VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

VIZUÁLNÍ VÝUKOVÝ SYSTÉM FRAKTÁLŮ

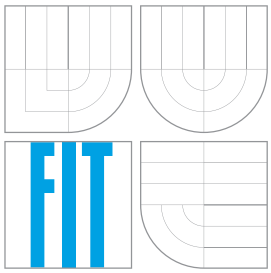
BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

TOMÁŠ FRIEDRICH

BRNO 2007

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

VIZUÁLNÍ VÝUKOVÝ SYSTÉM FRAKTÁLŮ

VISUAL EDUCATIONAL SYSTEM OF FRACTALS

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — TOMÁŠ FRIEDRICH
AUTHOR

VEDOUCÍ PRÁCE — Ing. ZBYŠEK GAJDA
SUPERVISOR

BRNO 2007

Abstrakt

Cílem této práce je přiblížit problematiku fraktálů. Nejprve zde jsou uvedeny základy fraktální geometrie a dále jsou postupně vysvětleny a na příkladech demostrovány některé typy fraktálů. Pro snadnější pochopení všech typů fraktálů byla vytvořena grafická aplikace dostupná na webu, ve které si uživatel může vyzkoušet vlastnosti a chovaní jednotlivých fraktálů.

Klíčová slova

Fraktál, fraktální geometrie, systém, Dynamický systém, IFS, L-sytém, Stochastický systém, komplexní rovina, grafické uživatelské rozhraní

Abstract

The target of this work is to explain basics of fractals generation and theory. The introdaction to the fractal geometry is mentioned first. Certain types od fractals are explained and demostrated by examples next.For easier understanding of all mentioned types of fractals a grafical aplication was designed where it is possible to experience fractals characteristics and behaviour. The aplication is available on the internet.

Keywords

Fractal, fractal geometry, system, dynamic systems, IFS, L-system, stochastic system, graphics user interfase

Citace

Tomáš Friedrich: Vizuální výukový systém fraktálů, bakalářská práce, Brno, FIT VUT v Brně, 2007

Vizuální výukový systém fraktálů

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Zbyška Gajdy. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Tomáš Friedrich
11. května 2007

Obsah

Kapitola 1

Úvod

Problematika kolem fraktálů je velice rozsáhlé odvětví, které v této době zažívá veliký rozvoj. Ovšem tento rozvoj, stejně jako na začátku jiných vědných odvětví, je spíše v řadách domácích kutilů, vědců a lidí zajímajících se o grafiku. Z mých vlastních zkušeností vyplývá, že pro širokou veřejnost i pro většinu studentů na technických školách jsou fraktály buď naprosto neznámá oblast nebo o ní mají jen velice matné a někdy i velice zkreslené informace. Hlavním cílem této práce je tedy nastínění základních oblastí týkajících se fraktálů veřejnosti a dále demonstrace všech základních druhů fraktálů.

V kapitole (2) je celkem podrobně popsaná fraktální geometrie [5] a všechny její základní oblasti, které jsou pro pochopení chování fraktálů potřebné. Bez těchto znalostí není možné pochopit, jak se jednotlivé druhy fraktálů chovají ani vykreslují na počítači.

V 3. kapitole jsou postupně vysvětleny všechny čtyři základní typy fraktálů a u každého typu jsou uvedeny a popsány jednotlivé příklady fraktálních obrazců. U většiny příkladů je uveden i obrázek nebo více obrázků, pro větší pochopení toho, jak se konkrétní fraktál chová.

V kapitole (4) je prezentována samotná aplikace, která byla součástí této práce. Aplikace je rozdělená na demonstrační a teoretickou část, kde v demonstrační části si uživatel může vybrat konkrétní fraktál, který chce zobrazit a dále s ním může pracovat a v teoretické části se dozví veškerou teorii, která je pro pochopení chování fraktálu potřebná.

Poslední kapitolou (5) je závěr, kde je celá práce zhodnocena a je zde i uveden možný budoucí rozvoj aplikace.

Kapitola 2

Fraktální geometrie

Fraktály jsou velice široký pojem, který zasahuje do velkého množství jiných vědních odvětví. Fraktály mají spoustu podob, jsou všude kolem nás a obsahují v sobě samotný princip našeho světa. Pro člověka neznalého problematiky (ale i pro vědce který se jimi zabývá už léta) jsou to úžasné obrazce, které mají nepochopitelnou, ale krásnou strukturu. Jejich zdánlivá nepochopitelnost, ale má svá pravidla a ty je potřeba před pochopením samotných fraktálů znát. Obyčejné předměty jako je stůl nebo židle se dají popsat pomocí jednoduchých útvarů (čtverec, úsečka), přičemž tyto útvary jsou definované pomocí staré řecké euklidovské matematiky. Fraktály se ale pomocí této geometrie popsat nedají. Postupem času bylo zjištěno, že pro jejích vysvětlení je potřeba vytvořit nový vědní obor – *Fraktální geometrii* [5]

2.1 Historie

Fraktální geometrie je samostatná a dnes již poměrně rozsáhlá vědní disciplína zasahující do mnoha dalších oborů, která je intenzivně rozvíjena zhruba od šedesátých let minulého století. Jak je vidět, je to poměrně hodně mladá disciplína, ale její základy byly položeny již dávno. Ostatně, fraktály jsou už odnedávna vidět všude, kam se jen podíváme. V okolní živé i neživé přírodě. Mnoho vědců a také umělců se pokoušelo definovat, co je to fraktál, ale to se podařilo až francouzskému vědci polského původu Benoit B. Mandelbrotovi. Ten jako první matematicky definoval pojem *fraktál* [7] a je tak právem považován za zakladatele fraktální geometrie.

Protože velká část fraktálů je využívána v počítačové grafice a fraktály lze nejlépe popsat jako geometrické objekty, lze fraktál nejjednodušeji definovat jako *nekonečně členitý útvar* . Pro vysvětlení pojmu nekonečně členitý útvar je třeba definovat pojem *geometricky hladký útvar*, který je jistým způsobem pravým opakem útvaru nekonečně členitého.

2.2 Geometricky hladký útvar

Běžná tělesa, ale především umělé útvary v našem okolí se dají popsat nebo zobrazit pomocí určitého počtu parametrů, které tato tělesa z hlediska jejich tvaru plně charakterizují. Pro základní geometrické tvary, například krychli, kouli, válec, přímku, úsečku, známe vzorce, díky nimž můžeme vypočítat například délku, plochu nebo objem. Samozřejmě, že výsledek je pokaždé stejný, ať už počítáme v jakýchkoli jednotkách. Je nepodstatné, zda je poloměr koule zadaný v milimetrech nebo metrech. Po převodu jednotek se výsledek nezmění. Vlast-

nost, kterou mají všechny tyto útvary společnou, je jejich dimenze. Tzn. počet rozměrů, kterými lze daný předmět definovat. Například úsečka má dimenzi 1, protože na určení přesné pozice bodu ležícího na ní stačí pouze jeden parametr, a to jeho souřadnice. Stejně tak jakákoli hladká plocha má dimenzi 2, těleso dimenzi 3. Je ale potřeba upozornit na fakt, že i když má úsečka dimenzi 1, neznamená to, že je zobrazována v jednorozměrném prostoru. Dimenze jen udává počet parametrů potřebných k určení bodu na úsečce.

2.3 Nekonečně členitý útvar

Pro běžné předměty vystačíme z dimenzemi 0, 1, 2 a 3. To znamená, že dimenze je přirozené číslo. Jenže pro většinu předmětů vyskytujících se v přírodě s těmito dimenzemi nevystačíme. Jako příklad mohu uvést břehy potoku nebo pobřeží ostrovu. Uvedený příklad se reálně vyskytl při kartografickém měření pobřeží Bretaně. *L. F. Richardson* jako první zjistil, že délka pobřeží, kterou naměřil, je závislá na délce měřidla, které na odkrokování pobřeží použil. Když si vezmeme mapu nějakého ostrova a budeme chtít změřit délku jeho pobřeží, použijeme například kružítko, pomocí kterého odkrokujeme celý obvod ostrova a po přepočtu na měřítko skutečné dostaneme výslednou délku. Pokud ale použije přesnější mapu a provedeme stejný pokus, výsledek bude jiný. Naměřená délka bude větší. Protože při použití přesnější mapy se objevili podrobnosti pobřeží ostrova, které na předchozí mapě nebyli patrné. Takto bychom mohli postupovat do nekonečna, stále měřit s větší přesností a délka by byla stále větší a větší. Z toho vyplývá že objekt ostrova má nekonečnou délku, ale přitom konečný obsah.

Richardson tak naprosto empiricky bez jakýchkoli matematických důkazu odvodil vztah

$$K = N(\varepsilon)\varepsilon^D$$

kde K značí délku celkového počtu $N(\varepsilon)$ úseček nutných k aproximaci (tj. nejtěsnějšímu pokrytí) dané křivky. Délka pobřeží se ukázala být závislá na konstantě D, jejíž význam si však Richardson nedokázal vysvětlit. Až Benoit B. Mandelbrot dokázal souvislost mezi touto konstantou a *Hausdorffovou-Besicovicovou dimenzí*[7].

2.4 Hausdorffova-Besicovicova dimenze

Délka geometricky hladké křivky, která má topologickou dimenzi rovnu jedné, je při různých měřítkách stále stejná. Délka pobřeží (což je také křivka s topologickou dimenzí rovnou jedné) se při zmenšování měřítka bude neustále prodlužovat až do nekonečna. Pobřeží je tedy plošně větší než hladká křivka. Nevyplňuje ale celou rovinu. Jeho "pravá" dimenze je tedy větší než topologická dimenze hladké křivky, ale zároveň je menší než topologická dimenze roviny (ta je rovna dvěma). Z toho plyne, že dimenze pobřeží nemůže být celé číslo a obecně je nazývána *fraktální dimenzí*.

Objekty s necelořísenou (fraktální dimenzí) se tedy dají považovat za fraktální objekty. Po svých objevitelích je dimenze fraktálních objektů nazývána *Hausdorffova-Besicovicova dimenze.*[7]

Rozdíl mezi fraktální a topologickou dimenzí určuje, jak členitý daný útvar je. Když se hodnoty těchto dimenzí budou lišit velmi málo, objekt bude jen málo členitý. Bude-li fraktální dimenze značně větší než dimenze topologická, bude objekt hodně členitý.

Tyto rozdíly mezi fraktální a topologickou dimenzí využívá asi nejznámější definice fraktálu, kterou definoval Benoit B. Mandelbrot [7].

Fraktál je množina či geometrický útvar, jehož Hausdorffova-Besicovicova dimenze dimenze je (ostře) větší než dimenze topologická.

2.5 Výpočet fraktální dimenze

Pojem fraktální dimenze jsme si vysvětlili v předchozím oddíle 2.4. Ale bude nám k ničemu, když nebudeme vědět jak ji spočítat. Výpočet topologické dimenze je snadný: počet parametrů potřebných pro její popis.

Výpočet fraktální dimenze je ale trochu složitější. Proto bude vysvětlen na jednoduchých příkladech .

2.5.1 Úsečka

Nejjednodušším příkladem výpočtu je úsečka jednotkové délky. Úsečku si rozdělíme na N dílů. Toto rozdělení odpovídá tomu, jako bychom se na úsečku podívali N-násobným zvetšením. Měřítko nové úsečky je tedy:

$$\varepsilon = 1/N \tag{2.1}$$

ε - měřítko, N - počet dílů, na které se úsečka rozdělila. Pro Hausdorffovu dimenzi D obecně platí:

$$N\varepsilon^D = 1 \tag{2.2}$$

Dimenzi D tedy vypočítáme následujícími úpravami:

$$\begin{aligned} N\varepsilon^D &= 1 \\ \log N\varepsilon^D &= \log 1 \\ \log N + D \log \varepsilon &= 0 \\ D \log \varepsilon &= -\log N \\ D &= \log N / \log (1/\varepsilon) \\ D &= \log N / \log N \\ D &= 1 \end{aligned}$$

Topologická dimenze úsečky je rovna jedné, stejně jako vypočítaná fraktální dimenze. Úsečka tedy není fraktál.

