Metody fraktalne w grafice komputerowej

Zestaw 3

We wszystkich zadaniach z zestawu będzie należało użyć mapy kolorów. Kilka przykładowych map kolorów znajduje się w pliku *colormaps.zip*. W archiwum każda mapa kolorów jest w dwóch formatach: (1) obraz, który może być wykorzystany jako tekstura 1D, (2) plik tekstowy, w którym w kolejnych liniach podane są składowe R, G i B kolorów tworzących mapę.

W każdym zadaniu należy zadbać, aby renderowany obraz był poddany anty-aliasingowi.

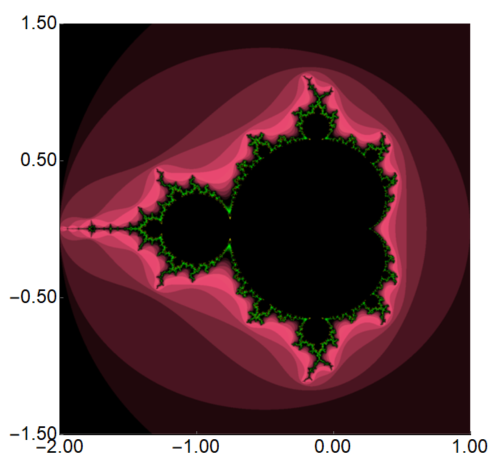
1. Zaimplementować algorytm generowania zbioru Mandelbrota (Algorytm 1).

Algorytm 1: Zbiór Mandelbrota.

Dane: $K \in \mathbb{N}$ – maksymalna liczba iteracji; $A \subset \mathbb{C}$ – obszar.
Wynik: Aproksymacja zbioru Mandelbrota w obszarze A.

```
for c ∈ A do
    R = max{|c|, 2}
    z = c
    i_c = 0
    for i = 1 to K do
        z = z^2 + c
        if |z| > R then
            i_c = i
            break
    Pokoloruj punkt c kolorem i_c
```

Przykład zbioru Mandelbrota dla $K = 50$, $A = [-2, 1] \times [-1.5, 1.5]$ oraz mapy kolorów *neon.map*:



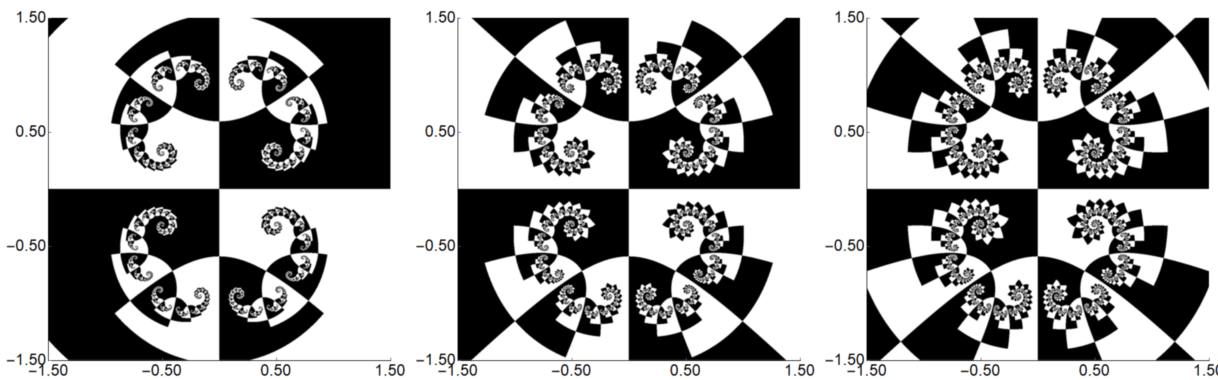
2. Zaimplementować algorytm generowania zbiorów Julii, w którym kolorowanie odbywa się za pomocą binarnej dekompozycji (Algorytm 2).

Algorytm 2: Zbiór Julii.

Dane: $c \in \mathbb{C}$ – parametr; $A \subset \mathbb{C}$ – obszar; $K \in \mathbb{N}$ – maksymalna liczba iteracji; $R \in \mathbb{R}_+$ – próg ucieczki.
Wynik: Aproksymacja zbioru Julii w obszarze A.

for $z_0 \in A$ **do**
    $i_c = 0$
    for $i = 1$ **to** K **do**
        $z_i = z_{i-1}^2 + c$
        if $|z_i| > R$ **then**
            $i_c = i$
            break
    Pokoloruj punkt z_0 używając binarnej dekompozycji dla z_i

Przykłady zbioru Julii dla $c = 0.35$, $A = [-1.5, 1.5]^2$, $K = 50$ i różnych wartości progu R, od lewej 2, 5, 10:



3. Zaimplementować algorytm wielomianografii z iteracjami (Algorytm 3). W programie mamy mieć możliwość wyboru metody znajdowania pierwiastków (co najmniej dwie metody, obie różne od metody Newtona), dowolnego wielomianu, rodzaju iteracji (co najmniej dwie iteracje, obie różne od iteracji Picarda) i testu zbieżności (co najmniej dwa różne testy zbieżności). Wartość wielomianu dla danego punktu należy obliczyć za pomocą schematu Hornera.

