VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

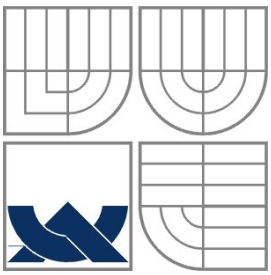
FRAKTÁLY A JEJICH APLIKACE V POČÍTAČOVÉ GRAFICE

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

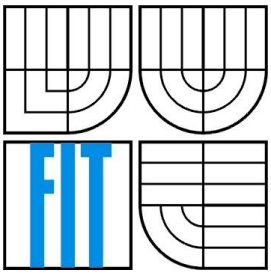
AUTOR PRÁCE
AUTHOR

MARTIN TESAŘ

BRNO 2009



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

FRAKTÁLY A JEJICH APLIKACE V POČÍTAČOVÉ GRAFICE

FRACTALS AND THEIR APPLICATIONS IN COMPUTER GRAPHICS

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

MARTIN TESAŘ

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. JIŘÍ KOUTNÝ

BRNO 2009

Abstrakt

První část bakalářské práce zahrnuje teorii fraktálů, vysvětluje Hausdorffovu dimenzi a obsahuje průřez uplatnění fraktálů v počítačové grafice. V druhé časti je zvolena jejich oblast užití, kterou se stala interaktivní tvorba fraktálu typu „flame fractal“. Je zde dopodrobna popsána matematická podstata systémů iterovaných funkcí, ze kterých algoritmus „flame fractal“ vychází. Na matematický popis navazuje rozbor originálního algoritmu na tvorbu výše zmíněného fraktálu. V práci jsou nastíněny metody vylepšení obrazu a navrhnuta nová rozšíření.

Abstract

First part of this bachelors thesis consists of fractal theory, Hausdorff dimension and their applications in computer graphics. In second part is choosen idea for application. It is interactive creation of flame fractals. In this section, there is defined mathematical system of iterated functions, that is the heart of flame fractals. After mathematical examination follows analysis of original algorithm for creation of flame fractals. Thesis continues with methods for image improvement and suggests new enhancements.

Klíčová slova

Fraktál, soběpodobnost, Hausdorffova dimenze, systémy iterovaných funkcí, flame fractal, interaktivní tvorba fraktálů, transformační matice, konvoluce, supersampling

Keywords

Fractal, self-similarity, Hausdorff dimension, iterated function system, flame fractal, interactive fractal creation, transformation matrix, convolution, supersampling

Citace

Martin Tesař: Fraktály a jejich aplikace v počítačové grafice, bakalářská práce, Brno, FIT VUT v Brně, 2009

Fraktály a jejich aplikace v počítačové grafice

Prohlášení

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

……
Martin Tesař
13. května 2009

Poděkování

Rád bych poděkoval vedoucímu bakalářské práce panu Ing. Jiřímu Koutnému, který mi poskytl odbornou pomoc a neocenitelné rady, díky kterým mohla tato práce vzniknout.

Obsah

1 Úvod

Fraktály se v počítačové grafice těší stále větší pozornosti. Nebýt počítačové grafiky dalo by se říci, že by neexistovala ani fraktální geometrie. Je to zajímavý poznatek, jelikož fraktální objekty byly objeveny již v dobách, kdy počítače ještě neexistovaly. Vlastně nás fraktální objekty obklopují odjakživa, mraky, sněhové vločky, rostliny, ráz krajiny, všechny tyhle objekty a je jich ještě více, mají fraktální podstatu. Fraktály sice obvykle mají jednoduchý popis, ale jejich konstrukce je pro jednoho člověka velice zdlouhavá. Ilustračním příkladem nám může být francouzský matematik Pierre Fatou, který již v roce 1905 definoval „zvláštní" množinu a pokusil se jí ručně vykreslit, bohužel to bylo nad jeho síly. Až Benoit B. Mandelbrot o několik desítek let později vytvořil pomocí výpočetní techniky první obrázky této množiny a položil tak základy fraktální geometrie. Nová vědní disciplína se nevídaným tempem rozšířila do mnoha oblastí moderní vědy. Její základ však zůstává i nadále v počítačové grafice, kde nalezla velká uplatnění a kterou se má práce zabývá.