2.5.2 Kochova křivka

Stejný postup lze aplykovat i na nejjednodnodušší fráktál na ploše – *Kochovu křivku* (3.15). Při každé iteraci se délka každé hrany zmenší na třetinu (ε) své původní hodnoty a délka křivky se zvětší čtyřikrát (N).

$$\varepsilon = 1/3 \quad , \quad N = 4$$

Fraktální dimenze Kochovy křivky je pak:

$$D = \log N / \log (1/\varepsilon) = \log 4 / \log 3 = 1{,}2618595.$$

Topologická dimenze Kochovy křivky je rovna jedné, frákální je větší. Z toho vyplývá, že tato křivka je fraktálem.

2.6 Soběpodobnost

Dalším důležitým pojmem, který se při popisování abstraktních matematických fraktálů i přírodních útvarů s fraktální strukturou používá, je *soběpodobnost*, nebo také *invariance vůči změně měřítka* [6]. Soběpodobnou strukturu je možno rozložit na struktury, z nichž každá je zmenšenou kopií originálu. Například čtverec lze složit z libovolného počtu menších čverců. Klasickým příklad soběpodobnosti je například *sněhová vločka* (viz oddíl 3.3.3).

2.7 Soběpříbuznost

Naprostá většina fraktálních útvarů nesplňuje podmínky soběpodobnosti. Ty splňují jen uměle vytvořené fraktální struktury. Většina fraktálů je tedy pouze *soběpříbuzná*. To znamená, že při změně měřítka fraktál nevypadá stejně jako celek, ale pouze podobně. Soběpříbuzné jsou prakticky všechny fraktální tvary, které se vyskytují v přírodě. Dochází zde totiž k určitému zkreslení, které je způsobené náhodou a všudypřítomnými fyzikálními zákony. Mezi typické představitele soběpříbuzných fraktálů patři např. mraky, řeky, hory nebo kořeny stromů.

2.8 Atraktor

Atraktor (*attractor*) systému je množina stavů, do kterých systém směřuje. Například pro reálné kyvadlo platí, že atraktorem je stav, kdy se přestane houpat. Naproti tomu atraktorem pohybu planety (Země) je uzavřená elipsa. Některé systémy se ale do konečného stavu nikdy neustálí. Tyto systémy mají tzv. *podivný atraktor* a tento atraktor vykazuje fraktální chování. U některých systémů se přímo vykresluje právě jejich atraktor. Rozeznáváme tedy tři druhy atraktorů:

- Bodové
- Periodické (cyklické)
- Podivné

Kapitola 3

Typy fraktálů

Kvůli systematičnosti se ve fraktální geometrii po určitém čase začali rozlišovat jednotlivé typy fraktálů. Jednotlivé druhy pak mají podobné nejzákladnější charakteristiky a generují se podobným (stejným) způsobem. Toto rozdělení bylo zavedeno i z důvodu odlišné použitelnosti jednotlivých typů. Fraktály se podle nejobecnějšího měřítka dělí na čtyři skupiny:

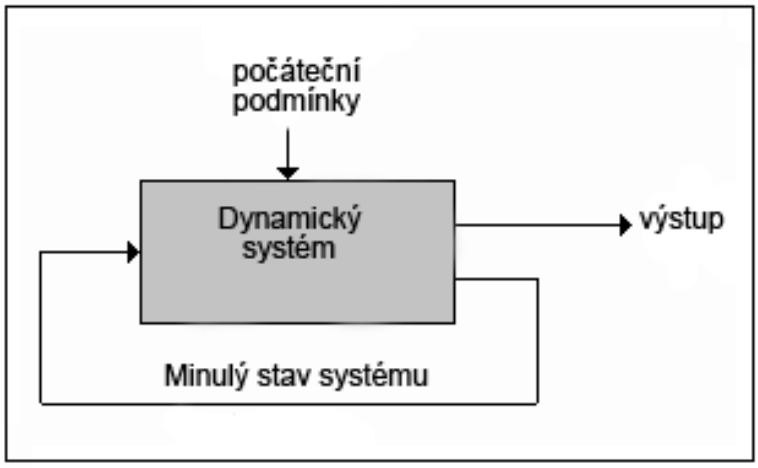
- Dynamické systémy
- Systémy iterovaných funkcí (IFS)
- L-systémy
- Stochastické (náhodné) systémy

3.1 Dynamické systémy

Dynamické systémy [1] tvoří kategorii fraktálů, která má v technické praxi největší uplatnění. Dynamický systém je matematický model, který je závislý na nějaké proměnné (např. na čase). Vychází z počátečních podmínek a je jimi determinován. Je popsán soustavou diferenciálních rovnic, které popisují změnu systému v čase. Stav systému v libovolném čase je potom reprezentován stavovým vektorem, který leží ve stavovém prostoru dynamického systému. Stavový vektor jsou vlastně výsledky všech rovnic, které systém popisují a stavový prostor je množina všech stavových vektorů, kterých systém může nabývat. Změna stavu systému se pak děje provedením diferenciálních rovnic, které systém popisují a nahrazením starého vektoru vektorem novým.

3.1.1 Dynamický systém se zpětnou vazbou

Z hlediska počítačové grafiky a fraktální geometrie je nejdůležitějším druhem těchto systémů *dynamický systém se zpětnou vazbou*. Zpětnou vazbu si může vysvětlit přibližně takto: Výstup systému, či jeho část (tzn. stavový vektor), je znovu přiveden na vstup tohoto systému. To znamená, že aktuální stav systému je přímo závislý na stavu předešlém. Tento jednoduchý princip je názorně vidět na následujícím obrázku (3.1).



Obrázek 3.1: Dynamický systém se zpětnou vazbou.

3.1.2 Dimenze a dynamické systémy

Jak už jsme si řekli, existuje nekonečný počet dimenzí. My si dokážeme představit ale jen první tři. Počet dimenzí úzce souvisí s počtem rovnic, které jsou pro dynamický systém potřeba. Tzn. pro jednu dimenzi je potřeba jedna proměnná, která se mění a ta je potřebná pro jednu rovnici. Dynamické systémy se tedy dají rozdělit na:

- Jednodimenzionální dyn. systémy
- Dvojdimenzionální dyn. systémy
- Vícedimenzionální dyn. systémy

3.1.3 Jednodimenzionální dynamické systémy

Tyto systémy jsou nejjednodušší a proto i nejsnazší pro pochopení. Zde budou uvedeny ty nejzajímavější, na kterých bude vysvětleno, jak se principy těchto systémů ve fraktální geometrii dají použít. Všechny tyto systémy jsou velice citlivé na počáteční podmínky a mají podivní atraktor (viz sekce 2.8).

Verhulstův proces

Při studiu populačního růstu v uzavřeném prostoru (např. počet ryb v rybníce) bylo zjištěno, že populační růst v jednom roce je přímo závislý na růstu v roce předešlém. Je zde vidět zřejmá analogie s principem zpětné vazby dynamického systému. Populační růst klesá jakmile populace dosáhne určité hodnoty (málo prostoru, nedostatek jídla) a stoupá pokud je jídla i prostoru přebytek. Tento dynamický proces je všeobecně označován jako *Verhulstův proces* a popisuje ho vzorec:

$$x_{n+1} = G_r * x_r * (1 - x_n) \tag{3.1}$$

kde G_r je velikost růstu a x_n je počet jedinců. Zajímavé je, že při různých velikostech růstu se systém chová dosti odlišně.

1. Pokud je G_r menší než 300 %, systém se postupně ustálí do jednoho bodu. Tzn. že populace se po určité době zastaví na nějakém počtu a v něm zůstane. Systém takzvaně konverguje do jednoho bodu a tento bod je jeho atraktorem.

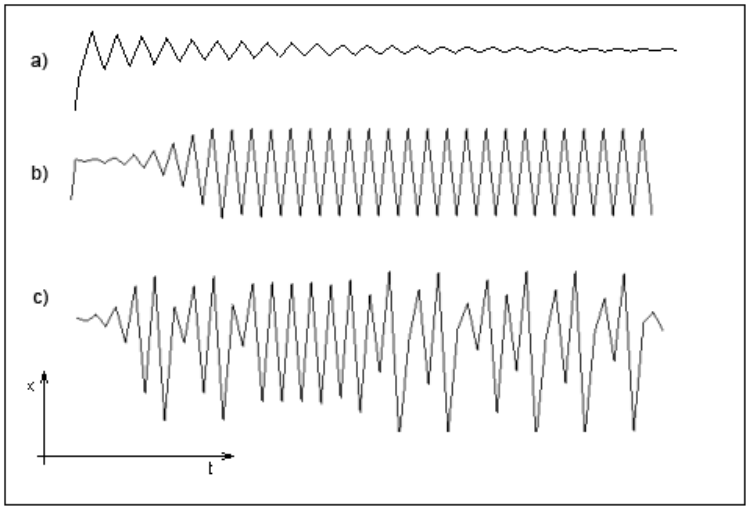
2. Pokud je G_r přesně 300 %, populace se bude každý rok pravidelně měnit mezi dvěma hodnotami. Které budou stále stejné a budou přibližně stejně vzdálené od průměrné populace. tyto dvě hodnoty jsou atraktorem systému, který je periodický.

3. Pro některé další hodnoty dochází také periodickému opakování, ale perioda už není 2 roky, ale více. Např. 4, 8, 16 atd. Dochází k takzvanému zdvojování period. Např. pro 4-periodu je to 345 %.

4. Nejzajímavější je ale poslední možnost, která už vykazuje fraktální chování. Pokud je velikost populace větší než 357 %, systém se stává chaotickým. Velikost populaci je nepředvídatelná a nikdy se neustálí na nějaké hodnotě. V tomto případě má systém podivný atraktor.

Logistická mapa

Verhulstův proces je možné velmi snadno zobrazit na plošném grafu. Přímým zobrazením funkce $f(x)$ popisující proces získáme *logistickou mapu*. Pro různé hodnoty G_r bude mapa vykazovat přesně to chovaní, které bylo popsáno výše. Při změně počátečního počtu populace x_n se výsledek změní jen když je systém chaotický. Následující grafy na obrázku 3.3 názorně vše demostrují.

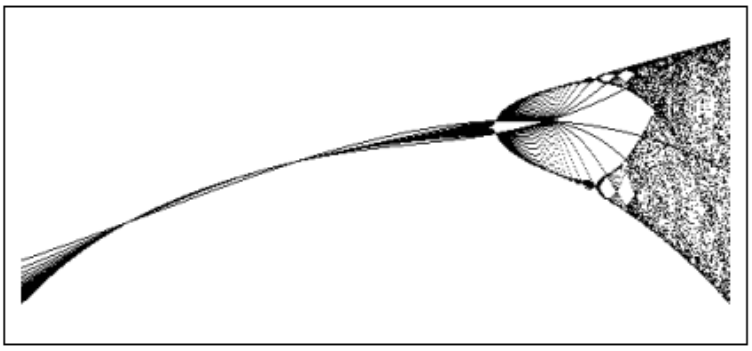


Obrázek 3.2: Logistická funkce s a) bodovým, b) periodickým, c) podivným atraktorem.

Bifurkační diagram

Z přímého zobrazení pomocí logistické funkce není jasně patrné, při jakých počátečních podmínkách je systém stabilní, kdy dochází ke zdvojování period a pro jaké hodnoty nastává chaotické chování systému. Pro názornější představu se používá *bifurkační diagram*, který

zobrazí všechny stavové prostory pro všechny hodnoty jedné proměnné najednou. Na horizontální osu (x) jsou naneseny hodnoty proměnné (Gr) a na vertikální osu (y) jsou naneseny všechny stavy (stavový prostor), kterých systém nabývá pro určitý počet kroků.



