

MASARYKOVA UNIVERZITA
FAKULTA INFORMATIKY



Metody vykreslování chaotických atraktorů

BAKALÁŘSKÁ PRÁCE

Lenka Racková

Brno, květen 2009

Prohlášení

Prohlašuji, že tato bakalářská práce je mým původním autorským dílem, které jsem vypracovala samostatně. Všechny zdroje, prameny a literaturu, které jsem při vypracování používala nebo z nich čerpala, v práci řádně cituji s uvedením úplného odkazu na příslušný zdroj.

Vedoucí práce: Mgr. Jiří Chmelík

Poděkování

Děkuji Mgr. Jiřímu Chmelíkovi za odborné vedení práce, jeho rady a připomínky, a především za čas, který mi věnoval.

Shrnutí

Tato bakalářská práce se zabývá speciálním případem fraktálů, kterým jsou chaotické atraktory. Obsahuje jejich popis od matematického základu po estetickou stránku. Zvláštní důraz je v této práci kladen na různé metody, kterými lze chaotické atraktory vykreslovat. Písemná část také obsahuje popis přiloženého programu – jeho implementaci, rozhraní, popis práce s programem, ukázky a srovnání grafických výstupů.

Klíčová slova: Chaotické atraktory, Lyapunův exponent, fraktální dimenze, fraktály, algoritmické umění, teorie chaosu

Obsah

1	Úvod	2
2	Algoritmické umění	3
3	Chaotické atraktory	4
3.1	Atraktor dynamického systému	4
3.2	Lyapunovův exponent	6
3.3	Fraktální dimenze	7
3.4	Jednoduché atraktory	8
3.5	Atraktory v rovině	9
3.6	Atraktory v prostoru	10
3.7	Čtvrtá dimenze a hyperprostor	10
4	Vykreslování atraktorů	11
4.1	Nalezení chaotického řešení a náhodný atraktor	12
4.2	Stupně šedi	13
4.3	Obarvování atraktorů	14
4.4	Rozmísťování dvojrozměrných objektů	16
4.5	Složitost	17
5	Popis programu	18
5.1	Použité programové prostředky	18
5.2	Uživatelské rozhraní aplikace	19
5.3	Implementace vykreslovacích metod	20
6	Závěr	23
	Literatura	24
A	Obsah CD	26
B	Použité chaotické funkce	27
C	Ukázky grafických výstupů	29

Kapitola 1

Úvod

Chaotické atraktory se z estetického hlediska vyznačují nápadně univerzální vizuální elegancí. Díky této vlastnosti a přístupnosti tvůrčího generování pro umělce – relativního laika – se staly jedním z nástrojů celého uměleckého proudu, takzvaného *algoritmického umění*. Téma chaotických atraktorů je však zajímavé nejen kvůli vizuální atraktivitě a nevšednosti, ale i pro jejich matematickou stránku.

Nutnou podmínkou využití atraktorů k tvorbě uměleckých děl je schopnost jejich generování, zobrazení a následně vykreslení do obrázku, který může být dále zpracováván. Cílem práce bylo popsat některé metody vykreslování chaotických atraktorů a tyto následně implementovat v programu.

Kapitola 2 se v krátkosti zmiňuje o vývoji algoritmického umění tak, aby byl čtenář zasazen do kontextu. Chaotickými atraktory se pak zabývá kapitola 3, ve které jsou objasněny základní pojmy, popsány některé důležité vlastnosti atraktorů, a definovány základní vztahy, které budou v dalších částech dále používány. Navazující kapitola 4 se podrobněji zabývá způsoby vykreslování atraktorů, především možnostem praktické implementace.

Hlavní částí práce je program „aTraktor“, jehož účelem je generování a zobrazování vybraných typů chaotických atraktorů. Program implementuje většinu z metod popsaných v této práci. Jeho popisu je věnována kapitola 5, která se více než detaily implementace zabývá principem jeho funkce. Je v ní také stručně popsáno uživatelské rozhraní aplikace a její ovládání.

Kapitola 2

Algoritmické umění

Již od dávných dob je umění ovlivněno matematikou. V posledním půlstoletí je to například algoritmické umění (*algorithmic art*), které se objevilo spolu s nástupem počítačů.

Vývoj algoritmického umění (uvědoměle algoritmického, nikoli pouze abstraktně ornamentálního, jako například umění islámských kultur) lze ale v jeho principu sledovat až k Černému čtverci Kazimíra Maleviče – symbolickému ground zero výtvarného umění. Za vývojový stupeň by se dala považovat gestická malba Jacksona Pollocka, kde algoritmus nemá matematickou podobu, ale výsledná malba vykazuje vlastnosti fraktálu. Za první příklady skutečného algoritmického umění lze považovat práce George Neese a Friedera Nakeho.

Umělec generující svá díla podle nějakého algoritmu se nazývá algorista (*algorist*). Takto vypadá definice algoritmického umění podle Jeana-Pierra Héberta [11] ve formě algoritmu:

```
if (creation && object of art && algorithm && one's own algorithm)
{
    include * an algorist *
}
elseif (!creation || !object of art || !algorithm || !one's own algorithm)
{
    exclude * not an algorist *
}
```

Chaotické atraktory v současné době nacházejí uplatnění i v architektuře, kde mají vedle estetické také materiální rovinu. Atraktor není jen ladný útvar, ale singularita rovnovážného stavu systému s určitými parametry v prostoru. Této vlastnosti využívá *parametrická architektura* – nástroj hledání forem jak pro digitální, tak skutečné stavby.

Kapitola 3

Chaotické atraktory

Pro pochopení chaotických atraktorů je nezbytné alespoň intuitivní seznámení s některými pojmy. Základ tvoří teorie chaosu, která popisuje chování nelineárních dynamických systémů, které za určitých podmínek vykazují jev zvaný chaos. Hlavní vlastností takového chaotického systému je velká citlivost na počáteční podmínky, jeho chování se jeví jako náhodné, i když ve skutečnosti je deterministické.

Zvolíme-li dva nekonečně blízké body, které budou reprezentovat počáteční podmínky systému, budou se od sebe následně exponenciálně vzdalovat, takže budoucí stav systému není možné žádným způsobem předpovědět. Příkladem může být počasí. To je řízeno atmosférou, a atmosféra se podřizuje deterministickým fyzikálním zákonům. Ovšem dlouhodobé předpovědi počasí se příliš nezdokonalily ani díky zpřesňování měření a nasazení výpočetní techniky. Bylo dokázáno, že počasí nelze předpovědět v horizontu větším, než dva až tři týdny [1]. Důvodem je to, že počasí vykazuje extrémní citlivost na počáteční podmínky. Velmi malá změna dnes zapříčiní větší změnu zítra a ještě mnohem více ovlivní počasí následující den. Tento jev je častěji znám jako *motýlí efekt*, podle kterého například mávnutí motýlích křídel v Brazílii může vyvolat tornádo v Texasu [4]. Jelikož počáteční podmínky nebudeme nikdy s absolutní přesností znát, dlouhodobé předpovědi jsou nemožné, dokonce i když fyzikální zákony jsou známé a chovají se deterministicky. Zde je nutné si uvědomit, že *deterministický* neznamena to samé, co *předvídatelný*.

3.1 Atraktor dynamického systému

Jedním ze způsobů vizualizace chaotického pohybu je vytvoření abstraktního prostoru stavů zvaného *fázový prostor*. V něm každá osa představuje jednu dimenzi stavu a čas je zde implicitní. Necháme-li systém vyvíjet, vznikne ve fázovém prostoru křivka. Po dostatečně dlouhé době začne tato křivka zvyrazňovat strukturu, které se říká *atraktor*.

Pod slovem atraktor si tedy můžeme představit konečný stav nějakého systému. Je to množina, ke které systém z okolních stavů vždy konverguje. Tento jev můžeme ilustrovat na jamce a kuličce, která se do ní z určitého okolí vždy skutálí. V tomto případě je atraktorem nejhlubší bod jamky a oblast, ze které se do jamky kulička skutálí, se nazývá *oblast přitažení*. Atraktor však nemusí být pouze bod. Někdy je to varieta, jindy jednoduchá

nebo dvojitá smyčka. Předmětem této práce bude nejsložitější z nich, takzvaný *chaotický* nebo také *podivný atraktor* (*strange attractor*).

Termín *chaotický atraktor* není ještě přesně matematicky definován, ale jeho vlastnosti jsou téměř shodné s vlastnostmi fraktálů. Chaotický atraktor může vzniknout jen tehdy, je-li systém popsán minimálně třemi navzájem souvisejícími diferenciálními rovnicemi. Chaotické systémy lze obecně popsat deterministickým vztahem

$$x_t = f(x_{t-1}, x_{t-2}, \dots), \quad (3.1)$$

kde x_t je skalár nebo vektor. Pokud má systém vykazovat chaotické chování, funkce f musí být nelineární. Pro chaos je nelinearita funkcí nutná, nikoliv však postačující podmínka – mnoho nelineárních funkcí chaos negeneruje.

Nejstarším a přitom nejznámějším takovým atraktorem je Lorenzův atraktor, který je odvozen ze zjednodušených rovnic vynucené konvekce v atmosféře [5]. Na něm byla numericky i analyticky ověřena velká citlivost na počáteční podmínky [10]. Skládá se ze třech navzájem závislých diferenciálních rovnic:

$$\begin{aligned} \frac{dx}{dt} &= \sigma(y - x), \\ \frac{dy}{dt} &= -xz + rx - y, \\ \frac{dz}{dt} &= xy - bz, \end{aligned} \quad (3.2)$$

kde σ , r a b jsou konstanty. Pro účely programového zpracování lze předchozí rovnici upravit následovně:

$$\begin{aligned} x_{n+1} &= x_n + (-a \cdot x_n \cdot dt) + (a \cdot y_n \cdot dt), \\ y_{n+1} &= y_n + (b \cdot x_n \cdot dt) - (y_n \cdot dt) - (z_n \cdot x_n \cdot dt), \\ z_{n+1} &= z_n + (-c \cdot z_n \cdot dt) + (x_n \cdot y_n \cdot dt), \end{aligned} \quad (3.3)$$

kde a , b , c a dt budou určovat počáteční podmínky systému.

Většinu atraktorů lze generovat iterativními technikami, následující bod počítáme na základě znalosti bodu předchozího. Počítačová implementace takového způsobu je velmi jednoduchá. Doba vykreslování závisí na požadavcích na kvalitu výstupu, čím více iterací je provedeno, tím je obraz ve výsledku ostřejší a obsahuje méně šumu.

