KAUNO TECHNOLOGIJOS UNIVERSITETAS

FUNDAMENTALIŲJŲ MOKSLŲ FAKULTETAS

TAIKOMOSIOS MATEMATIKOS KATEDRA

Paulius Černiauskas

Poslinkių dinamikos panaudojimo fraktalinių vaizdų sintezės procedūrose analizė

Magistro darbas

Vadovas

prof. dr. J. Valantinas

KAUNAS, 2007

KAUNO TECHNOLOGIJOS UNIVERSITETAS

FUNDAMENTALIŲJŲ MOKSLŲ FAKULTETAS

TAIKOMOSIOS MATEMATIKOS KATEDRA

TVIRTINU
Katedros vedėjas
prof. dr. J.Rimas
2007 05 24

Poslinkių dinamikos panaudojimo fraktalinių vaizdų sintezės procedūrose analizė

Taikomosios matematikos magistro baigiamasis darbas

Vadovas
prof. dr. J. Valantinas
2007 05 24

Recenzentas
doc. dr. S. Maciulevičius
2007 05 24

Atliko
FMM 1/2 gr. stud.
P. Černiauskas
2007 05 24

KAUNAS, 2007

Paulius Černiauskas. Analysis of the application of shift dynamics to synthesizing fractal images: Master's Thesis in Applied Mathematics / supervisor prof. dr. J. Valantinas; Department of Applied Mathematics, Faculty of Fundamental Sciences, Kaunas University of Technology. – Kaunas, 2007. –37 p.

Summary

The contribution of this work is a new version of the escape time algoritm adapted for synthesizing fractal images, indentified with atractors of iterated functions systems (IFS). The proposesd synthesis algorithm is based on the use of shift dynamics, associated with one or another IFS. The strategy for the seperation of extended domains of the inverse affine transformations, specified by IFS, is developed.

In the field of computerized real word image models (digital images) the fractal approach is of outmost importance, because it facilitates perception and understanding of the information content of an image. To say more, it provides us with a powerful means to catch sight of a fundamental real word image property generally known as self-similarity. Due to this property, the research and development of algorithms („fractal techniques") to extract imortant fractal parameters from appropriate digital data has received significant attention in recent years.

In this work, the basic concepts and ideas that are needed to describe, state and solve the problem of synthesizing fractal images, identified with attractors of IFS, are introduced and explored. A new original approach (idea), leading to practical implementation of the shift dynamical system, associated with a particular IFS, is proposed (part 3). Some experimental results are given (Part 4).

Turinys

Summary 3
Turinys 4
Paveiksliukų turinys 5
Lentelių turinys 5
1. Įvadas 6
2. Iteruotųjų funkcijų sistemos, fraktalai, dinaminės poslinkių sistemos 7
2.1. Fraktalas – iteruotojų funkcijų sistemos (IFS) atraktorius 7
2.2. Fraktalų tipai 9
2.3. Fraktalinės dimensijos sąvoka 10
2.4. Su IFS susietos dinaminės poslinkių sistemos formavimo ypatumai 13
2.5. IFS atraktoriaus sintezės algoritmai 16
2.5.1. Determinuotasis algoritmas 16
2.5.2. Atsitiktinių iteracijų algoritmas 16
2.5.2. PL-algoritmas ir jo ribotumai 17
3. Poslinkių dinamikos panaudojimo fraktalinių vaizdų sintezės procedūrose efektyvumo tyrimas 19
3.1. IFS sudarančių afiniųjų transformacijų veikimo zonų atskyrimas 19
3.1.1. Kriterijaus parinkimas 20
3.1.2. Zonų nustatymas 20
3.2. Sintezuojamo fraktalinio vaizdo tolygus spalvinis užpildymas 22
4. Eksperimentinis tyrimas 24
4.1 Trumpas programinio įrankio, su kuriuo bus atliekami ekspermentai, aprasymas. 24
4.2. Afiniųjų transformacijų veikimo zonų atskyrimo kriterijaus tyrimas 27
4.3. Sintezuojamo fraktalinio vaizdo tolygaus spalvinio užpildymo tyrimas 32
4.3. Sugeneruotų fraktalinių vaizdų (IFS atraktorių) palyginimas 34
Išvados 36
Literatūros sąrašas 37
Priedai 38

Paveiksliukų turinys

2.1 pav. IFS atraktorių (fraktalų) pavysdžiai: 8
2.2 pav. Mandelbroto ir Žulija aibės susiję su parametrizuota dinamine sistema $\{C; z^2-\lambda\}$ (centre—Mandelbroto aibė) 9
2.3 pav. Van Kocho kreivė IFS $\{R^2; \omega_1, \omega_2, \omega_3, \omega_4\}$ atraktorius (fraktalas) 11
2.4 pav. Sierpinskio trikampis 12
2.5 pav. Dinaminė poslinkių sistema $\{A;S\}$, atitinkanti IFS $\{R^2; \omega_1, \omega_2, \omega_3\}$; IFS atraktorius – Sierpinskio trikampis 15
2.6 pav. PL-algoritmo interpretacija; 18
4. Eksperimentinis tyrimas 24
4.1 pav. Pagrindinis langas 24
4.2 pav. Pasirinkti IFS iš failo 25
4.3 pav. Atidaryti pasirinktą IFS failą 25
4.4 pav. Afiniųjų transformacijų veikimo zonos 26
4.5 pav. IFS atraktorius 26
4.6 pav. Pirmosios IFS $\{R^2; \omega_1, \omega_2\}$ atraktorius, gautas atsižvelgiant į išskirtas veikimo zonas: 28
4.7 pav. Antroios IFS $\{R^2; \omega_1, \omega_2\}$ atraktorius, gautas atsižvelgiant į išskirtas veikimo zonas: 29
4.8 pav. Trečiosios IFS $\{R^2; \omega_1, \omega_2, \omega_3, \omega_4\}$ atraktorius, gautas atsižvelgiant į išskirtas veikimo zonas: 30
4.9 pav. IFS atraktorius (be iteracijų skaičiaus korekcijos) 32
4.10 pav. IFS atraktorius (su iteracijų skaičiaus korekcija) 33
4.11 pav. Antrosios IFS atraktorius (be iteracijų skaičiaus korekcijos) 33
4.12 pav. Antrosios IFS atraktorius (su iteracijų skaičiaus korekcija) 34
4.13 pav. Sugeneruotų fraktalinių viazdų (IFS atraktorių) palyginimas. 35

Lentelių turinys

3.1 lentelė. Iteracijų skaičiaus korekcijų lentelė 23
4.1 lentelė. Pirmos IFS $\{R^2; \omega_1, \omega_2\}$ skaičiavimo laikinės sąnaudos 31
4.2 lentelė. Antros IFS $\{R^2; \omega_1, \omega_2\}$ skaičiavimo laikinės sąnaudos 31
4.3 lentelė. Trečios IFS $\{R^2; \omega_1, \omega_2, \omega_3, \omega_4\}$ skaičiavimo laikinės sąnaudos 32

1. Įvadas

Šiame darbe nagrinėjami fraktalų – iteruoųjų funkcijų sistemų (IFS) atraktorių – sintezės algoritmai. Pagrindinis dėmesys skiriamas pabėgimo laiko (PL)-algoritmui. PL-algoritmas yra pakankamai universalus. Pagrindinės šio algoritmo panaudojimo sričitys – netiesinių dinaminių sistemų, veikiančių kompleksinėje plokštumoje, analizė, kompleksinių daugianarių šaknų pritraukimo baseinų vizualizavimas ir kt. Geometrinių fraktalų (IFS atraktorių) sintezei šis algoritmas iki šiol nebuvo naudojamas, nors tokia galimybė, kaip teorinis rezultatas, yra žinoma. Pagrindinė to priežastis – IFS sudarančių afiniųjų transformacijų veikimo zonų atskyrimo kriterijaus nebuvimas. Tokio kriterijaus paieškai ir realizacijai darbe skiriamas didžiausias dėmesys. Rezultatas – nauja adaptyvi IFS sudarančių afiniųjų transformacijų veikimo zonų atskyrimo procedūra. Lygegrečiai spendžiama tolygaus spalvinio sintezuojamo (PL-algoritmo pagalba) fraktalinio vaizdo užpildymo problema. Pasiūlytas originalus sprendimas – problemiškai oriantuota iteracijų skaičiaus (sintezės metu) korekcija.

Darbe pristatomi ir preliminarūs su fraktalinių vaizdų (IFS atraktorių) sinteze susijusių eksperimentų rezultatai.

2. Iteruotųjų funkcijų sistemos, fraktalai, dinaminės poslinkių sistemos

2.1. Fraktalas – iteruotojų funkcijų sistemos (IFS) atraktorius

Imkime metrinę erdvę (X, d) ir joje apibrėžtą transformaciją $f: X\to X$. Pastebėsime, jog $f(S)=\{f(x)|\ x\in S\}$, kai $S\subset X$. Be to, transformacija f yra apgręžiama, kai ji yra abipusiškai vienareikšmė ir $f(X)=X$. Šiuo atveju, galima apibrėžti atvirkštinę transformaciją $f^{-1}: X\to X$ tokią, kad $f^{-1}(y)=x$, ir $x\in X$ yra vienintelis taškas, kuriam $f(x)=y$.

Transformacijos $f: X\to X$ iteracijomis pirmyn vadinamos transformacijos $f^{0n}: X\to X$, apibrėžiamos lygybėmis:

$$f^{00}(x)=x, f^{01}(x)=f(x), f^{0(n+1)}(x)=f(f^{0n}(x)), \text{ su visais } n=0, 1, 2, \ldots$$

Jeigu f yra apgręžiama, tai transformacijos f iteracijomis atgal vadinamos transformacijos $f^{0(-m)}: X\to X$, apibrėžiamos lygybėmis:

$$f^{0(-1)}(x)=f^{-1}(x), f^{0(-m)}(x)=(f^{0m})^{-1}(x), \text{ su visais } m=1, 2, 3, \ldots .$$

Praktiniuose taikymuose, svarbiausia akcentuoti ryšį tarp transformacijas apibūdinančių parametrų ir jų poveikyje atsirandančių geometrinių pasikeitimų (ištempimų, poslinkių, lenkimų ir pan.) erdvėse. Svarbu ir tai, kaip transformacijos veikia ne atskirus erdvės X taškus, o jos poaibius.

Kalbant apie geometrinius fraktalus (IFS atraktorius) paprastai, imamos Euklido metrinės erdvės — (R, d), (R^2, d) arba (R^3, d) —ir jose veikiančios afiniosios transformacijos.

Bendru atveju, afinioji transformacija $\omega: R^2\to R^2$, veikianti (tarkime, dvimatėje) Euklido erdvėje (R^2, d), apibrėžiama lygybe $\omega(x)=\omega(x_1, x_2)=(ax_1 + bx_2 + e, cx_1 + dx_2 + f)$, su visais $x=(x_1, x_2)\in R^2$; čia a, b, c, d, e, f yra realieji skaičiai (afiniosios transformacijos ω parametrai). Afiniosios transformacijos turi daug svarbių geometrinių ir algebrinių savybių. Jų pagalba galima realizuoti posūkio, atspindžio, panašumo, pražulniąsias ir kitas transformacijas Euklido erdvėje (R^2, d),[8].

Jeigu su visais $x, y\in R^2$ teisinga nelygybė $d(\omega(x), \omega(y))\le s\cdot d(x,y)$, $0\le s<1$, tai afinioji transformacija $\omega: R^2\to R^2$ vadinama suspaudžiančiąja, o skaičius s — afiniosios transformacijos suspaudimo koeficientu.

Imkime afiniųjų suspaudžiančiųjų transformacijų $\omega_i: R^2\to R^2$, $i=1, 2, \ldots, N$, rinkinį; atskirų afiniųjų transformacijų suspaudimo koeficientus pažymėkime s_i, $i=1, 2, \ldots, N$. Tada, Euklido erdvė (R^2, d) su joje veikiančių suspaudžiančiųjų afiniųjų transformacijų rinkiniu vadinama

iteruotųjų funkcijų sistema, ir žymima — IFS{R^2; ω_1, ω_2, …, ω_N}. IFS suspaudimo koeficientu laikomas skaičius s= max{s_1, s_2, …, s_N}.

Apibrėžkime dar vieną metrinę erdvę (Ή(R^2), h) tokiu būdu: Ή(R^2)— visų netuščių uždarųjų aibės R^2 poaibių aibė; h — metrika, nusakanti atstumą tarp bet kurių dviejų aibės Ή(R^2) elementų (aibės R^2 poaibių), būtent:

$$h(A, B)= \max\{d(A, B), d(B, A)\};$$

kai $d(A, B)= \max\limits_{x\in A}\{\min\limits_{y\in B}\{d(x, y)\}\}$ ir $d(B, A)= \max\limits_{y\in B}\{\min\limits_{x\in A}\{d(x, y)\}\}$.

Įvestoji erdvė dažnai vadinama fraktaline erdve. Perkelkime IFS sudarančias afiniąsias transformacijas į erdvę (Ή(R^2), h). Tada ω_i: Ή(R^2)→ Ή(R^2), apibrėžiama lygybe $\omega_i(B)=\{\omega_i(y)|y\in B\}$ ($\forall B\in$ Ή(R^2)), yra suspaudžiančioji transformacija erdvėje (Ή(R^2), h), ir jos suspaudimo koeficientas lygus s_i (i={1, 2, …, N}).

Pagaliau, apibrėžkime dar vieną transformaciją W: Ή(R^2) → Ή(R^2) fraktalinėje erdvėje (Ή(R^2), h), tokiu būdu: čia $W(B)=\omega_1(B)\cup \omega_2(B)\cup \ldots \cup\omega_N(B)= \bigcup\limits_{i=1}^{N}\omega_i(B)$, $\forall B\in$Ή(R^2).

Galima įsitikinti, jog pastaroji transformacija irgi yra suspaudžiančioji, t.y.

$h(W(B), W(C))\le s\cdot h(B, C)$, su visais B, $C\in$Ή(R^2); be to, s= max{s_1, s_2, …, s_N}.Vienintelis nejudamasis transformacijos W: Ή(R^2) → Ή(R^2) taškas A ($A\in$Ή(R^2)) toks, kad A= $W(A)$= $\bigcup\limits_{i=1}^{N}\omega_i(A)=\lim\limits_{n\to\infty}W^{0n}(B)$, $\forall B\in$Ή(R^2), vadinamas IFS atraktoriumi (arba fraktalu). Daug įvairių iteruotųjų funkcijų sistemų bei jų atraktorių pavyzdžių galima rasti literatūroje [1] (2.1 pav.).

(a) **(b)** **(c)**

2.1 pav. IFS atraktorių (fraktalų) pavysdžiai:

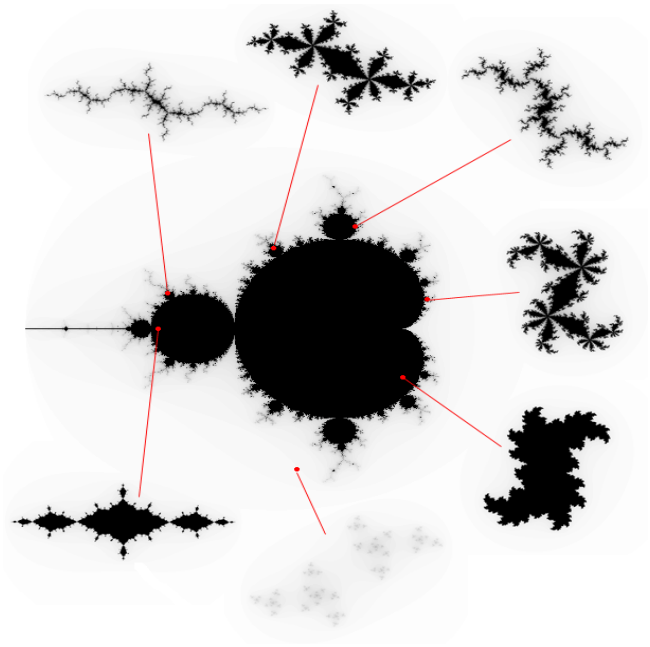
(a) IFS {R^2; ω_1, ω_2, ω_3} atraktorius – Sierpinskio trikampis; (b) IFS {R^2; ω_1,ω_2,ω_3,ω_4} atraktorius – pušis; (c) IFS {R^2; ω_1, ω_2, ω_3, ω_4} atraktorius – paparčio lapas

2.2. Fraktalų tipai

Fraktalinėje geometrijoje išskiriami šie fraktalų tipai — geometriniai fraktalai, algebriniai fraktalai, stochastiniai fraktalai.

Geometriniai fraktalai siejami su ankstyvuoju fraktalinės geometrijos (kaip tyrimo krypties) vystymosi periodu. Faktiškai, tai iteruotųjų funkcijų sistemų, charakterizuojamų įvairiais afiniųjų transformacijų (veikiančių Euklido erdvėse) rinkiniais, atraktorių aibė. Šio tipo fraktalai yra vaizdžiausi ir, galima sakyti, geriausiai išanalizuoti.

Algebriniai fraktalai priklauso pačiai didžiausiai fraktalų grupei. Jie siejami su netiesinėmis dinaminėmis sistemomis, veikiančiomis kompleksinėje plokštumoje. Šio tipo fraktalų sintezei gana efektyviai panaudojamas anksčiau paminėtas PL-algoritmas. Vienas ryškiausių šio tipo fraktalų yra Mandelbroto aibė ir su ja susijusios Žulija aibės (2.2 pav.). Šios aibės gaunamos, tiriant netiesinės parametrizuotos sistemos $\{C;\ f_\lambda(z)= z^2-\lambda\}$, dinamiką. Pastebėsime, jog algebriniai fraktalai nėra pilnai ištirti, jie vis dar slepia savyje daug paslapčių.



2.2 pav. Mandelbroto ir Žulija aibės susiję su parametrizuota dinamine sistema $\{C; z^2-\lambda\}$ (centre—Mandelbroto aibė)

Stochastinius fraktalus, tam tikra prasme, galima būtų priskirti geometriniams fraktalams. Skirtumas tas, jog vietoj įprastinių iteruotųjų funkcijų sistemų imamos parametrizuotos sistemos. Iteracinio proceso metu keičiamos parametrų reikšmės, ir tai daroma atsitiktinai (atsitiktine tvarka). Rezultatas — gaunami fraktalai daugumoje atvejų yra labai panašūs į realaus pasaulio objektus

(„asimetriniai“ medžiai, karpytos pakrantės linijos ir pan.). Dvimačiai stochastiniai fraktalai naudojami, modeliuojant vietovės reljefą, jūros paviršių ir kt.

2.3. Fraktalinės dimensijos sąvoka

Skaičiai, charakterizuojantys fraktalus, paprastai, vadinami fraktalinėmis dimensijomis. Tai gana svarbios kiekybinės fraktalų charakteristikos. Jos leidžia įvertinti intuityvų pojūtį apie tai, kaip tirštai („tankiai“) fraktalas užpildo erdvę, kurios poaibis jis pats yra. Fraktalinė dimensija—tai objektyvi priemonė fraktalų palyginimui.

Istoriškai, svarbu apibrėžti Hausdorfo ir Bezicovičiaus matą bei dimensiją, ir kartu atskleisti su jų apskaičiavimu susijusius sunkumus, [8].

Imkime poaibį $A \subset \mathrm{R}^n$; čia (R^n, d) —n-matė Euklido erdvė. Tegu $|A|=\mathrm{diam}(A)=\sup\{d(x,y) |\ x, y \in A\}$ žymi poaibio A diametrą.

Fiksuokime teigiamą skaičių $\varepsilon>0$. Imkime skaičią poaibių A_i ($|A_i| \le \varepsilon$) aibę $\{A_i\}$ ir tokią, kad ji padengtų poaibį A, t.y. $A \subset \bigcup_{i=1}^{\infty} A_i$; rinkinys (aibė) $\{A_i\}$ vadinamas poaibio A ε-denginiu.

Toliau, tarkime, kad $p>0$, ir apibrėžkime dydį $H_\varepsilon^p(A)$ tokiu būdu:

$$H_\varepsilon^p(A) = \inf\{ \sum_{i=1}^{\infty} |\ A_i\ |^p \mid \{A_i\} \text{ yra poaibio } A \ \varepsilon\text{-denginys}\}.$$

Faktiškai, tai minimizavimo uždavinys, kai stengiamasi minimizuoti dengiančiųjų aibių diametrų, pakeltų p—tuoju laipsniu, sumą (pagal visus įmanomus poaibio A ε-denginius). Kita vertus, dydis $H_\varepsilon^p(A)$, kaip funkcija, yra nemažėjantis, kai ε artėja prie nulio. Pereidami prie ribos, kai $\varepsilon \to 0$, apibrėšime Hausdorfo ir Bezicovičiaus matą (poaibiui A)

$$H^p(A) = \lim_{\varepsilon \to 0} H_\varepsilon^p(A).$$

Dydžio $H^p(A)$ priklausomybės nuo parametro p grafikas rodo, jog yra kritinė p reikšmė, kai $H^p(A)$ „šoka“ nuo ∞ prie 0 [8]. Tą kritinį tašką atitinkanti p reikšmė vadinama poaibio A Hausdorfo ir Bezicovičiaus dimensija žymėsime $D_H(A)$, t.y.