Obrázek 3.3: Bifurkační diagram pro kladné hodnoty G_r.

3.1.4 Dvojdimenzionální dynamické systémy

U 1D systémů byla vyzualizována jejich *mapa*. Pro vybrané dynamické systémy s podivným atraktorem je výhodné zobrazovat jejich *orbit*. Orbit je obraz stavového prostoru systému (množina všech stavů, které systém pro určitý počet iterací nabývá) ve dvojdimenzionálním prostoru. Pro vykreslení orbitu do roviny (2D) je použit velmi jednoduchý algoritmus platný pro všechny další příklady.

```
Zjisti počáteční hodnoty systému x0, y0
Zjisti počet iterací
Zjisti hodnoty všech parametrů p0 ... pn
for (n=0 to počet_iterací) do
    x(n+1)=f1(xn, yn, p0 ... pn)
    y(n+1)=f2(xn, yn, p0 ... pn)
vykresli_bod(x(n+1), y(n+1)) do plochy
```

Tento algoritmus je sice naprosto jednoduchý, nicméně pro některé nezasvěcené mohou být některá slova neznámá. Tak se ho pokusím vysvětlit i slovně.

Nejprve je nutné si určit počáteční podmínky proměnných nutných pro vykreslení.Dále je třeba dopředu vědět kolik iterací (stavů) systém bude mít. Vzhled většiny obrazců, které si zde ukážeme, se dá více či méně měnit pomocí parametrů, se kterými se pak počítá při provádění dílčích diferenciálních rovnicích. Když jsou všechny potřebné věci známé, je možné začít s výpočtem. Algoritmus počítá stále dokola dif. rovnice, dokud počet cyklů nedosáhne stanoveného počtu iterací. V jednom cyklu jsou počítány rovnice vždy pro jeden stav systému, kde v každé rovnici se mohou vyskytovat hodnoty předešlého stavu (zpětná vazba) a dále parametry, které určitým způsobem mění vzhled výsledného obrazce. Na konci každého cyklu se vykreslí bod do roviny, kde jeho souřadnice tvoří výsledky dif. rovnic (tzn. hodnoty jednoho stavu).

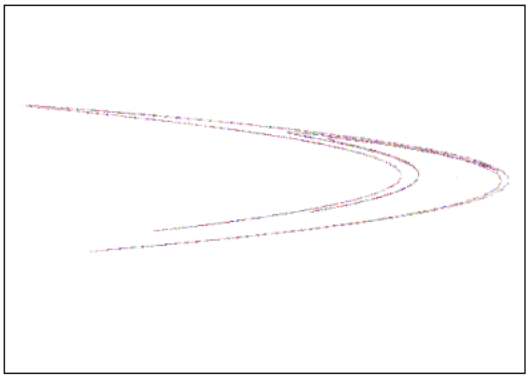
Z toho vyplývá, že každý bod (pixel) obrazce znázorňuje jeden stav systému a celý obrazec pak tvoří stavový prostor. Pokud by systémy, které se takto zobrazují, měli bodový nebo periodický atraktor, výsledkem by byl pouze jeden bod, nebo více v případě zdvojování period. Ale protože tyto systémy mají podivný atraktor, výsledkem je někdy možná nudný ale většinou velice zajímavý obrazec. Zde je jasně patrná vlastnost těchto fraktálních systémů: I když se na první pohled může zdát, že systém je naprosto chaotický, po zobrazení jeho orbitu je vidět, že tento chaos má svůj pořádek.

Henónův atraktor

Patří mezi jeden z nejjednodušších 2D dynamických systémů s podivným traktorem. Rovnice potřebné k jeho zobrazení odvodil *Michel Henón* při studiu pohybu astronomických těles. Vzorec:

$$\begin{aligned} x_{n+1} &= 1 + y_n - a x_n^2 & x_0 &= 0.0 \\ y_{n+1} &= b x_n & y_0 &= 0.0 \end{aligned}$$

Výsledkem je komplikovaný, i když trochu nudný, obrazec připomínající eliptické dráhy astronomických těles. Jeho Hausdorffova dimenze je rovna 1,261.

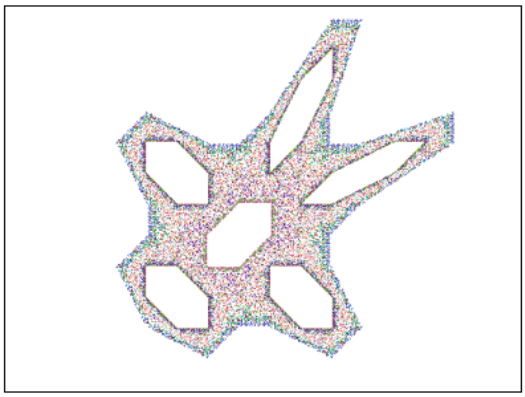


Obrázek 3.4: Henónův atraktor.

Gingerbreadman

Velice zajímavý je i tento obrazec. Vzorec pro jeho výpočet je také velice jednoduchý:

$$\begin{aligned} x_{n+1} &= 1 - y_n + |x_n| & x_0 &= -0.1 \\ y_{n+1} &= x_n & y_0 &= 0.0 \end{aligned}$$



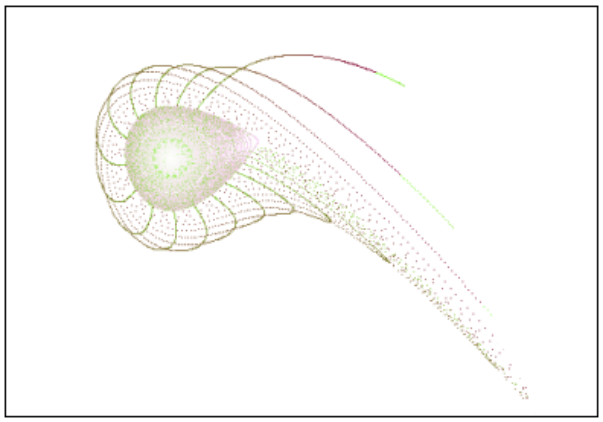
Obrázek 3.5: 2D dynamický systém Gingerbreadman.

Dynamický systém kamtorus

U tohoto dynamického obrazce se nevykresluje pouze jeden orbit (stavový prostor pro určité počáteční podmínky), ale celá sada na sobě nezávislých orbitů. Výpočet každého orbitu vždy vychází ze stejných rovnic. Pouze se mění počáteční podmínky. Celé vykreslení obrazce tedy probíhá ve dvou nezávislých smyčkách, kde ve vnější smyčce se určují počáteční hodnoty proměnných x a y každého orbitu a ve vnitřní je vypočítán a zobrazen každý orbit zvlášť. Všechny orbity dohromady pak tvoří výsledný obrazec. Vzorec:

$$\begin{aligned} x_{n+1} &= x_n \cos\alpha + (x_n^2 - y_n)\sin\alpha & x_0 &= orbit/3 \\ y_{n+1} &= x_n \sin\alpha + (x_n^2 - y_n)\cos\alpha & y_0 &= orbit/3 \end{aligned}$$

Orbit zde značí hodnotu, která se pro každý dílčí orbit mění. Je v rozsahu předem určeném, např. od 1,0 do 5,0 a postupně se mění například po jedné desetině. V takovém případě by se vykreslilo 40 orbitů do jednoho obrázku.



Obrázek 3.6: 2D dynamický systém Kamtorus.

Pickover

Tento systém je pojmenován po známém matematikovi a fyzikovi Cliffordu Pickoverovi. Je definován soustavou tří diferenciálních rovnic:

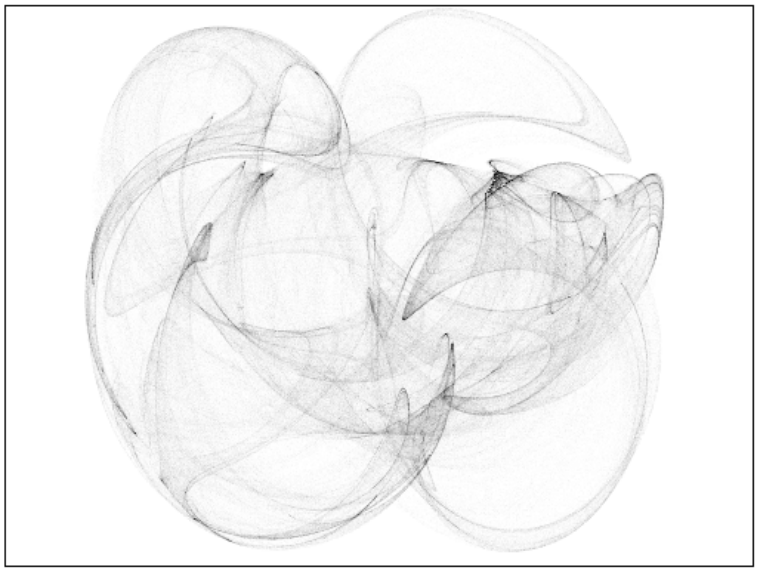
$$\begin{aligned} x_{n+1} &= \sin(a \cdot y_n) - z_n \cos(b \cdot x_n) \\ y_{n+1} &= z_n \sin(c \cdot x_n) - \cos(d \cdot y_n) \\ z_{n+1} &= \sin x_n \end{aligned}$$

počáteční podmínky:

$$x_0 = y_0 = z_0 = 0.0$$

Kde parametry a, b, c, d určují výsledný vzhled obrazce s počátečními hodnotami x_0, y_0 a z_0. Tento systém je tvořený soustavou 3 rovnic. To znamená, že je ideální pro zobrazení v prostoru (3D). Naštěstí je možné systém zobrazit i v rovině, a to tak, že vynecháme poslední z-souřadnici. Ve výpočtu ale 3. rovnice chybět nesmí. Protože na výsledku poslední rovnice je závislý další stav systému.

U Tohoto systému barva pixelu neodpovídá provedené iteraci, jak tomu bylo v předešlých případech, ale na aktuální pozici se pouze zvýší intenzita (světlost) toho pixelu. Tzn. že čím častěji je výsledkem vypočtu jeden pixel, tím vyšší intenzitu má. Pokud ale tohoto efektu chceme dosáhnout, musíme provést řádově stovky tisíc až miliony iterací. Což má samozřejmě za následek delší dobu výpočtu. Následující obrázek 3.7 demonstruje tento systém. Bylo u nej použito 900 000 iterací.



Obrázek 3.7: 2D dynamický systém Pickover.

3.1.5 Mapy dynamických systémů vykreslované v komplexní rovině

Asi nejzajímavější a nejznámější obrázky s tematikou fraktálů byly vytvořeny vizualizací dynamických systémů v komplexní rovině. Až do této chvíle se u dynamických systémů zobrazovali jejich orbity. U následujících příkladů se budou zobrazovat jejich mapy. Mapu představuje rastrový obrázek, ve kterém barva každého pixelu (bodu) odpovídá vybranému stavu dynamického systému v bodě, jehož souřadnice jsou do pixelu mapovány. Každému pixelu tedy odpovídá komplexní hodnota s reálnou a imaginární složkou. Zdaleka nejpoužívanějším způsobem vizualizace dynamických systémů v komplexní rovině je zvýraznění počtu iterací, tj. počtu opakování funkce dynamického systému do té doby, než je splněna nějaká předem známá podmínka.

Všechny dále uvedené druhy fraktálů jsou založeny na postupné iteraci funkce komplexní paraboly.

Komplexní parabola

Funkce komplexní paraboly je dána vzorcem:

$$z_{n+1} = z_n^2 + c$$

Kde z i c leží v komplexní rovině. Pokud by c bylo rovno nule, je chování systému předvídatelné:

1. Pro $z_0 = 1$, atraktorem systému bude kružnice.
2. Pro $z_0 < 1$, posloupnost bodů by tvořila spirálu směřující k nule (tedy 0 + 0i), která by byla i atraktorem systému.
3. pro $z_0 > 1$, posloupnost bodů by také tvořila spirálu, která by ovšem směřovala opačným směrem, tedy do nekonečna.