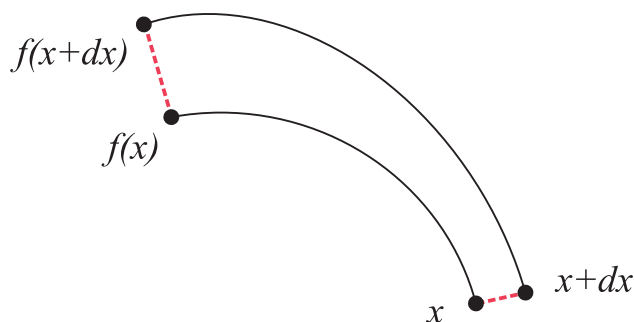
Při simulaci na počítači získáme periodický atraktor. K zobrazení číselných hodnot je použito konečného počtu bitů, po určitém počtu kroků se začne dráha atraktoru překrývat. V matematicky přesném modelu taková situace nenastane. Každá smyčka bude mít unikátní tvar, dráhy se nebudou překrývat, pouze protínat [10].

3.2 Lyapunovův exponent

Chaotické atraktory charakterizuje jejich velká citlivost na počáteční podmínky, tedy jejich parametry. Stupeň této *chaotičnosti* lze číselně vyjádřit parametrem, který se označuje jako *Lyapunovův exponent*. Je pojmenován po ruském matematikovi, jehož jméno se z azbuky přepisuje různě; proto se v některých zdrojích lze setkat i s označením *Ljapunov*, *Liapunov*, *Lyapunof* nebo dokonce *Liapunoff*.

Lyapunovův exponent je základním nástrojem pro popis dynamického systému. Jeho výpočet je poměrně obtížné numericky zvládnutelný, problémy s hledáním této charakteristiky jsou však vyváženy informacemi, které jejich nalezením získáme. Lyapunovovy exponenty totiž přímo svými hodnotami říkají, jak se systém chová. Jinými způsoby se klasifikace jeho chování provádí obtížněji [13].

Lyapunovův exponent vyjadřuje, zda blízké dráhy atraktoru konvergují nebo divergují. Takových exponentů existuje pro každý systém hned několik, přesněji řečeno právě jeden pro každou jeho dimenzi. Nejdůležitější je však většinou ten největší. Zjednodušeně lze říci, že maximální Lyapunovův exponent nejvíce ovlivňuje dlouhodobé chování systému.



Obrázek 3.1: divergující dráhy atraktoru

Pokud je exponent záporný, pak dráhy v čase konvergují a dynamický systém není citlivý vůči počátečním podmínkám. Když je ale exponent kladný, pak vzdálenosti mezi blízkými dráhami v čase exponenciálně rostou a takový systém vykazuje citlivost na počáteční podmínky. Příklad takových divergujících drah je uveden na obrázku 3.1. Chaotický systém musí mít alespoň jeden Lyapunovův exponent kladný, tedy alespoň v jednom směru se od sebe musí sousední trajektorie exponenciálně vzdalovat.

Existují dva základní způsoby, jak Lyapunovův exponent vypočítat. Jeden způsob je zvolit si několik blízkých bodů, které necháme v čase rozvíjet a přitom sledujeme rychlost růstu jejich vzájemné vzdálenosti. Tomuto postupu se také říká *Wolfův algoritmus*.

Další způsob lze shrnout do několika základních bodů [13]:

- Na určitém úseku je řešena soustava nelineárních rovnic a s ní spjatá trojice lineárních soustav, čímž je určen vývoj os.
- Nelineární soustava určuje středovou trajektorii a každá z trojice lineárních soustav

určuje vývoj sféry počátečních podmínek (což představuje trojici ortonormálních vektorů).

- Dále jsou stanoveny přírůstky všech tří os a je provedena reortonormalizaci vektorů os. K ortonormalizaci se používá *Gram-Schmidtův postup* [12].
- Nakonec jsou vypočítány lokální Lyapunovovy exponenty v čase t ze vztahu:

$$\lambda_t = \frac{1}{t} \cdot \log_2 p_t \quad (3.4)$$

kde t je dosažený čas a p_t je přírůstek vektoru osy v čase t .

Pro potřeby této práce je plně dostačující pouze hodnota maximálního Lyapunova exponentu, který se získá jako limita posloupnosti lokálních Lyapunovových exponentů:

$$\lambda = \lim_{N \rightarrow \infty} \frac{1}{N} \sum_{n=1}^N \ln \left(\frac{dx_{n+1}}{dx_n} \right). \quad (3.5)$$

3.3 Fraktální dimenze

Všechny chaotické atraktory jsou zároveň fraktály. Dva atraktory se od sebe liší mimo jiné podle toho, jak hustě plochu zaplňují. Je to tím, že každý z nich má jinou fraktální dimenzi. Existuje hned několik specifických definic fraktální dimenze, žádná z nich by tedy neměla být chápána jako obecná [6].

Tuto fraktální dimenzi je možné spočítat přesně. V úvahu vezmeme dva jednoduché případy, v jednom jsou následné iterace rovnoměrně rozloženy podél diagonály stránky, ve druhém iterace zaplní celou plochu. V prvním případě je fraktální dimenze 1, ve druhém případě 2.

Jednou z možností, jak vypočítat fraktální dimenzi, je nakreslit někde na ploše malou kružnici, která bude obklopovat alespoň jeden bod. Potom nakreslíme druhou kružnici se shodným středem, ale dvojnásobným poloměrem. Spočítáme počet bodů uvnitř každé z kružnic. Řekněme, že uvnitř menší kružnice leží N_1 bodů a uvnitř větší N_2 . Je zřejmé, že N_2 je větší nebo rovno N_1 . Jestliže je mezi jednotlivými body velká vzdálenost, pak $N_1 = N_2$. Pokud body tvoří rovnou linii, větší kružnice obsahuje v průměru dvakrát více bodů, než menší kružnice. Jsou-li body částí plochy, pak větší kruh uzavře průměrně čtyřikrát více bodů, než menší – obsah kružnice je úměrný druhé mocnině poloměru. Pro tyto jednoduché případy je fraktální dimenze dána vztahem [1]:

$$L = \log_2 \frac{N_2}{N_1}. \quad (3.6)$$

Podobnou a velmi často používanou metodou je metoda počítání čtverců (Box-Counting Method), která umožňuje stanovení závislosti počtu čtverců pokrývajících objekt (vyplněnou plochu) na jejich velikosti [14].

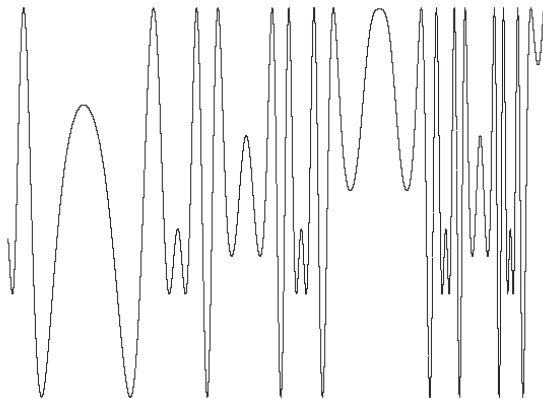
Lyapunovův exponent a fraktální dimenze nejsou zcela nezávislé. Usuzuje se, že vztah mezi fraktální dimenzí a dvěma Lyapunovými exponenty ve dvou dimenzích je

$$F = 1 - \frac{\lambda_1}{\lambda_2}, \quad (3.7)$$

kde λ_1 je větší ze dvou Lyapunovův exponentů [1]. Za předpokladu, že je znám i menší Lyapunovův exponent, může být tento vztah použit k výpočtu takzvané *Lyapunovy dimenze*.

3.4 Jednoduché atraktory

Tato část se zabývá jednoduchými rovnicemi s jednou proměnnou, které mají chaotické řešení. Tímto řešením jsou části křivek, které mohou být ve výsledku i velmi složité. Příklad takové křivky ukazuje obrázek 3.2.



Obrázek 3.2: jednodimenzionální kubická mapa

Nejobecnější jednorozměrná kvadratická iterační mapa je

$$x_{n+1} = a_1 + a_2 \cdot x_n + a_3 \cdot x_n^2, \quad (3.8)$$

kde a_1 , a_2 a a_3 jdou *kontrolní parametry*. Prohledáváním všech kombinací je možno získat všechna chaotická řešení, které rovnice má.

Mohlo by se zdát, že x_0 je čtvrtým kontrolním parametrem, ale řešením chaotického systému je obecně chaotický atraktor a žádná z počátečních podmínek spadající do oblasti přitažení se ani po mnoha iteracích nemění. Není ale žádná záruka, že volba hodnoty x_0 leží uvnitř oblasti přitažení, ovšem pro hodnoty x_0 blízké nule je přibližně poloviční pravděpodobnost, že budou ležet v oblasti přitažení. Vzhledem k tomu, že existuje obrovské množství chaotických řešení v rozsahu zbývajících tří parametrů a_1 , a_2 a a_3 , vyřazením poloviny z nich volbou hodnoty x_0 nebude pestrost produkovaných atraktorů příliš negativně poznamenána.

3.5 Atraktory v rovině

Narozdíl od předchozí části, kde byly pomocí jednodimenzionální chaotické funkce tvořeny části křivek, tato část se zabývá dvoudimenzionálními mapami, jejichž grafem jsou části ploch a které produkují o dost zajímavější grafické výstupy. Takto vzniklé atraktory již mohou být považovány za umění.

Dříve uvedené iterační funkce zahrnovaly jedinou proměnnou x , jejíž hodnota se měnila s každou iterací. Zajímavější je ale situace, kdy iterační mapa zahrnuje proměnné dvě, x a y . V tomto případě je s každou iterací získán nový bod na ploše, kde x představuje horizontální souřadnici a y vertikální souřadnici. Při více iteracích začnou tyto body vyplňovat části plochy.



Obrázek 3.3: Hénonův atraktor pro $a = 1.4$ a $b = 0.3$

Nejznámější chaotickou dvoudimenzionální mapou je *Hénonova mapa*:

$$\begin{aligned}x_{n+1} &= 1 + a \cdot x_n^2 + b \cdot y_n, \\ y_{n+1} &= x_n,\end{aligned}\tag{3.9}$$

kde a a b představují kontrolní parametry.

Na obrázku 3.3 je příklad Hénonovy mapy pro parametry $a = 1.4$ a $b = 0.3$, který uvádí snad každá správná literatura zabývající se atraktory, v této práci tedy nemůže chybět.

3.6 Atraktory v prostoru

Výše popsané techniky je možné jednoduše rozšířit tak, aby produkovaly trojrozměrné atraktory, které mají stejně jako šířku a výšku také hloubku. Zobrazování takového atraktoru je ale limitováno dvojrozměrností počítačové obrazovky nebo stránky papíru – hlavním úkolem se stává, jak takový trojrozměrný objekt vůbec zobrazit.