V kapitole druhé se seznámíme se základními pojmy fraktální geometrie, bez které by jen stěží bylo možné chápat složité zapeklitosti fraktální geometrie. Osvojíme si pojem Hausdorffovy dimenze, jejíž existence je hlavním indikátorem fraktálových objektů. V závěru kapitoly si nastíníme pojmy soběpodobnosti a atraktoru.

Třetí kapitola nás provede mezi základními typy fraktálů. Pochopíme v ní, jak je fraktální geometrie rozsáhlou vědní disciplínou. U každého typu si probereme jeho charakteristiku a pro ilustraci si fraktály ukážeme na několika vybraných obrázcích.

Uplatnění fraktálů v počítačové grafice nám předvede čtvrtá kapitola. Dozvíme se, že fraktály jsou dobrým prostředkem komprimace obrazu, též i v rozpoznávání obrazu. Dále, že jsou nepostradatelné při tvorbě rázu krajin, což mimochodem bylo jedno z jejich prvních uplatnění. Ukážeme si, že fraktály nejsou jen aparátem pro vědce, ale že i prostý člověk si v nich může najít své a to zejména v obrazcích, které spadají do odvětví fractal art. Právě vytváření těchto obrázků se práce z velké šíře věnuje.

Pátá kapitola je matematickým odrazovým můstkem pro interaktivní vytváření systémů iterovaných funkcí. Také obsahuje popis použitého algoritmu, který je postaven na náhodných číslech a je označován chaotickou hrou.

V následující šesté kapitole si probereme dopodrobna typ algoritmu na tvorbu fraktálů typu „flame fractal", který je zobecněním systémů iterovaných funkcí. Tímto fraktálem a jeho interaktivní tvorbou se má práce zabývá.

Jelikož se práce věnuje i kvalitě grafického výstupu je zde zahrnuta kapitola metody vylepšení obrazu, ve které si popíšeme vybrané metody, které jsem si zvolil pro moji aplikaci.

Osmá kapitola pojednává o hledání rozšíření fraktálu typu „*flame fractal*“ a nastiňuje mnou navržená rozšíření, které jsou v aplikaci implementovány. Text je doprovázen obrázky, které demonstrují použitá rozšíření.

Devátá kapitola je zasvěcena aplikaci MyFlame, která je určena k interaktivní tvorbě fraktálu popsaného v šesté kapitole. Je zde popsán postup mojí práce a její následná implementace spolu s navrženými rozšířeními.

Celou práci uzavírá desátá kapitola, která hodnotí dosažené výsledky a nastiňuje další možný vývoj celého projektu.

2 Základní pojmy

Dříve než se seznámíme s pojmem *fraktál* [10], je nutné pochopit pojmy:

- topologická dimenze
- fraktální dimenze

2.1 Topologická dimenze

Jedná se o dimenzi, se kterou se běžně setkáváme v Euklidovské geometrii. Tato dimenze je celočíselná. Lze jí popsat *geometrické hladké objekty* jako jsou např. úsečka, kruh, kvádr. Zjednodušeně lze říci, že topologická dimenze určuje počet parametrů, který je nutný pro popis bodu v tělese [17]. Například bod má dimenzi nulovou, jelikož pro popis bodu v něm samém není nutný žádný parametr.