Pokud je ale c různé od nuly není pro většinu bodů možné zjistit, jestli výsledky funkce budou konvergovat do jednoho bodu, divergovat do nekonečna, nebo jestli se budou periodicky opakovat. Množina bodů, které nedivergují pak tvoři množinu s fraktálními vlastnostmi - soběpodobností a nezávislostí na změně měřítka.

Pro zjištění, zda-li posloupnost $z_{n+1} = z_n^2 + c$ diverguje se použije tvrzení:

Pokud platí $|c| <= |z|$ a současně $|z| > 2$, pak posloupnost diverguje.

Pomocí trojúhelníkové nerovnosti pro funkci komplexní paraboly a předešlého tvrzení se ukazuje, že hraniční oblastí, za kterou posloupnost vždy diverguje je kružnice o poloměru 2.0 [9].

Uvnitř této kružnice není možné analyticky dokázat, zda posloupnost konverguje, diverguje či osciluje. Proto je nutné provádět další iterace.

Tohoto principu využívá *algoritmus TEA*, pomocí kterého se obrázky vytvářejí.

Algoritmus TEA

Algoritmus TEA (*Time Escape Algorithms*) [1] provádí iterace dané posloupnosti komplexní paraboly až do překročení hraniční oblasti nebo do vyčerpání maximálního počtu iterací. Algoritmus pracuje tak, že bere pod po bodu a využívá je jako startovní (počáteční) hodnoty posloupnosti. Provede příslušnou transformaci (vyčíslí rovnici paraboly) a zjistí jestli se vypočítaný bod vyskytuje uvnitř kružnice. Pokud je uvnitř, nově vypočítané komplexní číslo se použije pro následující iteraci a opět se provádí kontrola překročení. Výpočet se opakuje, dokud nedojde k překročení hranice (hodnota diverguje), nebo není vyčerpán maximální počet iterací. Pokud posloupnost nediverguje (zůstane v oblasti), startovnímu bodu se přiřadí určitá, předem stanovená barva. Pokud dojde k překročení, iterační proces se zastaví a danému startovnímu bodu se přiřadí barva odpovídající počtu iterací, které byly potřeba pro překročení hranice. Jakou barvou bude příslušný bod obarven, záleží čistě na programátorovi.

Jistě se také musí počítat s tím, že pokud má algoritmus nastavený určitý počet maximálních iterací, výpočet není přesný. Po ukončení iterací by se klidně mohlo hned při další iteraci rozhodnout o uniku z oblasti. Proto čím vyšší počet iterací bude, tím větší detaily se na obrázku budou zobrazovat.

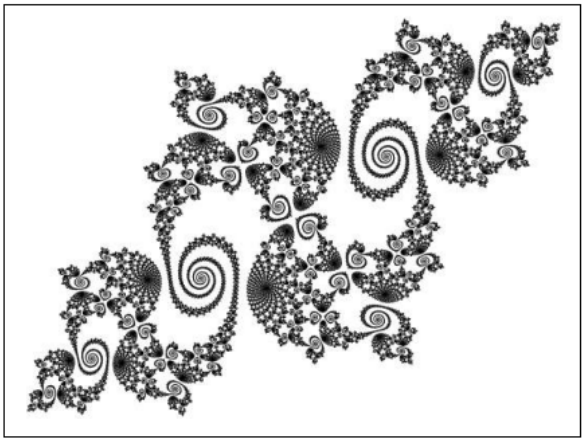
Juliovy množiny

Juliovy množiny jsou vytvářeny pomocí jednoduchého dynamického systému založeného na postupné iteraci komplexní paraboly 3.1.5 a jejich vykreslení je prováděno TEA algoritmem 3.1.5. Iterační vztah pro komplexní parabolu:

$$z_{n+1} = z_n^2 + c$$

kde z_n a c samozřejmě leží v komplexní rovině. Iterační TEA algoritmus si jako startovní body z_0 bere souřadnice pixelů výsledného obrázku namapovaného do konkrétní komplexní roviny. Komplexní parametr c je zvolen libovolně a pro celou množinu je stále stejný. Samozřejmě, že parametr musí ležet uvnitř kružnice úniku. Jinak nemá výpočet smysl.

Diky parametru c je tvar každé Juliovy množiny jiný a je jich nekonečně mnoho.



Obrázek 3.8: pohled na Juliovu množinu.

Mandelbrotova Množina

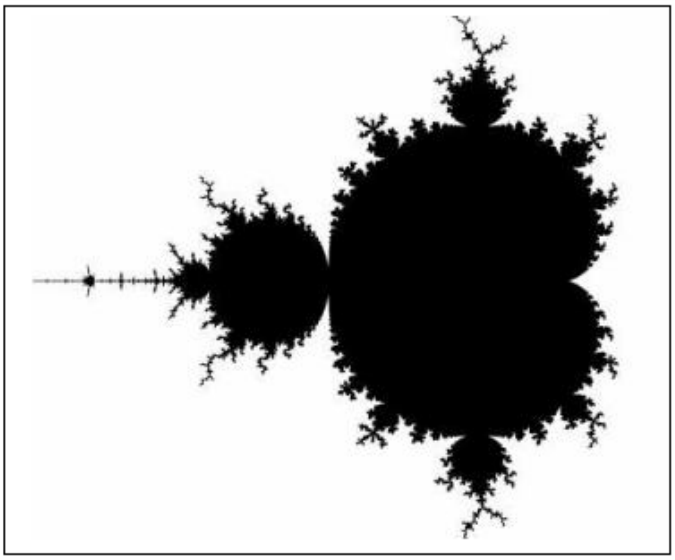
Tento fraktál je asi nejznámější ze všech a je tedy považován za jakýsi symbol fraktální geometrie. Jeho autorem je francouzský matematik Benoit Mandelbrot.

Mandelbrotova množina je velice blízkou příbuznou množin Juliových, je stejně jako ony založena na postupné iteraci komplexní paraboly, pro své vykreslení využívá algoritmus TEA a platí pro ní stejné podmínky konvergence. Hlavní rozdíl mezi ni Juliovými množinami je v tom, že proměnné z iteračního vzorce využívá opačně. Iterační vztah pro komplexní parabolu:

$$z_{n+1} = z_n^2 + c$$

Počáteční hodnota z_0 je vždy stejná (nulová) a za parametr c je dosazena souřadnice pixelu (namapovaného do komplexní roviny), pro který TEA algoritmus hledá barvu. A naopak při konkrétním výpočtu se mění počítané zn a c zůstává konstantní.

Protože se při výpočtu mandelbrotovy množiny využívají všechna c ($|c| < 2$), existuje pouze jedna.



Obrázek 3.9: Celkový pohled na Mandelbrotovu množinu.

3.2 IFS Systémy

Název IFS systémů je odvozen z anglického označení *Iterated Function System* [1]. Česky se tento termín překládá jako *Systém iterovaných funkcí*. Tvorba obrázků pomocí IFS systémů patří mezi generativní metody vytváření fraktálů (výsledná podoba fraktálu je postupně generována z nějakého vzoru). Algoritmus pro tvorbu těchto fraktálů muže být jak deterministický (přesný), tak stochastický (náhodný).

Tento druh fraktálů má v počítačové grafice velké uplatnění, kde asi nejvýznamnější postavení zabírá *fraktální komprese dat* (podprobne viz [5]). Dále jsou tyto systémy používány v grafických nástrojích a zejména ve hrách, kde slouží ke generování krajin.

Jednou ze základních vlastností fraktálů je soběpodobnost. Pokud je fraktál soběpodobný, je možné najít zobrazení, které transformuje (mapuje) celek na jednotlivé části. Pokud je

toto zobrazení prováděno iterativně, transformace postupně konvergují (přibližují se) k atraktoru fraktálu.

IFS fraktál je tedy popsán množinou *transformaci* [6]. Mezi nejčastěji používané transformace patří *posun, zkosení, rotace a změna měřítka.*

Tyto transformace jsou lineární. To znamená, že při aplikaci této transformace na úsečku bude výsledkem opět úsečka. Aby se zaručilo, že atraktorem výsledného fraktálu nebude nekonečno, transformace musí být *kontraktivní.* To znamená, že vzdálenost mezi dvěma po sobě transformovaných bodů se bude stale zmenšovat.

3.2.1 Algoritmus náhodné procházky (RWA)

Algoritmus náhodné procházky (RWA - *random walk algorithm* [9]) je asi nejznámějším algoritmem pro generování fraktálních objektů pomocí systémů iterovaných funkcí.

Generování fraktálu začíná zvolením náhodného bodu, ze kterého se budou provádět následují transformace. Poloha tohoto bodu nemá žádný vliv na tvar výsledného fraktálu, protože po určitém počtu transformací se body stejně začnou přibližovat atraktoru systému. Poté je náhodně vybrána jedna z transformací, pomocí kterých se obrazec vykresluje a je aplikována na bod (jeho souřadnice). Výsledkem je bod s odlišnými souřadnicemi, který je poté vykreslen na plochu. Na tento bod je posléze iterativně aplikována další vybraná transformace. Iterace probíhají tak dlouho, dokud není dosaženo maximálního počtu iterací.

Protože počáteční bod může mít jakékoli souřadnice, nemusí ležet v atraktoru systému. Pokud by byl tedy tento bod zobrazen, výsledný obrazec by nesplňoval fraktální vlastnosti. Protože se systém po několika iteracích do svého atraktoru dostane, stačí když těchto prvních pár bodu prostě nebude vykresleno.

Jak je z algoritmu patrné, kvalita fraktálu je přímo závislá na počtu vykreslených bodů (počtu iterací). Pokud je počet iterací malý, výsledný obrázek bude tvořit malé množství bodů a obrázek tedy bude špatně viditelný. Pokud naopak bude počet iterací příliš velký, výpočet souřadnic bodu bude trvat příliš dlouho a některé body se mohou i překrývat.

3.2.2 Transformační matice

V algoritmu náhodné procházky nejsou transformace používány přímo. Jejich koeficienty jsou uloženy do transformačních matic [6], pomocí kterých se pak body transformují. Transformace je dána vztahem 3.2:

$$w(x) = w\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} r_1 \cos\phi & -r_2 \sin\vartheta \\ r_1 \sin\phi & r_2 \cos\vartheta \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} + \begin{pmatrix} e \\ f \end{pmatrix} \tag{3.2}$$

V této transformaci mají jednotlivé parametry následující význam. Úhel ϕ určuje otočení osy x, v jejímž směru je útvar přeškálován parametrem r_1 a současně ϑ určuje úhel otočení osy y, v jejímž směru je útvar přeškálován parametrem r_2. Parametry e a f určují translaci útvaru podle jednotlivých (neotočených) os. Parametry x_1 a x_2 jsou souřadnicemi transformovaného bodu.

Abychom takto definovaná transformace mohla být převedena na transformační matici, je potřeba ji trochu zjednodušit.

Prvky matice, ve kterých se vyskytují goniometrické funkce (sin, cos) se nahradí konstantami:

$$w(x) = w\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} + \begin{pmatrix} e \\ f \end{pmatrix} \tag{3.3}$$

Takto upravenou transformaci už můžeme převést.

Transformační matice, která nic netransformuje je jednotková [6] a má tvar

$$\mathbf{T} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.4}$$

Do této matice se dosadí na správná místa koeficienty transformace:

$$\mathbf{T} = \begin{bmatrix} a & b & e \\ c & d & f \\ 0 & 0 & 1 \end{bmatrix} \tag{3.5}$$

A pomocí této matice se pak transformují jednotlivé body fraktálu.