Nejjednodušší způsob jak získat trojrozměrný chaotický atraktor je ze vztahu:

$$\begin{aligned}
 x_{n+1} &= a_1 + a_2 \cdot x_n + a_3 \cdot x_n^2 + a_4 \cdot x_n \cdot y_n + a_5 \cdot x_n \cdot z_n + a_6 \cdot y_n \\
 &\quad + a_7 \cdot y_n^2 + a_8 \cdot y_n \cdot z_n + a_9 \cdot z_n + a_{10} \cdot y_n^2, \\
 y_{n+1} &= a_{11} + a_{12} \cdot x_n + a_{13} \cdot x_n^2 + a_{14} \cdot x_n \cdot y_n + a_{15} \cdot x_n \cdot z_n + a_{16} \cdot y_n \\
 &\quad + a_{17} \cdot y_n^2 + a_{18} \cdot y_n \cdot z_n + a_{19} \cdot z_n + a_{20} \cdot y_n^2, \\
 z_{n+1} &= a_{21} + a_{22} \cdot x_n + a_{23} \cdot x_n^2 + a_{24} \cdot x_n \cdot y_n + a_{25} \cdot x_n \cdot z_n + a_{26} \cdot y_n \\
 &\quad + a_{27} \cdot y_n^2 + a_{28} \cdot y_n \cdot z_n + a_{29} \cdot z_n + a_{30} \cdot y_n^2,
 \end{aligned} \tag{3.10}$$

ve kterém se nachází 30 koeficientů $a_1, a_2, a_3, \dots, a_{30}$. Díky takovému počtu koeficientů je možno získat velmi rozmanité spektrum atraktorů.

Dále je řešena otázka zobrazení objektu složeného z bodů ve třech dimenzích. Nejjednodušší je jednu ze souřadnic ignorovat a vykreslovat jen zbývající dvě. Dalšími způsoby se podrobněji zabývají další části práce.

3.7 Čtvrtá dimenze a hyperprostor

Ačkoliv většinou lidé o prostoru uvažují jako o třírozměrném, matematika třemi dimenzemi omezená není. Chaotické atraktory lze rozšířit i do čtvrté dimenze a dokonce do dimenzí vyšších.

Se čtvrtou dimenzí souvisí pojem *hyperprostor*, který označuje obecně n -rozměrný prostor – například *hyperkostka* je rozšíření obyčejné trojrozměrné kostky do čtyřech dimenzí. Má 16 vrcholů, 32 hran a 24 stěn.

Je-li k dispozici atraktor ve čtyřech dimenzích, lze tuto čtvrtou dimenzi použít jako další proměnnou používanou v nějaké vykreslovací metodě a zkombinovat několik různých metod dohromady.

Kapitola 4

Vykreslování atraktorů

Základní vykreslování je velmi jednoduché: je zvolena nějaká chaotická funkce, která je v každém kroku vykreslování použita pro výpočet nových souřadnic. Za počáteční hodnoty x , y a z jsou zvoleny hodnoty blízké nule. První vypočítané body je vhodné ignorovat a počkat, až se systém ustálí, teprve potom začít s vykreslováním.

Tento postup můžeme zapsat následujícím kódem:

```
for (i = 0, i < MAX_ITERATIONS, i++)
{
    (x,y,z) = f(x,y,z);
    if (i > SETTLE_ITERATIONS)
        plot(x,y,z);
}
```

například pro Lorenzův atraktor je $f(x, y, z)$

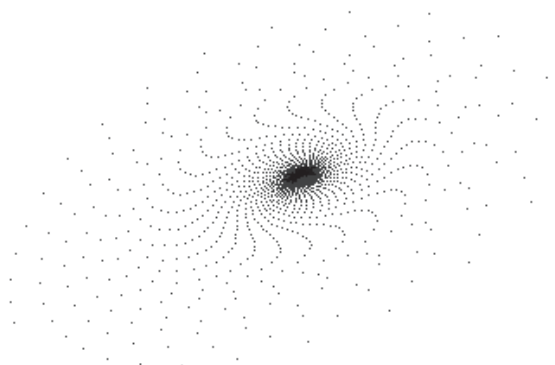
```
f(x,y,z)
{
    return (x + dT * A * (y - x)),
           y + dT * (x * (B - z) - y),
           z + dT * (x * y - C * z));
}
```

Aby mohl být výsledný atraktor vhodně umístěn na plátno, je potřeba zjistit ještě před vykreslováním hodnoty x_{min} , x_{max} , y_{min} a y_{max} . Pomocí nich pak mohou být atraktory posunouty a zmenšeny na odpovídající velikost plátna.

4.1 Nalezení chaotického řešení a náhodný atraktor

Jen malá část ze všech možných kombinací hodnot parametrů atraktoru produkuje skutečně chaotické řešení. Takové atraktory nejsou zdaleka tak vizuálně zajímavé, jako ty chaotické – na obrázku 4.1 je uveden příklad bodového atraktoru, což je případ atraktoru, ve kterém je každá počáteční hodnota po opakované iteraci vždy přitahována k nějakému pevnému bodu – podobně jako v příkladu s kuličkou uvedeném v části 3.1. Proto je velmi užitečné umět podstatnou část řešení vedoucích k nechaotickému atraktoru odhalit, některé třeba i po několika málo iteracích. K tomuto účelu je využit Lyapunovův exponent (λ), který byl již podrobněji popsán v části 3.2:

- $\lambda > 0$ systém diverguje – je nestabilní, což je dobrý předpoklad pro vznik chaosu
- $\lambda = 0$ stabilní systém
- $\lambda < 0$ periodický atraktor



Obrázek 4.1: bodový atraktor

Postup hledání chaotického řešení, který používá přiložený program vypadá následovně:

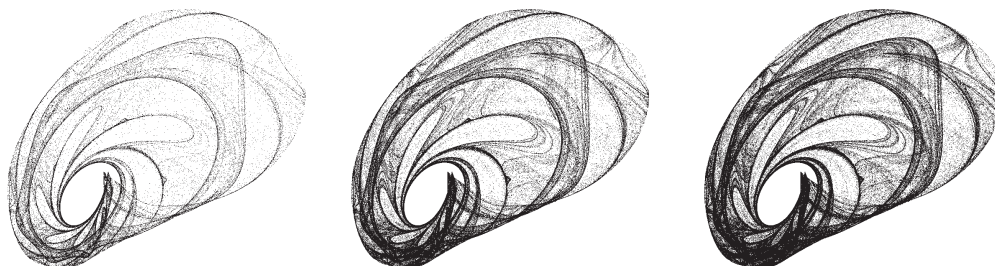
- Za hodnoty x_0 a y_0 , případně z_0 jsou dosazena čísla blízká nule a začneme iterovat přes vybranou chaotickou funkci.
- Pokud výsledek během výpočtu přesáhne nějaké extrémně velké číslo, je výsledný atraktor pravděpodobně neohraničený.
- Pokud změna polohy bodu určeného souřadnicemi x , y , z je velmi malá (téměř nulová), jde s největší pravděpodobností o bodový atraktor.
- Když se během výpočtu Lyapunovův exponent stane záporným nebo je to velmi malé číslo, výsledek je pravděpodobně pevný bod nebo limitní cyklus.
- Pokud je (po několika tisících iteracích) Lyapunovův exponent stále kladný, pak byl pravděpodobně nalezen chaotický atraktor.

Je-li cílem vygenerovat náhodný atraktor, k výběru parametrů atraktoru může být využit generátor (pseudo)náhodných čísel, který většina programovacích jazyků poskytuje. Je vygenerováno tolik náhodných čísel, kolik je parametrů; tato čísla musí pochopitelně být ve vhodném rozsahu. Začneme iterovat přes danou rovnici a pokud je během výpočtu zjištěno, že řešení není chaotické, vygenerujeme nové náhodné parametry a postup opakujeme, dokud nenajdeme nějaké chaotické řešení.

Podle testu provedeného na Lorenzově chaotické funkci v programu, který je součástí práce, bylo přibližně 1 % atraktorů chaotických, dále 74 % atraktorů neohraničených, 20 % bodových, 3 % stabilních a 2 % periodických. Je tedy patrné, že naprostou většinu tvoří neohraničené a bodové atraktory, k jejichž detekci nepotřebujeme počítat Lyapunovův exponent. Hledání náhodného chaotického řešení se tak výrazně urychlí, pokud nám ovšem nevadí, že teprve zhruba každý šestý atraktor bude chaotický. Detekovat nekonečný atraktor se ve většině případů podařilo již během prvních deseti až třiceti iterací, oproti tomu k rozpoznání bodového atraktoru často nestačilo ani deset tisíc iterací.

4.2 Stupně šedi

Během vykreslování může jedním pixelem dráha atraktoru projít vícekrát. Tento fakt můžeme ignorovat a atraktor vykreslovat jen jednou barvou, výsledky ale nebudou z estetického hlediska příliš uspokojivé – viz obrázek 4.2.



Obrázek 4.2: Lorenzův atraktor po 10^5 / 5×10^5 / 10^6 iteracích

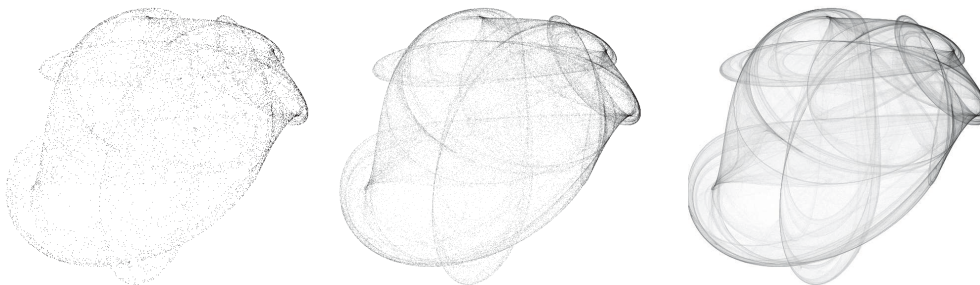
Při každém „zasazení“ pixelu můžeme zvýšit jeho intenzitu o nějakou konstantní hodnotu. Při použití této metody je ale potřeba zvolit vhodný poměr mezi konstantním přírůstkem a počtem iterací. Jinak se může stát, že se intenzita pixelu dostane za hranici zobrazení a na obrázku začnou vznikat místa bez kresby (ve fotografii takzvané *přepaly*), v opačném případě zase bude obrázek příliš nevýrazný, jak ukazuje obrázek 4.3.

Abychom se vyhnuli těmto nepříjemným efektům, budeme výsledné intenzity pixelů normalizovat do rozsahu zobrazení tak, aby na výsledném obrázku barva s maximální intenzitou reprezentovala čistě bílou (při vykreslování na černé pozadí) a minimální intenzitu bude představovat barva pozadí. Rozložení odstínů bude tedy pořád stejné, nebude záviset na počtu provedených iterací. Se zvyšujícím se počtem vykreslených bodů

se bude měnit pouze kvalita obrázku, začnou vznikat nové detaily a bude se postupně ztrácet zrnitost, jak je ukázáno na obrázku 4.4.



