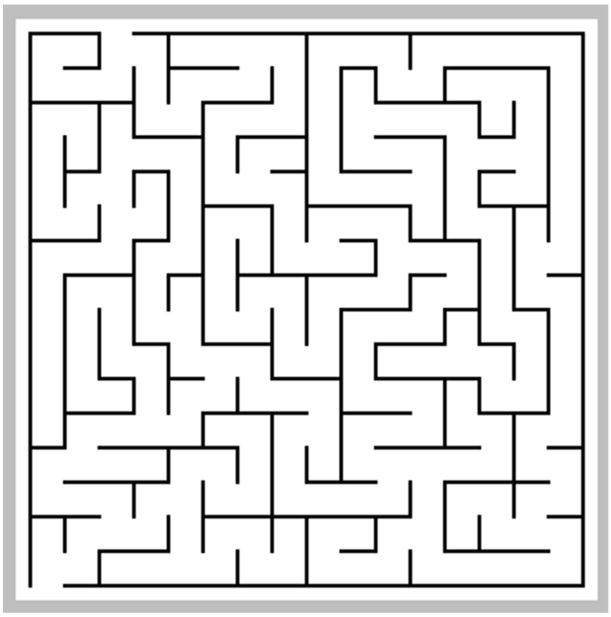
Napisać program rysujący przybliżony obraz trójkąta Sierpińskiego. Program powinien umożliwiać rysowanie trójkąta przy pomocy dwóch algorytmów. Pierwszy opisano powyżej natomiast drugi, pozwalający na narysowanie obrazu trójkąta z kolejno wyliczanych punktów trzeba odszukać w literaturze. W obu przypadkach należy przy w sposób losowy (mniej więcej tak, jak to opisano w instrukcji) dobierać kolory rysowanych elementów.

Zadanie 2.6

Labirynt (*Maze*)

Założenia:

W literaturze opisanych jest wiele algorytmów budowy labiryntów. Najczęściej rozpoczyna się od regularnej siatki kwadratowych komórek, zamkniętych z czterech stron ścianami i następnie różnymi metodami usuwa się niektóre ściany. W efekcie powstaje plan labiryntu na przykład taki, jak pokazano poniżej.



Zadanie do wykonania

Na podstawie literatury wybrać jeden z algorytmów generacji labiryntu (wykorzystujący losowość) i napisać program rysujący plan labiryntu.

Jacek Jarnicki
Instytut Informatyki, Automatyki i Robotyki Politechniki Wrocławskiej