Křivky mají obecně dimenzi rovnu jedné, pro popis polohy bodu na křivkách postačuje jeden parametr. Pro názornější ukázku ve funkci sinus:

$$y = \sin(t) \tag{2.1}$$

postačuje jediný parametr t, který jednoznačně určuje polohu na této křivce. Křivky je možné popsat v prostoru, přičemž počet parametrů nutný pro popis bodu uvnitř se nezmění. Dále je definována jejich délka, která může být i nekonečná, avšak jejich plocha zůstává nulová (jsou nekonečně tenké). Pro popis hladké plochy je proto nutné přidat jeden parametr navíc, tedy dimenze jakékoliv hladké plochy je rovna 2. Přidáním dalšího parametru lze popsat třídimenzionální hladká tělesa, jako je například koule.

2.2 Hausdorffova-Besicovitchova dimenze

Ve *fraktální geometrii* [16] si nevystačíme s topologickou dimenzí. Fraktály nejsou útvary, které lze vyjádřit celočíselnou dimenzí. Tyto objekty popisuje fraktální dimenze, přesněji nazývaná *Hausdorffova-Besicovitchova dimenze* [10] (přesná definice pochází od Hausdorffa 1919 a propracoval ji Besitcovitch). Hodnota této dimenze je neceločíselná a udává úroveň členitosti objektu (jak rychle roste délka, plocha, či další odpovídající veličina daného rozměru, do nekonečna). Poprvé na obdobnou neceločíselnou konstantu narazil Lewis Frye Richardson(1961) [10], který měřil délku pobřeží ostrova Korsiky. Zjistil, že zvětšováním měřítka se také prodlužuje délka pobřeží. Ovšem této konstantě (která závisela na zvoleném úseku pobřeží) nevěnoval větší pozornost. Později až

B. Mandelbrot zjistil, že Richardsonovou konstantou je Hausdorffa-Besicovitchova dimenze a začal používat její vzorec pro výpočet dimenze fraktálových objektů [10]:

$$D = \frac{\log(n)}{\log(1/r)} \tag{2.2}$$

kde

1/r .. měřítko

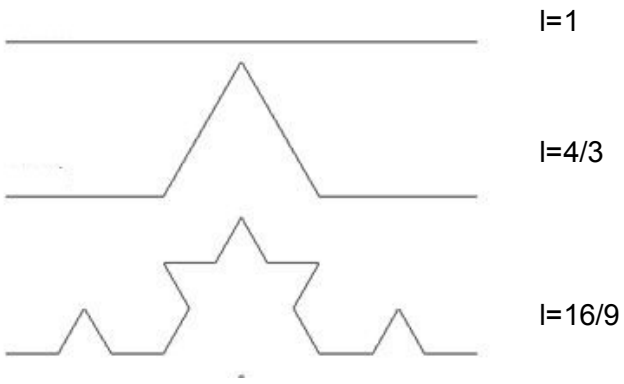
n .. počet částí ne které je útvar rozložitelný.

2.2.1 Výpočet Hausdorffovy dimenze

Jako příklad výpočtu uvedu výpočet fraktálové dimenze pro Kochovu křivku [17], která je jedna z prvně známých fraktálních křivek. Pojmenována je po švédském matematikovi Helge von Kochovi, ten ji poprvé popsal v roce 1904.

Sestrojení této křivky je jednoduché, postupujeme ve třech krocích, které donekonečna aplikujeme:

1. Úsečku rozdělíme na třetiny.
2. Nad prostřední třetinou sestrojíme rovnostranný trojúhelník.
3. Základnu trojúhelníku (bývalou prostřední třetinu úsečky) odstraníme.



Vidíme, že délka von Kochovy křivky roste do nekonečna. Do vzorce pro výpočet fraktální dimenze dosadíme hodnotu n, která je v našem případě rovna počtu úseček, ze kterých je křivka tvořena, tedy n = 4. A nový dílek je třetinou původní délky, tedy r = 1/3.

Vlastní výpočet:

$$D = \frac{\log(n)}{\log(1/r)} = \frac{\log(4)}{\log(3)} = 1{,}2618595$$

Hausdorffa-Besicovitchova dimenze Kochovy křivky je poté přibližně 1,26.