Obrázek 4.3: Lorenzův atraktor po 10^6 / 10^7 / 10^8 iteracích



Obrázek 4.4: Atraktor po 10^4 / 10^5 / 10^7 iteracích po provedení normalizace

4.3 Obarvování atraktorů

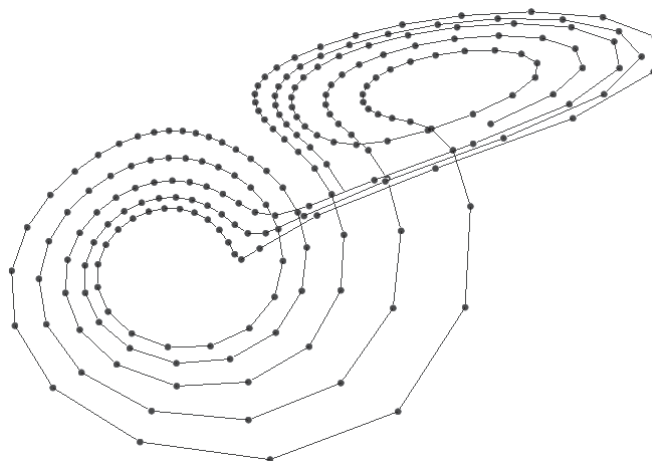
Protože tato část již bude pracovat s barvami, které jsou reprezentovány více barevnými složkami a jednotlivé body atraktoru budou moci nabývat různých barev, nemůžeme k vykreslování dále používat metodu přičítání konstantních přírůstků popsanou v části 4.2.

Řešením může být takzvaný *alfa blending*. Jedná se o konvexní kombinaci dvou barev umožňující v počítačové grafice vytvářet efekty jako průhlednost. Hodnota alfa nabývá hodnot 0,0 až 1,0. Nejnížší hodnota reprezentuje plně transparentní barvu, nejvyšší naopak barvu se stoprocentním krytím. Máme-li barvu $value_1$ s alfa hodnotou α a chceme ji vykreslit na neprůhledné pozadí s barvou $value_0$, bude výsledná barva dána vztahem

$$value = (1 - \alpha) \cdot value_0 + \alpha \cdot value_1. \quad (4.1)$$

V případě vykreslování jednotlivých bodů atraktoru na pozadí volíme za hodnotu α nějaké malé číslo.

K jednoduchému barevnému přechodu můžeme použít jednu ze souřadnic x , y nebo z . Pro barevný přechod mezi barvami c_1 a c_2 a jeho mapování na souřadnici z bude platit,



Obrázek 4.5: Lorenzův atraktor po prvních 300 iteracích se zvýrazněním spojníc mezi body jdoucími po sobě

že $color(x, y, z_{max}) = c_1$ a $color(x, y, z_{min}) = c_2$. Pak například pro střed atraktoru podle osy z bude platit

$$color(x, y, \frac{z_{max} + z_{min}}{2}) = \frac{c_1 + c_2}{2}. \quad (4.2)$$

Na obrázku 4.5 vidíme, jak se v čase mění vzdálenost mezi jednotlivými body. Tomuto jevu můžeme říkat rychlost dráhy atraktoru – v místech, kde jsou jednotlivé body blízko sebe je rychlost malá, u vzdálenějších bodů je naopak velká. Vzdálenost mezi dvěma po sobě jdoucími body můžeme použít podobně jako v předchozím příkladě k mapování na barvy z gradientu. Aktuální rychlost vypočítáme jako vzdálenost dvou bodů v prostoru:

$$velocity = \sqrt{(x_n - x_{n+1})^2 + (y_n - y_{n+1})^2 + (z_n - z_{n+1})^2} \quad (4.3)$$

Je vhodné si ještě před začátkem obarvování atraktoru spočítat největší vzdálenost mezi dvěma body v obarvovaném atraktoru (tedy největší rychlost), která bude dále vyjadřovat maximální barvu z gradientu. Podobně můžeme učinit i pro nejmenší vzdálenost, nebo pro jednoduchost minimální barvu z gradientu mapovat na hodnotu 0.

Dalším způsobem obarvování bodů atraktoru je podle úhlu mezi třemi po sobě jdoucími body. Barva bodu $[x_n, y_n, z_n]$ bude určena úhlem α , pro který bude platit

$$\cos \alpha = \frac{a^2 + c^2 - b^2}{2 \cdot a \cdot c}, \quad (4.4)$$

kde a je vzdálenost mezi body $[x_{n-1}, y_{n-1}, z_{n-1}]$ a $[x_n, y_n, z_n]$, b je vzdálenost mezi body $[x_{n-1}, y_{n-1}, z_{n-1}]$ a $[x_{n+1}, y_{n+1}, z_{n+1}]$ a c je vzdálenost mezi body $[x_n, y_n, z_n]$ a $[x_{n+1}, y_{n+1}, z_{n+1}]$.

Barevný přechod můžeme mapovat do rozsahu $0^\circ - 180^\circ$, nezobrazíme tak ale všechny barvy z barevného přechodu. U některých atraktorů, jako je například Lorenzův atraktor

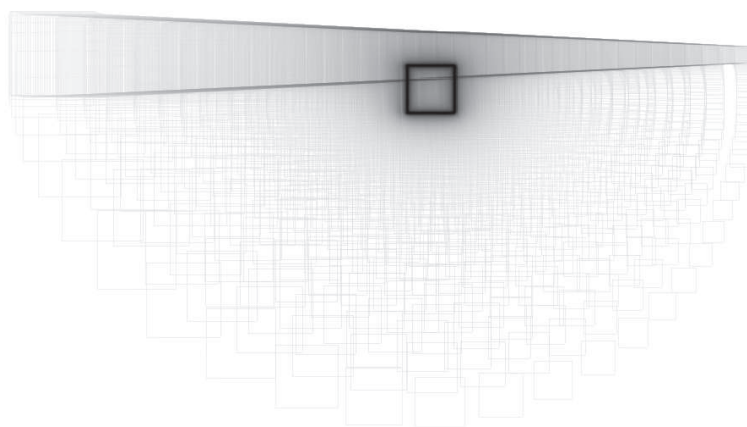
z obrázku 4.5, se nevyskytují žádné ostré úhly, v jiných jsou naopak všechny úhly ostré. Podobně jako v předchozím případě je tedy vhodné ještě před začátkem obarvování vypočítat maximální a minimální hodnoty úhlů, které se v atraktoru vyskytují.

4.4 Rozmísťování dvojrozměrných objektů

Až dosud byla pozornost věnována pouze metodám vykreslování, kde každý vygenerovaný bod byl vykreslen jako jeden pixel. Zajímavým rozšířením je nevykreslovat bod přímo na dané souřadnice, ale použít ho například jako střed nějakého grafického objektu. Tím může být kružnice, čtverec, nebo i libovolný další objekt.



Obrázek 4.6: Atraktor vykreslený metodou rozmísťování kružnic



Obrázek 4.7: Bodový atraktor vykreslený metodou rozmísťování čtverců

Při zadání velkého počtu iterací začnou vykreslované objekty tvořit souvislejší strukturu, jako na obrázku 4.6. Metoda může být také použita k „chaotickému“ rozmístění objektů po ploše – v tomto případě nevolíme příliš velký počet iterací. Tento způsob je mimo jiné vhodný pro vykreslování atraktorů, které jinak nepůsobí příliš atraktivně – i nechaotický atraktor (například atraktor bodový) může ve výsledku vypadat velmi zajímavě, jak ukazuje obrázek 4.7.

Vykreslování dvojrozměrných objektů je možno dále kombinovat s dříve popsányými metodami – změnami barev nebo i velikosti či otočením v závislosti na některé charakteristické hodnotě atraktoru.

4.5 Složitost

Rychlost, jakou lze chaotické atraktory generovat, hraje důležitou roli v případě, že nároky na kvalitu výstupního obrazu jsou velké a tedy nutný počet generovaných bodů je obrovský. Je zřejmé, že asymptotická složitost vlastního výpočtu atraktoru je lineární v počtu iterací – v každém kroku proběhne jednoduchý výpočet následníka aktuálního bodu.

V praxi je však doba výpočtu atraktoru zanedbatelná, neboť generované bodu musí být obvykle dále zpracovány (například vykresleny), což většinou trvá mnohem déle. Například přiložená aplikace stráví při výpočtu barvy pomocí úhlu mezi posledními třemi body zhruba dvakrát více času, než potřebuje na jejich vygenerování.

Toto dodatečné zpracování lze obvykle snadno paralelizovat, hrozba, že se vlastní generování stane úzkým hrdlem, tedy zůstává. Je tedy zcela na místě hledat i pro toto generování paralelní algoritmus. Je nutné si uvědomit, že přesné pozice vygenerovaných bodů nejsou tak důležité, jako jejich vizuální rozložení v prostoru. Proces tedy můžeme paralelizovat tak, že zvolíme dva různé, avšak blízké, počáteční body a sekvenčním algoritmem vygenerujeme dvě různé množiny bodů. Tyto množiny budou disjunktní, budou však svým rozložením vizuálně vyznačovat stejný atraktor. Analogicky můžeme úlohu rozložit na libovolný počet procesorů.

Atraktor může být výše uvedeným způsobem paralelně spočítán i přímo na některé z moderních grafických karet, nárůst výkonu by v takovém případě byl řádový.

Kapitola 5

Popis programu

Program aTraktor slouží k prohlížení a vykreslování chaotických atraktorů a také k jejich náhodnému generování. Okno s náhledem atraktoru umožňuje provádění transformací jako je otáčení, posun a přiblížení trojrozměrného atraktoru. V okně náhledu je možno nastavovat několik různých parametrů zobrazení, jako je velikost jednotlivých bodů nebo zobrazení spojnic mezi nimi. Náhled je dále možno vykreslit v několika různých režimech a uložit do souboru ve formátu PNG.

5.1 Použité programové prostředky

Program je vytvořen v jazyce C++, k jeho vývoji bylo použito prostředí Microsoft Visual Studio (verze 2005). Tento programovací jazyk byl zvolen proto, že je to kompilovaný jazyk umožňující silné optimalizace – generování jednotlivých bodů atraktoru v interpretovaném jazyce by trvalo příliš dlouho. Zároveň C++ ve své standardní knihovně nabízí vhodné datové struktury a poskytuje možnost objektového programování.

Kromě standardních knihoven je programový kód dále doplněn o další knihovny, které slouží ke specifickým účelům. Při výběru těchto knihoven byla jednou z hlavních priorit možná přenositelnost aplikace mezi různými platformami.

Nejdůležitější pro celou funkci programu je knihovna OpenGL (Open Graphics Library), která se stará o vykreslení základních grafických primitiv (bodů, úseček a mnohoúhelníků) do obrazového rámce (framebufferu) v různých režimech. Kromě OpenGL je dále použita její nadstavbová knihovna GLUT (OpenGL Utility Toolkit), která umožňuje činnosti, které nejsou kvůli snaze o zachování co největší platformní nezávislosti v OpenGL přímo podporovány. Základními takovými funkcemi je podpora pro práci s okny a zpracování událostí.