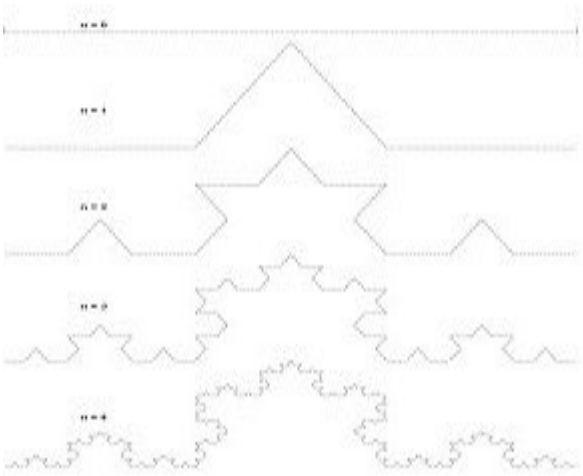
$$D_H(A)=\sup\{p \mid H^p(A)=\infty\} = \inf\{p \mid H^p(A) =0\}.$$

Pastebėsime, jog šią dimensiją apskaičiuoti netgi „paprastoms“ aibėms (fraktalams) yra pakankamai sudėtinga. Kaip pavyzdį, panagrinėkime Kocho kreivę (2.3 pav.; [8]). Po k iteracijų kreivę sudaro 4^k segmentai, kurių kiekvieno ilgis 3^{-k}. Todėl kreivę galima padengti (tiksliai) 4^k aibėmis, kurių diametrai lygūs 3^{-k}. Tokiu būdu,

$$H_{\varepsilon}^{p}(A) = H_{3^{-k}}^{p}(A) = 4^{k}(3^{-k})^{p}.$$

Pertvarkykime šią išraišką taip: $H_{\varepsilon}^{p}(A) = H_{3^{-k}}^{p}(A) = 3^{k(D-p)}$; čia D=ln4/ln3. Pereidami prie ribos, kai dengiančiųjų aibių diametrai artėja prie nulio (t.y. $k \to \infty$), gauname

$$H^{p} = \lim_{k\to\infty} H_{3^{-k}}^{p} = \begin{cases} \infty, p < D \\ 1, p = D \\ 0, p > D \end{cases}.$$



2.3 pav. Van Kocho kreivė IFS $\{R^2; \omega_1, \omega_2, \omega_3, \omega_4\}$ atraktorius (fraktalas)

Taigi, galima teigti, jog Van Kocho kreivės Hausdorfo ir Bezicovičiaus dimensija lygi D=ln4/ln3≅1.262. Vėliau, B. Mandelbrotas šią dimensiją pavadino fraktaline dimensija. Faktiškai, tai nauja erdvės (poaibių) matavimo priemonė. Daugelyje šaltinių, fraktalinės dimensijos sąvokos įvedimui naudojamas šiek tiek supaprastintas požiūris (interpretacija).

Imkime metrinę erdvę (R^n, d). Tegul $A \in Ή(R^n)$. Tarkime, kad $\varepsilon>0$ ir $N(A, \varepsilon)$ žymi mažiausią uždarųjų rutulių $B(x, \varepsilon)=\{y \in R^n \mid d(x, y)\le\varepsilon\}$, sudarančių baigtinį poaibio A denginį, skaičių, t.y. $N(A, \varepsilon)$—mažiausias sveikasis skaičius toks, kad $A \subset \bigcup_{i=1}^{N(A,\varepsilon)} B(x_i, \varepsilon)$;

čia $\{x_1, x_2, \ldots, x_{N(A,\varepsilon)}\} \subset R^n$.

Fraktalinės dimensijos sąvoka įvedama, remiantis intuityvia idėja, jog aibė A ($A \subset R^n$) turi fraktalinę dimensiją D, jeigu $N(A,\varepsilon) \approx C\varepsilon^{-D}$; čia: C — tam tikra konstanta; „≈“ reiškia, jog $\lim_{\varepsilon\to 0}(\ln f(\varepsilon)/\ln g(\varepsilon))=1$, kai $f(\varepsilon)\approx g(\varepsilon)$. Dabar, išsprendę ($N(A,\varepsilon) \approx C\varepsilon^{-D}$) „lygybę“ ($D$ atžvilgiu), gauname

$$D \approx \frac{\ln N(A,\varepsilon) - \ln C}{\ln\left(\frac{1}{\varepsilon}\right)}.$$

Perėję prie ribos ($\varepsilon \to 0$), turime:

$$D = \lim_{\varepsilon \to 0} \frac{\ln N(A, \varepsilon)}{\ln\left(\frac{1}{\varepsilon}\right)}.$$

Ši riba vadinama poaibio A ($A \subset \mathrm{R}^n$) fraktaline dimensija ir žymima $D(A) = D$.

Žemiau pateikiame keletą svarbesnių teiginių, skirtų eksperimentiniam fraktalinės Euklido erdvės poaibių dimensijos nustatymui.

1 teorema. Tarkime, kad $A \subset$ Ĥ(R^m); be to, erdvė R^m padengta nesusikertančiais m—mačiais „kubiukais“ (dėžutėmis), kurių briaunos ilgis lygus $1/2^n$; tegul $N_n(A)$ žymi „kubiukų“, persidengiančių su poaibiu A, skaičių. Tada skaičius

$$D = \lim_{n \to \infty} \left\{ \frac{\ln N_n(A)}{\ln(2^n)} \right\}$$

vadinamas poaibio A fraktaline dimensija.

Būtent pastarasis teiginys sudaro (dažniausiai) pagrindą eksperimentiniam realaus pasaulio (fraktalinio) objekto dimensijos nustatymui (dimensijos įverčio gavimui).

Kaip pavyzdį, panagrinėkime fraktalą A (2.4 pav. Sierpinskio trikampis). Nesunku pastebėti, jog:

$N_1(A)=3$, kai $n=1$;

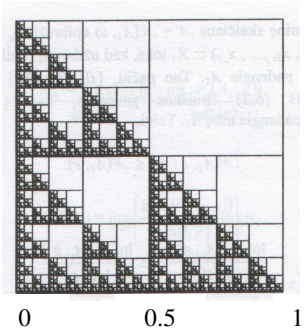
$N_2(A)=9$, kai $n=2$, t.y. kai $\varepsilon_2 = \frac{1}{2^2} = \frac{1}{4}$;

$N_3(A) = 3^3 = 27$, kai $n = 3$ ir t.t.

Pagaliau, $N_n(A) = 3^n$, kai $\varepsilon_n = \frac{1}{2^n}$

Taigi, fraktalinė Sierpinskio trikampio dimensija lygi:

$$D = \lim_{n \to \infty} \frac{\ln 3^n}{\ln 2^n} = \frac{\ln 3}{\ln 2} \cong 1.585.$$



2.4 pav. Sierpinskio trikampis

2 teorema. Tarkime, kad $A \subset \mathrm{R}^n$ sutampa su visiškai nejungios IFS $\{\mathrm{R}^2;\ \omega_1,\ \omega_2,\ \ldots \omega_N\}$, sudarytos iš afiniųjų panšumo transformacijų, atraktoriumi. Tada, poaibio A fraktalinė dimensija D yra lygties

$$\sum_{i=1}^{N} s_i^D = 1$$

sprendinys; čia s_i yra i—tosios afiniosios transformacijos suspaudimo koeficientas, (i=1, 2, …, N).

Pavydžiui, Kocho kreivės atveju (2.3 pav.), pastarojo teiginio taikymas leidžia gauti tokį rezultatą – kadangi Kocho kreivė yra IFS $\{\mathrm{R}^2;\ \omega_1,\ \omega_2,\ \omega_3,\ \omega_4\ \}$ atraktorius, be to, s_i=1/3, su visais i=1, 2, 3, 4, tai $\sum_{i=1}^{4}\left(\frac{1}{3}\right)^D = 1$; iš čia $4\cdot\left(\frac{1}{3}\right)^D = 1$; $\left(\frac{1}{3}\right)^D = \frac{1}{4}$; $D = \frac{\ln 4}{\ln 3} \cong 1.262$

3 teorema. Tarkime, kad $D(A)$ ir $D_H(A)$ žymi kokio nors aprėžto poaibio $A \subset \mathrm{R}^n$ atitinkamai fraktalinę bei Hausdorfo ir Bezicovičiaus dimensijas. Tada

$$0 \le D_H(A) \le D(A) \le n.$$

Galima būtų teigti, jog $D_H(A)$ „šiek tiek subtiliau“ charakterizuoja poaibį A, negu $D(A)$. Yra ir daugiau požiūrių bei interpretacijų, pateikiant fraktalines Euklido erdvės poaibių dimensijas. Tai — koreliacijos dimensija, informacijos dimensija, Liapunovo dimensija ir panašiai. Plačiau apie jas čia nekalbėsime.

Pagaliau, pastebėsime, kad fraktalinę dimensiją galima panaudoti kaip kriterijų, palyginant įvairius geometrinių fraktalų (IFS atraktorių) sintezės algoritmus, siekiant įvertinti sintezės algoritmų „darbo“ kokybę.

2.4. Su IFS susietos dinaminės poslinkių sistemos formavimo ypatumai

Kaip jau buvo minėta, dinamine sistema vadinama transformacija $f: X \to X$ metrinėje erdvėje (X,d); paprastai, dinaminė sistema žymima $\{X;f\}$; be to, seka $\left\{f^{0n}(x)\right\}_{n=0}^{\infty}$ vadinama taško $x \in X$ orbita.

Dinaminės sistemos yra įdomesnės tais atvėjais, kai transformacijos f nėra suspaudžiantieji atvaizdžiai, kaip rodo patirtis, tada atskirų taškų orbitos būna geometriškai gana sudėtingos.

Imkime dinaminę sistemą $\{X;\ f\}$. Taškas $x \in X$ vadinamas periodiniu f tašku, jeigu yra skaičius $n \in \{1,2,...\}$ toks, kad $f^{0n}(x) = x$. Skaičius n vadinamas taško x periodu. Mažiausias iš tokių skaičių vadinamas minimaliu periodinio taško x periodu.

Periodinio f taško orbita vadinama f ciklu; f ciklo periodas yra taško cikle periodas; minimalus f ciklo periodas – tai skirtingų ciklo taškų skaičius.

Grįžkime prie dinaminės sistemos $\{X;f\}$. Tarkime, kad $x_f \in X$ yra nejudamasis f taškas, t.y. $f(x_f)=x_f$. Taškas x_f vadinamas pritraukiančiuoju nejudamuoju f tašku, jeigu yra teigiamas skaičius $\varepsilon>0$ toks, kad $f: B(x_f;\varepsilon) \to B(x_f;\varepsilon)$, t.y. rutulys B yra atvaizduojamas į save; be to, f yra suspaudžiantysis atvaizdis aibėje $B(x_f;\varepsilon)$ (beje, $B(x_f;\varepsilon)=\{y \in X \mid d(x_f;y) \le \varepsilon\}$).

Taškas x_f vadinamas atstumiančiuoju f tašku, jeigu yra skaičiai $\varepsilon>0$ ir $c>1$ tokie, kad $d(f(x_f),f(y)) \ge c \cdot d(x_f,y)$, su visais $y \in B(x_f,\varepsilon)$.

Periodinis f taškas (periodas n) yra pritraukiantysis, jeigu jis yra transformacijos $f^{\circ n}$ pritraukiantysis nejudamasis taškas. Periodo n ciklas yra pritraukiantysis f ciklas, jeigu į ciklą įeina periodinis (periodas n) pritraukiantysis f taškas. Periodinis f taškas (periodas n) yra atstumiantysis, jeigu jis yra transformacijos $f^{\circ n}$ atstumiantysis nejudamasis taškas. Periodo n ciklas yra atstumiantysis f ciklas, jeigu į jį įeina periodinis (periodas n) atstumiantys f taškas.

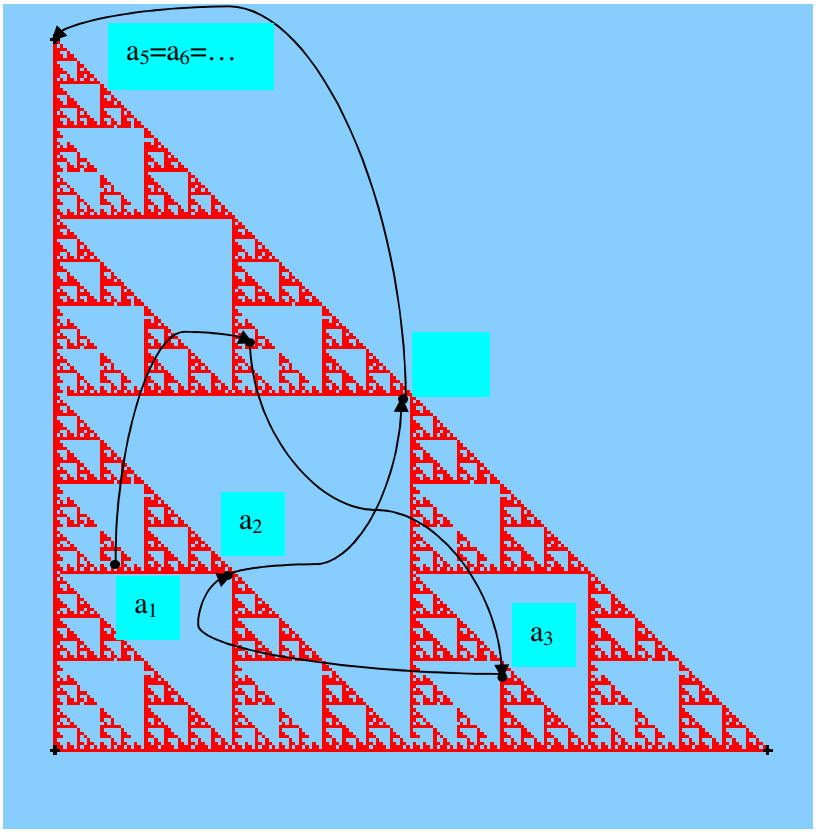
Tarkime, kad $\{X;\ \omega_1,\ \omega_2,\ \dots\ ,\ \omega_n\}$ yra IFS. Tada $\{\acute{\mathrm{H}}(X),\ W\}$ yra dinaminė sistema; čia $W(B)=\bigcup_{n=1}^{N} w_n(B)$, $\forall B \in \acute{\mathrm{H}}(x)$ (2.1 skyrelis). Tokio tipo sistemos kartais vadinamos aibes veikiančiomis dinaminėmis sitemomis.

Dinaminės sistemos savoką susiejame su iteruotojų funkcijų sistemomis, veikiančiomis metrinėse erdvėse.

Jei IFS $\{X;\ \omega_1,\ \omega_2,\ \dots\ ,\ \omega_n\}$ atraktorius yra aibė A ir IFS yra visiškai nejungi (t.y. $\omega_i(A) \cap \omega_j(A) = \varnothing$), kai $i \ne j$) tai transformacija $\omega_n: A \to A$, $n \in \{1,2,\dots,N\}$, yra abipusiškai vienareikšmė. Šią IFS atitinkanti poslinkio transformacija $S: A \to A$ apibrėžiama taip:

$S(a) = \omega_n^{-1}(a), \quad \forall a \in \omega_n(A); \quad n \in \{1,2,\dots,N\}$. Dinaminė sistema $\{A;S\}$ vadinama dinamine poslinkių sistema, atitinkančia duotąją IFS.

Dinaminės poslinkių sistemos $\{A;S\}$, atitinkančios iteruotųjų funkcijų sistema $\mathrm{IFS}\left\{R^2;\ 0.47\begin{pmatrix}x_1\\x_2\end{pmatrix},\ 0.47\begin{pmatrix}x_1\\x_2\end{pmatrix}+\begin{pmatrix}1\\0\end{pmatrix},\ 0.47\begin{pmatrix}x_1\\x_2\end{pmatrix}+\begin{pmatrix}0\\1\end{pmatrix}\right\}$, kurios atraktorius yra Sierpinskio trikampis, veikimo rezultatas (taško a_0 orbita) parodytas 2.5 pav.



2.5 pav. Dinaminė poslinkių sistema {*A;S*}, atitinkanti IFS {R^2; ω_1, ω_2, ω_3}; IFS atraktorius – Sierpinskio trikampis

Matome , jog taško a_0 orbita baigiasi nejudamajame taške. Taškas $a_4 \in A$ yra atstumiantysis nejudamasis dinaminės poslinkių sistemos taškas.

Imkime IFS {X; ω_1, ω_2,}, kurios atraktorius yra aibė A. Tarkime, kad abi transformacijos, ω_1: $A \to A$ ir ω_2: $A \to A$, yra apgręžiamos. Aibės A taškų seka $\{x_n\}_{n=0}^{\infty}$ vadinama dinaminės poslinkių sistemos, atitinkančios duotąją IFS, orbita, jeigu

$$x_{n+1} = \begin{cases} \omega_1^{-1}(x_n), kai \quad x_n \in \omega_1(A) \quad ir \quad x_n \notin \omega_1(A) \cap \omega_2(A), \\ \omega_2^{-1}(x_n), kai \quad x_n \in \omega_2(A) \quad ir \quad x_n \notin \omega_1(A) \cap \omega_2(A), \\ bet \quad kuris \quad aibėi \quad \{\omega_1^{-1}(x_n), \omega_2^{-1}(x_n)\} \quad taškas, \quad kai \quad x_n \in \omega_1(A) \cap \omega_2(A) \end{cases},$$

su visais $n \in \{1,2,...\}$; žymėsime $x_{n+1}=S(x_n)$; $S:A \to$A. Dinaminė poslinkių sistemą {*A;S*} vadinsime dinamine atsitiktinių poslinkių sistema, atitinkančia IFS.

2.5. IFS atraktoriaus sintezės algoritmai

Yra žinomi ir plačiai praktikoje taikomi keli frkatalinių vaizdų (IFS atraktorių) sintezės (generavimo) algoritmai. Tai — determinuotasis algoritmas, atsitiktinių iteracijų algoritmas bei PL-algoritmas. Trumpai pakomentuosime šiuos algoritmus.

2.5.1. Determinuotasis algoritmas

Determinuotojo fraktalų (IFS atraktorių) generavimo algoritmo „veikimas“ tiesiogiai remiasi IFS atraktoriaus apibrėžimu.

Tarkime, kad $\{R^2; \omega_1, \omega_2, \ldots, \omega_N\}$ yra iteruotųjų funkcijų sistema (IFS); čia ω_i (i= 1, 2, …, N) yra suspaudžiančiosios afiniosios transformacijos. Parenkame (pradinę) uždarąją aibę $A_0 \in H(R^2)$ ir nuosekliai formuojame aibių seką $\{A_n\}$, būtent:

$A_n=W^{0n}(A_0)=W(A_{n-1})=\bigcup_{i=1}^{N}\omega_i(A_{n-1})$, su visais n= 1, 2, … Įrodoma, jog tai (Koši) seka, kuri konverguoja į IFS atraktorių A. Kai n yra pakankamai didelis skaičius, aibė A_n vizualiai yra „artima“ (metrikos h prasme, tuo pačiu, ir vizualiai) aibei (fraktalui, IFS atraktoriui) A.

Pastebėsime, jog visiškai nesvarbu, kokia yra pradinė uždaroji aibė A_0. Iteracinės procedūros rezultatas yra vienas ir tas pats — IFS atraktorius A. Kitaip tarus, aibė A pilnai nusakoma afiniųjų transformacijų ω_i , (i= 1, 2, …, N), veikiančių fraktalinėje erdvėje $(H(R^2), h)$, išraiškomis [8].

2.5.2. Atsitiktinių iteracijų algoritmas

Vėlgi, tarkime, kad $\{R^2; \omega_1, \omega_2, \ldots, \omega_N\}$ yra iteruotųjų funkcijų sistema. Kiekvienai suspaudžiančiajai afiniajai transformacijai ω_i (i= 1, 2, …, N) yra priskiriamas teigiamas skaičius (tikimybė) p_i taip, kad $p_1+ p_2+ \ldots + p_N=1$. Jeigu $\omega_i=\omega_i(x)=\omega_i(x_1,x_2)=(a_ix_1+b_ix_2+e_i,\ c_ix_1+d_ix_2+f_i)$ (i=1,2,…, N), tai apytikslės tikimybių p_i reikšmės, paprastai, randamos iš formulės

$$p_i \cong \frac{|\, a_i d_i - b_i c_i \,|}{\sum_{j=1}^{N} |\, a_j d_j - b_j c_j \,|};$$

jeigu su kuria nors reikšme i, $a_id_i-b_ic_i$= 0, tai p_i prilyginama pakankamai mažam teigiamam skaičiui (tarkim, p_i=0,001).

Parenkamas pradinis taškas x_0 ($x_0 \in R^2$). Konstruojama erdvės R^2 taškų seka $\{x_n\}_{n=0}^{\infty}$; čia $x_n \in \{\omega_1(x_{n-1}), \omega_2(x_{n-1}), \ldots, \omega_N(x_{n-1})\}$, su visais $n= 1, 2, \ldots$, ir įvykio, jog x_n įgys reikšmę $\omega_i(x_{n-1})$, tikimybė yra lygi p_i, t. y. $P\{x_n= \omega_i(x_{n-1})\}= p_i$, $i \in \{1, 2, \ldots, N\}$.

Šis atsitiktinių iteracijų algoritmas yra gana greitas, lengvai realizuojamas ir efektyvus. Atsitiktinių iteracijų algoritmo „veikimas“ yra grindžiamas fraktalo (IFS atraktoriaus) taškams būdinga chaotiška judesio dinamika, [1].

2.5.2. PL-algoritmas ir jo ribotumai

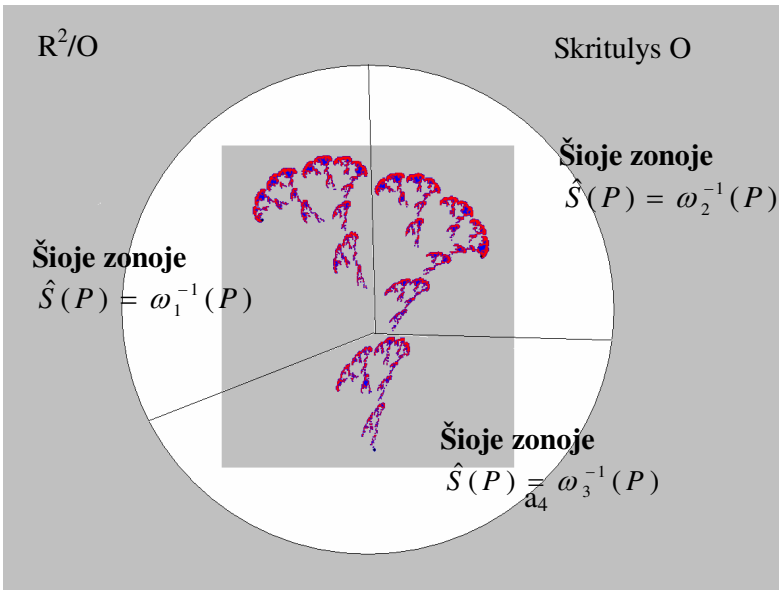
PL(pabėgimo laiko)-algoritmas yra pakankamai universalus. Iš pagrindinių šio algoritmo panaudojimo sričių galima paminėti šias: netiesinių dinaminių sistemų, veikiančių kompleksinėje plokštumoje, analizė, parametrizuotų dinaminių sistemų tyrimas, kompleksinių daugianarių šaknų pritraukimo baseinų vizualizavimas ir pan. Geometrinių fraktalų (IFS atraktorių) sintezei šis algoritmas iki šiol nebuvo naudojamas, nors tokia galimybė, kaip teorinis rezultatas, yra žinoma. Praktinė algoritmo realizacija vis dar kelia tam tikrų problemų.