3.2.3 Pravděpodobnost výběru transformace

V algoritmu náhodné procházky nejsou transformace vybírány úplně náhodně, jak to bylo popsáno v části věnující se tomuto algoritmu, ale jednotlivé transformace jsou vybírány s určitou pravděpodobností. Každá transformace má určitou pevně danou pravděpodobnost použití a všechny pravděpodobnosti dávají dohromady 1. Transformace se tedy sice vybírají náhodně, ale některé transformace mají větší pravděpodobnost že budou vybrány.

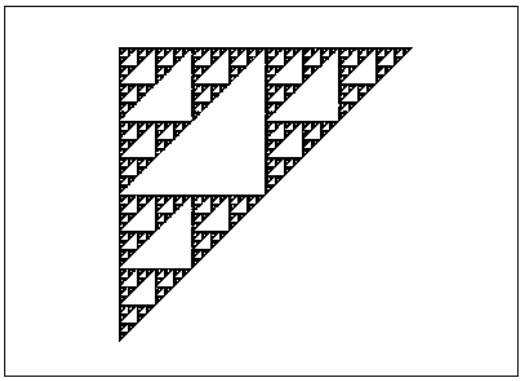
3.2.4 Příklady IFS systémů

Nyní budou uvedeny některé známé a zajímavé fraktály tvořené IFS systémy. U každého budou vždy uvedeny koeficienty transformací potřebné pro jejích generování a pravděpodobnosti použití těchto transformací.

Sierpinského trojúhelník

Tento fraktál je asi nejznámější ze všech je uveden snad v každé publikaci, která se zabývá problematikou fraktálů.

Sierpinského trojúhelník						
a	**b**	**c**	**d**	**e**	**f**	**p**
0.5	0	0	0.5	0	0	0.33
0.5	0	0	0.5	0	1	0.33
0.5	0	0	0.5	1	1	0.33

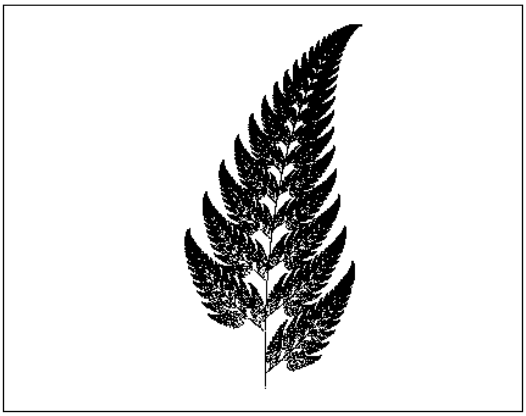


Obrázek 3.10: Sierpinského trojúhelník.

Kapradina Michaela Barnsleye

Tento fraktální obrazec je také velice známý a je nádhernou ukázkou toho, jak se dají přírodní útvary vykreslit naprosto jednoduše.

Kapradina Michaela Barnsleye						
a	**b**	**c**	**d**	**e**	**f**	**p**
0	0	0	0.16	0	0	0.01
0.2	-0.26	0.23	0.22	0	1.6	0.07
-0.15	0.28	0.26	0.24	0	0.44	0.08
0.85	0.04	-0.04	0.85	0	1.6	1.0

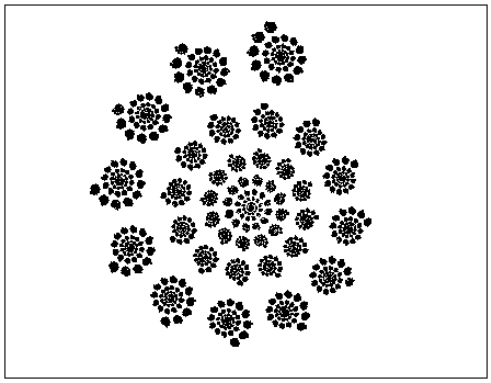


Obrázek 3.11: Kapradina Michaela Barnsleye.

Spirála

Na této spirále je velice dobře patrná její soběpodobnost.

Spirála						
a	**b**	**c**	**d**	**e**	**f**	**p**
0.83	-0.48	0.48	0.83	0	0	0.9
0.17	-0.1	0.1	0.17	0.2	1.0	1.0

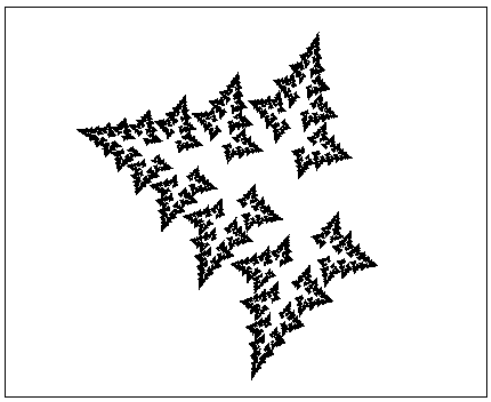


Obrázek 3.12: Spirála.

Krystal

Jako posledni je zde uveden fraktální obrazec připomínající krystalickou strukturu nerostů.

Krystal						
a	**b**	**c**	**d**	**e**	**f**	**p**
0.7	-0.49	-0.39	-0.66	2.15	10.31	0.75
0.09	-0.44	0.52	-0.1	4.27	2.93	1.0



Obrázek 3.13: Krystal.

3.3 L-systémy

L-systémy (*Lindenmayerovy systémy* [8]) jsou naprosto odlišnou skupinou než předchozí dynamické nebo IFS systémy. Jsou definovány pomocí *regulárních* nebo *bezkontextových gramatik*. Tvorba obrazců spočívá v přepisování řetězců podle určitých pravidel (gramatika). Každý symbol v řetězci má nějaký geometrický význam. Například nakreslení objektu, posun nebo rotaci. Pomocí L-systémů se velice snadno generují objekty, které se podobají rostlinám, stromům, řekám a dalším přírodním útvarům. Pokud přepisování symbolů v řetězci není přesně dané, ale je v něm zakomponována i náhoda, výsledné obrazce vypadají velice realisticky.

3.3.1 Gramatiky

Abychom mohli L-systémy pochopit, je nutné vědět alespoň základy užívání gramatik. Gramatika je definována uspořádanou čtveřicí symbolů:

$$G = [N, \sum, P, S]$$

Kde:

- N je konečná abeceda nonterminálních symbolů
- $\sum$ je konečná abeceda terminálních symbolů
- P je konečná množina přepisovacích pravidel ve tvaru $A \rightarrow B; ; A \in N; B \in (N \cup \sum)^*$
- S je *axiom*: neprázdná posloupnost symbolů ve tvaru $S \in (N \cup \sum)^+$

Přepisovací pravidla mohou vypadat například takto $S \rightarrow aBC$, $B \rightarrow XY$, $C \rightarrow a$, aj.

Nonterminální symboly jsou označovány velkými písmeny a jsou určeny pro rozložení pomocí nějakého pravidla na posloupnost terminálních nebo nonterminálních symbolů. Terminální symboly jsou pro přehlednost označovány malými písmeny a tyto symboly se už dále nerozepisují (terminální symbol nesmí být na levé straně pravidla). Podle tvaru přepisovacích pravidel se gramatiky rozdělují na regulární, bezkontextové, kontextové a obecné. L-systémy jsou většinou popsány pomocí regulárních gramatik, které jsou ale mírně pozměněné.

L-systémy nepotřebují terminální symboly, protože počet derivací (použití pravidla) je libovolný (daný programem) a po proběhnutí zadaného počtu derivací se všechny nonterminální symboly stanou terminálními.

Tyto systémy jsou deterministické. Aby to bylo splněno, nesmějí existovat dvě pravidla se stejnou levou stranou. Nebylo by pak možné určit které se má vybrat.

Příklad použití gramatiky

L-systém G je definován takto:

$G = [V, P, S]$

$V = \{A, F, +, -\}$

$P = \left\{ \begin{array}{l} A \rightarrow F - -F \\ F \rightarrow F + F - -F + F \end{array} \right\}$

$S = A$

Pro první dvě derivace (použití pravidel) pak vypadají takto:

$$A \rightarrow F--F--F \rightarrow F+F--F+F--F+F--F+F--F+F--F+F$$

3.3.2 Želví grafika

Želví grafika tvoří základní nástroj pro tvorbu L-systémů (obrazců) z použité gramatiky. Základem je takzvaná *želva*, která je definována svým stavem a množinou akcí, které může provádět. Stav želvy se skládá ze dvou částí, z polohy želvy a z její orientace. Želva sekvenčně čte řetězec vytvořený gramatikou a pomocí tabulky akcí interpretuje jednotlivé symboly. Každý symbol tedy znázorňuje určitou akci, kterou želva vykoná když ho načte.

Nejčastějším způsobem tvorby fraktálních obrazců pomocí želví grafiky je značení cesty, kudy želva *prochází*, když čte posloupnost řídících symbolů. Nejjednodušší forma L-systému interpretovaného želvou v ploše může používat následující symboly:

- F – posun želvy dopředu s kreslením úsečky
- G – posun želvy dopředu bez kreslení úsečky
- B – posun želvy dozadu s kreslením úsečky
- $+$ – natočení želvy doleva o předem známý počet stupňů
- $-$ – natočení želvy doprava o předem známý počet stupňů

3.3.3 Fraktální útvary vzniklé pomocí L-systémů

Cantorova množina (Cantorův prach)

Tato množina je pojmenována po německém matematikovi ruského původu Georg Cantorovi a byla publikována už v roce 1883. Konstrukce Kantorovi množiny je velice jednoduchá.

Množinu tvoří úsečka ležící v intervalu $< 0, 1 >$. Úsečka se rozdělí na třetiny a prostřední třetina se vyjme. Zbudou tedy dvě úsečky třetinové délky té původní.

V dalším kroku (iteraci) se celý postup opakuje. Po provedení nekonečně mnoha iterací, vznikne množina bodu (úseček s nulovou délkou), které nejsou propojeny, protože mezi každou dvojicí bodů se nachází mezera (s nulovou délkou).

Pomocí gramatiky je možné Kantorovu množinu popsat následujícím způsobem:

$G = [V, P, S]$

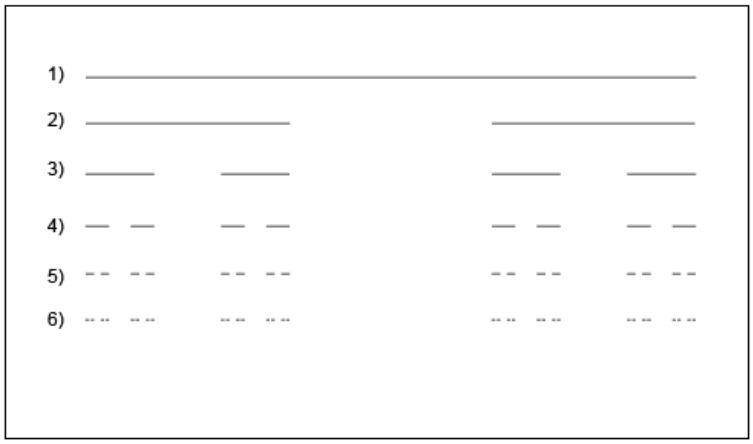
$V = \{F, G\}$

$P = \left\{ \begin{array}{l} F \rightarrow FGF \\ G \rightarrow GGG \end{array} \right\}$

$S = F$

Po čtyřech derivacích bude posloupnost vypadat takto:

$$S \rightarrow F \rightarrow FGF \rightarrow FGFGGGFGF \rightarrow FGFGGGFGFGGGGGGGGGFGFGGGFGF$$



Obrázek 3.14: Prvních 6 iterací Cantorovy množiny.

Kochova křivka

Tento fraktální obrazec je dílem švédského matematika Helge von Kocha, který jej představil v roce 1904. Celý obrazec vzniká následujícím způsobem.