2.3 Fraktál

Pojem *fraktál*, či *fraktálový objekt* [10] poprvé zavedl Benoit B. Mandelbrot, který se těmito útvary zabýval. Název vytvořil z latinského adjectiva *fractus* (v překladu rozlámaný, rozbitý). Nejjednodušeji lze fraktál popsat jako nekonečně členitý útvar. Fraktály se také vyznačují soběpodobností (viz oddíl 2.3). Už jsme si popsali Hausdorffovu dimenzi, takže si můžeme fraktál definovat matematicky, stejně jak to udělal Benoit B. Mandelbrot [10]:

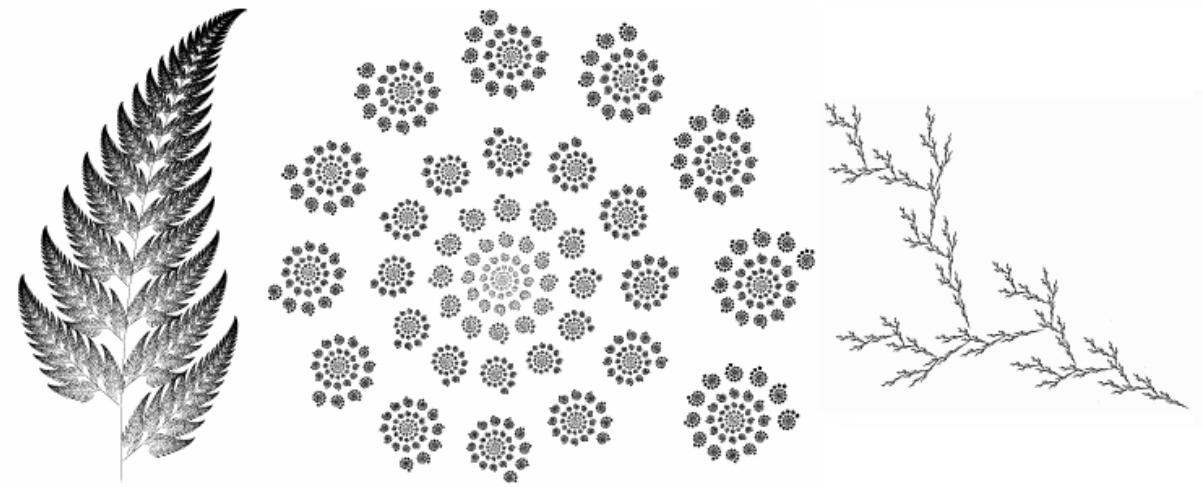
„*Fraktál je množina či geometrický útvar, jejíž Hausdorffova dimenze je (ostře) větší než dimenze topologická* .“

Fraktálové objekty byly známy už před B. Mandelbrotem, jako například *Sierpinského trojúhelník* [6], nebo výše zmíněná Kochova křivka, avšak Mandelbrot byl první, kdo jim dal společné jméno. Jeho objevem se také začala rozvíjet nová vědecká disciplína *fraktální geometrie*. Fraktály najdete i v přírodě, výše zmiňované pobřeží ostrovů, mraky, stromy, blesky, či vločky vykazují fraktální vlastnosti.

Obrázek 2.1: Romanesco broccoli „věžatý květák“ vykazuje fraktální vlastnosti. Převzato z http://www.beart.org.uk/Emergent/.

2.4 Soběpodobnost

Mezi další důležité fraktální vlastnosti patří *soběpodobnost (self-similarity)* [7][17]. V matematice více známa jako *invariace vůči změně měřítka*. Soběpodobnost je vlastnost, která zajišťuje, že objekt na který se díváme, vypadá podobně v jakémkoliv měřítku. Tedy soběpodobná struktura je taková struktura, kterou je možné rozložit na struktury, z nichž každá je zmenšenou kopií originálu [7]. Ačkoliv se jedná o vlastnost, která je pozorována na fraktálech, soběpodobnost není podmínkou postačující k identifikaci fraktálního charakteru, jelikož existují i jiná soběpodobná tělesa [7]. Klasickými soběpodobnými fraktály je například Barnsleyho kapradina (viz kapitola 5) .