Pateikiame pačią PL-algoritmo, skirto geometriniams fraktalams (IFS atraktoriams) generuoti, idėją. Tarkime, kad IFS$\{R^2; \omega_1, \omega_2, \ldots, \omega_N\}$ atraktorius yra aibė A (beje, $A=\omega_1(A)\cup\omega_2(A)\cup \ldots\cup\omega_N(A)$). Tarkime, taipogi, jog visos afiniosios transformacijos $\omega_i(i=1,2,\ldots,N)$ yra apgręžiamosios. Šiai IFS konstruojama dinaminė poslinkių sistema $\{A; S\}$, kur poslinkio transformacija S: $A \to A$ apibrėžiama (kiekvienam taškui $a \in A$) taip: $S(a) = \omega_i^{-1}(a)$, jeigu $a \in \omega_i(A)$ ($i \in \{1, 2, \ldots, N\}$) ir $a \notin \bigcup_{j=1(j\neq i)}^{N} \omega_j(A)$ (sakoma, jog taškas a yra transformacijos ω_i^{-1} veikimo zonoje); $S(a)= \omega_t^{-1}(a)$, jeigu $a \in \bigcap_{s=1}^{r} \omega_{i_s}(A)$; čia $t \in \{i_1, i_2, \ldots, i_r\} \subset \{1, 2, \ldots, N\}$ (sakoma, jog taškas a yra bet kurios iš transformacijų ω_t^{-1} ($t \in \{i_1, i_2, ..., i_r\}$) veikimo zonoje).

Duotąjį IFS atitinkanti dinaminė poslinkių sistema $\{A; S\}$ „praplečiama“ į visą erdvę R^2, t.y. konstruojama nauja dinaminė sistema $\{R^2; \hat{S}\}$. Naujoji poslinkio transformacija $\hat{S}$: $R^2 \to R^2$ apibrėžiama lygybe $\hat{S}(x)= \omega_i^{-1}(x)$, kai taškas $x=(x_1, x_2) \in R^2$ patenka į i-tosios afiniosios transformacijos ω_i^{-1} ($i \in \{1, 2, \ldots, N\}$) veikimo zoną. Aišku, kad $\hat{S}$ sutampa su S, kai taškas $x \in A$.

Toliau, tarkime, kad sintezuojamas (ieškomas) IFS atraktorius A patenka į stačiakampį B, t. y. $A \subset B \subset R^2$. Kiekvienam stačiakampio B taškui x konstruojama orbita $\{\hat{S}^{0n}(x)\}_{n=1}^{\Im}$; čia: $\hat{S}^{0n}(x) = \hat{S}(\hat{S}^{0n-1}(x))$, $n=1,2,...$; $\Im$ - iš anksto apibrėžtas iteracijų skaičius. Fiksuokime skritulį su centru

stačiakampio įstrižainių susikirtimo taške (pažymėkime O) ir spinduliu R ; skritulys talpina savyje stačiakampį B (2.6 pav.).



2.6 pav. PL-algoritmo interpretacija;
stačiakampyje pavaizduotas IFS{R^2; ω_1, ω_2, ω_3} atraktorius;
tiesėmis atskirtos afiniųjų transformacijų veikimo zonos

Jeigu euklidinis atstumas $d(\hat{S}^{0\Im}(x), O) \le R$, t. y. po $\Im$ iteracijų taško $x \in B$ orbita nepalieka skritulio, daroma išvada, jog taškas x priklauso IFS atraktoriui A ($x \in A$); priešingu atveju ($d(\hat{S}^{0\Im}(x), O) > R$), $x \notin A$.

PL-algoritmo „veikimas“ remiasi tuo, jog IFS atraktorius (aibė A) yra transformacijos $\hat{S}$: $H(R^2) \to H(R^2)$ atstumiantysis nejudamasis taškas. Kitaip tarus, taškų, esančių arčiau atraktoriaus A, orbitos ilgiau „užsibūna“ skritulyje, negu orbitos taškų, labiau nutolusių nuo atraktoriaus.

Tarkime, IFS { R^2; ω_1, ω_2, ... , ω_n} atraktorius yra aibė $A \in H(R^2)$. IFS nusakančių afiniųjų transformacijų $\omega_i (i \in \{1,2,...,N\})$ suspaudimo taške $P \in R^2$ (kryptimi $\overrightarrow{PM_i}$; M_i – nejudamasis transformacijos ω_i taškas) koeficientai $s_i(P)$ apibrėžiami formule

$$s_i(P) = \frac{d(\omega_i(P), M_i))}{d(P, M_i)}.$$

Nagrinėkime {R^2; $\hat{S}$} - dinaminės poslinkių sistemos {A;S} plėtinį į erdvę R^2. Transformacija $\hat{S}$ perveda bet kokį tašką $P \in B \subset R^2$ į naują tašką P_i, t.y. $P_i = \hat{S}(P) = \omega_i^{-1}(P)$, $i \in \{1,2,...,N\}$. Teoriškai, transformacijų ω_i^{-1} $(i \in \{1,2,..., N\})$ veikimo zonų atskyrimui galime naudoti kriterijų:

$$\min_{i \in \{1,2,...,N\}} \left\{ \frac{d(P, \omega_i(A))}{s_i(P)} \right\} = \frac{d(P, \omega_{i^\circ}(A))}{s_{i^\circ}(P)}$$

(indeksas i° - rodo, kokios transformacijos veikimo zonai priklauso taškas *P*). Deja, praktiškai šiuo kriterijumi pasinaudoti negalime, nes nežinome nei atraktoriaus *A*, nei jo kopijų $\omega_i(A)$, $i=1,2,\ldots,N$.

Pabandysime formalizuoti PL-algoritmą. Iteruotųjų funkcijų sistemai IFS$\{R^2;\omega_1,\omega_2, \ldots ,\omega_n\}$, konstruokime sutvarkytų indeksų rinkinių aibę $I(\Im)=\{(i_1,i_2,\ldots,i_\Im) | i_k \in \{1,2,\ldots,N\},\ k=1,2,\ldots,\ \Im\}$. Visų galimų aibės $I(\Im)$ elementų skaičius yra $N^\Im$. Konkrečiam indeksų rinkiniui $(i_1,i_2,\ldots,i_\Im)$ ir bet kokiam taškui $P\in B\subset R^2$ galima sudaryti $\Im$ taškų orbitą, būtent $\{P_{i_1}, P_{i_1,i_2},\ldots, P_{i_1,i_2,\ldots,i_\Im}\}$, kur $P_{i_1,i_2,\ldots,i_k} = (\omega_{i_k}^{-1} \circ \omega_{i_2}^{-1} \circ \ldots \circ \omega_{i_1}^{-1})(P)$, su visais $k\in\{1,2,\ldots \Im\}$.

Teisingai parinkę skritulio spindulį *R* ir iteracijų skaičių $\Im$, galime teigti, kad taškas $P(P\in R^2)$ priklauso atraktoriui tada ir tik tada, kai

$$\min\{\ d(P_{i_1,i_2,\ldots,i_\Im}, A) \mid (i_1,i_2,\ldots,i_\Im) \in I(\Im)\} = d(P_{i_1^\circ,i_2^\circ,\ldots,i_\Im^\circ}, A) \le R\ ; \qquad (1.1)$$

priešingu atveju $(d(P_{i_1^\circ,i_2^\circ,\ldots,i_\Im^\circ}, A) > R)$, $P\notin A$. Akivaizdu, kad $\left(i_1^\circ,i_2^\circ,\ldots,i_\Im^\circ\right)\in I(\Im)$ nurodo atvejį, kada afiniųjų transformacijų veikimo zonos tiksliai fiksuojamos visiems orbitos $\{\ \hat{S}^{0n}(P)\}_{n=1}^{\Im}\}$ taškams. Kadangi dim(*A*)<<*R,* tai (1.1) sąlyga gali būti pertvarkyta taip

$$\left(P \in A\right) \Leftrightarrow (\min\{\ d(P_{i_1,i_2,\ldots,i_\Im}, O) \mid (i_1,i_2,\ldots,i_\Im) \in I(\Im)\} = d(P_{i_1^\circ,i_2^\circ,\ldots,i_\Im^\circ}, O) \le R)\ ; \qquad (1.2)$$

čia *O* yra skritulio centras (stačiakampio *B* istržainių susikirtimo taškas). Tačiau ir šios (1.2) sąlygos panaudojimas praktikoje yra problematiškas, nes $I(\Im)$ galia auga labai greitai, didėjant $\Im$.

3. Poslinkių dinamikos panaudojimo fraktalinių vaizdų sintezės procedūrose efektyvumo tyrimas

3.1. IFS sudarančių afiniųjų transformacijų veikimo zonų atskyrimas

Šiame skyrelyje yra pateikiamas naujas kriterijus atvirkštinių afiniųjų transformacijų, sudarančių IFS$\{R^2;\ \omega_1,\ \omega_2,\ \ldots,\ \omega_N\}$, veikimo zonoms atskirti (Euklido erdvėje (R^2,d)). Sudarytasis kriterijus formuluoja sąlygas praktiniam PL-algoritmo įgyvendinimui.

3.1.1. Kriterijaus parinkimas

Siūlomo kriterijaus esmė tokia - taškas $P_{i_1^\circ, i_2^\circ, \ldots, i_k^\circ}$ $\left(k \in \{1,2,\ldots,\tau-1\}\right)$, gautas skaičiuojant taško $P \in B$ orbitą $\{\hat{S}^{0n}(P)\}_{n=1}^{\Im}$, priklauso afiniosios transformacijos $\omega_{i_{k+1}^\circ}^{-1}$ ($i_{k+1}^\circ \in \{1,2,\ldots,N\}$) veikimo zonai tada ir tik tada, kai:

$$\min_{i_{k+1},\ldots,i_{k+\tau} \in \{1,2,\ldots,N\}} \{d(P_{i_1^\circ,\ldots,i_k^\circ,i_{k+1},\ldots,i_{k+\tau}}, P_{i_1^\circ,\ldots,i_k^\circ})\} = \min_{i_{k+2},\ldots,i_{k+\tau} \in \{1,2,\ldots,N\}} \{d(P_{i_1^\circ,\ldots,i_{k+1}^\circ,i_{k+2},\ldots,i_{k+\tau}}, P_{i_1^\circ,\ldots,i_k^\circ})\} \qquad (3.1)$$

($\tau \geq 1$). Kitaip tarus, tam kad nustatyti, kokiai veikimo zonai priklauso taškas $P_{i_1^\circ, i_2^\circ, \ldots, i_k^\circ}$, reikia remtis „τ žingsnių į priekį“ strategija (kriterijumi). Teorinis šio kriterijaus aiškinimas grindžiamas tokiais teiginiais:

a) Jei taškas $P(P \in B \subset \mathrm{R}^2)$ yra transformacijos ω_i^{-1} ($i \in \{1,2,\ldots,N\}$) veikimo zonoje, tada

$d(\omega_i^{-1}(P), A) \leq d(\omega_j^{-1}(P), A)$, su visais $j \in \{1,2,\ldots,N\}$, $j \neq i$;

b) Bet kokiam taškui $P \in \mathrm{R}^2$ ir jo vaizdui $P_i = \omega_i^{-1}(P)$ ($i \in \{1,2,\ldots,N\}$) teisinga nelygybė:

$d(P_i, P) \geq \dfrac{d(P,A)\cdot(1-s(P))}{s(P)} \geq \dfrac{d(P,A)\cdot(1-s)}{s}$; čia: $s(P)=\max\{s_j(P) | j=1,2,\ldots,N\}$; s - IFS suspaudimo koeficientas. Ši nelygybė reiškia, kad atstumas tarp dviejų taškų P ir P_i didėja, kai P tolsta nuo atraktoriaus A;

c) Teisinga lygybė

$$d(P_{i_1^\circ,\ldots,i_\Im^\circ}, A) = \min_{i_1^\circ,\ldots,i_k^\circ,i_{k+1},\ldots,i_\Im \in I(\Im)} \{d(P_{i_1^\circ,\ldots,i_k^\circ,i_{k+1},\ldots,i_\Im}, A)\}$$

su visais k ($k \in \{1,2,\ldots,\Im-1\}$).

Koks žingsnių skaičius τ reikalingas, kad kiekviename orbitos taške $P_{i_1, i_2, \ldots, i_k}$ ($k \in \{1,2,\ldots,\Im\}$) teisingai būtų fiksuojama afiniųjų transformacijų veikimo zonos? Kaip žingsnių skaičius priklauso nuo IFS sudarančių afiniųjų transformacijų skaičiaus bei jas charakterizuojančius parametrų? Siekiant gauti atsakymus į šiuos klausimus, buvo atliktas teorinis ir eksperimentinis tyrimas.

3.1.2. Zonų nustatymas

Žinant IFS sudarančių afiniųjų transformacijų skaičių N bei parinkus fiksuotą parametro τ reikšmę „žingsnių pirmyn“ skaičius, pirmiausia konstruojamos visos galimos afiniųjų transformacijų superpozicijos, tiksliau, afiniųjų transformacijų eilės numerių (indeksų) perstatiniai su pasikartojimais (kombinacijos) $<k_0, k_1, \ldots, k_\tau>$. Bendras kombinacijų skaičius yra N^τ; i-tają indeksų

kombinaciją pažymėkime $S_i (i \in \{0,1,....,N^\tau - 1\})$, t.y $S_i = < k_0 k_1 ... k_{\tau-1} >$, $k_j \in \{0,1,...,N-1\}$, $j = 0,1,...,\tau - 1$.

Kombinacijoms S_i generuoti siūloma taikyti naujai sudarytą procedūra, kurios aprašą pateikiame žemiau. Įveskime parametrą L_j ($j \in \{0,1,...,\tau\}$). Norėdami sugeneruoti *i*–tąją kombinaciją (perstatinį), parametrui L_τ priskiriame reikšmę *i*. Tada: $L_j=[L_{j+1}/N]$, kai $L_{j+1} \geq N$, $L_j=0$, kai $L_{j+1}<N$; $k_j=\{L_{j+1}/N\}$, kai $L_{j+1} \neq 0$, $k_j=0$, kai $L_{j+1}=0$, su visais $j= 0,1,...,\tau$-1.

Toliau, kiekvienai sugeneruotai indeksų kombinacijai S_i (i=0,1,...,N^τ-1) apskaičiuojama afiniųjų transformacijų superpozicija $(\omega^{-1}_{k_{\tau-1}+1} \circ \omega^{-1}_{k_{\tau-2}+1} \circ ... \circ \omega^{-1}_{k_0+1})$. Turėdami visas afiniųjų transformacijų superpozicijas, susijusias su IFS ir pasirinktuoju „žingsnių pirmyn“ skaičiumi τ, kiekviename taško $P \in B \subset R^2$ orbitos vietoje galime operatyviai ir tiksliai (remdamiesi 3.1 kriterijumi) nustatyti afiniosios transformacijos veikimo zoną.

Pastebėsime, jog įvestas kriterijus ((3.1) išraiška) veikia su fiksuotu žingsnių skaičiumi τ, kuris skirtingoms IFS yra skirtingas. Taigi, susiduriama su „reikiamo“ žingsnių skaičiaus τ nustatymo problema. Teorinis šios problemos sprendimas nėra žinomas. Žemiau siūloma afiniųjų transformacijų veikimo zonų atskyrimo kriterijaus modifikacija, kai žingsnių skaičius τ parenkamas adaptyviai, priklausomai nuo analizuojamos IFS. Modifikacijos esmė – vertinamos konkretaus orbitos taško $P_{i_1,...,i_\tau}$ atstumas ne nuo pradinio taško $P \in B$, bet nuo IFS atraktoriaus *A*. Pastarojo atstumo įvertinimui naudojami IFS sudarančių afiniųjų transformacijų nejudamieji taškai $M_1,...,M_N$. Maksimalų atstumą tarp nejudamųjų taškų pažymėję *L* ($L=\max\{d(M_i,M_j) | i \neq j\}$), galime teigti, jog $D<m \cdot L$ kai *D* – IFS atraktoriaus diametras, o $m \in [1;3] \subset R$. Toliau, įvedame kriterijaus galios sampratą – kriterijaus galia *G* apskaičiuojama, remiantis išraiška:

$$G = \min(G_k), \text{ kur } G_k = \left| \min_{i \in \{0,1,...,N^\tau - 1\}} \{d(P_{j,i_2,...,i_\tau}, P\} - \min_{i \in \{0,1,...,N^\tau - 1\}} \{d(P_{k,i_2,...,i_\tau}, P\} \right|, \ (j \neq k; \quad k \in \{1,2,...,N\}).$$

Jeigu $G < m \cdot L$, sakysime, kad kriterijaus galia yra maža ir pasirinktas „žingsnių į priekį“ (taške *P*) skaičius τ yra nepakankamas; padidiname (taške *P*) iteracijų skaičių τ ($\tau := \tau + 1$), apskaičiuojame S_i ir naują kriterijaus galios reikšmę *G*, šį žingsnį kartojame tol, kol $G > m \cdot L$. Pastebėsime, jog adaptyviam afiniųjų transformacijų veikimo zonų nustatymui kriterijaus galia *G* fiksuojama kiekviename stačiakampio $B \subset R^2$ taške.

Kaip parodė vėlesni eksperimentiniai tyrimai, pateiktoji afiniųjų transformacijų (sudarančių IFS) veikimo zonų atskyrimo kriterijaus modifikacija netik paspartino sintezės procesą, bet ir pagerino sintezuojamo fraktalinio vaizdo (IFS atraktoriaus) kokybę.

3.2. Sintezuojamo fraktalinio vaizdo tolygus spalvinis užpildymas

Nagrinėkime IFS { R^2; ω_1, ω_2, ... , ω_N}, kurios suspaudimo koeficientas s ($s = \max\{s_1, s_2, \ldots, s_N\}$). Tegu $M_1, M_2, \ldots, M_N$ yra atitinkamai afiniųjų transformacijų ω_1, ω_2, ... ,ω_N nejudamieji taškai. Veikiant afiniąjai transformacijai ω_i ($i \in \{1,2,\ldots,N\}$), taško P, nutolusio nuo taško M_i atstumu δ (δ-mažas teigiamas skaičius), orbita išeina už skritulio (su spinduliu R ir centru taške M_i) ribų tada ir tik tada kai iteracijų skaičius (pažymėkime $\Im_i$) tenkina sąlygą $\Im_i \geq \log(R/\delta)/\log(1/s_i)$. Be to, jei $s_i > s_j$ ($i,j \in \{1,2,\ldots,N\}$) , tai taškai iš M_j aplinkos palieka skritulį greičiau nei taškai iš M_i aplinkos. Dėl to, taikant vienodą iteracijų skaičių (sintezės metu) visoms IFS sudarančioms transformacijoms, gauname netolygiai užpildytą vaizdą (IFS atraktorių). Akivaizdu, jog reikia įvesti iteracijų skaičiaus korekciją, atsižvelgiant į tai, kurios afiniosios transformacijos veikimo zonoje konkrečiu momentu yra vienas ar kitas orbitos taškas.

Pradinis iteracijų skaičius apdorojamai IFS apskaičiuojamas, remiantis nelygybe

$$\Im_{max}(P) \geq -\log(R/\delta)/\log(s_{i_1^0}); \qquad (4.1)$$

čia R yra skritulio, talpinančio stačiakampį B, spindulys (2.5.1 skyrelis; 2.6 pav.). Jeigu $P_{i_1^0,i_2^0,\ldots,i_k^0}$ yra k–tasis taško $P \in B \subset R^2$ orbitos taškas, $\Im_{left}$ - likęs iteracijų skaičius (pradiniu momentu $\Im_{left}=\Im_{max}(P)$) ir $P_{i_1^0,i_2^0,\ldots,i_k^0}$ patenka į i-tosios ($i \in \{1,2,\ldots,N\}$) atvirkštinės afiniosios transformacijos veikimo zoną, tai $\Im_{left} = \Im_{left} + k$, kur k reikšmė imama iš 3.1 lentelės. Pastebėsime, jog duomenys pateikti 3.1 lentelėje, gauti empyriniu būdu, atlikus gana daug eksperimentų su įvairaus tipo IFS.