1. Tzv. iniciátorem je zde opět úsečka. Na její délce nezáleží.
2. Úsečka se rozdělí, stejně jako v případě kontorovi množiny na třetiny a prostřední se vyjme. Tato mezera se ale nahradí trojúhelníkem. To znamená, že se na jejím místě sestrojí dvě stejně dlouhé ramena rovnostranného trojúhelníku. Vznikne tedy lomená čára která má (v případě, že úhel který ramena svírají je 60 stupňů) 4/3 původní délky.
3. V dalším kroku se pravidlo aplikuje na každou nově vzniklou úsečku.

Po nekonečně mnoha iteracích vznikne obrazec který je nekonečně členitý. Jak je vidět z obrázku 3.15, křivka je velice členitá už po pár iteracích a je zde na 1. pohled vidět základní vlastnost fraktálů, soběpodobnost. Protože je křivka nekonečně členitá je i nekonečně dlouhá. Přitom ale zabírá konečnou plochu.

Gramatika použitá pro vykreslení Kochovy křivky je následující:

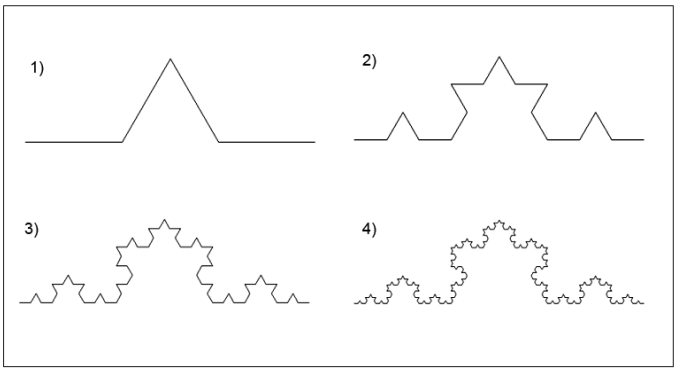
$G = [V, P, S]$

$V = \{F, +, -\}$

$P = \{\ F \rightarrow F + F - -F + F\ \}$

$S = F$

$\alpha = 60°$



Obrázek 3.15: První 4 iterace Kochovy křivky.

Sněhová vločka

Autorem tohoto fraktálu je také výše zmíněný Helge von Koch a je tvorba je naprosto stejná jako tvorba Kochovy křivky. Jediný rozdíl je v iniciatoru. Není jím úsečka, ale rovnostranný trojúhelník. Gramatika tedy vypadá takto:

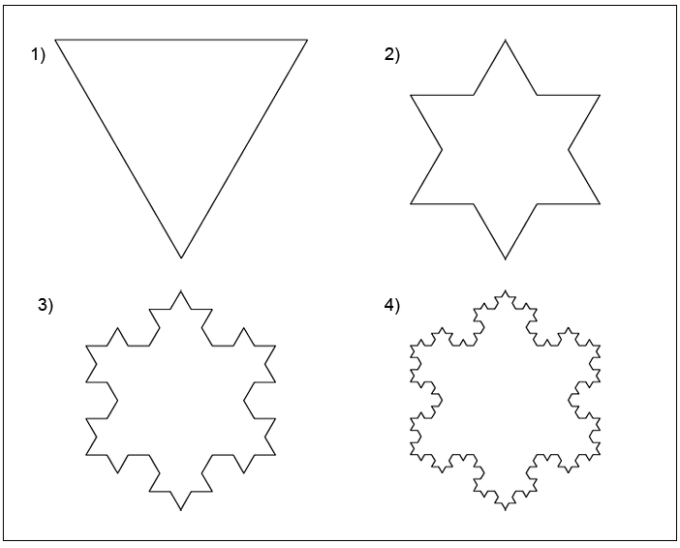
$G = [V, P, S]$

$V = \{F, +, -\}$

$P = \{\ F \rightarrow F + F - -F + F\ \}$

$S = F - -F - -F$

$\alpha = 60°$



Obrázek 3.16: První 4 iterace sněhové vločky.

Hilbertova křivka

Tento fraktální útvar je velice významný, protože se jedná o jeden z nejjednodušších útvarů, který je sice tvořen jen soustavou na sebe navazujích úseček (topologická dimenze je tedy rovna jedné), ale přitom zabírá celou plochu (hausdorffova dimenze je rovna dvěma). Její gramatika má následující tvar:

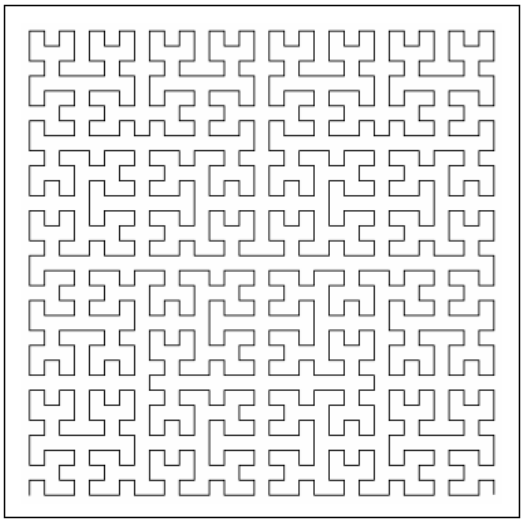
$G = [V, P, S]$

$V = \{X, Y, F, +, -\}$

$P = \left\{ \begin{array}{l} X \to YF + XFX + FY- \\ Y \to XF - YFY - FX+ \end{array} \right\}$

$S = X$

$\alpha = 90°$



Obrázek 3.17: Pátá iterace Hibertovy křivky.

Sierpinského trojúhelník

Tento útvar byl uveden již v příkladech IFS (3.2.4), ale pomocí L-systémů ho lze sestavit také. Gramatika pro jeho sestrojení je následující:

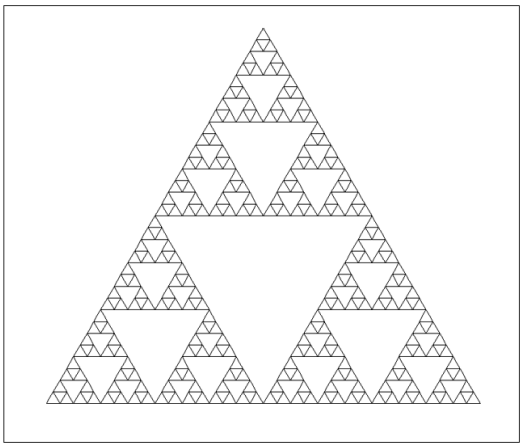
$G = [V, P, S]$

$V = \{X, F, +, -\}$

$P = \left\{ \begin{array}{l} F \to FF \\ X \to ++FXF--FXF--FXF++ \end{array} \right\}$

$S = FXF++FF++FF$

$\alpha = 60°$



Obrázek 3.18: Pátá iterace Sierpinského trojúhelníku.

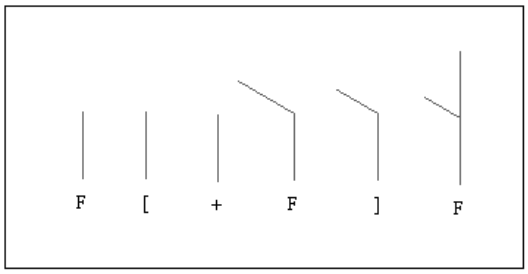
3.3.4 Závorkové L-systémy

Předchozích kapitolách byli ukázány obrazce, které měli jednu společnou vlastnost, netvořili rozvětvené struktury. Většinou se jednalo jen o spojitou křivku (kochova, hibertova). Vlastnost tvorby větvících se struktur nemohla být splněna, protože želva si neuměla zapamatovat svůj stav, tj. pozici v rovině a její orientaci. Problém se vyřeší velice jednoduše zavedení zásobníku, do kterého je při interpretaci přepisovaného řetězce ukládán stav či více stavů želvy. Stavy želvy pak mohou být kdykoli vybrány. Želva se pak vrátí na zapamatované místo a provede danou operaci.

Aby se ale zásobník dal používat, musí se rozšířit i gramatika. Gramatika se rozšíří o dva symboly:

1. [– želva si svůj aktuální stav uloží na zásobník.
2.] – želva vybere ze zásobníku poslední uložený stav a přesune se na aktuální pozici a provede natočení.

Celý princip demonstruje následující obrázek 3.19:



Obrázek 3.19: Demostrace použití zásobníku pro želví grafiku.

Pomocí takto rozšířených L-systémů se velice snadno tvoří různé útvary připomínající rostliny, keře a stromy. Některé z nich budou ukázány v následujících příkladech

Binární strom

Ukázkovým příkladem tohoto druhu fraktálů je *binární strom* (po více jak 10 iteracích se ale spíše podobá květu pampelišky). Na jeho jednoduchosti je krásně vidět funkce závorek a zásobníku, který tyto systémy používají. Jeho gramatika má následující tvar:

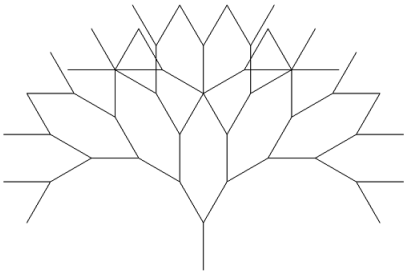
$G = [V, P, S]$

$V = \{F, X, +, -, [,]\}$

$P = \{\ X \rightarrow [-FX] + FX\ \}$

$S = + + + FX$

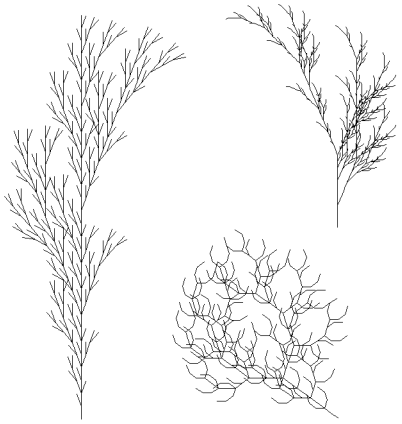
$\alpha = 30°$



Obrázek 3.20: Šestá iterace binárního stromu.

Simulace keřů, rostlin a stromů

Následující příklady už nemají nějaký konkrétní význam a jsou zde uvedeny jako ukázky toho, co se dá pomocí L-systémů vytvořit.



Obrázek 3.21: Příklady modelů keřů.

3.4 Stochastické systémy

Velkou část fraktálního světa zabírají takzvané *stochastické* (náhodné) fraktály [9].

Zatímco všechny předchozí typy fraktálních útvarů byly v určitém smyslu měřítkově symetrické, tj. soběpodobné (viz oddíl 2.6), při generování nepravidelných fraktálů je využita náhoda. Tyto útvary jsou tedy pouze soběpříbuzné (viz oddíl 2.7). Tato vlastnost dává tomuto typu fraktálů nejlepší možnosti pro popis přírodních útvarů, přičemž míra, se kterou se náhodnost bude podílet na procesu generování fraktálů, bude vždy určovat tvar fraktálu a současně i jeho Hausdorffovu dimenzi (viz oddíl 2.4).

Náhodné fraktály je možné vytvářet několika způsoby. například *simulací brownova pohybu* nebo *metodou přesouvání středního bodu*.

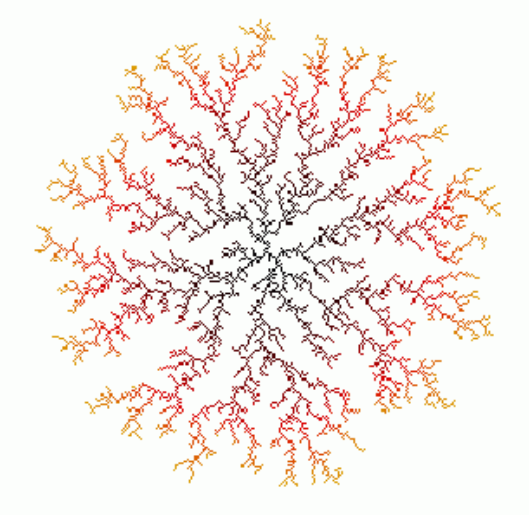
3.4.1 Simulace brownova pohybu

Simulace Brownova pohybu vytváří fraktální objekt, jehož Hausdorffova dimenze je úměrná absolutní velikosti změny při jednom kroku iterace. Tato metoda se používá například při generování toků řek.