K tvorbě grafického uživatelského rozhraní byla použita knihovna Fast Light Toolkit (FLTK), která umožňuje velmi dobrou podporu dříve zmíněné knihovny GLUT a zároveň nabízí všechny potřebné komponenty uživatelského rozhraní, jako jsou například okna pro výběr barev.

Pro ukládání obrázků program využívá knihovnu libpng (původně nazývanou pnglib), která je oficiální knihovnou formátu PNG (Portable Network Graphics). Knihovna libpng

pro svou práci vyžaduje další knihovnu zlib užívanou pro kompresi dat. Formát PNG byl zvolen s ohledem na další možné zpracování výstupních obrázků v grafických aplikacích.

5.2 Uživatelské rozhraní aplikace

Grafické uživatelské rozhraní aplikace se skládá ze dvou hlavních částí: uživatelského menu a okna náhledu (viewportu). Menu umožňuje uživateli nastavovat parametry konkrétního atraktoru a způsob jeho zobrazení ve viewportu. Pomocí myši lze provádět transformace jako je rotace, posun a zvětšení či zmenšení atraktoru. Uživateli je dále umožněno používat několik přednastavených klávesových zkratk.

Viewport umožňuje přepínání mezi několika režimy zobrazení – nastavitelný je počet iterací (bodů) zobrazovaných v okně náhledu. Větší počet těchto iterací zvyšuje kvalitu zobrazeného náhledu, ale má větší nároky na výkon počítače. Toto nastavení tedy umožňuje zvolit vhodný poměr mezi rychlostí vykreslování náhledu a jeho kvalitou v závislosti na dostupných hardwarových možnostech. Postupné zvyšování počtu iterací také umožňuje zvědavému uživateli sledovat, jak se celý chaotický systém s přibývajícimi iteracemi vyvíjí.

S nastavením počtu iterací viewportu souvisí další položka menu *opacity*, která nastavuje alfa hodnotu zobrazovaných bodů. Čím je tato hodnota nižší, tím budou vykreslované body průhlednější. Při velkém počtu iterací je vhodné tuto hodnotu snížit.

Velikost zobrazovaných bodů je nastavitelná položkou *point size* a *point smooth* vypíná nebo zapíná jejich vyhlazování. Uživatel také může přepínat mezi perspektivní a ortogonální projekcí.

Dále je umožněno ve viewportu zobrazit spojnice mezi následujícími body a trojúhelník, jehož vrcholy jsou vždy určeny třemi po sobě jdoucími body. Tyto zobrazovací režimy umožňují demonstrovat metody použité pro obarvování bodů – *display joins* pro metodu, která body obarvuje podle jejich vzájemné vzdálenosti a *display triangles* pro metodu, která pro stanovení barvy používá velikost úhlu mezi třemi body (tedy úhel v zobrazeném trojúhelníku).

Na tomto místě je důležité upozornit, že nastavení viewportu, jako je velikost bodů a zobrazení jejich spojníc, nijak neovlivňuje vykreslený obrázek. Samotné renderování probíhá odlišným způsobem než zobrazování náhledu. Jediné nastavení viewportu, které má na vykreslování vliv, je zapnutí nebo vypnutí perspektivní projekce, která přímo ovlivňuje transformační matice používané při rasterizaci bodů.

Program dokáže zobrazovat několik typů atraktorů, které lze v menu vybírat. Konkrétně je to jedna polynomiální chaotická funkce, atraktory Ikeda, Lorenz-84, a Lorenzův, Cliffordův a Pickoverův atraktor. Chaotické funkce, které tyto typy atraktorů generují, jsou uvedeny v příloze B. Samozřejmě existuje celá řada dalších chaotických funkcí, zmíněných šest typů bylo vybráno pro vizuální rozmanitost produkovaných atraktorů a byl také brán ohled na jednoduchost uživatelského rozhraní – některé chaotické funkce vyžadují i více než 30 vstupních parametrů, jako například polynomiální funkce uvedená v kapitole 3.6.

Pro nastavování parametrů atraktoru program používá posuvník, kterým lze hodnoty nastavovat v rozsahu, který je pevně daný pro každý typ atraktoru.

Menu dále nabízí výběr metody vykreslování. V nabídce jsou metody *Basic*, *Smoke*, *Depth*, *Velocity*, *Angle*, *Circle* a *Square*. U metod umožňujících použití barevného přechodu je zobrazena paleta pro nastavení až šesti barev, u zbylých metod se nastavuje pouze jedna barva. Vykreslování atraktoru se zahájí tlačítkem *Render view*. Postup této rasterizace bude podrobně popsán v další kapitole.

5.3 Implementace vykreslovacích metod

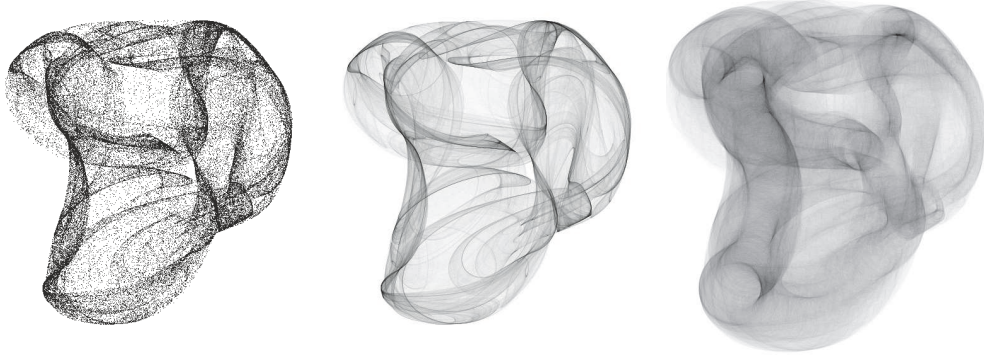
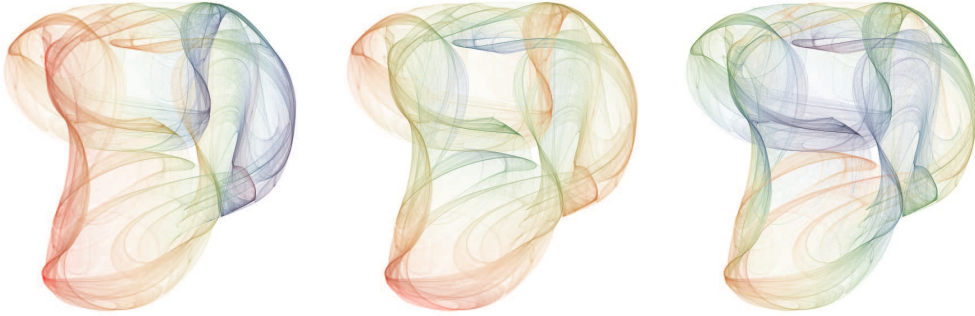
Před výstupem obrázku si uživatel zvolí parametry vykreslování, mezi něž patří velikost obrázku v pixelech, počet iterací, nastavení barev a barevných přechodů, metodu vykreslení jednotlivých bodů, koeficient gamma korekce a jméno výstupního souboru.

Během renderování se pomocí postupů uvedených v kapitole 4 postupně generují jednotlivé body ve fázovém prostoru. Na každý bod jsou uplatněny stejné transformace, jako na atraktor v náhledu, a je pomocí nich zobrazen do roviny. V závislosti na zvolené metodě je pak bodu přiřazena charakteristická hodnota, ze které se podle nastavení barevných přechodů vypočítá barva.

Zobrazený a obarvený bod je poté rasterizován v závislosti na zvolené metodě.

- V nastavení *Basic* se pixel na daných souřadnicích nastaví na zadanou barvu, vznikne tedy dvoubarevný obrázek. Výsledky nebudou příliš vizuálně atraktivní, jedná se však o nejrychlejší metodu.
- Nastavení *Smoke* překonává nedostatky předchozího popsaného nastavení a barvy v pixelech míchá standardním způsobem (alfa blending [9]). Výsledný atraktor potom působí poloprůhledně.
- Nastavení *Depth*, *Velocity* a *Angle* se chovají stejně jako předchozí popsaná metoda *Smoke*, barvy však zde nejsou konstantní, ale závisí na souřadnici z , okamžité rychlosti bodu, případně na poloměru křivosti, který odpovídá úhlu mezi předchozím, aktuálním a následujícím bodem.
- Body nemusíme vždy reprezentovat jen jedním pixelem. Metoda *Circle* reprezentuje vygenerovaný bod kružnicí se středem odpovídajícím souřadnicím bodu, a poloměrem daným souřadnicí z . K rasterizaci kružnice je použit modifikovaný Bresenhamův algoritmus (midpoint circle algorithm [7]). Ten pracuje jen s celočíselnou aritmetikou a je proto velmi rychlý. Rychlost je v tomto případě důležitá – k dosažení hezkého obrázku je často třeba vykreslit velký počet kružnic.
- Nastavení *Square* je podobné metodě *Circle* s tím rozdílem, že vygenerovaný bod je reprezentován čtvercem, jeho velikost opět závisí na souřadnici z . Vykreslením velkého počtu čtverců se na atraktoru vytvoří poměrně zajímavá textura.

Srovnání některých metod je uvedeno na obrázcích 5.1 a 5.2.

Obrázek 5.1: Srovnání atraktoru vykresleného metodou *Basic* / *Smoke* / *Circle*Obrázek 5.2: Srovnání atraktoru vykresleného metodou *Depth* / *Velocity* / *Angle* při stejném barevném přechodu

Po skončení renderování je obrázek normalizován způsobem popsáným v části 4.2. Dále je na výsledný obrázek aplikována gamma korekce podle vztahu

$$V_{out} = V_{in}^{\gamma}, \quad (5.1)$$

kde V_{in} je původní hodnota pixelu. Konstanta γ je volitelná uživatelem a je přednastavena na hodnotu 0,45; při tomto nastavením je obrázek na monitoru zobrazen lineárně – běžné monitory provádí korekci $\gamma = 2,2$. Výsledný obraz je po skončení renderování a provedení normalizace a korekce uložen ve formátu PNG v režimu RGB s barevnou hloubkou 16 bitů (bez průhlednosti).

Pro některé popsané metody byl proveden srovnávací test jejich rychlosti. Doba, kterou trvalo provést 5×10^7 iterací při použití jednotlivých vykreslovacích metod byla následující:

- *Basic* – 9.883335ms
- *Depth* – 15.286986ms
- *Velocity* – 18.207331ms
- *Angle* – 27.854073ms

Test byl proveden na procesoru Intel Core 2 Duo E8400, 3.0 GHz; při výpočtu se používalo jen jedno jádro procesoru. Výsledky nejsou nijak překvapivé – metoda *Basic* nepracuje s žádným barevným přechodem, je proto nejrychlejší. Metoda *Depth* ke stanovení barvy bodu nemusí provádět žádné speciální výpočty (souřadnice z je vypočítána vždy bez ohledu na použitou vykreslovací metodu), oproti metodě *Basic* ale pro souřadnici z stanovuje odpovídající barvu z barevného přechodu. Metoda *Velocity* musí při každé iteraci vypočítat rychlost aktuálního bodu, je tedy o něco pomalejší. Metoda *Angle* má v tomto srovnání největší výpočetní nároky, pro stanovení velikosti aktuálního úhlu musí totiž nejprve zjistit vzájemné vzdálenosti mezi třemi body v prostoru.