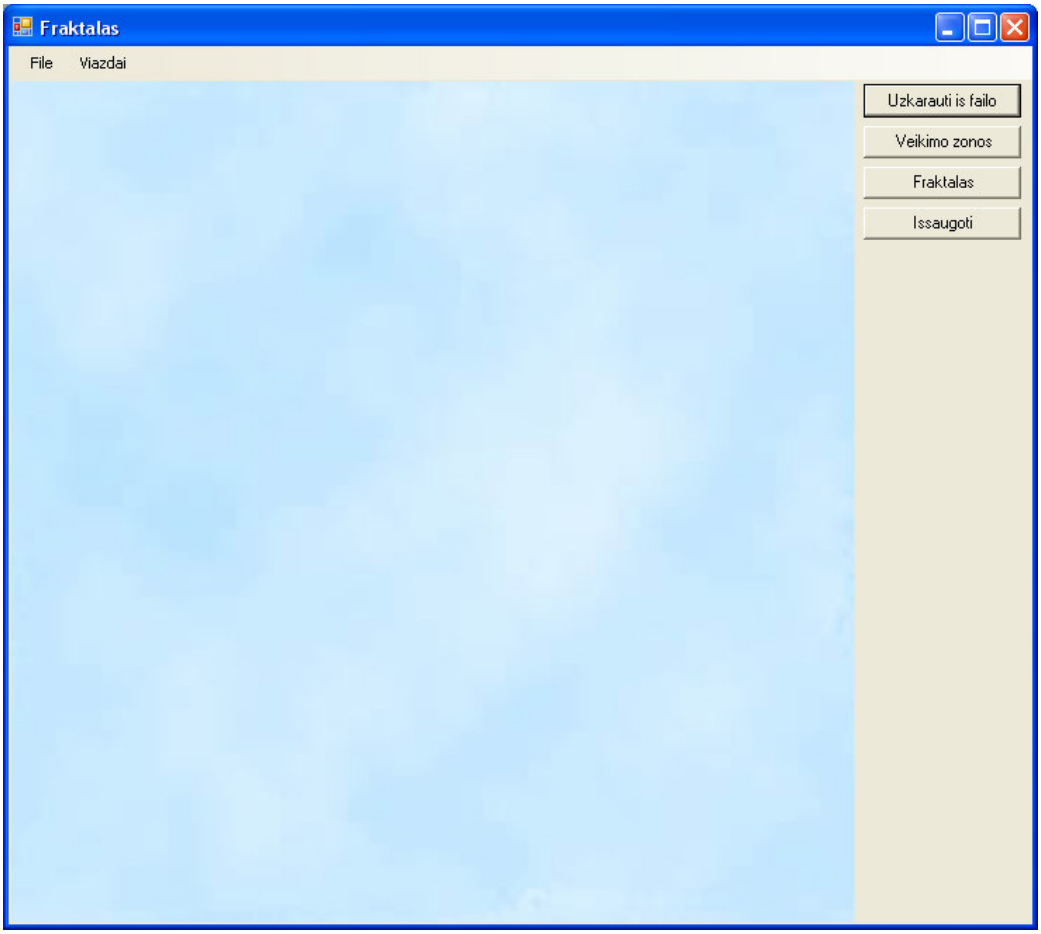
Afiniosios transformacijos ω_i suspaudimo koeficientas, s_i	Iteracijų skaičiaus korekcija, k
0 – 0.1	-1.4
0.1 – 0.2	-1.1
0.3 – 0.4	-0.8
0.4 – 0.5	-0.4
0.5 – 0.6	0.6
0.7 – 0.8	0.8
0.9 – 1	0.9

3.1 lentelė. Iteracijų skaičiaus korekcijų reikšmės

4. Eksperimentinis tyrimas

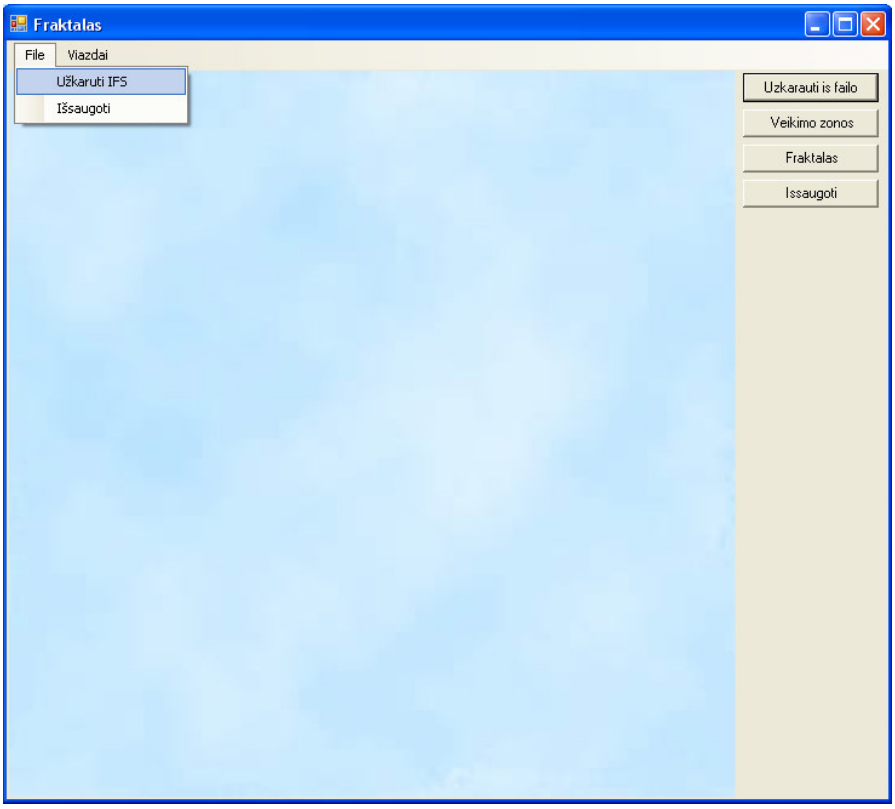
4.1 Trumpas programinio įrankio, su kuriuo bus atliekami ekspermentai, aprasymas.

Paleidus IFS atraktoriams sintezuoti skirtą programą, ekrane bus matomas toks vaizdas (langas; 4.1 pav.)



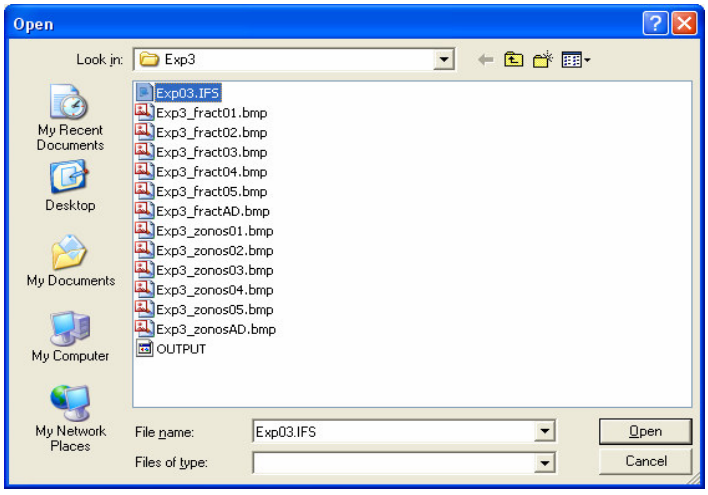
4.1 pav. Pagrindinis langas

Norint pradėti darbą, reikia turėti IFS parametrus, išsaugotus tekstiniame, faile. Galime pasinaudoti paketo Fractal Vision teikiamomis galimybėmis, t.y. galime norimą IFS ir ją išsaugoti faile (4.2 pav).



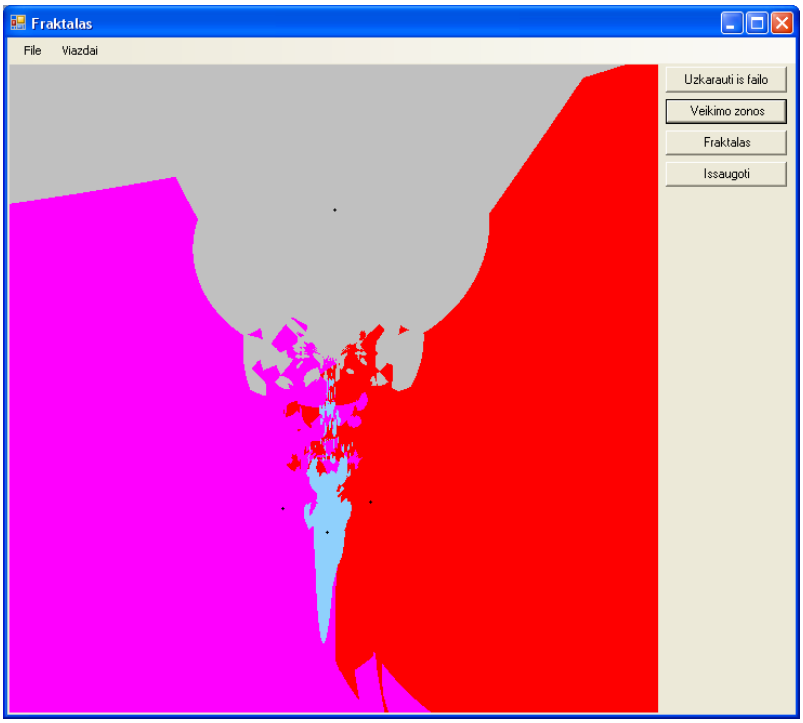
4.2 pav. Pasirinkti IFS iš failo

Pasirinkime sukurtą IFS parametrų failą.

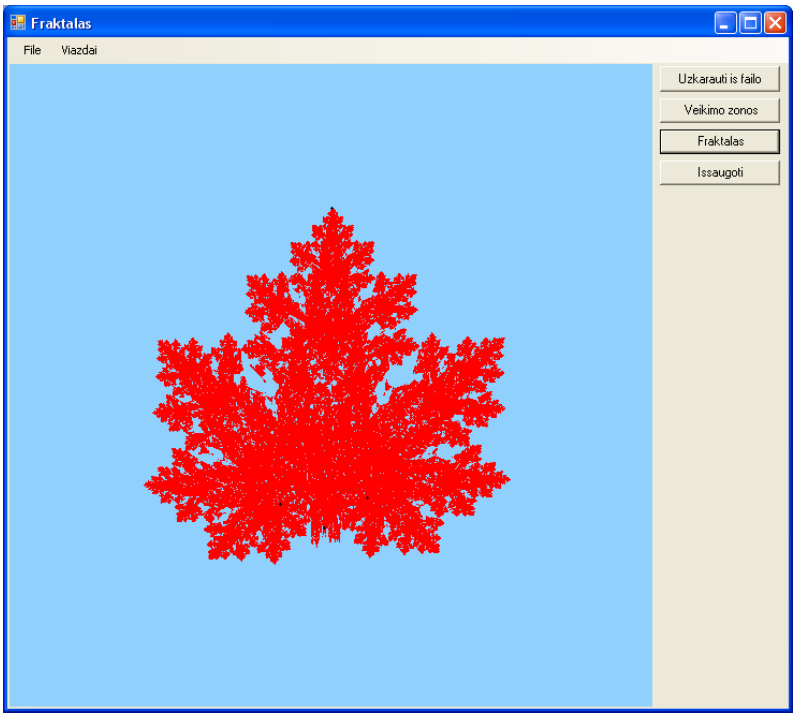


4.3 pav. Atidaryti pasirinktą IFS failą

Atsidarius failą, galimi du tęsiniai – „veikimo zonos“ arba „fraktalas“. Atitinkamai gausime rezultatą – afiniųjų transformacijų veikimo zonas (4.4 pav.) arba IFS atraktorių (4.5 pav.).



4.4 pav. Afiniųjų transformacijų veikimo zonos



4.5 pav. IFS atraktorius

4.2. Afiniųjų transformacijų veikimo zonų atskyrimo kriterijaus tyrimas

Pirmoje eksperimentinio tyrimo dalyje įvertinsime modifikuoto afiniųjų transformacijų (sudarančių IFS) veikimo zonų atskyrimo kriterijaus (3.1 išraiška; 3.1.1 skyrelis) panaudojimo tikslingumą bei efektyvumą.

Imkime tris skirtingas iteruotųjų funkcijų sistemas IFS:

a) IFS $\{R^2; \omega_1, \omega_2\}$;

$$\omega_1=\begin{pmatrix}0.12 & -0.61\\ 0.61 & 0.12\end{pmatrix}\begin{pmatrix}x\\ y\end{pmatrix}+\begin{pmatrix}-0.69\\ 0.32\end{pmatrix};\ \omega_2=\begin{pmatrix}0.53 & 0.37\\ -0.37 & 0.53\end{pmatrix}\begin{pmatrix}x\\ y\end{pmatrix}+\begin{pmatrix}0.33\\ 0.72\end{pmatrix}$$

b) IFS $\{R^2; \omega_1, \omega_2\}$;

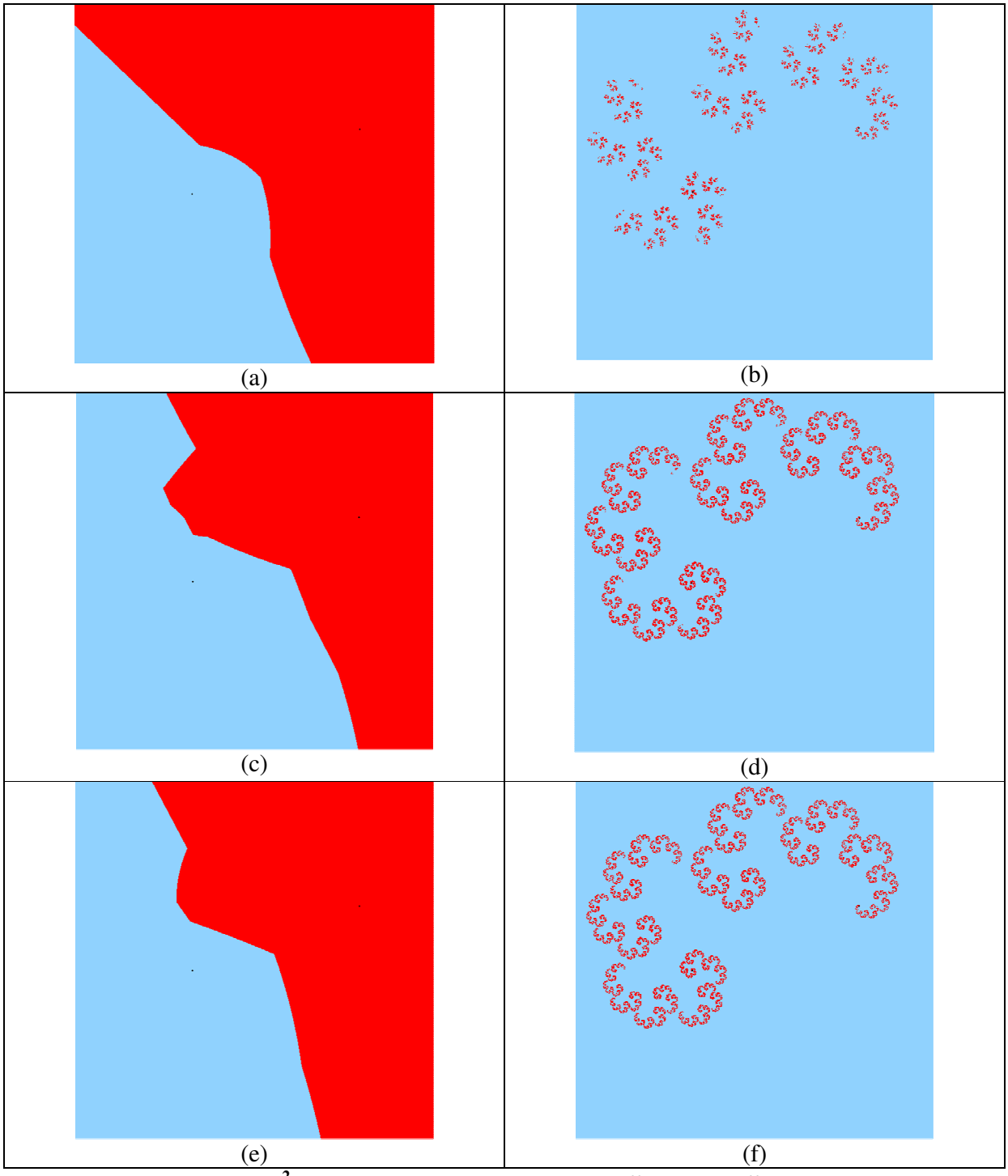
$$\omega_1=\begin{pmatrix}0.5 & 0\\ 0 & 0.5\end{pmatrix}\begin{pmatrix}x\\ y\end{pmatrix}+\begin{pmatrix}1.03\\ 0.91\end{pmatrix};\ \omega_2=\begin{pmatrix}-0.76 & 0.15\\ -0.15 & -0.76\end{pmatrix}\begin{pmatrix}x\\ y\end{pmatrix}+\begin{pmatrix}-0.52\\ -0.16\end{pmatrix}$$

c) IFS $\{R^2; \omega_1, \omega_2, \omega_3, \omega_4\}$;

$$\omega_1=\begin{pmatrix}0.14 & 0.01\\ 0 & 0.51\end{pmatrix}\begin{pmatrix}x\\ y\end{pmatrix}+\begin{pmatrix}-0.08\\ -1.31\end{pmatrix};\ \omega_2=\begin{pmatrix}0.43 & 0.52\\ -0.45 & 0.5\end{pmatrix}\begin{pmatrix}x\\ y\end{pmatrix}+\begin{pmatrix}1.49\\ 0.75\end{pmatrix}$$

$$\omega_3=\begin{pmatrix}0.45 & -0.49\\ 0.47 & 0.47\end{pmatrix}\begin{pmatrix}x\\ y\end{pmatrix}+\begin{pmatrix}-1.62\\ -0.74\end{pmatrix};\ \omega_4=\begin{pmatrix}0.49 & 0\\ 0 & 0.51\end{pmatrix}\begin{pmatrix}x\\ y\end{pmatrix}+\begin{pmatrix}0.02\\ 1.62\end{pmatrix}$$

Šių IFS veikimo zonos bei atraktoriai pateikti 4.1, 4.2, 4.3 paveikslėliuose; detalesnius veikimo zonų vaizdus bei atraktorius galime rasti priede.



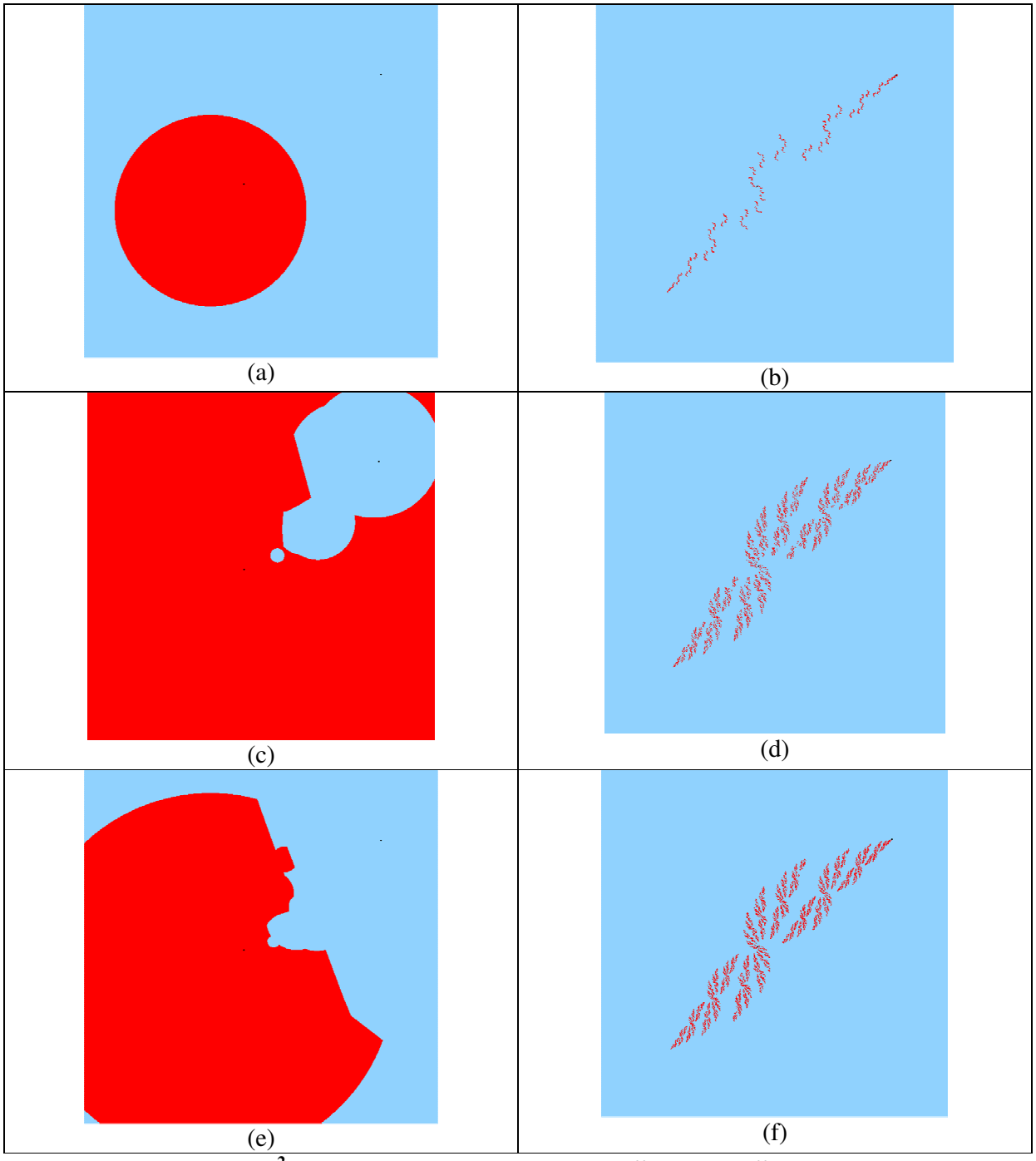
4.6 pav. Pirmosios IFS $\{R^2; \omega_1, \omega_2\}$ atraktorius, gautas atsižvelgiant į išskirtas veikimo zonas:

(a) veikimo zonos, (τ=2); (b) IFS atraktorius, kai τ=2;

(c) veikimo zonos, (τ=3); (d) IFS atraktorius, kai τ=3;

(e) veikimo zonos, naudojant naujai sudarytą adaptyvųjį kriterijų;

(f) IFS atraktorius, kai taikomas adaptyvus kriterijus;