Difúzí omezená agregace

Jednou z přímých aplikací Brownova pohybu je difúzí omezená agregace (*diffusion limited aggregation* [6]), označovaná jako DLA. DLA je fyzikálním jevem, při kterém rostou například obrazce na zamrzlých sklech. V počítačové grafice se DLA používá například pro modelování struktur a růstu korálů.

Předpokládejme, že máme roztok, ve kterém volně plavou molekuly nějaké látky, a v roztoku je zároveň kondenzační jádro (oblast, ze které se začne obrazec tvořit). Jakmile se některá z okolních molekul dostatečně přiblíží jádru, je jím zachycena a stane se jeho součástí. Tímto způsobem difúze molekul postupně přidává nová jádra a agreguje tak nový tvar.



Obrázek 3.22: Model obrazce na zamrzlém skle vytvořený algoritmem DLA.

Simulace DLA

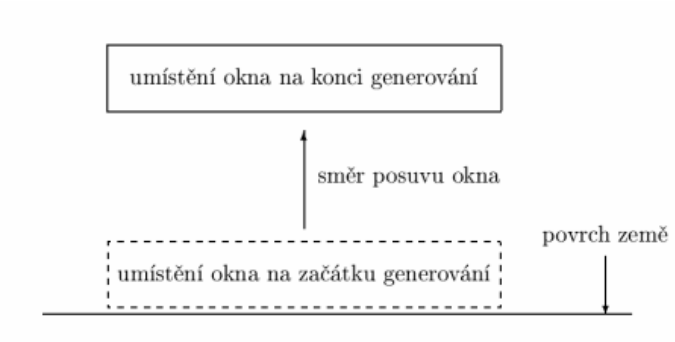
Princip uvedený v předchozím oddíle se dá velice snadno simulovat.

Nejprve se vytvoří částice ze kterých bude fraktál vyrůstat - *semínka*. Na jejich počtu a rozmístění je přímo závislý výsledný tvar obrazce. Poté se začnou generovat náhodné souřadnice částic a tyto částice se pak pohybují po trajektorii určené Brownovým pohybem. Jakmile se částice při svém pohybu dostatečně přiblíží k jádru, je k němu připojena. Pokud částice při svém pohybu opustí plochu pro vykreslení, zanikne. Generování končí, pokud je dosažen maximální počet částic, které tvoří fraktál, nebo se některá z připojených částic dostane na okraj obrázku. Výsledek simulace je vidět na obrázku 3.22

Simulace pohybu každé částice pomocí Brownova pohybu je ale velice náročná na výpočetní výkon a je tedy velice pomalá. Existuje proto mnoho rozšíření algoritmu DLA, které tento algoritmus urychlují.

Jednou z nich je omezení prostoru, ve kterém se částice tvoří. Pokud jsou částice generovány naprosto náhodně, pravděpodobnost, že se bod dotkne vytvářeného fraktálu, je velice nízká.

Pokud chceme simulovat růst korálů, které rostou vzhůru, stačí když budou částice generovány v kvádru, který má šířku celého obrázku a výšku velmi malou. Tento kvádr je pak postupně posouván s růstem fraktálu (viz obrázek 3.23).



Obrázek 3.23: Obdélníková oblast, ve které se generují částice algoritmem DLA.

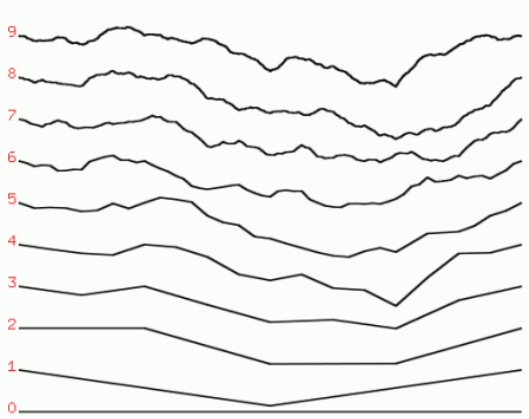
3.4.2 Metoda náhodného přesouvání středního bodu

Pravděpodobně nejznámější a nejpoužívanější metodou určenou pro generování stochastických fraktálů je metoda nazvaná *Náhodné přesouvání středního bodu* (random midpoint displacement) [6]. Tato metoda se používá pro tvorbu přírodní krajiny, povrchů vesmírných těles apod. Často se také používá v počítačových hrách a filmech.

Rekurzivní dělení úsečky

Nejlépe se princip MDM metody vysvětluje na obyčejné úsečce.

Vytváření obrazce začíná na jednoduché úsečce určité délky. U této úsečky se najde bod ležící uprostřed a posune se ve vertikálním směru o určitou hodnotu.Vznikne tak lomená čára, složená ze dvou úseček. Pro každou nově vzniklou úsečku se rekurzivně aplikuje stejný postup, dokud není dosaženo určitého počtu iterací (viz obrázek 3.24).



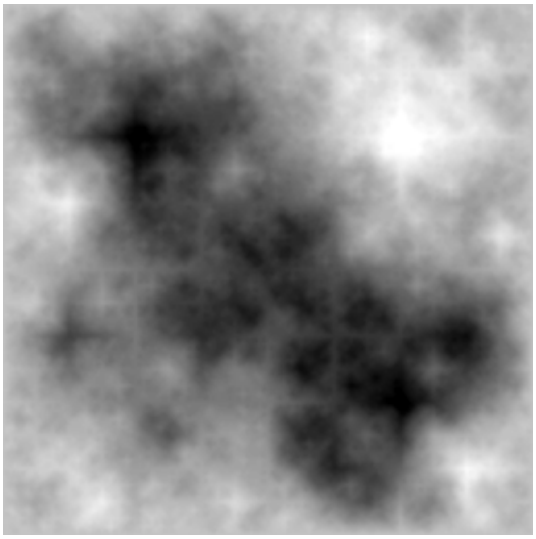
Obrázek 3.24: Průběh postupného dělení úsečky.

Rekurzivní dělení čtverce

Výše uvedené dělení úsečky je možné aplikovat i na čtverec.

Vše začíná u obyčejného čtverce. Na něm se naleznou pozice středů všech čtyř stran, podle kterých se najde i střed čtverce. Ten se pak posune v ose Z o určitou hodnotu, stejně jako v případě úsečky. Vzniknou tak další čtyři čtverce na které se rekurzivně aplikuje stejný postup.

Toto rekurzivní dělení čtverce je možné využít jak pro vytváření modelů krajin, tak pro tvorbu obrázků nazývaných - *plazma*. Čtverec je zde nahrazen rastrovým obrázkem, který obsahuje barvy v odstínech šedi. Souřadnice pixelů představují pozice bodu ve čtverci (osy x a y) a barva pixelu znázorňuje z-ovou souřadnici každého bodu. Obrázek je dělen stále na čtvrtiny, na které se aplikuje posunutí středu, dokud se nedosáhne úrovně pixelů. Zde se rekurzivní vnořování zastaví a obrázek se při zpětném vracení rekurzí obarví (viz obrázek 3.25).



Obrázek 3.25: Model plazmy vytvořený pomocí rekurzivního dělení čtverce.

Kapitola 4

Implementace aplikace

Úkolem této práce nebylo pouze základní přiblížení do problematiky fraktálů, ale také, jak už z názvu této práce vyplývá, vytvoření grafické aplikace s jednoduchých uživatelským rozhraním, která názorně demonstruje všechny podrobně vysvětlené typy fraktálů.

4.1 Systém FracViz

Název systému FracViz je odvozen od jeho účelu. Tedy **viz**uální výukový systém **frak**tálů. Systém slouží jako výukový nástroj. Jeho účelem je tedy přiblížit problematiku fraktálů široké veřejnosti a je koncipován takovým způsobem, aby celkem komplexní problematiku fraktálů uživatelé pochopili i bez větších technických znalostí, které s touto problematikou jistě souvisí. Celý system je založen na interaktivitě s uživatelem. Ten má tedy možnost měnit nejrůznější nastavení a systém okamžitě reaguje na změnu.

4.1.1 Popis aplikace

Po zpuštění aplikace je vidět rozdělení aplikace na dvě základní části.

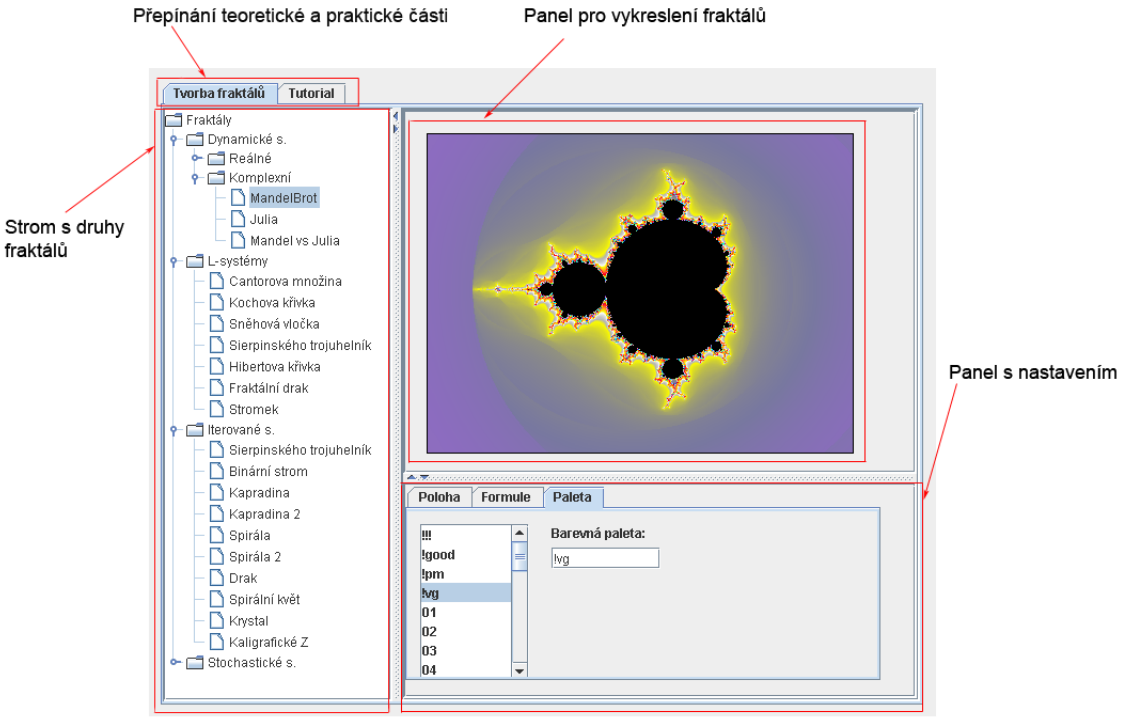
Teoretická část – Tato část kapitolu po kapitole postupně vysvětluje problematiku fraktálů od těch nejzákladnějších věcí, jakými jsou například soběpodobnost nebo dimenze fraktálů, přes o něco složitější dynamické systémy, až po stochastické fraktály, kterými je tutoriál zakončen.

Praktická část – Tato je rozdělena stejně jako část první na kapitoly a uživatel si v ní může ověřovat znalosti, které mu byly vysvětleny. Každá velká kapitola popisuje jeden druh fraktálů, pokud je kapitola širší, může být rozdělena i na více podkapitol a konkrétní příklady fraktálů jsou na nejnižší úrovni.