Kapitola 6

Závěr

Hlavní součást práce, program „aTraktor“, produkuje kvalitní grafické výstupy a byl dokončen do formy, která je použitelná pro tvůrčí generování chaotických atraktorů – ať už jako samostatného výtvarného díla, nebo jen jako základ pro další zpracování. Umožňuje volbu různých vykreslovacích metod, při jejichž implementaci byl větší důraz kladen na kvalitu výsledných obrázků, než na počet možných způsobů vykreslování. Přesto aplikace umožňuje velkou variabilitu výstupů.

Program ještě zdaleka nevyčerpal veškeré možnosti, které chaotické atraktory nabízejí; aplikace je ale připravena pro další rozšiřování. Novou vykreslovací metodou by mohla například být možnost propojení jednotlivých bodů do dvojrozměrné, nebo i prostorové křivky s následným vektorovým výstupem. Zajímavé by také bylo využít možnosti použitého výstupního formátu PNG v podobě vícebitové průhlednosti. Atraktor by se mohl vykreslovat na průhledné pozadí, což by přineslo širší možnosti dalšího zpracování – například jednoduché použití při tvorbě různých koláží.

Přestože byla podstatná část času při vývoji programu věnována návrhu multiplatformního uživatelského rozhraní, je v tomto směru ještě co zlepšovat. Program by dále mohl nabízet funkci uložení parametrů atraktoru i jejich zpětné načtení. Velmi uživatelsky příjemná by byla možnost předčasného přerušení renderování, případně zobrazení jeho průběhu v podobě progress baru.

Literatura

- [1] SPROTT, J.C. *Strange Attractors: Creating Patterns in Chaos*. M & T Books, New York, 2003. ISBN 1-55851-298-5.
- [2] PICKOVER, C.A. *Computers, Pattern, Chaos, and Beauty*. Dover Publications, 2001. ISBN 0486417093.
- [3] BOURKE, P. *Random Attractors Found Using Lyapunov Exponents*, [online]. [citováno 2009-05-23]. <<http://local.wasp.uwa.edu.au/~pbourke/fractals/lyapunov/>>
- [4] *Chaos theory*, [online]. Wikipedia, The Free Encyclopedia, c2008 [citováno 2008-11-16]. <http://en.wikipedia.org/wiki/Chaos_theory>.
- [5] *Lorenz attractor*, [online]. Wikipedia, The Free Encyclopedia, c2008 [citováno 2008-11-16]. <http://en.wikipedia.org/wiki/Lorenz_attractor>.
- [6] *Fractal dimension*, [online]. Wikipedia, The Free Encyclopedia, c2008 [citováno 2008-11-16]. <http://en.wikipedia.org/wiki/Fractal_dimension>.
- [7] *Midpoint circle algorithm*, [online]. Wikipedia, The Free Encyclopedia, c2008 [citováno 2009-05-23]. <http://en.wikipedia.org/wiki/Midpoint_circle_algorithm>.
- [8] *Gamma correction*, [online]. Wikipedia, The Free Encyclopedia, c2008 [citováno 2009-05-23]. <http://en.wikipedia.org/wiki/Gamma_correction>.
- [9] *Alpha compositing*, [online]. Wikipedia, The Free Encyclopedia, c2008 [citováno 2009-05-22]. <http://en.wikipedia.org/wiki/Alpha_compositing>.
- [10] *Fraktály v počítačové grafice III*, [online]. 2005-11-09 [citováno 2008-11-16]. <<http://www.root.cz/clanky/fraktaly-v-pocitacove-grafice-iii>>.
- [11] *Jean-Pierre Hébert: Mac-Controlled Algorithmic Art*, [online]. [citováno 2009-05-23]. <<http://www.apple.com/science/profiles/hebert/>>.
- [12] *Gram-Schmidt Orthonormalization*, [online]. Wolfram MathWorld [citováno 2009-05-21]. <<http://mathworld.wolfram.com/Gram-SchmidtOrthonormalization.html>>.

- [13] FRANTÍK, P. *Spektrum Ljapunovových exponentů pro vybraný atraktor*, [online]. [citováno 2009-05-16]. <http://kitnarfovo.misto.cz/_MAIL_/htm/le/le.htm>.
- [14] ZMEŠKAL, O. – NEŽÁDAL, M. – BUCHNÍČEK, M. – FEDÁK, J. *Fraktální analýza tiskových struktur*, [online]. [citováno 2009-05-18]. <www.fch.vutbr.cz/lectures/imagesci/download/cz02_brat00.pdf>

Příloha A

Obsah CD

Součástí této práce je také přiložené CD, obsahující:

- text práce ve formátu PDF,
- zdrojové kódy programu aTraktor,
- spustitelnou aplikaci aTraktor,
- ukázky chaotických atraktorů vytvořených pomocí aplikace aTraktor.

Příloha B

Použité chaotické funkce

V programu aTraktor byly použity následující chaotické funkce:

Lorenz:

$$\begin{aligned}x_{n+1} &= x_n + dT \cdot a \cdot (y_n - x_n) \\y_{n+1} &= y_n + dT \cdot (x_n \cdot (b - z_n) - y_n) \\z_{n+1} &= z_n + dT \cdot (x_n \cdot y_n - c \cdot z_n)\end{aligned}\tag{B.1}$$

Lorenz-84:

$$\begin{aligned}x_{n+1} &= x_n + dT \cdot (-a \cdot x_n - y_n \cdot y_n - z_n \cdot z_n + a \cdot f) \\y_{n+1} &= y_n + dT \cdot (-y_m + x_n \cdot y_n - b \cdot x_n \cdot z_n + g) \\z_{n+1} &= z_n + dT \cdot (-z_n + b \cdot x_n \cdot y_n + x_n \cdot z_n)\end{aligned}\tag{B.2}$$

Polynomial:

$$\begin{aligned}x_{n+1} &= p_0 + y_n - z_n \cdot y_n \\y_{n+1} &= p_1 + z_n - x_n \cdot z_n \\z_{n+1} &= p_2 + x_n - y_n \cdot x_n\end{aligned}\tag{B.3}$$

Pickover:

$$\begin{aligned}x_{n+1} &= \sin(a \cdot y_n) - z_n \cdot \cos(b \cdot x_n) \\y_{n+1} &= z_n \cdot \sin(c \cdot x_n) - \cos(d \cdot y_n) \\z_{n+1} &= \sin(x_n)\end{aligned}\tag{B.4}$$

Clifford:

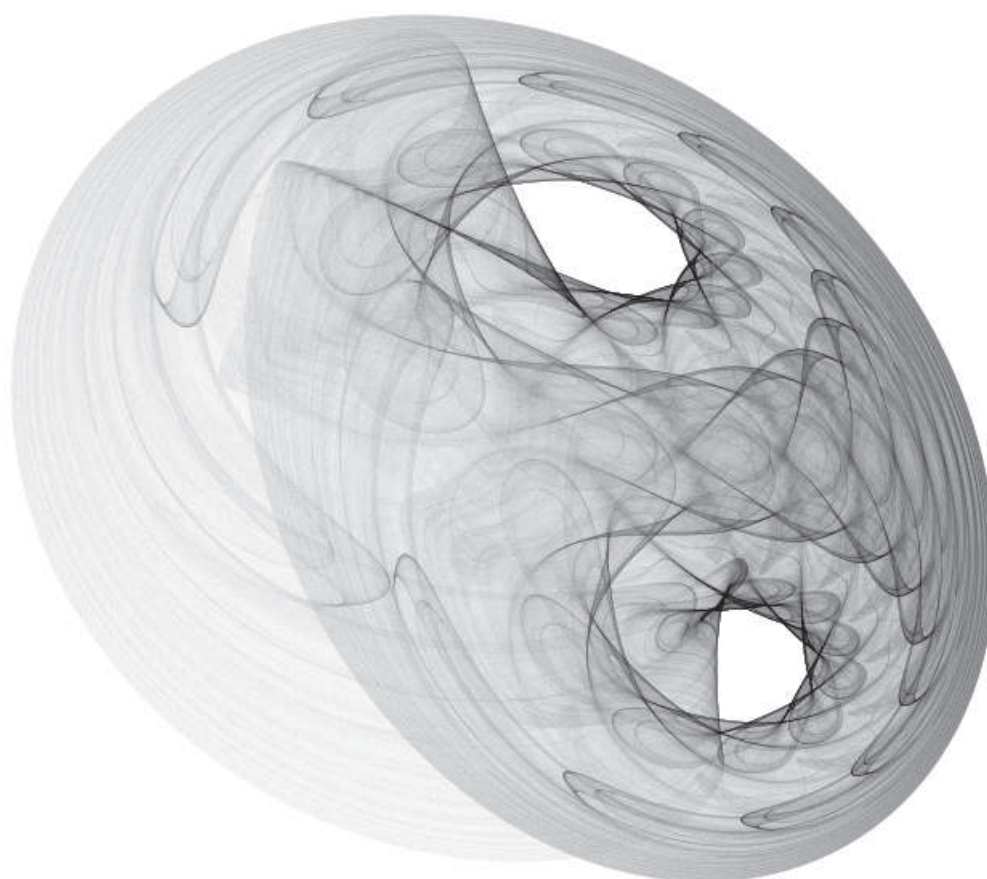
$$\begin{aligned}x_{n+1} &= \sin(a \cdot y_n) + c \cdot \cos(a \cdot x_n) \\y_{n+1} &= \sin(b \cdot z_n) + d \cdot \cos(b \cdot y_n) \\z_{n+1} &= \sin(c \cdot x_n) + a \cdot \cos(c \cdot z_n)\end{aligned}\tag{B.5}$$

Ikeda:

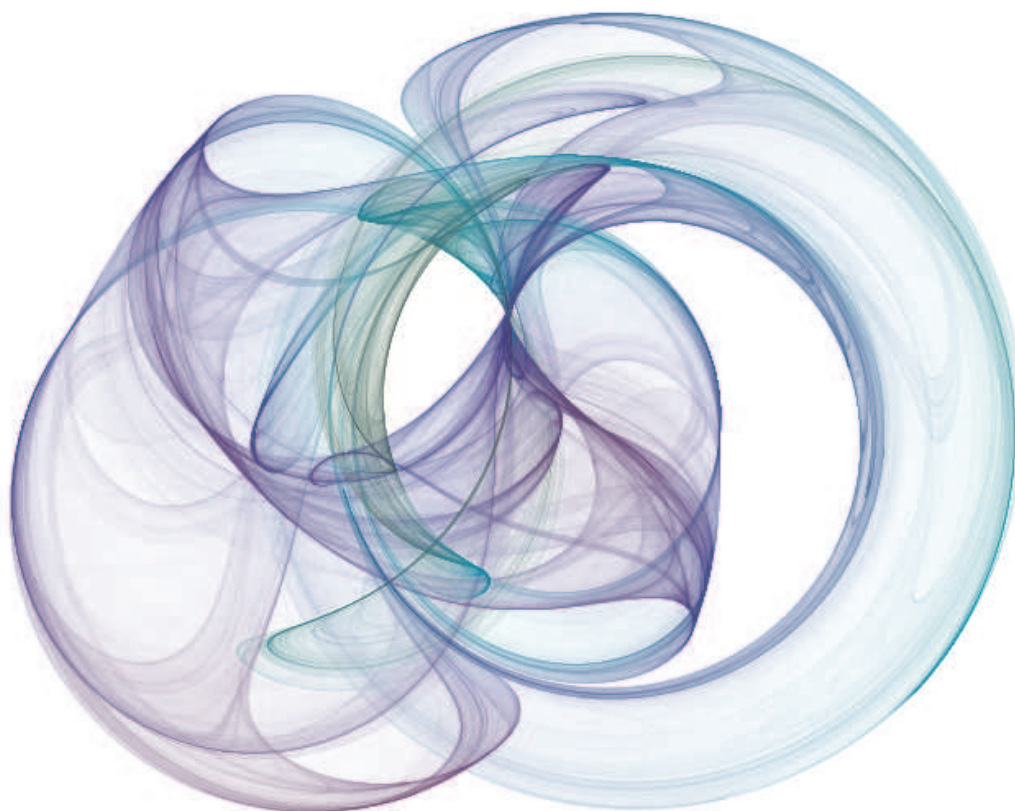
$$\begin{aligned}x_{n+1} &= a + b \cdot (x_n \cdot \cos(z_n) - y_n \cdot \sin(z_n)) \\y_{n+1} &= b \cdot (x_n \cdot \sin(z_n) + y_n \cdot \cos(z_n)) \\z_{n+1} &= c - d/(1 + x_n \cdot x_n + y_n \cdot y_n)\end{aligned}\tag{B.6}$$

Příloha C

Ukázky grafických výstupů



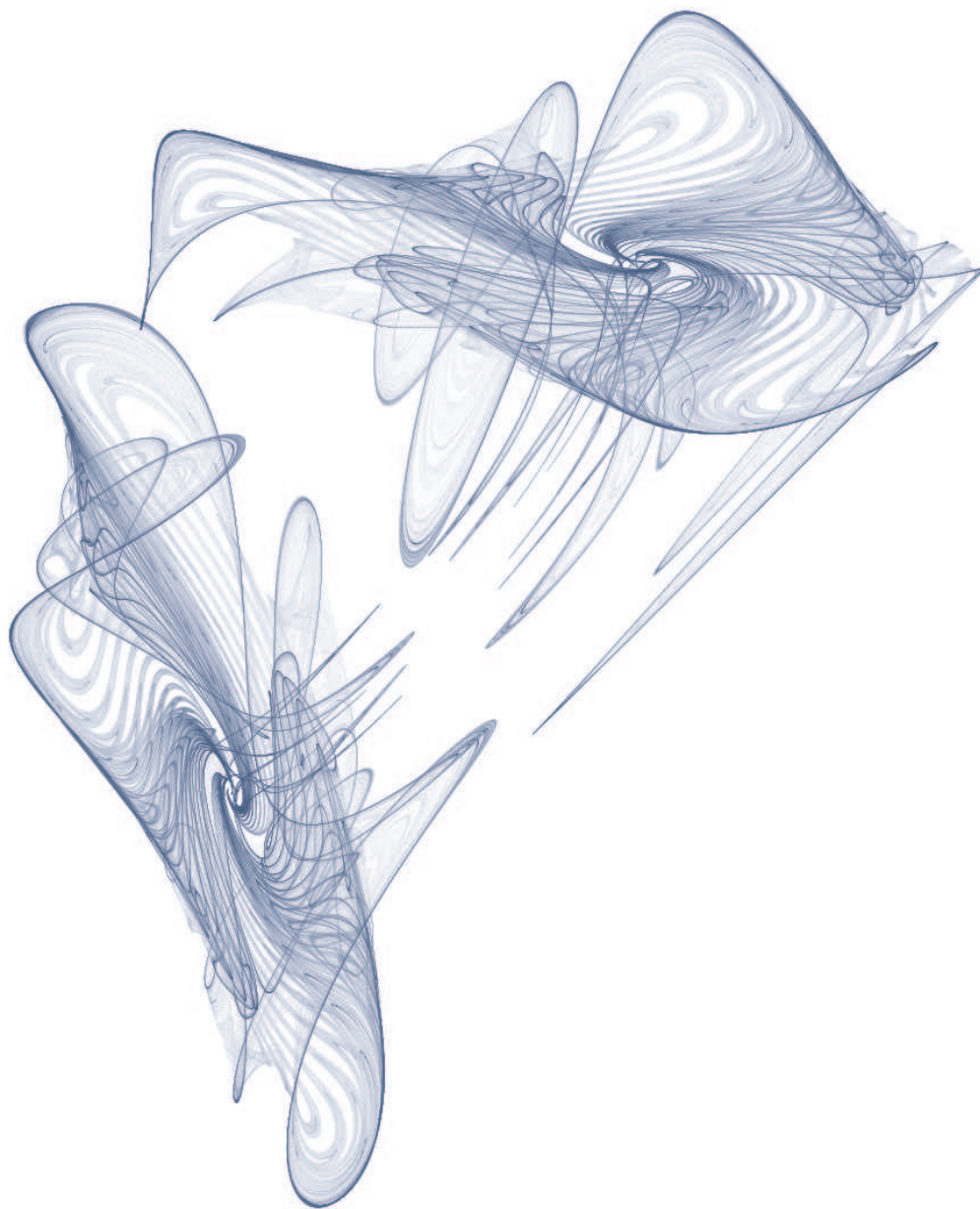
Obrázek C.1: Atraktor typu *Lorenz* s parametry $a = 8.45$, $b = 6.25$, $c = 2.35$, $dT = 0.14$ vykreslený metodou *Smoke*.



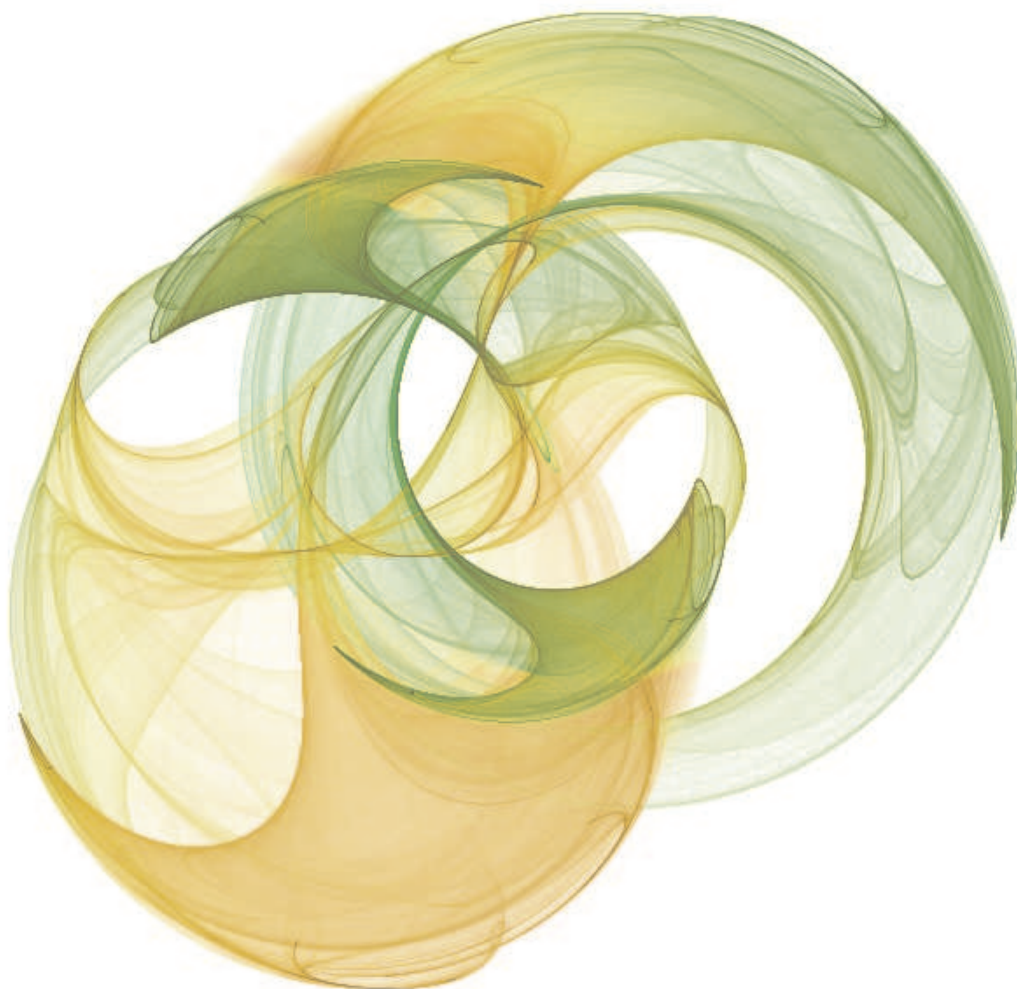
Obrázek C.2: Atraktor typu *Ikeda* s parametry $a = 0.55$, $b = 0.97$, $c = 7.74$, $d = 7.28$ vykreslený metodou *Angle* s dvoubarevným přechodem.