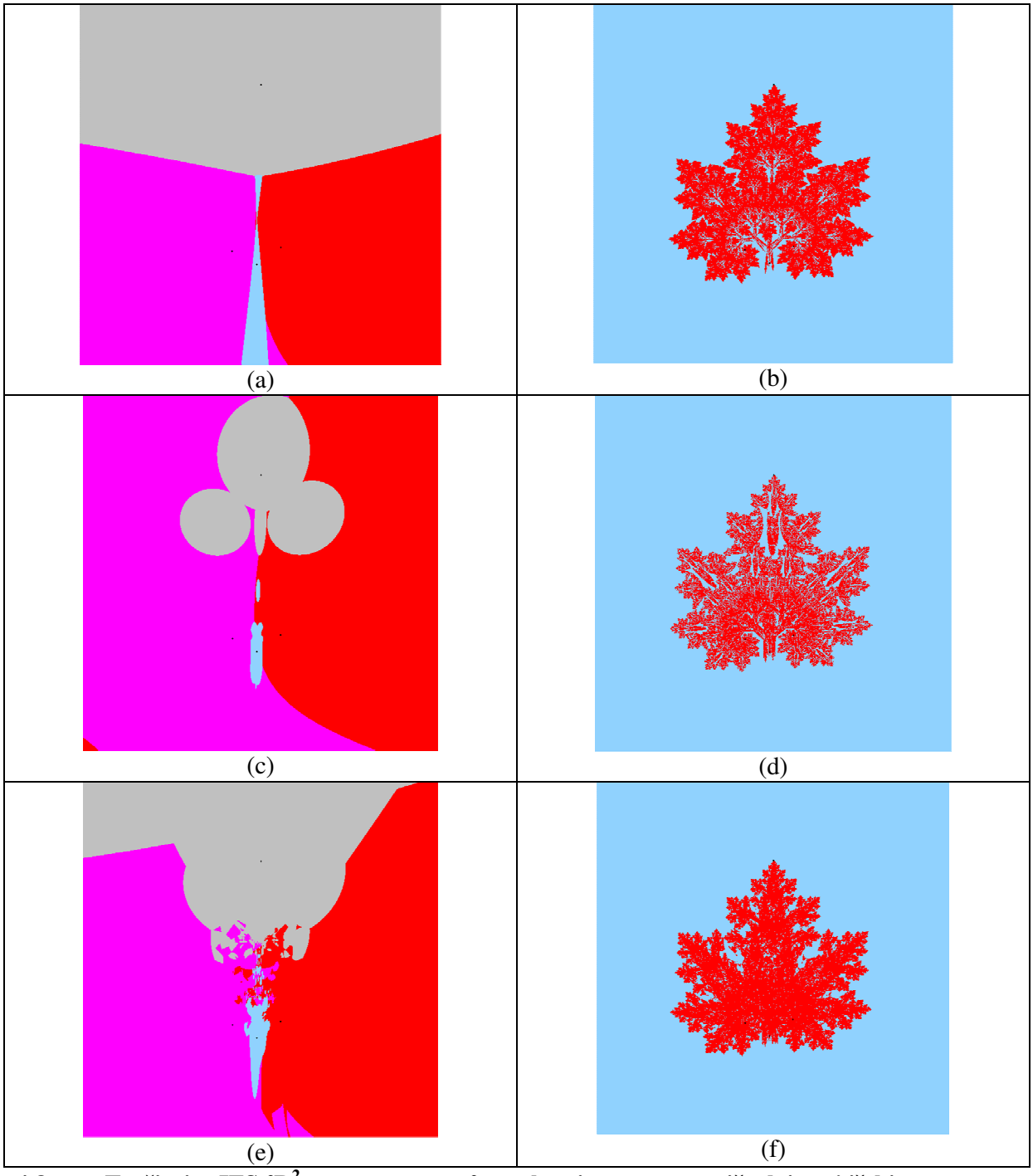
4.7 pav. Antroios IFS $\{R^2; \omega_1, \omega_2\}$ atraktorius, gautas atsižvelgiant į išskirtas veikimo zonas:

(a) veikimo zonos, (τ=1); (b) IFS atraktorius, kai τ=1;

(c) veikimo zonos, (τ=4); (d) IFS atraktorius, kai τ=4;

(e) veikimo zonos, naudojant naujai sudarytą adaptyvųjį kriterijų;

(f) IFS atraktorius, kai taikomas adaptyvus kriterijus;



4.8 pav. Trečiosios IFS $\{R^2; \omega_1, \omega_2, \omega_3, \omega_4\}$ atraktorius, gautas atsižvelgiant į išskirtas veikimo zonas:

(a) veikimo zonos, (τ=1); (b) IFS atraktorius, kai τ=1;

(c) veikimo zonos, (τ=4); (d) IFS atraktorius, kai τ=4;

(e) veikimo zonos, naudojant naujai sudarytą adaptyvųjį kriterijų;

(f) IFS atraktorius, kai taikomas adaptyvus kriterijus;

Kaip matome iš pavysdžių skirtingoms IFS reikia imti skirtingą iteracijų (žingsnių pirmyn) skaičių τ, Pastebėsime, jog fiksavus (pakankamai didelį) iteracijų skaičių (kad tiktų visoms IFS) susidutume su didelėmis laikinėmis sąnaudomis (4.1, 4.2, 4.3 lentelės).

IFS atraktoriaus sintezės procese panaudojus adaptyviąją iteracijų skaičiaus τ nustatymo procedūrą (kriterijaus galią G), afiniųjų transformacijų veikimo zonas pavyko atskirti kur kas efektyviau. Taškuose, kur kriterijaus galia G buvo didelė, iteracijų skaičius buvo parenkamas mažas, o ten, kur galia maža, iteracijų skaičius buvo didinamas.

Laikas (sek.)	**Iteracijų skaičius, τ**
0.266	1
0.500	2
0.968	3
1.750	4
3.375	5
1.172	Kintamas (adaptyvus kriterijus)

4.1 lentelė. Pirmosios IFS $\{R^2; \omega_1, \omega_2\}$ atraktoriaus generavimo laikinės sąnaudos

Laikas (sek.)	**Iteracijų skaičius, τ**
0.265	1
0.516	2
1.148	3
1.935	4
3.398	5
1.188	Kintamas (adaptyvus kriterijus)

4.2 lentelė. Antrosios IFS $\{R^2; \omega_1, \omega_2\}$ atraktoriaus generavimo laikinės sąnaudos

Laikas (sek.)	Iteracijų skaičius, τ
0.532	1
1.843	2
6.828	3
28.735	4
98.390	5
11.266	Kintamas (adaptyvus kriterijus)

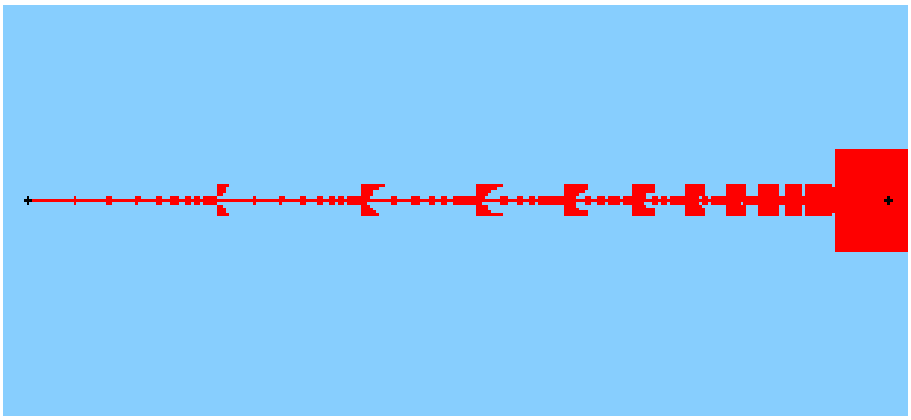
4.3 lentelė. Trečiosios IFS $\{R^2; \omega_1, \omega_2, \omega_3, \omega_4\}$ atraktoriaus generavimo laikinės sąnaudos

4.3. Sintezuojamo fraktalinio vaizdo tolygaus spalvinio užpildymo tyrimas

Neįvedus iteracijų skaičiaus (IFS atraktoriaus sintezės metu) korekcijos, gaunamas netolygiai užpildytas fraktalinis vaizdas (atraktorius).

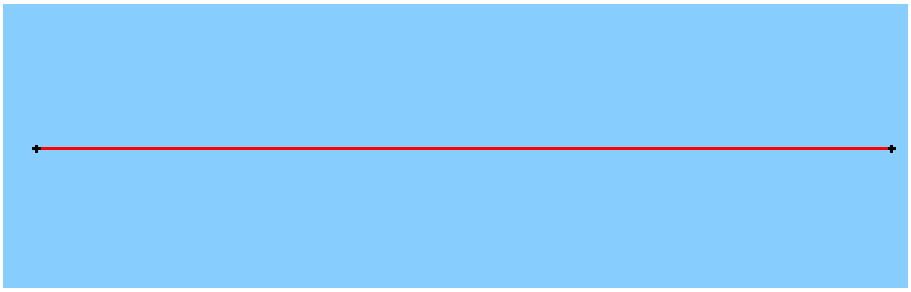
Ištikrųjų imkime IFS $\{R^2; \omega_1, \omega_2\}$:

$\omega_1 = \begin{pmatrix} 0.78 & 0 \\ 0 & 0.78 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} 2.02 \\ 0 \end{pmatrix}$; $\omega_2 = \begin{pmatrix} 0.24 & 0 \\ 0 & 0.24 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \end{pmatrix}$. Šios IFS atraktorius yra žinomas – tiesės atkarpa. Be iteracijų skaiėiaus korekcijos gautas fraktalinis vaizdas (IFS atrakroriaus aproksimacija) pavaizduotas 4.9 pav.



4.9 pav. IFS atraktorius (be iteracijų skaičiaus korekcijos)

Matome, jog tai (toli gražu) nėra tiesės atkarpa. Įvedus korekciją, t.y. atsižvelgiant į tai, kurios transformacijos veikimo zonoje yra vienas ar kitas orbitos taškas gaunamas kur kas tikslesnis IFS atraktorius (4.10 pav.).

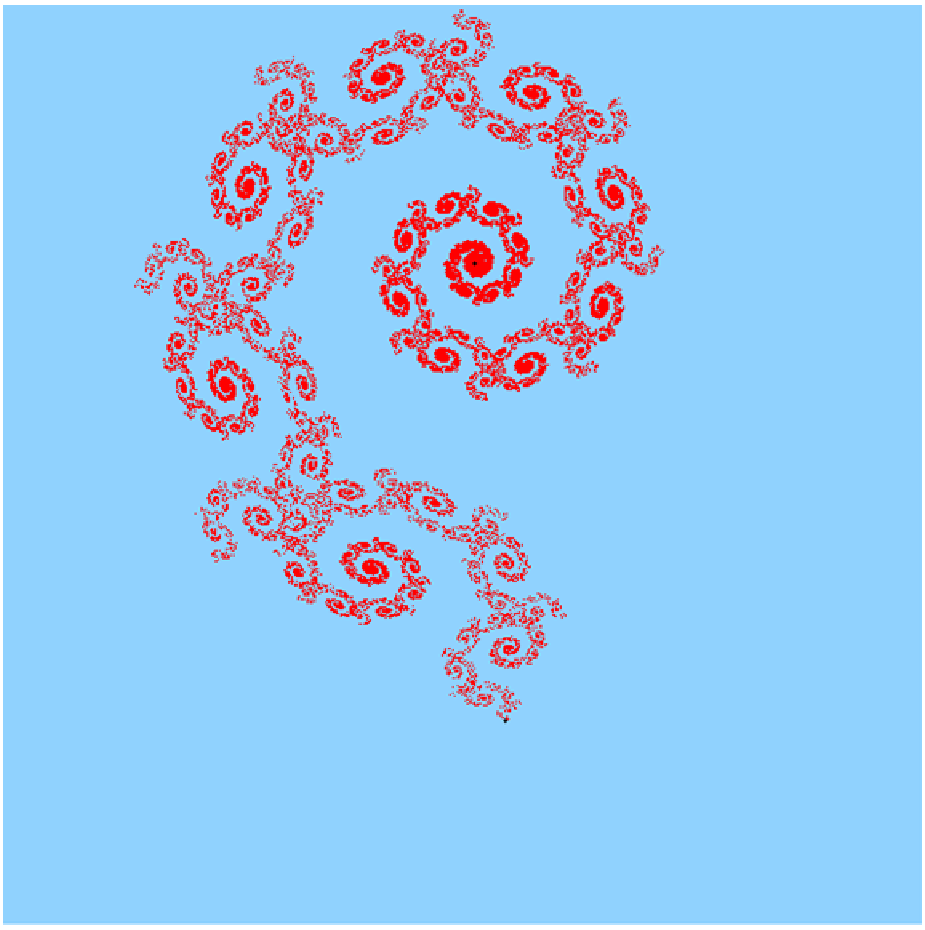


4.10 pav. IFS atraktorius (su iteracijų skaičiaus korekcija)

Imkime kitą pavyzdį – IFS $\{R^2; \omega_1, \omega_2\}$;

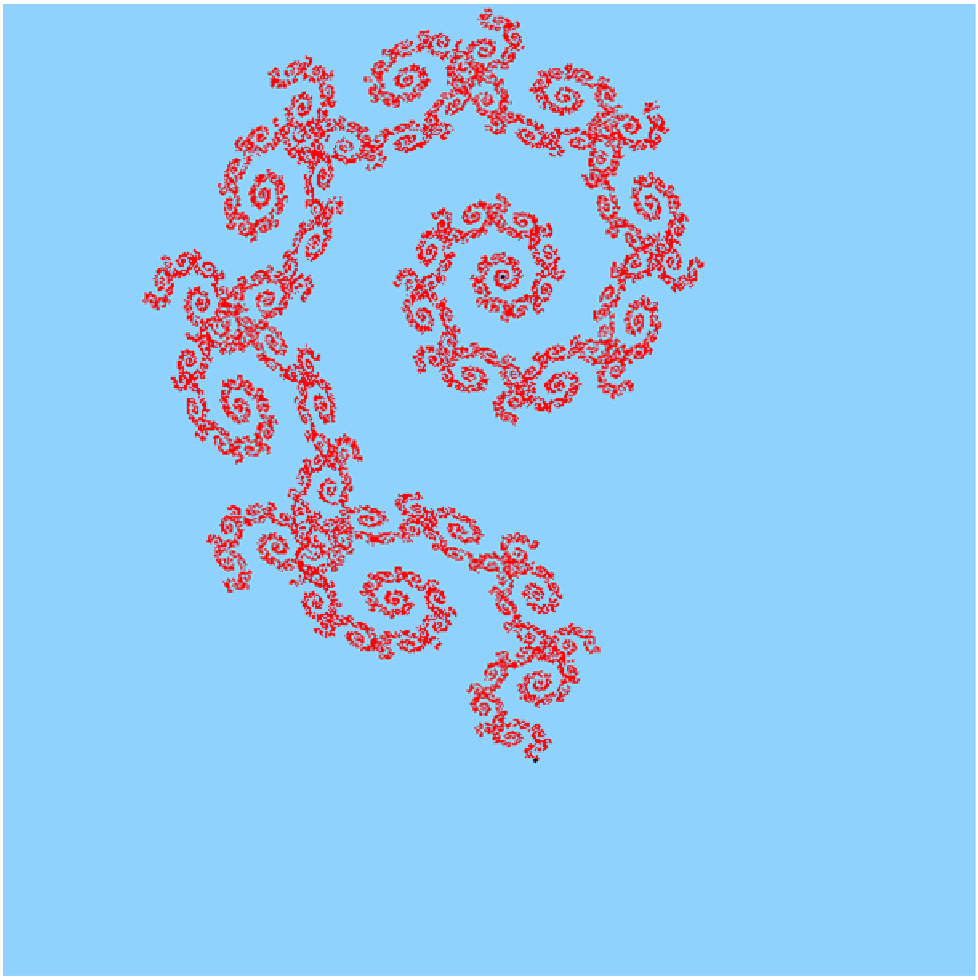
$$\omega_1=\begin{pmatrix} 0.38 & -0.32 \\ -0.18 & 0.32 \end{pmatrix}\begin{pmatrix} x \\ y \end{pmatrix}+\begin{pmatrix} -0.12 \\ -0.96 \end{pmatrix};\ \omega_2=\begin{pmatrix} 0.61 & 0.61 \\ -0.61 & 0.61 \end{pmatrix}\begin{pmatrix} x \\ y \end{pmatrix}+\begin{pmatrix} -0.68 \\ 0.5 \end{pmatrix}.$$

Skirtingai nei ankstesniajame pavyzdyje, neįvedant iteracijų skaičiaus korekcijos, gauname neryškų vaizdą tos afiniosios transformacijos veikimo zonoje kurios suspaudimo koeficientas yra mažesnis.



4.11 pav. Antrosios IFS atraktorius (be iteracijų skaičiaus korekcijos)

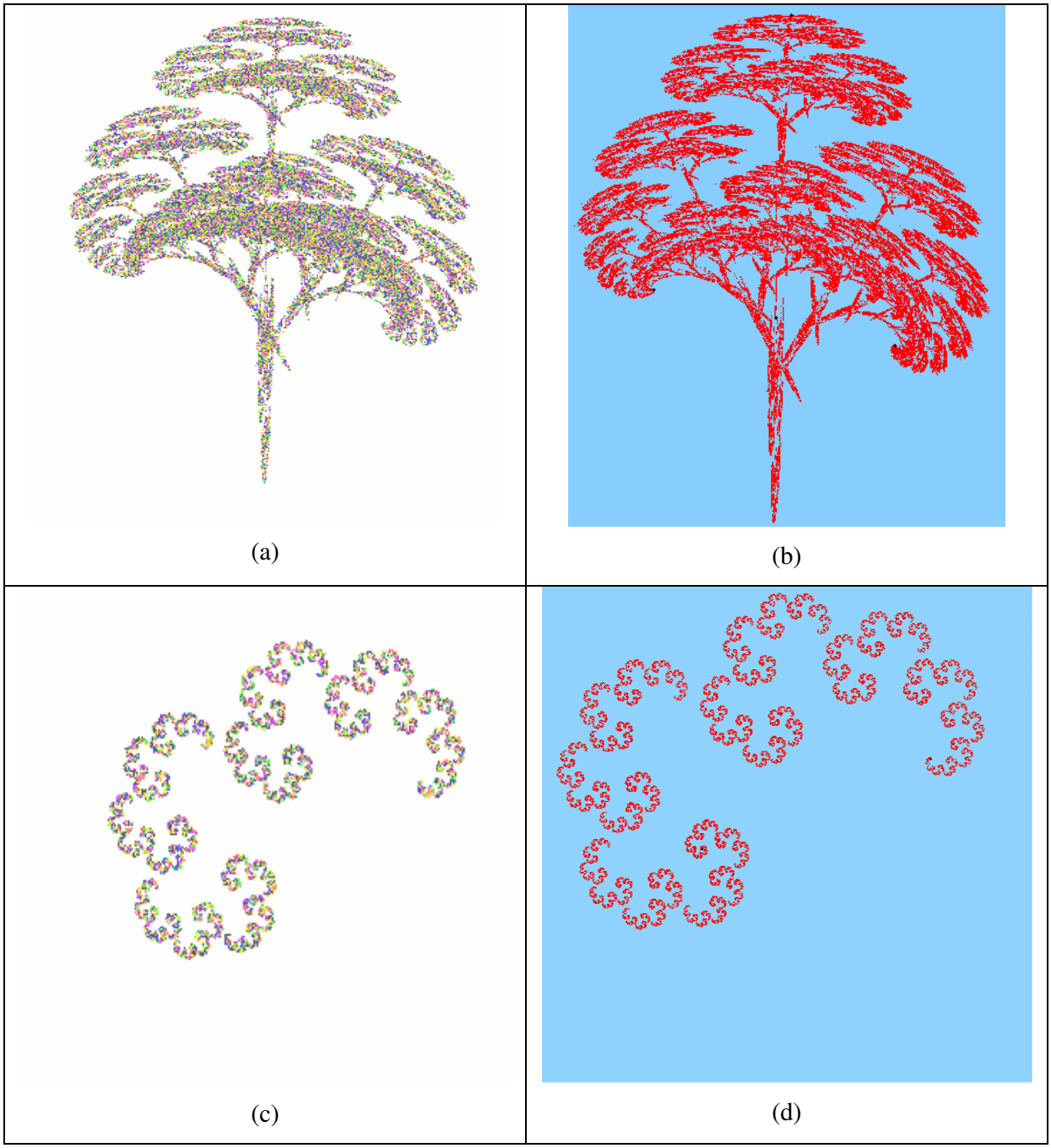
Įvedus iteracijų skaičiaus (sintezės metu) korekciją, netolygaus fraktalionio vaizdo užpildymo problema yra išsprendžiama (4.12 pav).



4.12 pav. Antrosios IFS atraktorius (su iteracijų skaičiaus korekcija)

4.3. Sugeneruotų fraktalinių vaizdų (IFS atraktorių) palyginimas

PL-algoritmas geometrinių fraktalų (IFS atraktorių) sintezei, dar nebuvo pritaikytas, todėl tenka lyginti skirtingų generavimo algoritmų sugeneruotus fraktalinius vaizdus (IFS atraktorius), t.y. tarp PL-algoritmo ir atsitiktinių iteracijų algoritmo, bet kokybiniam įvertinimui reiktų suskaičiuoti gautų IFS atraktorių fraktalines dimensijas ir palyginti jas su teorinėmis fraktalinėmis dimensijomis. Tačiau fraktalinėm dimensijom skaičiuoti reikalingas programinis įrankis, kuris dar nėra sukurtas. Todėl galimas tik vizualinis palyginimas, 4.13 pav. pateikiami keli fraktaliniai vaizdai (IFS atraktoriai) gauti skirtingais generavimo algoritmais.



4.13 pav. Sugeneruotų fraktalinių viazdų (IFS atraktorių) palyginimas.