Obrázek 2.2: Ukázky soběpodobných fraktál. Převzato z http://www.elektrorevue.cz/clanky/03019/kap_1.htm.

2.5 Atraktor

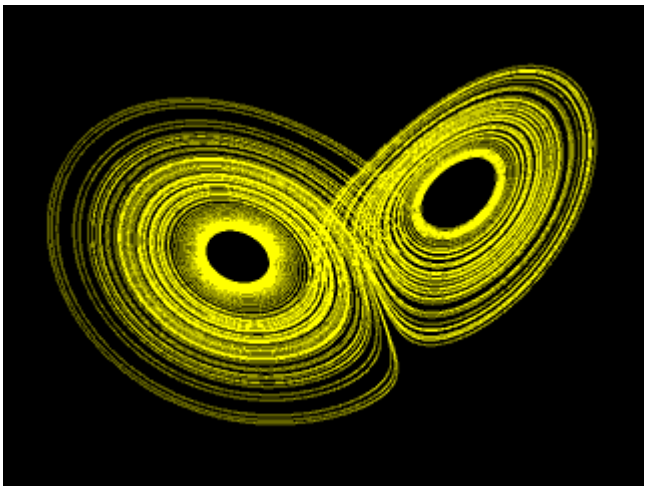
Atraktor [11] je stav, do kterého se dynamický systém dostane po dostatečně dlouhé době, kde dostatečná doba může být rovna nekonečnu. Slovo pochází z anglického pojmu *to attract* = přitahovat. Geometricky se může jednat o bod, křivku, nebo jakýkoliv jinak složitý útvar. Více studiem atraktorů se zabývá teorie chaosu [6].

Atraktory dělíme na [6]:

- Fixní body
- Cyklické atraktory
- Podivné atraktory

Příkladem atraktoru jako fixního bodu může být například stav, do kterého se dostane kyvadlo, které třením vzduchu zpomaluje. V tomto případě je atraktorem těžiště. Může ovšem nastat případ,

kdy není možné odhadnout, jak se bude systém chovat, v takovém případě je atraktor chaotický a může se jednat o takzvaný *podivný atraktor (strange attractor)* [6]. Mezi systémy, které vykazují takové vlastnosti lze zařadit modely atmosféry. Podivné atraktory vykazují fraktální vlastnosti, takže se jedná o fraktály. Nejznámějším podivným atraktorem je *Lorenzův atraktor* (viz Obrázek 2.1), který připomíná motýlí křídla a jež má Hausdorffovu dimenzi mezi 2 a 3.



Obrázek 2.1: Dynamický systém Lorenzův atraktor, převzato z [6].

3 Typy fraktálů

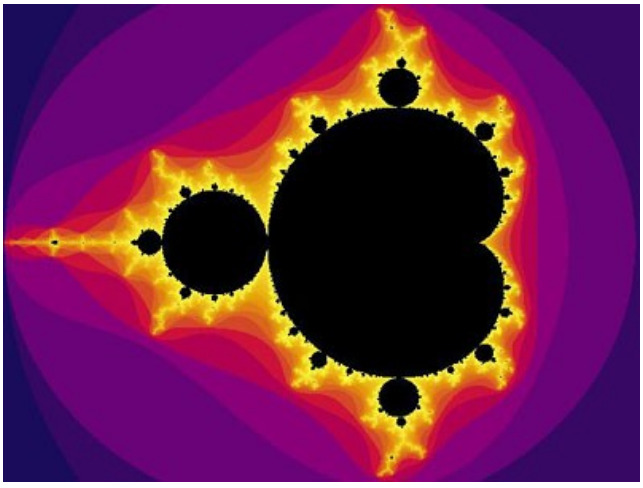
Za dobu existence fraktální geometrie docházelo k základnímu členění fraktálů do jednotlivých kategorií, vyhradily se základní typy. V počítačové grafice je důležitý fakt, že každý typ má podobné algoritmy pro jejich generování, avšak jednotlivé fraktální obrazce vygenerované stejnými algoritmy se liší. Nejhrubší dělení fraktálů je následující [14]:

- L-systémy
- Systémy iterovaných funkcí
- Dynamické systémy
- Stochastické fraktály

V následujících kapitolách uvedu základní poznatky těchto kategorií a později se zaměřím na *systémy iterovaných funkcí* [2], kterých se má práce dotýká.

3.1 Dynamické systémy

Nejrozšířenější kategorií fraktálů jsou s největší pravděpodobností *dynamické systémy* [11], které mají pravděpodobně i největší uplatnění v praxi (např. studium růstu populace bakterií). Dynamický systém je matematický model závislý na určité veličině. Každý dynamický systém je determinován svými počátečními podmínkami, proto se jedná o systém deterministický [11]. Tyto systémy jsou dále určeny dynamickými podmínkami popisující změnu systému v čase. Stav dynamického systému v čase je pak definován stavovým vektorem [11][14]. Dynamické podmínky často bývají zadávany jako diferencialní rovnice. Změna stavu systému je tedy nahrazení starého stavového prostoru novým právě za použití diferenciálních rovnic.



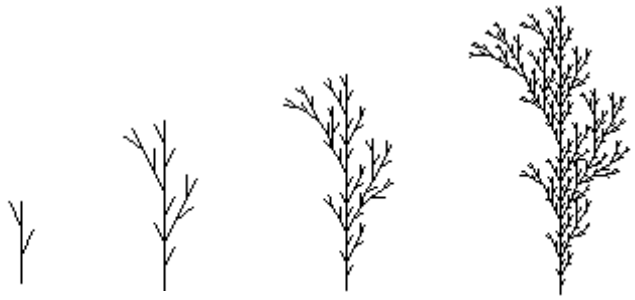
Obrázek 3.1: Mandelbrotova množina,
převzato z http://explorationscience.org/Articles/Chaos.htm.

Dynamické systémy existují i v komplexní rovině, mezi tyto systémy patří asi nejznámější fraktály, jako jsou Juliovy množiny [16], či Mandelbrotova množina (viz Obrázek 3.1), která je jedním ze symbolů počítačové grafiky.

3.2 L-Systémy

L-Systémy, neboli *Lindenmayerovy systémy* [12], jsou systémy tvořené na základě přepisovacích gramatik. Každému nonterminálním symbolu v gramatice je přiřazen geometrický význam a na tomto základě jsou vytvářeny fraktální obrazce.

L-systémy umožnují generovat plošné, či trojrozměrné objekty, které jsou velice podobné přírodním útvarům, jako jsou stromy, rostliny a jiné přírodní útvary. Obzvláště přidá-li se do algoritmů na jejich tvorbu náhoda, pak vznikají objekty velmi podobné reálným útvarům. Na stejném principu lze též vytvářet modely růstu rostlin.



Obrázek 3.2: Ukázka vygenerovaného závorkového L-systému popsaného řetezcem:
w: F, p: F -> F[-F]F[+F][F]
převzato z http://www.biologie.uni-hamburg.de/b-online/e28_3/lsys.html.

3.3 Systémy iterovaných funkcí

Název systémy iterovaných funkcí pochází z anglického *iterated function system* (IFS) [2][14]. Jde o generativní metodu, kterou publikoval Demko (1985) a Barnsley (1987) [14]. Jedná se vlastně o obrazce pokryté menšími kopiemi tohoto obrazce, takže jsou tyto fraktální objekty plně soběpodobné.