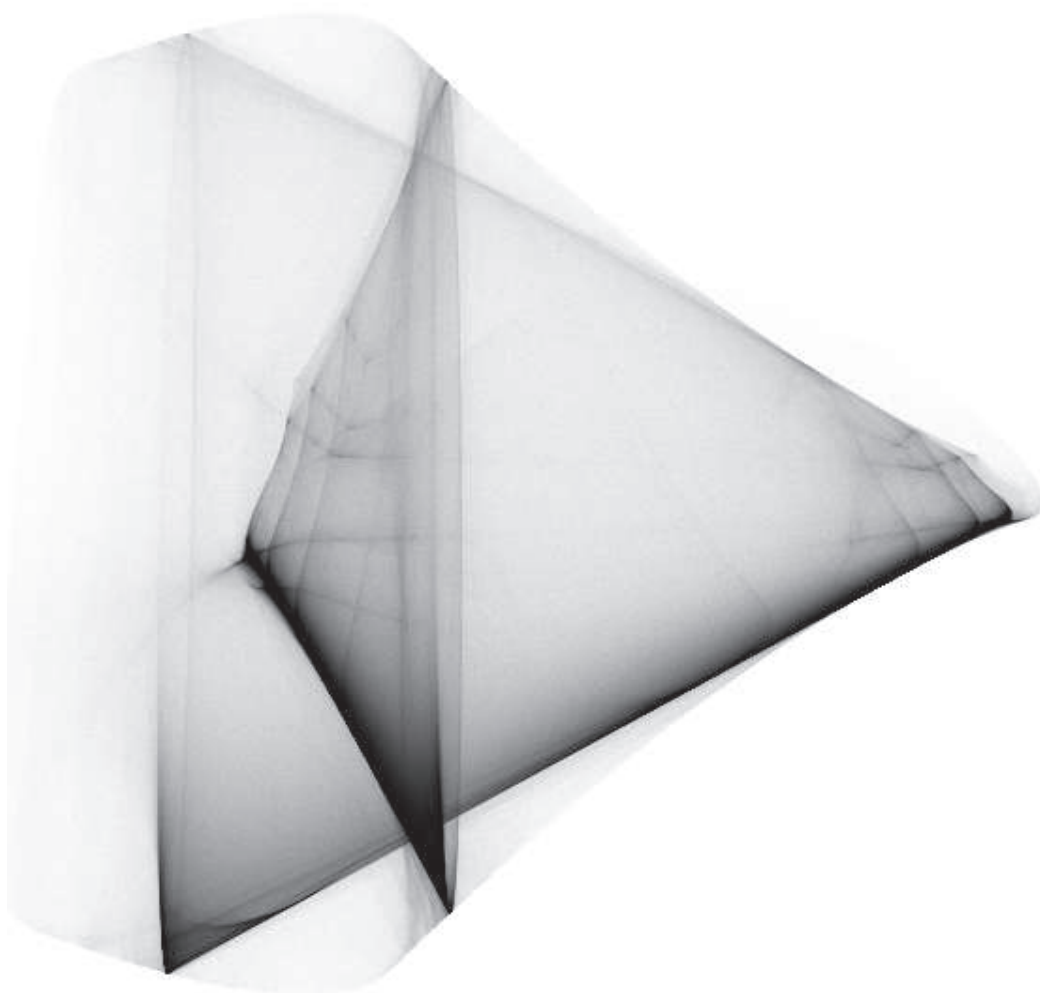
Obrázek C.3: Atraktor typu *Clifford* s parametry $a = -1.63$, $b = 1.25$, $c = -0.70$, $d = 1.64$ vykreslený metodou *Depth* s použitím šesti barev.



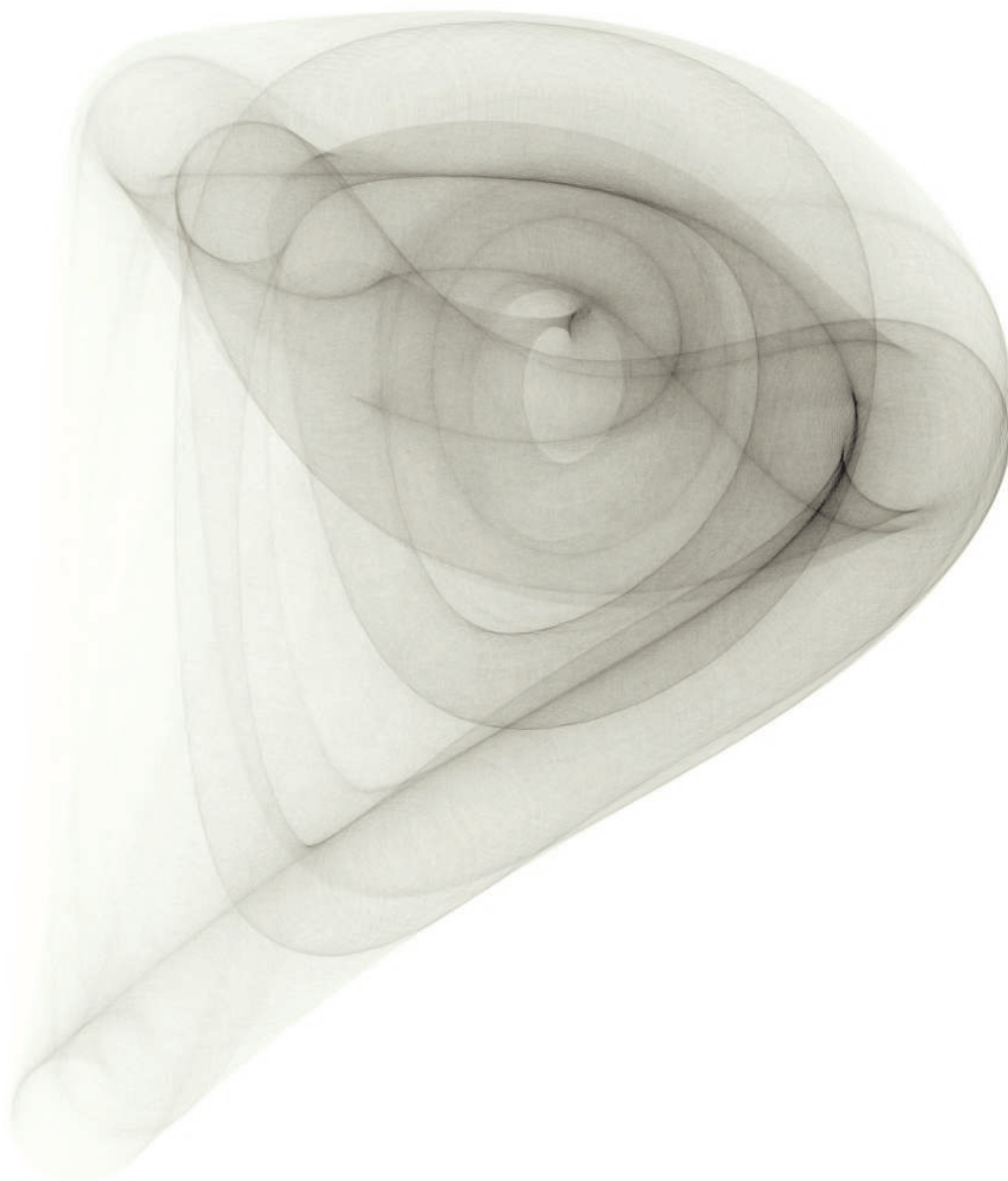
Obrázek C.4: Atraktor typu *Polynomial* s parametry $p_0 = 1.12$, $p_1 = 0.22$, $p_2 = 1.61$ vykreslený metodou *Smoke*.



Obrázek C.5: Atraktor typu *Ikeda* s parametry $a = 2.99$, $b = 0.81$, $c = 4.22$, $d = 7.39$ vykreslený metodou *Angle* s dvoubarevným přechodem.



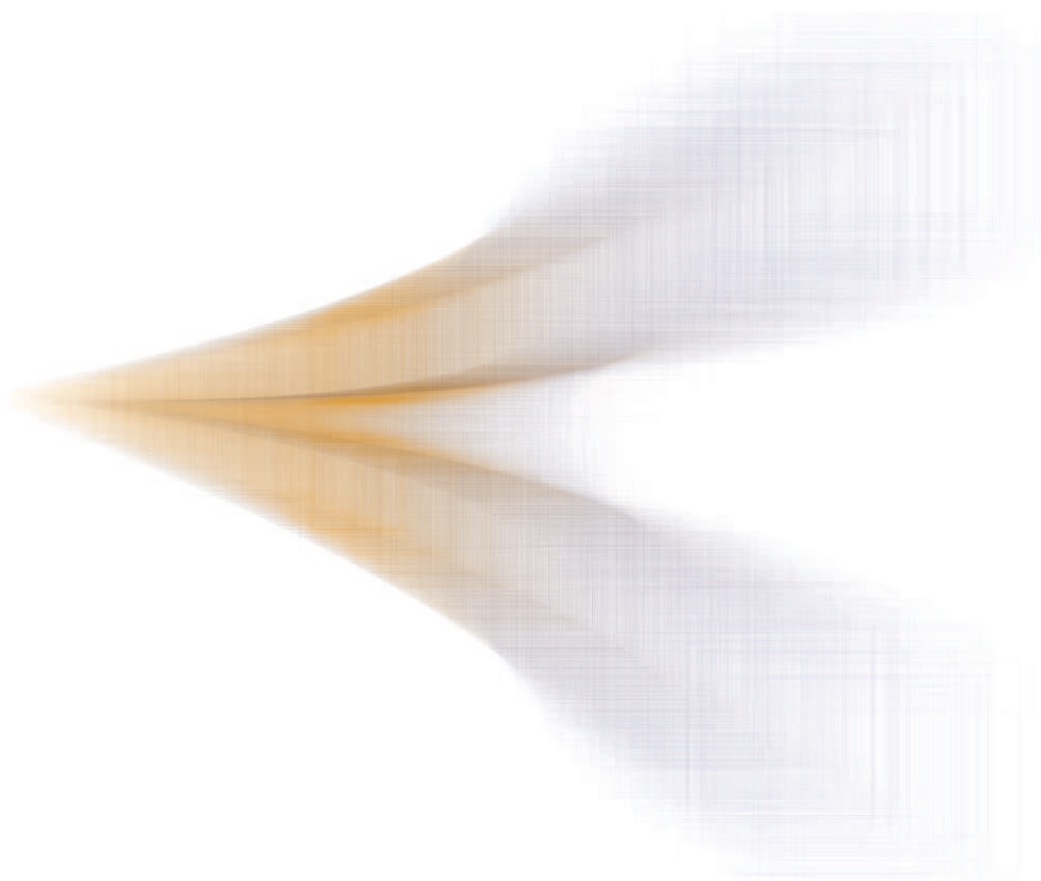
Obrázek C.6: Atraktor typu *Lorenz-84* s parametry $a = 5.68$, $b = 0.15$, $c = 3.36$, $d = 0.12$, $e = 0.35$ vykreslený metodou *Smoke*.



Obrázek C.7: Atraktor typu *Pickover* s parametry $a = 1.13$, $b = 0.45$, $c = 2.82$, $d = 1.17$ vykreslený metodou *Circle*.



Obrázek C.8: Atraktor typu *Lorenz-84* s parametry $a = 7.85$, $b = 1.47$, $c = 2.22$, $d = 1.46$, $e = 0.26$ vykreslený metodou *Circle*.



Obrázek C.9: Atraktor typu *Pickover* s parametry $a = 2.95$, $b = 1.45$, $c = -2.75$, $d = 0.30$ vykreslený metodou *Square* s použitím dvou barev.



Obrázek C.10: Atraktor typu *Clifford* s parametry $a = -1.74$, $b = -0.90$, $c = 0.95$, $d = 0.90$ vykreslený metodou *Circle* s použitím čtyř barev.