(a) IFS $\{R^2; \omega_1, \omega_2, \omega_3, \omega_4\}$ atraktorius sugeneruotas atsitiktinių iteracijų algoritmu (Fractal Vision paketu);

(b) IFS $\{R^2; \omega_1, \omega_2, \omega_3, \omega_4\}$ atraktorius sugeneruotas sugeneruotas PL-algoritmu;

(c) IFS $\{R^2; \omega_1, \omega_2,\}$ atraktorius sugeneruotas atsitiktinių iteracijų algoritmu (Fractal Vision paketu);

(d) IFS $\{R^2; \omega_1, \omega_2\}$ atraktorius sugeneruotas sugeneruotas PL-algoritmu;

Išvados

1. Realizuotas pirmasis „pabėgimo laiko“ algoritmo versijos, skirtos generuoti fraktaliniams vaizdams (IFS atraktoriams) etapas – IFS sudarančių afiniųjų transformacijų veikimo zonų atskyrimas. Pasiūlytas naujas afiniųjų transformacijų superpozicijų generavimo būdas.
2. Nepaisant visų priemonių, skirtų racionaliam afiniųjų transformacijų veikimo zonų atskyrimo procedūros darbui, reikalavimai laikinėms sąnaudoms nėra pilnai tenkinami (kai kurių IFS atraktorių generavimo laikai yra 1-2 minučių eilės). Darbe nagrinėjamas ir siūlomas adaptyvus požiūris į strategiją „τ žingsnių pirmyn“ taikymą, t.y. siūloma naudoti kintamą žingsnį τ, priklausomai nuo apdorojamos IFS atraktoriaus (fraktalinio vaizdo) dalies.
3. Sklandžiam PL-algoritmo veikimui, būtina naudoti iteracijų skaičiaus (sintezės procese) korekciją, priešingu atveju gaunamas netolygiai užpildytas fraktalinis vaizdas (IFS atraktorius).
4. Kriterijaus „τ žingsnių pirmyn“ efektyvumas (laikinių sąnaudų prasme ir ne tik) labai priklauso nuo IFS sudarančiųjų afiniųjų transformacijų parametrų. Atskiru atveju kai nė viena iš afiniųjų transformacijų nerealizuoja pasūkių koordinačių ašių atžvilgiu, nepriklausomai nuo afiniųjų transformacijų skaičiaus N, pakanka imti žingsnių skaičių $\tau=1$.
5. Tyrimai rodo, jog racionalus afiniųjų transformacijų veikimo zonų atskyrimo procedūros organizavimas betarpiškai įtakoja viso „pabėgimo laiko“ algoritmo darbą, kadangi antrasis šio algoritmo etapas (tolygus fraktalinio viazdo užpildymas) faktiškai atkartoja (daugelį kartų) pirmąjį.

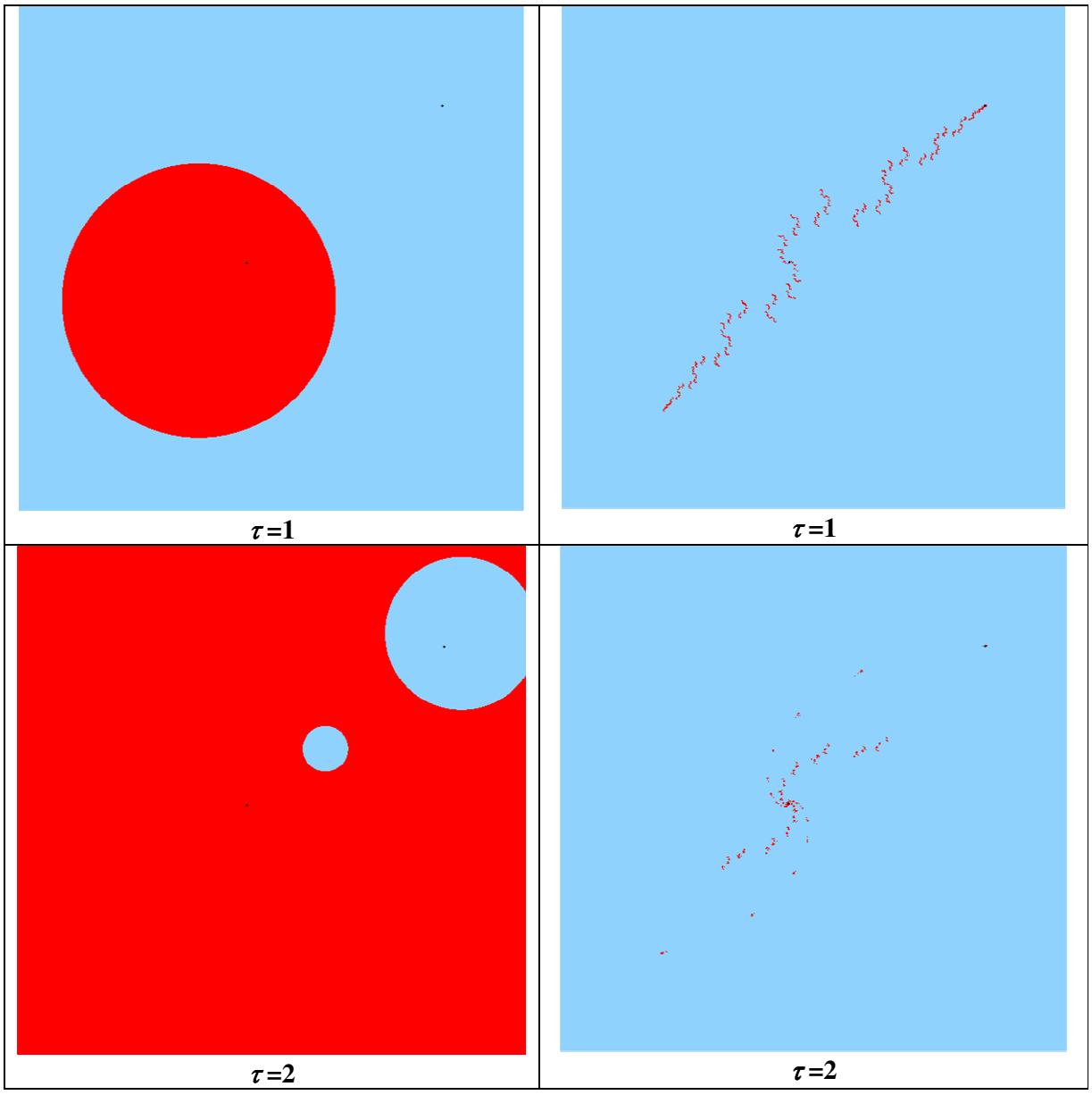
Literatūros sąrašas

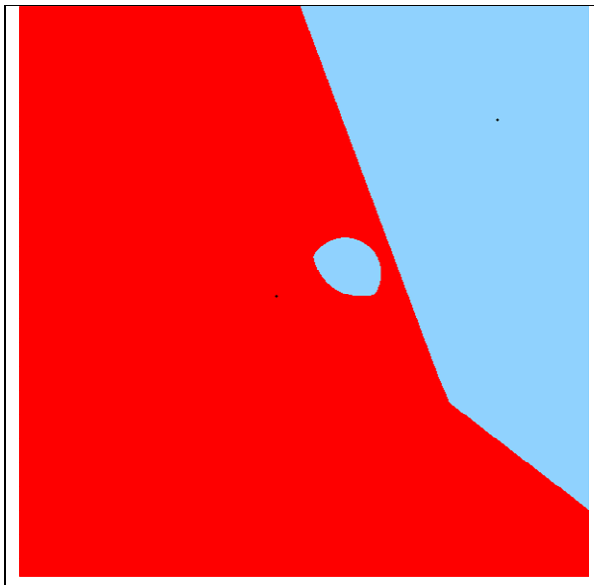
1. M. BARNSLEY, Fractals Everywhere, Second edition, Academic Press Professional, Cambridge 1993.
2. FractalVision: Put Fractals to Work For You Authors/ Dick Oliver: Publisher; 1992.— 485 p.
3. B. B. MANDELBROT, The Fractal Gheometry of Nature. 1977
4. D. OLIVER, Fractal Vision, Sams Publishing (A division of Prentice Hall Computer Publishing, Carmel, Indiana). 1992.
5. W. PAULSON, Introduction to Fractals and IFS; 1998
6. H. PEITGEN, P. RICHTER, Krasota fraktalov. Obrazy kompleksnych dinamičeskich sistem.—M.:Mir, 1993.
7. H. PEITGEN, P. RICHTER, The Beauty of Fractals, Springer—Verlag, Berlin, New York. 1986.
8. J. VALANTINAS, Fraktalinė geometrija.— K.: Technologija, 1999.—p.10— 69,121— 153

Priedai

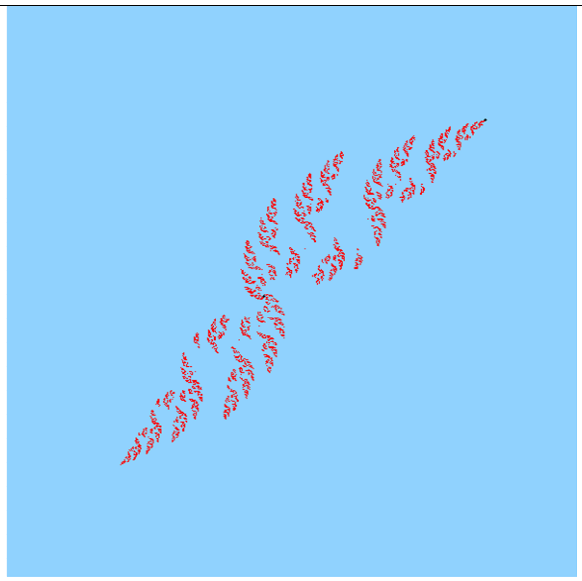
Detalesni eksperimento paveiksliukai

4.2 skyrelio antrosios IFS veikimo zonos bei atraktorius prie skirtingų afinijųjų transformacijų veikimo zonų.

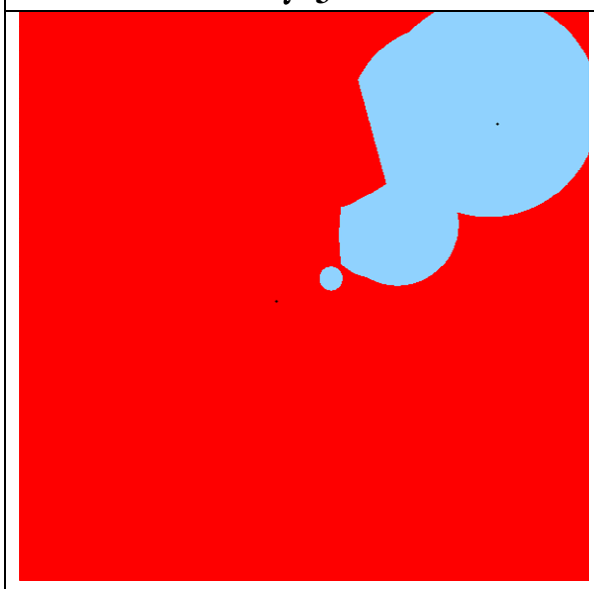




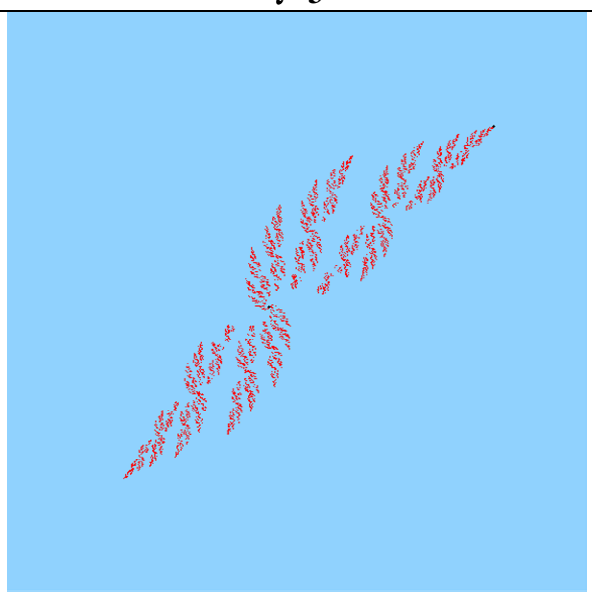
τ=3



τ=3

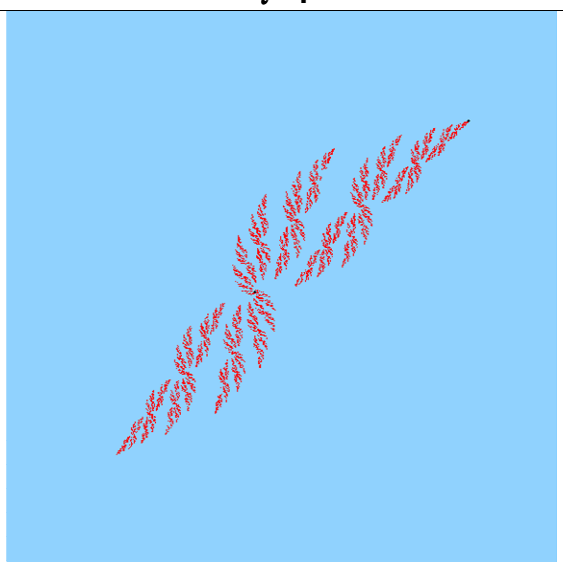


τ=4

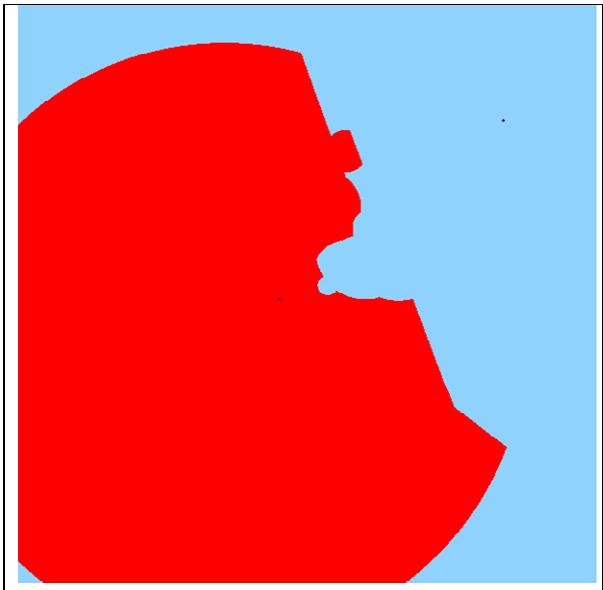


τ=4

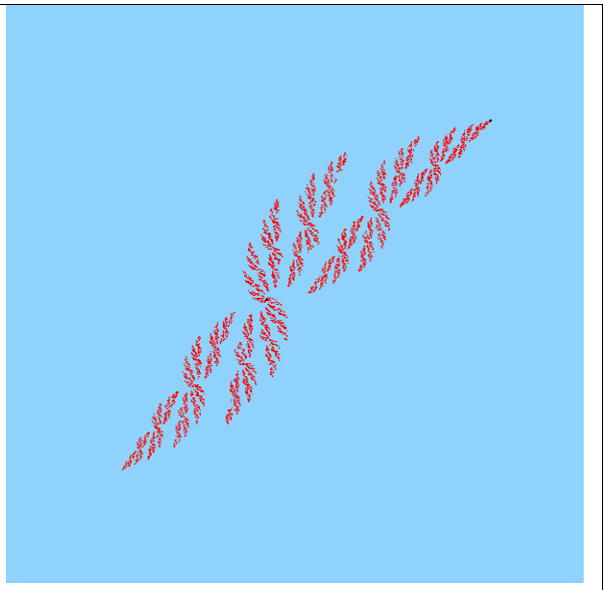
τ=5



τ=5

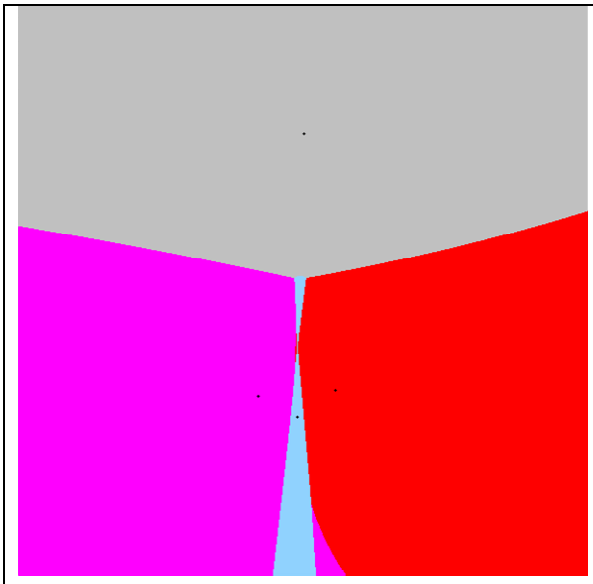


Kintamas τ

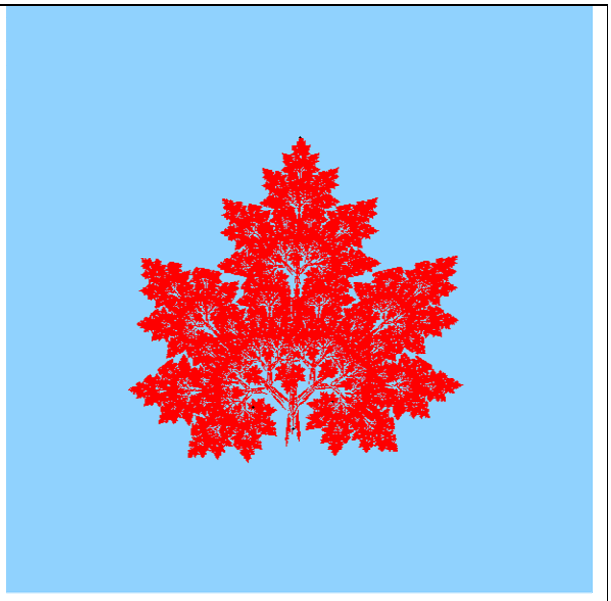


Kintamas τ

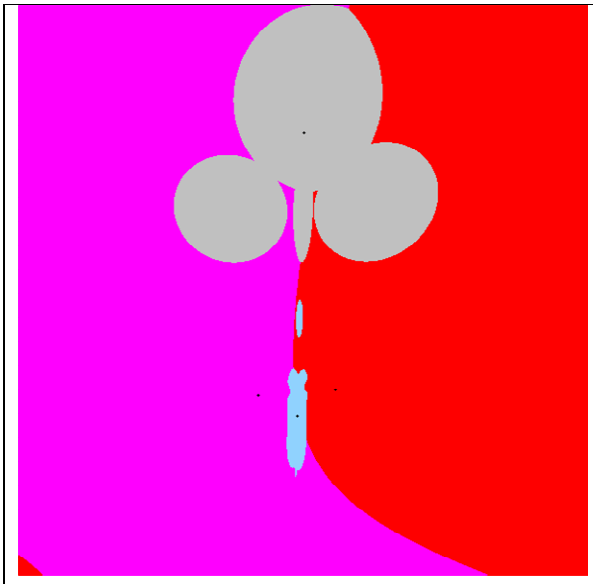
4.2 skyrelio trčiosios IFS veikimo zonos bei atraktorius prie skirtingų afinijųjų transformacijų veikimo zonų.