Soběpodobnost je klíčová vlastnost těchto fraktálů. Využívá se toho, že lze najít takové zobrazení, které mapuje celek na jednotlivé části. Pro jejich generování se využívá lineárních transformací, mezi něž patří zkosení, změna měřítka, posunutí a převrácení [7]. Transformace ovšem musí být *kontrakcemi* (viz oddíl 5.1), jinak by systém nebylo možné zobrazit, protože by divergoval [14]. Typickým IFS fraktálem je Barnsleyho kapradina (viz obr. 5.1).

IFS systémy našly použití v praxi, hlavně v počítačové grafice při modelování 3D scén počítačových her, nebo také v oboru komprese obrazu (viz kapitola 4).

Nastínil jsem zde základní pojmy IFS systémů, podrobněji se je pokusím přiblížit v šesté kapitole, jelikož moje práce úzce souvisí právě s těmito systémy.

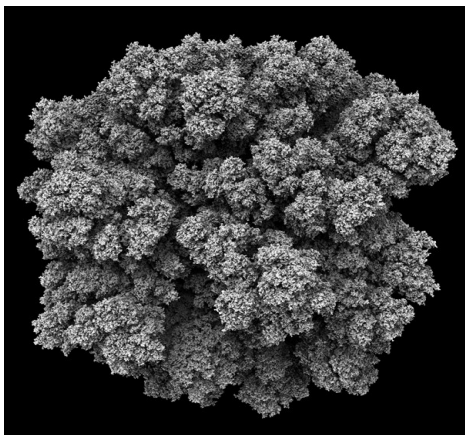
3.4 Stochastické fraktály

Stochastické (náhodné) fraktály [17], jsou další skupinou fraktálů. Název nám napovídá, že při jejich generování bude hrát velkou roli náhoda. Na rozdíl od IFS fraktálů tvoří stochastické fraktály objekty velmi podobné reálným útvarům v přírodě. IFS fraktály jsou sice schopné generovat přírodní útvary, ale jsou pravidelné, což se v přírodě málo kdy vidí. Proto použitím náhody při generování těchto objektů, dosáhneme vetší nepravidelnosti. Výsledné fraktály se po té už více podobají realitě. Zavedením náhody tyto objekty ztrácí jednu fraktální vlastnost, kterou je soběpodobnost, ale objevuje se u nich nová vlastnost, *soběpříbuznost* [17].

Stochastické fraktály lze generovat různými způsoby:

- Simulace Brownova pohybu
- Metodou přesouvání prostředního bodu

První způsob, simulace Brownova pohybu *(Brownian motion)* [17] je nejvhodnější pro generování toků řek, nebo také pro vytváření difúzních obrazců. Metoda přesouvání prostředního bodu (*midpoint displacement*) [17] zaujala své místo v počítačové grafice převážně při generování náhodných přirodních krajin, zejména hor. Změnou odchylky přesouvaného prostředního bodu, lze zcela změnit ráz generovaného výškového pole.