Obě části jsou v aplikaci umístěné v samostatné liště a přepínat mezi nimi se dá pomocí kliknutí na název lišty. Každá lišta je dále rozdělena na dvě části. Nalevo je umístěn strom s adresářovou strukturou, pomocí kterého se přepínají jednotlivé kapitoly a v pravé části je výukový obsah aplikace. V teoretické části je to stránka na které se zobrazují texty s vysvětlením dané problematiky a v praktické části je to panel, který vykresluje jednotlivé fraktály.

4.1.2 Práce s fraktály v praktické části

Jak již bylo řečeno, tato část se rozděluje na dvě hlavní části - strom s kapitolami a panel pro vykreslení. Pokud by ovšem aplikace pouze vykreslovala dané fraktály a neumožňovala jejich vlastnosti měnit, uživatel by jen těžko chápal podstatu vykreslování fraktálů. Část pro vykreslování fraktálů je tedy rozdělena také na dvě poloviny, kde horní polovina obsahuje samotný obrázek fraktálu a dolní polovina je plná všemožného nastavení, pomocí kterého může uživatel měnit podobu fraktálu. Podrobný popis panelů je uveden v uživatelské přiručce na konci práce (viz dodatek refprirucka).



Obrázek 4.1: Screenshot aplikace s vyznačenými oblastmi.

4.2 Způsob implementace

Systém byl implementován v programovacím jazyce *Java* (konkrétně ve verzi JAVA 2 SDK, Standart Edition 1.5.0 update 10). Důvodů proč byl tento jazyk vybrán je více. Nejvýznamnější je možnost zobrazit aplikace jako tzv. *applet* (viz [4]), který může být umístěn přímo na webovou stránku. Uživatel si tak nemusí vůbec nic instalovat a stačí, když do prohlížeče zadá adresu (samozřejmě musí mít nainstalovanou podporu Javy ve webovém prohlížeči, ale to je v dnešní době již samozřejmostí). Další velkou výhodou je nezávislost na platformě (na operačním systému). Program tedy pracuje stále stejně a je jedno jestli je spuštěný například ve Windows nebo Linuxu. Jazyk Java je také objektově orientovaný (viz [3]), což umožňuje rozdělit jednotlivé části aplikace na podčásti a zabývat

se pouze jimi. To také velice usnadňuje práci s grafickými komponentami, kterých je v tomto systému mnoho. Tyto komponenty byly implementovány pomocí grafických knihoven *AWT* a *JFC Swing* [4], které jsou součásti platformy *Java2SE*. Celá aplikace byla vytvořena v prostředí IDE NetBeans. Hlavním důvodem pro výběr tohoto prostředí byla snažší práce s grafickými komponentami, kterou například prostředí Eclipse vůbec nenabízí. Vzhledem k počtu těchto grafických kompenent (panelů, tlačítek, textových polí aj.) bylo nemyslitelné, vytvářet tyto komponenty ručně. IDE NetBeans mi v tomto ohledu velmi pomohlo, ale občas práci i docela stížilo. Podpora vytváření GUI v IDE NetBeans ještě není na takové úrovni, jako například ve Visual Studiu.

Pro vytváření složitých struktur bylo také zapotřebí použít jisté technologie, které s GUI nemají nic společného. Například celkem vélkým oříškem bylo upravit algoritmy pro vytváření Mandelbrotovi a Juliových množin pomocí komplexních čísel. Java totiž nemá implementovanou podporu komplexních čísel. Měl jsem dvě možnosti jak toto provést. Buď tuto podporu implementovat, což by později asi usnadnilo práci, ale protože tyto algoritmy jsou velice náročné na výpočetní výkon, bylo by toto řešení příliš pomalé. Rozhdl jsem se proto pro převod komplexních čísel na reálná. Algoritmy pak sice nejsou tolik názorné, ale jsou mnohem rychlejší.

Další technologie, kterou bylo nutné použít, jsou zasobníkové automaty pro vytváření L-systémů. Naštěstí Java tyto technologie podporuje, takže s tímto nebyl žádný problém.

Asi nejzajímavější je využití barevných palet pro vykreslení Mandelbrotovi množiny. Aby si uživatel mohl podle libosti měnit obarvení této množiny, implementoval jsem systém který z XML-souboru načte do HaspMapy (viz [2]) všechny barevné mapy, kde v každé mapě je 255 řádků a každý řádek představuje jednu barvu. Jakmile uživatel změní barevnou paletu kliknutím v panelu *Paleta* (viz dotatek A) je z této HaspMapy nahrána konkrétní paleta do třídy *ColorMap*, se kterou už třída generující mandelbrotovu množinu umí pracovat.

Ostatní typy fraktálů už využívají jen obyčejné v kládání pixelů do obrázku,kde barva pixelu je buď přednastavená, nebo je vypočítaná na základě počtu dané iterace.

Jednotlivé algoritmy, které jsou pro vykreslení obrazců použity jsou vysvětleny v teoretické části aplikace.

Kapitola 5

Závěr

Na závěr bych chtěl shrnout obsah a význam celé práce. Význam a rozsáhlost problematiky kolem fraktálů je tak široký, že není v silách jednoho člověka, vysvětlit do podrobna tuto problematiku na tak malém prostoru. Smyslem této práce tedy nebylo vyčerpávajícím způsobem vysvětlit vlastnosti a chování fraktálů, ale pouze je přiblížit uživateli a nabídnout mu východisko, pro případné další studium. Tato práce tento cíl splňuje a domnívám se, že v některých oblastech i převyšuje základní znalosti o fraktálech. Vysvětluje všechny základní druhy fraktálů a přibližuje čtenáři i oblasti, které s fraktály úzce nesouvisí (gramatiky).

Implementace webové aplikace podle mého názoru také převyšuje původní záměr, vytvořit jednoduchou aplikaci pro demonstraci fraktálů. Díky jejímu základnímu konceptu, tedy rozdělení na demonstrační a teoretickou část, si její uživatel může nejenom zkoušet, jak pracují jednotlivé fraktály, ale dozví se i všechno co potřebuje vědět, aby problematiku pochopil. Program tedy není vůbec závislý na této práci, protože teorie je obsažena i v něm.

Program umožňuje velice lehce a intuitivně nastavovat nejrůznější parametry pro vykreslení a dále využívá i externí soubory, ze kterých načítá tabulky barev (barevné palety) pro vybarvovaní mandelbrotovy množiny a juliových množin.

Jako rozšíření pro tento program by mohla být implementace dalších druhů fraktálů, zdokonalení algoritmu pro vykreslení a přidání další interaktivity pro uživatele, jako například vlastní vytváření barevných palet aj.

Aplikace by mohla být využívána i pro studijní účely, jako demonstrační aplikace na cvičeních počítačové grafiky a jiných předmětů z podobnou tématikou.

Literatura

[1] Michael F. BARNSLEY. *Fractals everywhere.* Academic Press, San Diego, 2 edition, 1993.

[2] Pavel HEROUT. *JAVA - Bohatství knihoven.* nakladatelství Kopp, České Budějovice, 1 edition, 2003.

[3] Pavel HEROUT. *Učebnice jazyka JAVA.* nakladatelství Kopp, České Budějovice, 1 edition, 2003.

[4] Pavel HEROUT. *JAVA - Grafické uživatelské prostředí a čestina.* nakladatelství Kopp, České Budějovice, 1 edition, 2006.

[5] Marek ČANDÍK Ivan ZELINKA, František VČELAŘ. *FRAKTÁLNÍ GEOMETRIE - principy a aplikace.* nakladatelstvi BEN - technická literatura, Praha, 1 edition, 2006.

[6] Petr FELKEL Jiří ŽÁRA, Bedřich BENEŠ. *Moderní počítačová grafika.* Computer Press, a.s., Brno, 2 edition, 2005.

[7] Benoît MANDELBROT. *Fraktály : tvar, náhoda a dimenze. / Benoît Mandelbrot.* Mladá fronta, Praha, 1 edition, 2003.

[8] Aristid LINDENMAYER Przemyslaw PRUSINKIEWICZ. *The Algorithmic Beauty of Plants.* Springer-Verlag, NY, 1 edition, 1990.

[9] Pavel TIŠNOVSKÝ. Seriál fraktály v počítačové grafice. [online], ©2005-2007. http://www.root.cz/serialy/fraktaly-v-pocitacove-grafice/, [cit. 2007-10-05].

Dodatek A

Uživatelská příručka

Systém FracViz funguje ve webovém prohlížeči s pluginem Java2SE (Java Virtual Machine) verze 1.5.0. Aplikace je dostupná na internetové stránce:

http://eva.fit.vutbr.cz/ xfried03/fracviz/Fracviz.html

Ovládání systému je velice jednoduché a intuitivní. Popis aplikace jako celku je uveden v implementaci (viz oddíl 4.1.1). Zde budou uvedeny jen podrobné popisy jednotlivých panelů pro nastavení parametrů jednotlivých fraktálů a dále interaktivní odezva chování systému na práci s myší a klávesnicí.

A.0.1 Panel nastavení

Panel s nastavením je rozdělen na pět části. Každá část obsahuje množství různých parametrů, které uživatel může měnit. Aby se změna projevila v obrázku, musí se buď kliknout na tlačítko použít nebo stisknout klávesu Enter. Některé panely obsahují tzv. spinner (textové pole s šipkami pro změnu obsahu), který automatický mění podobu fraktálu bez stisku klávesy Enter.

Formule – Zde je možné nastavovat různé parametry, které jsou potřeba při výpočtu, a které většinou významně změní výsledný tvar fraktálu. Jednotlivé druhy fraktálů mají ale různé požadavky a tento panel se tedy podle druhu vykreslovaného fraktálu mění. Popis jednotlivých parametrů je uveden v teoretické části příslušného druhu fraktálů.

Poloha – Tento panel slouží k změně velikosti obrázku (parametry šířka a výška), a dále ke změně polohy a velikosti útvaru (parametry poloha a přiblížení).

Paleta – Tento panel slouží ke změně barevné palety, pomocí které se některé fraktály vykreslují. Konkrétně se jedná o dynamické systémy v komplexní množině. Pro jiné druhy fraktálů tento panel není viditelný.

L-Systemy – Zobrazuje parametry gramatiky, kterou používá želví grafika L-systémů. Dále zobrazuje úhel natočení želvy a nejduležitějším prvkem je spinner, pomocí kterého se dá nastavovat počet iterací pro vykreslení. Tento panel je viditelný pouze u L-systémů.

Stoch. systemy – Na tomto panelu jsou zobrazena nastavení pro vykreslení stochastických systémů. Je samozřejmě viditelný pouze i nich.

A.0.2 Panel pro vykreslení

Pokud uživatel nechce parametry nastavovat v panelu nastavení, může některé z nich měnit přímo na vykreslovaném obrazci. Ve vykreslovacím panelu je možné měnit polohu obrázku a jeho přiblížení. Panel podporuje dva způsoby, jak toho dosáhnout.

1. pomocí myši
   - levým polohovacím tlačítkem myši se mění poloha obrázku a to tak, že se myší klikne na nějaké místo na obrázku (tlačítko se musí stále držet) a obrázek pak kopíruje pohyby myši. V momentě, kdy se tlačítko pustí, obrázek se ustálí na dané pozici.
   - pravým polohovací tlačítkem myši se mění přiblížení daného obrázku. Myší se klikne na nějaké místo na obrázku (na pozici nezáleží). Pokud se myší pohne nahoru, velikost přiblížení se sníží, pokud se myší pohne směrem dolů, přiblížení se zvýší.
2. pomocí klávesnice
   - poloha obrázku se mění pomocí směrových šipek a velikost přiblížení pomocí kláves `Page Up` a `Page Down`. Pokud jsou klávesy drženy déle obrázek se stále mění. Aby změna pomocí kláves fungovala, musí se přes panel alespoň jednou přejet myší. Tím se tato funkce aktivuje. Je to proto, že klávesy se dají použít i v jiných částech aplikace (přepínaní lišt apod.).