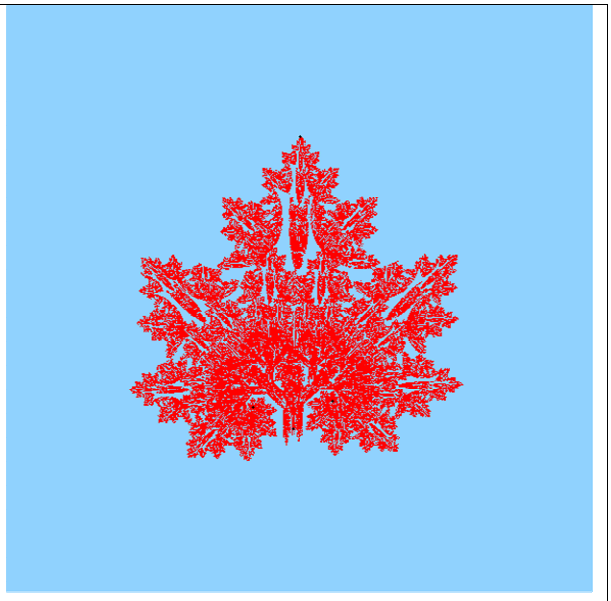
τ=1



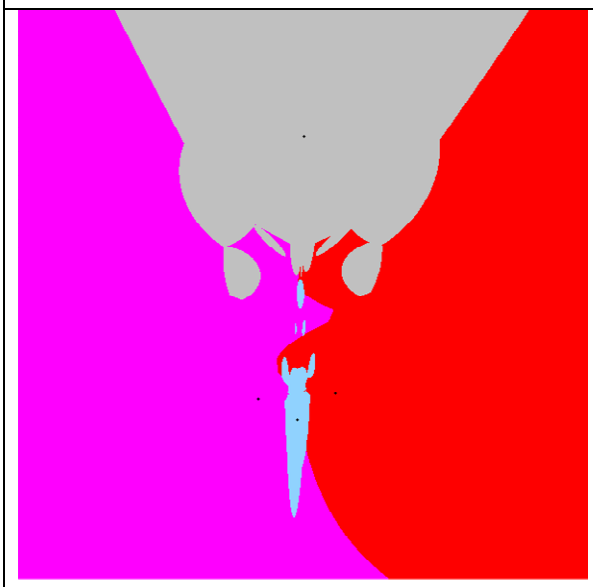
τ=1



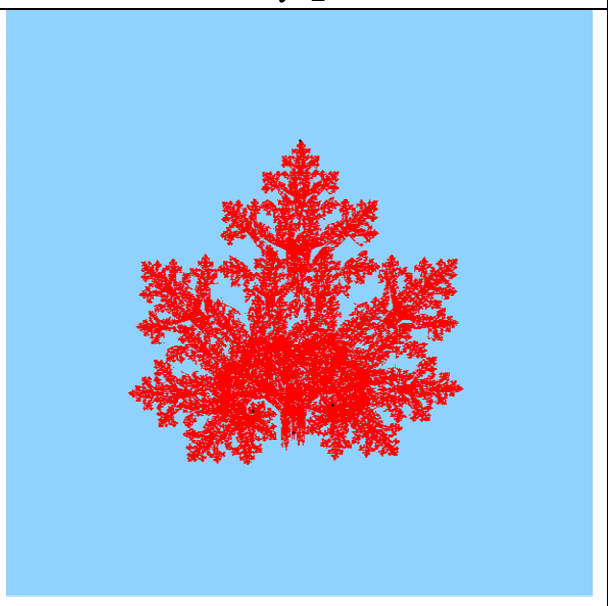
$\tau=2$



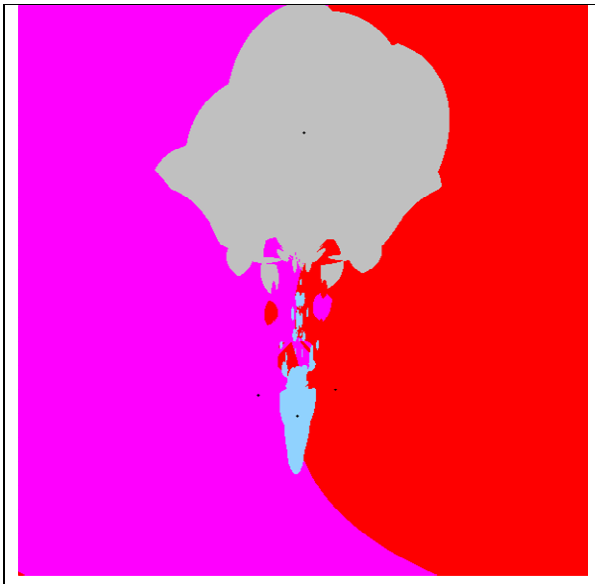
$\tau=2$



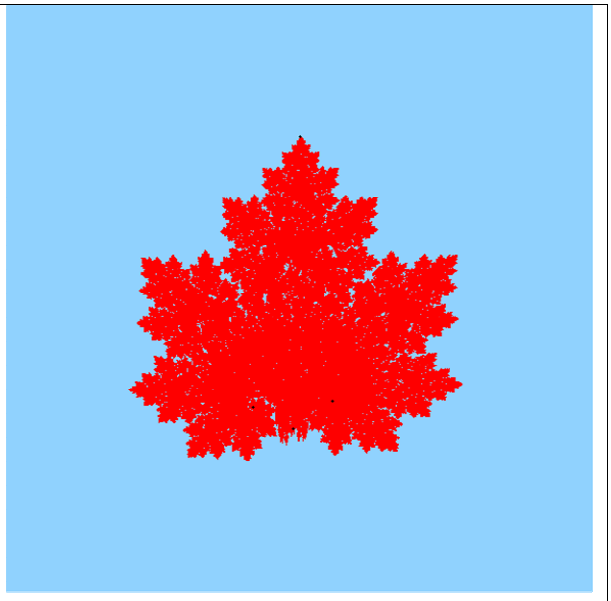
$\tau=3$



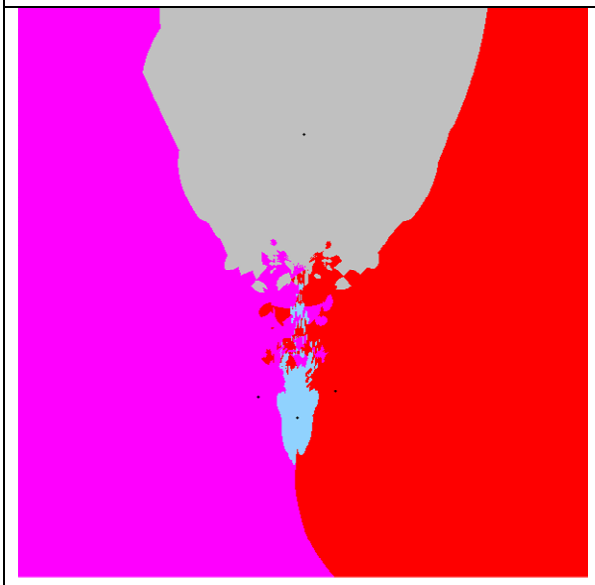
$\tau=3$



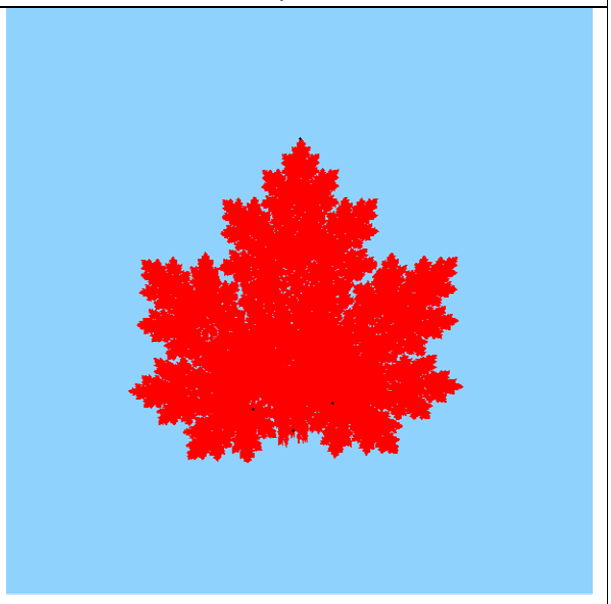
$\tau = 4$



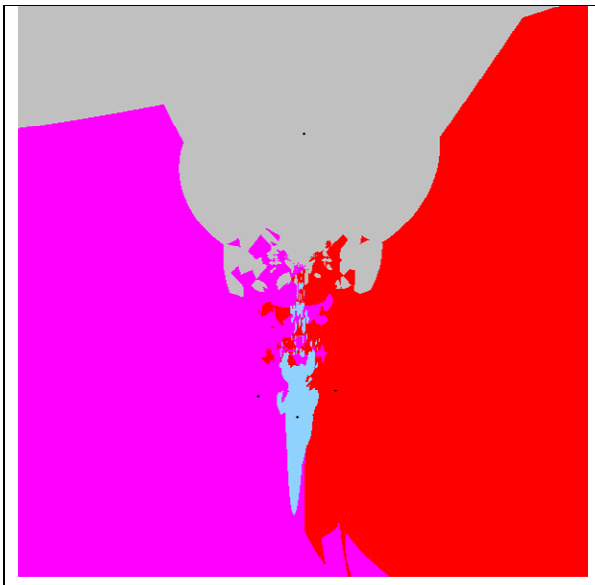
$\tau = 4$



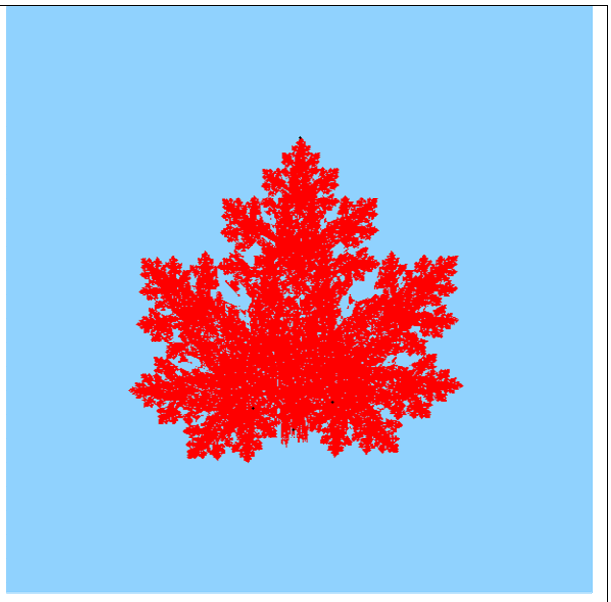
$\tau = 5$



$\tau = 5$



Kintamas τ



Kintamas τ

Programos kodas

```
using System;
using System.Drawing;
using System.Collections;
using System.Collections.Generic;
using System.ComponentModel;
using System.Windows.Forms;
using System.IO;
using System.Globalization;

using Pc.Fract.Calculations;
using Pc.Fract.Core;

namespace Pc.Fract.Main
{
                /// <summary>
                /// Summary description for frmMain.
                /// </summary>
                public class frmMain : System.Windows.Forms.Form
                {
                                private Calculate calcutale = null;
                                private frmNustatymai fn = new frmNustatymai();
                                private System.Windows.Forms.Button button_LoadFromFile;
                                private System.Windows.Forms.PictureBox pictureBox_outPut;
        private System.Windows.Forms.Button button_kombinacijos;
                                private System.Windows.Forms.Button button_veikimozonos;
                                private System.Windows.Forms.Button button_nustatymai;
                                private System.Windows.Forms.OpenFileDialog openFileDialog;
        private Button button1;
        private Button button_save;
        private SaveFileDialog saveFileDialog;
        private MenuStrip menuStrip1;
        private ToolStripMenuItem toolStripMenuItem1;
        private ToolStripMenuItem uzkarutiIFSToolStripMenuItem;
        private ToolStripMenuItem išsaugotiToolStripMenuItem;
        private ToolStripMenuItem viewToolStripMenuItem;
        private ToolStripMenuItem veikimoZonosToolStripMenuItem;
        private ToolStripMenuItem fraktalasToolStripMenuItem;
                                private System.ComponentModel.Container components = null;

                                public frmMain()
                                {
                                                InitializeComponent();
                                                calcutale= new Calculate(pictureBox_outPut.Size);
                                }

        [STAThread]
                                public static void Main(String[] args)
                                {
                                                Application.Run(new frmMain());

                                }
```

```
                                protected override void Dispose( bool disposing )
                                {
                                                if( disposing )
                                                {
                                                                if(components != null)
                                                                {
                                                                                components.Dispose();
                                                                }
                                                }
                                                base.Dispose( disposing );
                                }

                                #region Windows Form Designer generated code
                                /// <summary>
                                /// Required method for Designer support - do not modify
                                /// the contents of this method with the code editor.
                                /// </summary>
                                private void InitializeComponent()
                                {
            System.ComponentModel.ComponentResourceManager resources = new
System.ComponentModel.ComponentResourceManager(typeof(frmMain));
            this.button_LoadFromFile = new System.Windows.Forms.Button();
            this.pictureBox_outPut = new System.Windows.Forms.PictureBox();
            this.button_kombinacijos = new System.Windows.Forms.Button();
            this.button_veikimozonos = new System.Windows.Forms.Button();
            this.button_nustatymai = new System.Windows.Forms.Button();
            this.openFileDialog = new System.Windows.Forms.OpenFileDialog();
            this.button1 = new System.Windows.Forms.Button();
            this.button_save = new System.Windows.Forms.Button();
            this.saveFileDialog = new System.Windows.Forms.SaveFileDialog();
            this.menuStrip1 = new System.Windows.Forms.MenuStrip();
            this.toolStripMenuItem1 = new System.Windows.Forms.ToolStripMenuItem();
            this.uzkarutiIFSToolStripMenuItem = new System.Windows.Forms.ToolStripMenuItem();
            this.išsaugotiToolStripMenuItem = new System.Windows.Forms.ToolStripMenuItem();
            this.viewToolStripMenuItem = new System.Windows.Forms.ToolStripMenuItem();
            this.veikimoZonosToolStripMenuItem = new System.Windows.Forms.ToolStripMenuItem();
            this.fraktalasToolStripMenuItem = new System.Windows.Forms.ToolStripMenuItem();
            ((System.ComponentModel.ISupportInitialize)(this.pictureBox_outPut)).BeginInit();
            this.menuStrip1.SuspendLayout();
            this.SuspendLayout();
            //
            // button_LoadFromFile
            //
            this.button_LoadFromFile.Location = new System.Drawing.Point(608, 27);
            this.button_LoadFromFile.Name = "button_LoadFromFile";
            this.button_LoadFromFile.Size = new System.Drawing.Size(112, 24);
            this.button_LoadFromFile.TabIndex = 0;
            this.button_LoadFromFile.Text = "Uzkarauti is failo";
            this.button_LoadFromFile.Click += new System.EventHandler(this.button_LoadFromFile_Click);
            //
            // pictureBox_outPut
            //
            this.pictureBox_outPut.Image =
((System.Drawing.Image)(resources.GetObject("pictureBox_outPut.Image")));
            this.pictureBox_outPut.Location = new System.Drawing.Point(2, 25);
            this.pictureBox_outPut.Name = "pictureBox_outPut";
            this.pictureBox_outPut.Size = new System.Drawing.Size(600, 600);
            this.pictureBox_outPut.SizeMode = System.Windows.Forms.PictureBoxSizeMode.StretchImage;
            this.pictureBox_outPut.TabIndex = 1;
            this.pictureBox_outPut.TabStop = false;
            //
            // button_kombinacijos
            //
            this.button_kombinacijos.Enabled = false;
            this.button_kombinacijos.Location = new System.Drawing.Point(608, 283);
            this.button_kombinacijos.Name = "button_kombinacijos";
            this.button_kombinacijos.Size = new System.Drawing.Size(112, 23);
            this.button_kombinacijos.TabIndex = 2;
            this.button_kombinacijos.Text = "Kombinacijos";
            this.button_kombinacijos.Visible = false;
            this.button_kombinacijos.Click += new System.EventHandler(this.button_kombinacijos_Click);
            //
            // button_veikimozonos
            //
            this.button_veikimozonos.Location = new System.Drawing.Point(608, 57);
            this.button_veikimozonos.Name = "button_veikimozonos";
            this.button_veikimozonos.Size = new System.Drawing.Size(112, 23);
            this.button_veikimozonos.TabIndex = 4;
            this.button_veikimozonos.Text = "Veikimo zonos";
            this.button_veikimozonos.Click += new System.EventHandler(this.button_veikimozonos_Click);
            //
            // button_nustatymai
            //
            this.button_nustatymai.Location = new System.Drawing.Point(608, 254);
            this.button_nustatymai.Name = "button_nustatymai";
            this.button_nustatymai.Size = new System.Drawing.Size(112, 23);
            this.button_nustatymai.TabIndex = 5;
            this.button_nustatymai.Text = "Nustatymai";
            this.button_nustatymai.Visible = false;
            this.button_nustatymai.Click += new System.EventHandler(this.button_nustatymai_Click);
            //
```

```
            // button1
            //
            this.button1.Location = new System.Drawing.Point(608, 86);
            this.button1.Name = "button1";
            this.button1.Size = new System.Drawing.Size(112, 23);
            this.button1.TabIndex = 6;
            this.button1.Text = "Fraktalas";
            this.button1.Click += new System.EventHandler(this.button1_Click);
            //
            // button_save
            //
            this.button_save.Location = new System.Drawing.Point(608, 115);
            this.button_save.Name = "button_save";
            this.button_save.Size = new System.Drawing.Size(112, 23);
            this.button_save.TabIndex = 7;
            this.button_save.Text = "Issaugoti";
            this.button_save.Click += new System.EventHandler(this.button_save_Click);
            //
            // menuStrip1
            //
            this.menuStrip1.Items.AddRange(new System.Windows.Forms.ToolStripItem[] {
            this.toolStripMenuItem1,
            this.viewToolStripMenuItem});
            this.menuStrip1.Location = new System.Drawing.Point(0, 0);
            this.menuStrip1.Name = "menuStrip1";
            this.menuStrip1.Size = new System.Drawing.Size(728, 24);
            this.menuStrip1.TabIndex = 8;
            this.menuStrip1.Text = "menuStrip1";
            //
            // toolStripMenuItem1
            //
            this.toolStripMenuItem1.DropDownItems.AddRange(new System.Windows.Forms.ToolStripItem[] {
            this.uzkarutiIFSToolStripMenuItem,
            this.išsaugotiToolStripMenuItem});
            this.toolStripMenuItem1.Name = "toolStripMenuItem1";
            this.toolStripMenuItem1.Size = new System.Drawing.Size(35, 20);
            this.toolStripMenuItem1.Text = "File";
            //
            // uzkarutiIFSToolStripMenuItem
            //
            this.uzkarutiIFSToolStripMenuItem.Name = "uzkarutiIFSToolStripMenuItem";
            this.uzkarutiIFSToolStripMenuItem.Size = new System.Drawing.Size(152, 22);
            this.uzkarutiIFSToolStripMenuItem.Text = "Užkaruti IFS";
            this.uzkarutiIFSToolStripMenuItem.Click += new
System.EventHandler(this.uzkarutiIFSToolStripMenuItem_Click);
            //
            // išsaugotiToolStripMenuItem
            //
            this.išsaugotiToolStripMenuItem.Name = "išsaugotiToolStripMenuItem";
            this.išsaugotiToolStripMenuItem.Size = new System.Drawing.Size(152, 22);
            this.išsaugotiToolStripMenuItem.Text = "Išsaugoti";
            this.išsaugotiToolStripMenuItem.Click += new
System.EventHandler(this.išsaugotiToolStripMenuItem_Click);
            //
            // viewToolStripMenuItem
            //
            this.viewToolStripMenuItem.DropDownItems.AddRange(new System.Windows.Forms.ToolStripItem[] {
            this.veikimoZonosToolStripMenuItem,
            this.fraktalasToolStripMenuItem});
            this.viewToolStripMenuItem.Name = "viewToolStripMenuItem";
            this.viewToolStripMenuItem.Size = new System.Drawing.Size(52, 20);
            this.viewToolStripMenuItem.Text = "Viazdai";
            //
            // veikimoZonosToolStripMenuItem
            //
            this.veikimoZonosToolStripMenuItem.Name = "veikimoZonosToolStripMenuItem";
            this.veikimoZonosToolStripMenuItem.Size = new System.Drawing.Size(152, 22);
            this.veikimoZonosToolStripMenuItem.Text = "Veikimo zonos";
            this.veikimoZonosToolStripMenuItem.Click += new
System.EventHandler(this.veikimoZonosToolStripMenuItem_Click);
            //
            // fraktalasToolStripMenuItem
            //
            this.fraktalasToolStripMenuItem.Name = "fraktalasToolStripMenuItem";
            this.fraktalasToolStripMenuItem.Size = new System.Drawing.Size(152, 22);
            this.fraktalasToolStripMenuItem.Text = "Fraktalas";
            this.fraktalasToolStripMenuItem.Click += new
System.EventHandler(this.fraktalasToolStripMenuItem_Click);
            //
            // frmMain
            //
            this.AutoScaleBaseSize = new System.Drawing.Size(5, 13);
            this.ClientSize = new System.Drawing.Size(728, 626);
            this.Controls.Add(this.menuStrip1);
            this.Controls.Add(this.button_save);
            this.Controls.Add(this.button1);
            this.Controls.Add(this.button_nustatymai);
            this.Controls.Add(this.button_veikimozonos);
            this.Controls.Add(this.button_kombinacijos);
            this.Controls.Add(this.pictureBox_outPut);
            this.Controls.Add(this.button_LoadFromFile);
```

```
            this.MainMenuStrip = this.menuStrip1;
            this.MaximumSize = new System.Drawing.Size(736, 660);
            this.MinimumSize = new System.Drawing.Size(736, 660);
            this.Name = "frmMain";
            this.Text = "Fraktalas";
            ((System.ComponentModel.ISupportInitialize)(this.pictureBox_outPut)).EndInit();
            this.menuStrip1.ResumeLayout(false);
            this.menuStrip1.PerformLayout();
            this.ResumeLayout(false);
            this.PerformLayout();

                        }
                        #endregion

                        private void button_LoadFromFile_Click(object sender, System.EventArgs e)
                        {

                                    try
                                    {
                CultureInfo myCIintl = new CultureInfo("en-US", false);
                System.Threading.Thread.CurrentThread.CurrentCulture = myCIintl;

                                                if (openFileDialog.ShowDialog() == DialogResult.OK)
                                                {
                                                            string file =
openFileDialog.FileName;//@"D:\Documents and Settings\Paulius.DO-NOT-TOUCH-ME\My
Documents\pc_bakalaurinis\00pvz.txt";
                    List<float[]> ar = new List<float[]>();
                                                            TextReader fr = new
System.IO.StreamReader(file);
                                                            while(true)
                                                            {
                                                                        string line  = "";
                                                                        if((line =
fr.ReadLine())!=null)
                                                                        {

                string[] lineSplit = line.Split(' ');
                            float[] d = new float[lineSplit.Length];

                for(int i = 0; i<lineSplit.Length ;i++)

                {//System.In

                d[i] = (float)Double.Parse(lineSplit[i]);
                                                                                                    }

                ar.Add(d);
                                                                        }
                                                                        else
                                                                                    break;
                                                            }
                                                            fr.Close();
                    calcutale.LoadIFS(ar);
                }
                                    }
                                    catch( Exception ea)
                                    {
                                                ea.ToString();
                                    }

                        }

                        private void button_kombinacijos_Click(object sender, System.EventArgs e)
                        {
                                    frmKombinacijos fk = new frmKombinacijos(calcutale.Count);
                                    if( fk.ShowDialog() == DialogResult.OK)
                                    {
                                                //calcutale.LoadKombinacijos(fk.Kombinacijos);
                                                button_veikimozonos.Enabled = true;
                                    }
                        }

                        private void button_veikimozonos_Click(object sender, System.EventArgs e)
                        {
            pictureBox_outPut.Image = calcutale.Piesti(Drow.VeikimoZonas);
                        }

                        private void button_nustatymai_Click(object sender, System.EventArgs e)
                        {
                                    //calcutale.Mastelis = fn.Mastelis;
                                    fn.IteracijuSkaicius = calcutale.IteracijuSkaicius;
                                    if(fn.ShowDialog() == DialogResult.OK)
                                    {
                                                //calcutale.Mastelis = fn.Mastelis;
                                                calcutale.IteracijuSkaicius= fn.IteracijuSkaicius;

                                    }
                        }

        private void button1_Click(object sender, EventArgs e)
```

```
        {
            calcutale.LoadFraktalasNew();
            this.pictureBox_outPut.Image = calcutale.Piesti(Drow.Fraktala);
        }

        private void button_save_Click(object sender, EventArgs e)
        {
            try
            {
                if (saveFileDialog.ShowDialog() == DialogResult.OK)
                {
                    Bitmap b = pictureBox_outPut.Image as Bitmap;
                    b.Save(saveFileDialog.FileName);
                }
            }
            catch { }
        }

        private void uzkarutiIFSToolStripMenuItem_Click(object sender, EventArgs e)
        {
            button_LoadFromFile_Click(null, null);
        }

        private void išsaugotiToolStripMenuItem_Click(object sender, EventArgs e)
        {
            button_save_Click(null, null);
        }

        private void veikimoZonosToolStripMenuItem_Click(object sender, EventArgs e)
        {
            pictureBox_outPut.Image = calcutale.Piesti(Drow.VeikimoZonas);
        }

        private void fraktalasToolStripMenuItem_Click(object sender, EventArgs e)
        {
            calcutale.LoadFraktalasNew();
            this.pictureBox_outPut.Image = calcutale.Piesti(Drow.Fraktala);
        }

            }

}

using System;
using System.Drawing;
using System.Drawing.Imaging;
using System.Collections;
using System.Collections.Generic;

using Pc.Fract.Core;

namespace Pc.Fract.Calculations
{

            public class Calculate
            {
                        private Figura cp = new CompositeFigura("pagrindine Grupe",0,0,0,0);
                        private AbstractFabric fabrikas=null;
        private List<float[]> IFS = new List<float[]>();
        public static List<float[]> ATIFS = new List<float[]>();
        private List<float[]> nejudTaskai = new List<float[]>();
        private List<double> suspaudimoKoef = new List<double>();
        private List<double> zonuSvoriai = new List<double>();
                        private Size size = new Size(0,0);
        private int mastelisBig = 10;
        private int mastelisSmal = 10;
                        private int iteracijuSk = 27;
                        private int[][] masyvasBig = null;
        private int[][] masyvasSmal = null;
                        private int[][] fract = null;
                        protected Bitmap picture = null;
        private zoneList _zList = null;
        public static double maxRange = 0;
                        public Calculate(Size size)
                        {
                                this.size = size;
                                picture = NewPicture();
                                cp.Picture = picture;
                                fabrikas=new ConkreteFabrikas(cp);
                        }
                        private Bitmap NewPicture()
                        {
                                Bitmap b =new Bitmap(size.Width,size.Height);
            BitmapData bd = b.LockBits(new Rectangle(0, 0, b.Width, b.Height), ImageLockMode.WriteOnly,
b.PixelFormat);
            IntPtr ptr = bd.Scan0;

            int bytes = b.Width * b.Height * 4;
            byte[] rgbValues = new byte[bytes];
            System.Runtime.InteropServices.Marshal.Copy(ptr, rgbValues, 0, bytes);
```

```