   Przykładowe metody znajdowania pierwiastków, które można wykorzystać, można znaleźć np. tutaj

   - Sekcja 5 „Newton Fraktale” http://www.3d-meier.de/tut20/Seite1.html
   - Gościniak, I., Gdawiec, K.: PSO-based Newton-like Method and Iteration Processes in the Generation of Artistic Patterns. Lecture Notes in Computer Science, vol. 11241, pp. 47-56, (2018) wersja PDF

- Gdawiec, K., Kotarski, W., Lisowska, A.: On the Robust Newton's Method with the Mann Iteration and the Artistic Patterns from Its Dynamics. Nonlinear Dynamics 104(1), 297-331, (2021) wersja PDF

Przegląd 17 różnych iteracji, które można wykorzystać, można znaleźć tutaj:

- Gdawiec, K., Kotarski, W.: Polynomiography for the Polynomial Infinity Norm via Kalantari's Formula and Nonstandard Iterations. Applied Mathematics and Computation 307, 17-30, (2017) wersja PDF

Algorytm 3: Renderowanie wielomianografu.

Dane: $p \in \mathbb{C}[Z]$, $\deg p \geq 2$ – wielomian; $A \subset \mathbb{C}$ – obszar; M – liczba iteracji; $I_v : \mathbb{C} \to \mathbb{C}$ – iteracja z parametrami v; $C_u : \mathbb{C} \times \mathbb{C} \to \{true, false\}$ – test zbieżności; $colours[0..k]$ – mapa kolorów.

Wynik: Wielomianograf w obszarze A.

for $z_0 \in A$ **do**
    $[n, z]$ = ITERATEPOINT(z_0, p, I_v, C_u, M)
    Pokoloruj z_0 używając n, z i mapy kolorów *colours*

Algorytm 4: ITERATEPOINT

Dane: $z_0 \in \mathbb{C}$ – punkt; $p \in \mathbb{C}[Z]$, $\deg p \geq 2$ – wielomian; $I_v : \mathbb{C} \to \mathbb{C}$ – iteracja z parametrami v; $C_u : \mathbb{C} \times \mathbb{C} \to \{true, false\}$ – test zbieżności; M – liczba iteracji.

Wynik: Numer iteracji i ostatni obliczony punkt.

ITERATEPOINT(z_0, p, I_v, C_u, M)
    $n = 0$
    while $n < M$ **do**
        $z_{n+1} = I_v(z_n)$
        if $C_u(z_n, z_{n+1}) = true$ **then**
            break
        $n = n + 1$
    return $[n, z_{n+1}]$

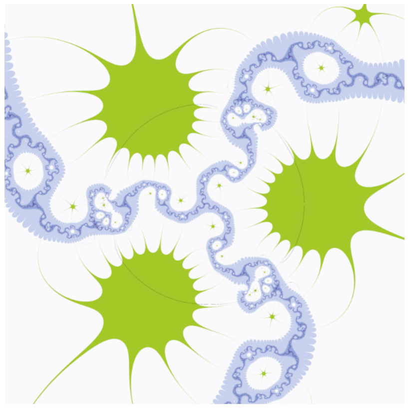
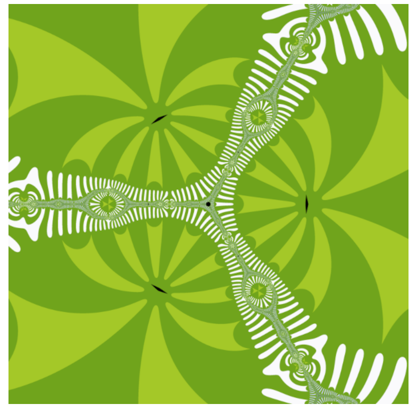
Przykład 1: wielomianografy dla metody Halleya, $p(z) = z^3 - 1$, $A = [-2, 2]^2$, $M = 30$, $\varepsilon = 0.001$, mapy kolorów z pliku *0408_093-s.map* oraz iteracji Picard–S z parametrami:

- $\alpha = 0.5 + 1.5\mathbf{i}$, $\beta = 0.8$ i testem zbieżności

$$||z_{n+1}|^2 - |z_n|^2| < \varepsilon,$$

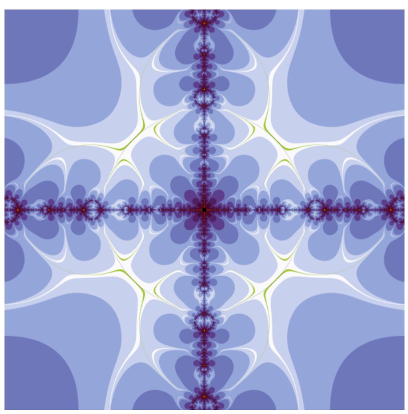
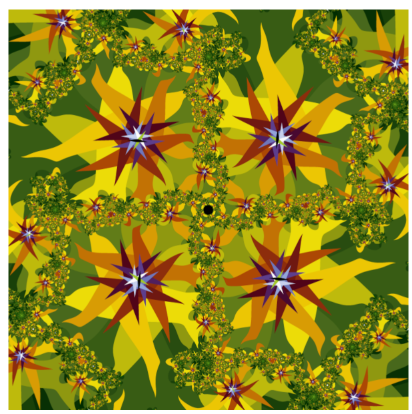
- $\alpha = 0.1$, $\beta = 0.9$ i testem zbieżności

$$|0.01(z_{n+1} - z_n)| + |0.029|z_{n+1}|^2 - 0.03|z_n|^2| < \varepsilon.$$

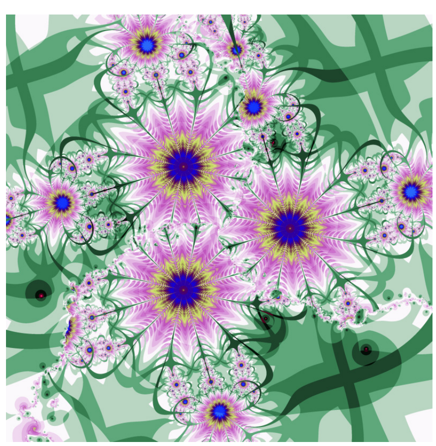
Przykład 2: wielomianografy dla metody Halleya, $p(z) = z^4 + 1$, $A = [-2,2]^2$, $M = 30$, $\varepsilon = 0.001$, mapy kolorów z pliku *0408_093-s.map* oraz iteracji Ishikawy z parametrami:

- $\alpha = 0.9$, $\beta = 0.1$ i testem zbieżności
$$||z_{n+1}|^2 - |z_n|^2| < \varepsilon,$$
- $\alpha = 0.6 - 0.75\mathbf{i}$, $\beta = 0.8$ i testem zbieżności
$$|0.01(z_{n+1} - z_n)| + |0.029|z_{n+1}|^2 - 0.03|z_n|^2| < \varepsilon.$$

4. Zaimplementować wielomianografię wielokrokową (Algorytm 5).

   Wielomianograf wielokrokowy:



Algorytm 5: Renderowanie wielomianografu wielokrokowego.

Dane: $A \subset \mathbb{C}$ – obszar; $\{p_1, p_2, \ldots, p_N\}$ – wielomiany; $\{I_{v_1}, I_{v_2}, \ldots, I_{v_N}\}$ – iteracje;
$\{M_1, M_2, \ldots, M_N\}$ – liczby iteracji dla poszczególnych kroków;
$\{C_{u_1}, C_{u_2}, \ldots, C_{u_N}\}$ – testy zbieżności dla poszczególnych kroków;
$\{f_1, f_2, \ldots, f_N\}$ – transformacje obszaru dla poszczególnych kroków;
$colours[0..k]$ – mapa kolorów.

Wynik: Wielomianograf wielokrokowy dla obszaru A.

for $z_0 \in A$ **do**
  $m = 0$
  $z = z_0$
  for $i = 1, 2, \ldots, N$ **do**
    $[n, u]$ = iteratePoint(z, p_i, I_{v_i}, C_{u_i}, M_i)
    $m = m + n$
    $z = f_i(u - z)$
  Wyznacz kolor dla z_0 używając m i mapy kolorów *colours*

został wygenerowany w obszarze $[-3, 3]^2$ używając mapy kolorów *jutemap.map*, $\varepsilon = 0.001$ oraz następujących parametrów:

- pierwszy krok: metoda Helleya, $p_1(z) = z^3 - 1$, $M_1 = 20$, iteracja Noora z $\alpha = 0.8 - 0.4\mathbf{i}$, $\beta = 0.2$, $\gamma = 0.9 - 0.5\mathbf{i}$, $f_1(z) = z/(0.75 + 2.5\mathbf{i})$ i testu zbieżności:

$$|z_{n+1} - z_n| < \varepsilon,$$

- drugi krok: metoda Newtona, $p_2(z) = z^4 + 1$, $M_2 = 20$, iteracja Ishikawy z $\alpha = 0.7$, $\beta = 0.7$, $f_2(z) = z/(0.75 + 2.5\mathbf{i})$ i testu zbieżności:

$$|0.04\Re(z_{n+1} - z_n)| < \varepsilon \vee |0.05\Im(z_{n+1} - z_n)| < \varepsilon,$$

gdzie $\Re(z)$, $\Im(z)$ oznaczają odpowiednio część rzeczywistą i urojoną liczby z.