 for (int i = 0; i < rgbValues.Length; i += 1)
     rgbValues[i] = 0;

 System.Runtime.InteropServices.Marshal.Copy(rgbValues, 0, ptr, bytes);

 b.UnlockBits(bd);

/* for (int i = 0; i < size.Width; i++)
     for (int j = 0; j < size.Height; j++)
         b.SetPixel(i, j, Color.White);*/
                                 return b;
                 }
                 void skaiciuotiATIFS()
                 {
                                 for(int i=0;i<IFS.Count;i++)
                                 {
     float[] lentele = new float[6];
     float a, b, c, d, e, f, detA;
                                                 a= IFS[i][0];
                                                 b= IFS[i][1];
                                                 c= IFS[i][2];
                                                 d= IFS[i][3];
                                                 e= IFS[i][4];
                                                 f= IFS[i][5];
                                                 detA=a*d-b*c;
                                                 lentele[0]=d/detA;
                                                 lentele[1]=-b/detA;
                                                 lentele[2]=-c/detA;
                                                 lentele[3]=a/detA;
                                                 lentele[4]=(f*b-e*d)/detA;
                                                 lentele[5]=(e*c-f*a)/detA;
                                                 ATIFS.Add(lentele);
                                 }
                 }

                 void nejudamiTaskai()
                 {
 float a, b, c, x, y, r;
                                 nejudTaskai.Clear();
 zonuSvoriai.Clear();
 foreach (float[] d in IFS)
 {
     a = -d[3] * d[4] + d[5] * d[1] + d[4];
     b = -d[0] * d[5] + d[4] * d[2] + d[5];
     c = d[0] * d[3] - d[0] - d[2] * d[1] - d[3] + 1;
     x = a / c;
     y = b / c;
     float[] lentele = new float[2];
     lentele[0] = x;
     lentele[1] = y;
     nejudTaskai.Add(lentele);
     float susp = Math.Abs(d[0] * d[3] - d[1] * d[2]);
     suspaudimoKoef.Add(susp);
     float xxx = (float)(2 / Math.Exp(susp - 0.5) - 1);
     if (xxx > 0.9)
         xxx = 0.9F;
     zonuSvoriai.Add(0);
     continue;
     if (susp < 0.05)
         zonuSvoriai.Add(-3.8);
     else if (susp < 0.1)
         zonuSvoriai.Add(-2.4);
     else if (susp < 0.15)
         zonuSvoriai.Add(-1.8);
     else if (susp < 0.2)
         zonuSvoriai.Add(-1.1);
     else if (susp < 0.3)
         zonuSvoriai.Add(-0.8);
     else if (susp < 0.4)
         zonuSvoriai.Add(-0.4);
     else if (susp < 0.6)
         zonuSvoriai.Add(0);
     else if (susp < 0.7)
         zonuSvoriai.Add(0.4);
     else if (susp < 0.8)
         zonuSvoriai.Add(0.6);
     else if (susp < 0.9)
         zonuSvoriai.Add(0.8);
     else if (susp < 1.1)
         zonuSvoriai.Add(0.9);
     //zonuSvoriai.Add(susp-0.5);
 }
 foreach (float[] d1 in nejudTaskai)
     foreach (float[] d2 in nejudTaskai)
      {
          r = (float)Math.Sqrt((d1[0]-d2[0])*(d1[0]-d2[0]) + (d1[1]-d2[1])*(d1[1]-d2[1]));
          if (r > maxRange)
              maxRange = r;
      }
  //maxRange *= 2;
```

```
            //(int)(((double[])(nejudTaskai[i]))[0]*mastelis)+size.Height/2,-
(int)(((double[])(nejudTaskai[i]))[1]*mastelis)+size.Height/2
            mastelisBig = (int)(size.Width / (maxRange * 20));
            mastelisSmal = (int)(size.Width / (maxRange * 2));
             //mastelis = mastelisSmal;
           // maxRange.ToString();
        }

        public void LoadIFS(List<float[]> ar)
                            {
            mastelisBig = 10;
            mastelisSmal = 10;
                                iteracijuSk = 25;
            maxRange = 0;
            IFS.Clear();
            ATIFS.Clear();
            nejudTaskai.Clear();
            if (_zList != null)
                _zList.Clear();
                                        IFS = ar;
                                        nejudamiTaskai();
                                        cp.GrazVek.Clear();
                                        ATIFS.Clear();
                                        skaiciuotiATIFS();
            _zList = new zoneList(9);
                                        veikimoZonosNew();

                            }

                            public Bitmap Piesti(Drow action)
                            {
                                        cp.GrazVek.Clear();
                                        cp.Picture = NewPicture();
                                        switch(action)
                                        {
                                                    case Drow.NejudamusTaskus:
                                                                PiestiNejudamusTaskus();
                                                                break;
                                                    case Drow.VeikimoZonas:
                                                                PiestiVeikimoZonas();
                                                                PiestiNejudamusTaskus();
                                                                break;
                                                    case Drow.Fraktala:
                                                                PiestiFraktala();
                                                                PiestiNejudamusTaskus();
                                                                break;

                                        }
                                        cp.piesti(null);
                                        return cp.Picture;
                            }
                            private void PiestiNejudamusTaskus()
                            {
                                        for(int i=0;i<nejudTaskai.Count;i++)
                                        {
                fabrikas.taskas("s", (int)(((float[])(nejudTaskai[i]))[0] * mastelisSmal) + size.Height / 2, -
(int)(((float[])(nejudTaskai[i]))[1] * mastelisSmal) + size.Height / 2, 0, 0);
                                        }
                            }
                            private void PiestiVeikimoZonas()
                            {
                                        fabrikas.funkcija(size.Width,size.Height,masyvasSmal);
                            }
                            private void PiestiFraktala()
                            {
                                        fabrikas.funkcija(size.Width,size.Height,fract);
                            }
                            public int Count
                            {
                                        get {return IFS.Count;}
                            }
                            public int IteracijuSkaicius
                            {
                                        get{return iteracijuSk;}
                                        set{iteracijuSk = value;}
                            }
                            /*public int Mastelis
                            {
                                        get{return mastelisSmal;}
            set { mastelisSmal = value; }
                            }*/

                            private void veikimoZonosNew()
                            {
                                        int si = size.Width;
                                        masyvasBig = new int[si+1][];

                                        for(int ii = 0; ii < si+1;ii++)
                                                    masyvasBig[ii] =                new int[si+1];
            TDebug.Start("Test", "Zone");
                                        for(int xp=1;xp<si;xp++)
```

```
                                                    for(int yp=1;yp<si;yp++)
                                                    {
                masyvasBig[xp][yp] = _zList.GetZones[8].nustatytiZona(xp, yp, si, mastelisBig);
                if (masyvasBig[xp][yp] == -1)
                {
                    masyvasBig[xp][yp] = _zList.GetZones[2].nustatytiZona(xp, yp, si, mastelisBig);
                    if (masyvasBig[xp][yp] == -1)
                    {
                        masyvasBig[xp][yp] = _zList.GetZones[4].nustatytiZona(xp, yp, si, mastelisBig);

                    }
                }
                                                    }
            TDebug.Reg("Test", "x1");

            masyvasSmal = new int[si + 1][];

            for (int ii = 0; ii < si + 1; ii++)
                masyvasSmal[ii] = new int[si + 1];
            for (int xp = 1; xp < si; xp++)
                for (int yp = 1; yp < si; yp++)
                {
                    masyvasSmal[xp][yp] = _zList.GetZones[8].nustatytiZona(xp, yp, si, mastelisSmal);
                    if (masyvasSmal[xp][yp] == -1)
                    {
                        masyvasSmal[xp][yp] = _zList.GetZones[2].nustatytiZona(xp, yp, si, mastelisSmal);
                        if (masyvasSmal[xp][yp] == -1)
                        {
                            masyvasSmal[xp][yp] = _zList.GetZones[4].nustatytiZona(xp, yp, si, mastelisSmal);

                        }
                    }
                }
            TDebug.Reg("Test", "x2");
            TDebug.Show("Test");
            masyvasSmal[0] = masyvasSmal[0];
                              }

        public void LoadFraktalasNew()
        {
            fraktalasNew();
        }

        void fraktalasNew()
        {
            fract = new int[size.Width + 1][];
            for (int ii = 0; ii < size.Width + 1; ii++)
                fract[ii] = new int[size.Height + 1];
            for (int xp = 1; xp < size.Width; xp++)
                for (int yp = 1; yp < size.Height; yp++)
                {
                    double isk = iteracijuSk;
                    double x = ((double)(xp - size.Width / 2)) / mastelisSmal;
                    double y = -((double)(yp - size.Height / 2)) / mastelisSmal;
                    double[] lentele = new double[5];
                    lentele[0] = x;
                    lentele[1] = y;
                    lentele[3] = 0;
                    double xx = x, yy = y;
                    for (int i = 0; i < isk; i++)
                    {
                        double x1, y1, min;
                        int aa = (int)(xx * mastelisSmal + size.Width / 2);
                        int bb = (int)(-yy * mastelisSmal + size.Width / 2);
                        if (aa <= 0 || aa >= 600 || bb <= 0 || bb >= 600)
                        {
                            fract[xp][yp] = fractX(i, xx, yy, x, y, isk);
                            break;
                        }
                        if (zonuSvoriai[masyvasSmal[aa][bb]] < -10)
                        {
                            fract[xp][yp] = 1;
                            break;
                        }
                        isk += zonuSvoriai[masyvasSmal[aa][bb]];
                        //if (i > 6 && bb == 300 && aa == 598)
                        //    isk = isk;
                        x1 = ATIFS[masyvasSmal[aa][bb]][0] * xx + ATIFS[masyvasSmal[aa][bb]][1] * yy +
ATIFS[masyvasSmal[aa][bb]][4];
                        y1 = ATIFS[masyvasSmal[aa][bb]][2] * xx + ATIFS[masyvasSmal[aa][bb]][3] * yy +
ATIFS[masyvasSmal[aa][bb]][5];
                        min = (x - x1) * (x - x1) + (y - y1) * (y - y1);
                        xx = x1;
                        yy = y1;
                        fract[xp][yp] = 1;
                    }

                }
        }

        private int fractX(int j, double xx, double yy,double x, double y, double isk)
```

```
        {
            for (int i = j; i < isk; i++)
            {
                double x1, y1, min;
                int aa = (int)(xx * mastelisBig + size.Width / 2);
                int bb = (int)(-yy * mastelisBig + size.Width / 2);
                if (aa < 0 || aa > 600 || bb < 0 || bb > 600)
                {
                    return 0;
                }
                isk += zonuSvoriai[masyvasBig[aa][bb]];
                x1 = ATIFS[masyvasBig[aa][bb]][0] * xx + ATIFS[masyvasBig[aa][bb]][1] * yy +
ATIFS[masyvasBig[aa][bb]][4];
                y1 = ATIFS[masyvasBig[aa][bb]][2] * xx + ATIFS[masyvasBig[aa][bb]][3] * yy +
ATIFS[masyvasBig[aa][bb]][5];
                min = (x - x1) * (x - x1) + (y - y1) * (y - y1);
                xx = x1;
                yy = y1;
            }
            return 1;
        }
            }
}

using System;
using System.Collections.Generic;
using System.Collections;
using System.Text;

namespace Pc.Fract.Calculations
{
    class zone
    {
        private List<float[]> afiniosTr = new List<float[]>();
        private List<string> kombinacijos = new List<string>();
        private List<float[]> ATIFS = Calculate.ATIFS;
        private double maxRange = 0;
        private int cc = 0;
        private int help1 = 1;

        public zone(int i)
        {
            maxRange = Calculate.maxRange * 10;// *Calculate.maxRange * 3;
            generuotiKombinacijas(0, ATIFS.Count, i, "");
            skaiciuotiAfiniasasTransformacijas();
            help1 = afiniosTr.Count / ATIFS.Count;
            cc = i;
        }

        private void generuotiKombinacijas(int iteracija, int N, int M, String s)
        {
            if (iteracija < M)
                for (int i = 0; i < N; i++)
                    generuotiKombinacijas(iteracija + 1, N, M, s + Convert.ToString(i + 1));
            else
                kombinacijos.Add(s);
        }

        private void skaiciuotiAfiniasasTransformacijas()
        {
            int i = kombinacijos[0].ToString().Length - 1;
            for (int j = 0; j < kombinacijos.Count; j++)
            {
                char[] eil = kombinacijos[j].ToString().ToCharArray();
                int g = (int)Char.GetNumericValue(eil[0]) - 1;
                float a = ((float[])(ATIFS[g]))[0],
                    b = ((float[])(ATIFS[g]))[1],
                    c = ((float[])(ATIFS[g]))[2],
                    d = ((float[])(ATIFS[g]))[3],
                    e = ((float[])(ATIFS[g]))[4],
                    f = ((float[])(ATIFS[g]))[5];
                for (int k = 0; k < i; k++)
                {
                    g = (int)Char.GetNumericValue(eil[k + 1]) - 1;
                    float aa, bb, cc, dd, ee, ff;
                    aa = a * ((float[])(ATIFS[g]))[0] + c * ((float[])(ATIFS[g]))[1];
                    bb = b * ((float[])(ATIFS[g]))[0] + d * ((float[])(ATIFS[g]))[1];
                    cc = a * ((float[])(ATIFS[g]))[2] + c * ((float[])(ATIFS[g]))[3];
                    dd = b * ((float[])(ATIFS[g]))[2] + d * ((float[])(ATIFS[g]))[3];
                    ee = e * ((float[])(ATIFS[g]))[0] + f * ((float[])(ATIFS[g]))[1]
                        + ((float[])(ATIFS[g]))[4];
                    ff = e * ((float[])(ATIFS[g]))[2] + f * ((float[])(ATIFS[g]))[3]
                        + ((float[])(ATIFS[g]))[5];
                    a = aa;
                    b = bb;
                    c = cc;
                    d = dd;
                    e = ee;
                    f = ff;
                }
                float[] lentele = new float[6];
```

```
                lentele[0] = a;
                lentele[1] = b;
                lentele[2] = c;
                lentele[3] = d;
                lentele[4] = e;
                lentele[5] = f;
                afiniosTr.Add(lentele);
            }
        }

        public int nustatytiZona(int xp, int yp, int lenght, int mastelis)
        {
            int rez = 0;
            float[] minimums = new float[ATIFS.Count];
            float x = ((float)(xp - lenght / 2)) / mastelis;
            float y = -((float)(yp - lenght / 2)) / mastelis;
            float x1 = afiniosTr[0][0] * x + afiniosTr[0][1] * y + afiniosTr[0][4];
            float y1 = afiniosTr[0][2] * x + afiniosTr[0][3] * y + afiniosTr[0][5];
            float min = (x - x1) * (x - x1) + (y - y1) * (y - y1);
            minimums[0] = min;
            for (int i = 1; i < kombinacijos.Count; i++)
            {
                x1 = afiniosTr[i][0] * x + afiniosTr[i][1] * y + afiniosTr[i][4];
                y1 = afiniosTr[i][2] * x + afiniosTr[i][3] * y + afiniosTr[i][5];
                float a = ((x - x1) * (x - x1) + (y - y1) * (y - y1));
                if (a < minimums[i / afiniosTr.Count] || minimums[i / help1] == 0.0)
                    minimums[i / help1] = a;
                if (a < min)
                {
                    min = a;
                    rez = i / (afiniosTr.Count / ATIFS.Count);
                }
            }
            if (cc == 9)
                return rez;
            else
            {
                float m1 = minimums[0];
                float m2 = minimums[1];
                for(int ii = 2; ii < minimums.Length; ii++)
                    if(minimums[ii] < m1)
                        m1 = minimums[ii];
                    else if(minimums[ii] < m2)
                        m2 = minimums[ii];
                if (Math.Abs(m1 - m2) < maxRange)
                {
                    return -1;
                }
            }
            return rez;
        }
    }
}

using System;
using System.Collections.Generic;
using System.Text;

namespace Pc.Fract.Calculations
{
    class zoneList
    {
        private List<zone> _zoneList;

        public zoneList(int Count)
        {
            _zoneList = new List<zone>();
            for (int i = 0; i < Count; i++)
            {
                _zoneList.Add(new zone(i+1));
            }
        }

        public List<zone> GetZones
        {
            get
            {
                return _zoneList;
            }
        }

        public void Clear()
        {
            _zoneList.Clear();
        }
    }

}

using System;
using System.Drawing;
```

```
using System.Drawing.Imaging;

namespace Pc.Fract.Core
{

public class Funkcija : Figura
{
  int[][] x;
  int i,j;
  public Funkcija(int i1,int j1, int[][] x1) :base("s",1,1,0,0)
  {
    i=i1;
    j=j1;
    x=x1;
  }
  public override void setXY(int x1,int y1,int l1,int h1)
  {
  }

  public override void piesti(Bitmap p)
  {
    ///g.drawImage()
  //  g.fillRect();
      try
      {
          BitmapData bd = p.LockBits(new Rectangle(0, 0, p.Width, p.Height), ImageLockMode.WriteOnly,
p.PixelFormat);
          IntPtr ptr = bd.Scan0;

          int bytes = p.Width * p.Height * 4;
          byte[] rgbValues = new byte[bytes];
          System.Runtime.InteropServices.Marshal.Copy(ptr, rgbValues, 0, bytes);

          for (int xp = 1; xp < i; xp++)
              for (int yp = 1; yp < j; yp++)
              {
                  if (x[xp][yp] == 0)
                  {
                      rgbValues[(xp - 1) * 4 + (yp - 1) * 4 * 600] = 255;
                      rgbValues[(xp - 1) * 4 + (yp - 1) * 4 * 600 + 1] = 206;
                      rgbValues[(xp - 1) * 4 + (yp - 1) * 4 * 600 + 2] = 135;
                      rgbValues[(xp - 1) * 4 + (yp - 1) * 4 * 600 + 3] = 235;
                  }
                  //p.SetPixel(xp,yp,Color.Red);
                  else if (x[xp][yp] == 1)
                  {
                      rgbValues[(xp - 1) * 4 + (yp - 1) * 4 * 600] = 0;
                      rgbValues[(xp - 1) * 4 + (yp - 1) * 4 * 600 + 1] = 0;
                      rgbValues[(xp - 1) * 4 + (yp - 1) * 4 * 600 + 2] = 255;
                      rgbValues[(xp - 1) * 4 + (yp - 1) * 4 * 600 + 3] = 255;
                  }
                  else if (x[xp][yp] == 2)
                  {
                      rgbValues[(xp - 1) * 4 + (yp - 1) * 4 * 600] = 255;
                      rgbValues[(xp - 1) * 4 + (yp - 1) * 4 * 600 + 1] = 0;
                      rgbValues[(xp - 1) * 4 + (yp - 1) * 4 * 600 + 2] = 255;
                      rgbValues[(xp - 1) * 4 + (yp - 1) * 4 * 600 + 3] = 255;
                  }
                  else if (x[xp][yp] == 3)
                  {
                      rgbValues[(xp - 1) * 4 + (yp - 1) * 4 * 600] = 192;
                      rgbValues[(xp - 1) * 4 + (yp - 1) * 4 * 600 + 1] = 192;
                      rgbValues[(xp - 1) * 4 + (yp - 1) * 4 * 600 + 2] = 192;
                      rgbValues[(xp - 1) * 4 + (yp - 1) * 4 * 600 + 3] = 255;
                  }
                      //p.SetPixel(xp,yp,Color.Blue);
                      //else
                      //    p.SetPixel(xp, yp, Color.Yellow);
              }
          System.Runtime.InteropServices.Marshal.Copy(rgbValues, 0, ptr, bytes);

          p.UnlockBits(bd);
      }
      catch (Exception e)
      {
          e.ToString();
      }
  }
}
}
```