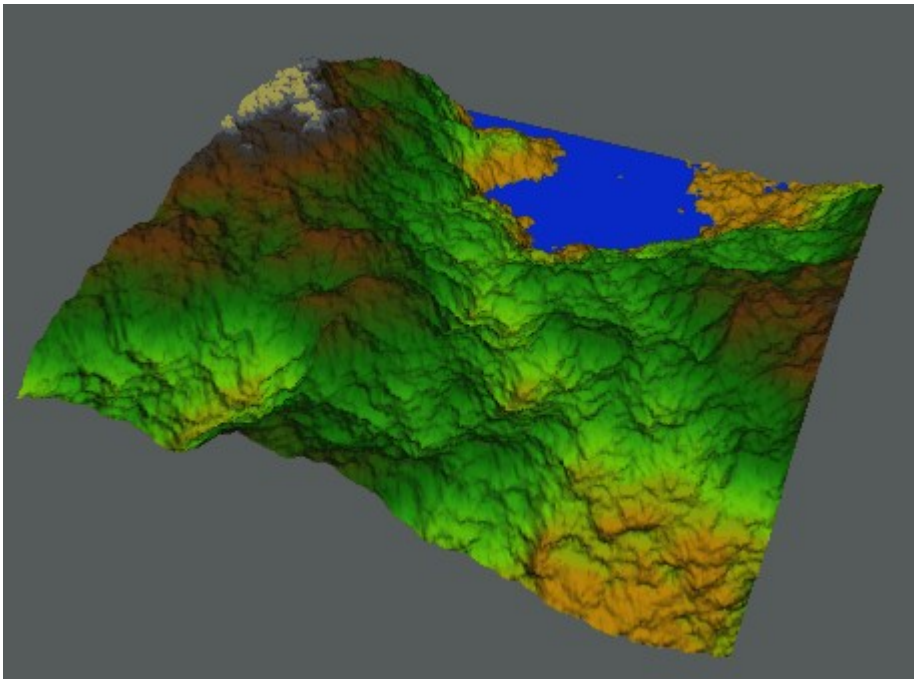


Stochastický fraktál vygenerovaný algoritmem DLA (technika založená na Brownovu pohybu), vytvořený Andy Lomasem (tvůrce visuálních efektů ve filmech jako je například Matrix). Převzato z http://www.andylomas.com/.

4 Využití fraktálů v počítačové grafice

Fraktály v dnešní době tvoří v počítačové grafice nezastupitelné místo. Sama počítačová grafika dává vědcům aparát pro studium fraktálů a hrála při jejich objevování podstatnou roli. Z těchto nových poznatků počítačová grafika těží. V následujících odstavcích uvedu základní uplatnění fraktálů v tomto oboru.

Jako jeden z hlavních přínosů fraktální geometrie vidím možnost generování přírodních útvarů. Stromy, ráz krajiny, či oblaka představují pro počítačovou grafiku velké množství dat, které fraktální geometrie napomáhá vysoce redukovat. Nevýhodou je nemožnost předem navrhnout výsledný tvar objektu, jelikož fraktály lze takto navrhovat jen velmi těžko, takže grafikům nezbývá nic jiného, než experimentovat s těmito objekty metodou pokus omyl. Ovšem při vhodně zvolených parametrech dochází ke kýženým výsledkům. Fraktály se staly nezbytnou součástí CAD systémů, které jsou podstatnou součástí počítačové grafiky. Pro tyto vlastnosti fraktály též pronikly do tvorby počítačových her.



Obrázek 8.1: Fraktální krajina vytvořená metodou přesouvání prostředního bodu, mechanismus vyvinutý Fournierem, Fusselem a Carpenterem, převzato z http://scienceblogs.com/goodmath/.

Další odvětví tvoří komprimace obrazu. V 80. letech minulého století se naskytla myšlenka, že pokud jde popsat obraz fraktální geometrií, nešlo by to opačně využít při kompresi obrazu? [9] Michael Barnsley (zakladatel IFS fraktálů), byl první kdo vyřešil tento inverzní problém, tj. nalezení

soustavy jednoduchých lineárních funkcí popisujících rastrový obraz. Postup si patentoval v roce 1987. Jeho první pokusy fraktální komprese potřebovaly dlouho dobu generování (až 100h) [9], vysoce výkonné počítače a znalého člověka, který by tento proces řídil, dosáhl tak ale vysoké komprimace 1:10 000. Postup byl však pro jeho náročnost těžko aplikovatelný. O rok později Barnsleyho student upravil jeho algoritmus tím, že se nehledala věrná kopie celého obrazu, ale jen jeho části. Modifikovaný algoritmus byl nazván Partitioned IFS [9]. Bohužel došlo ke zhoršení kompresního poměru, který se po té pohyboval kolem 8:1 až 50:1. všechny nynější algoritmy jsou založené na tomto modifikovaném algoritmu, který je též v patentovém vlastnictví [15] Michaela Barnsleyho.

Obrázek 8.2: Demonstrace principu metody Partitioned IFS (hledání soběpodobných částí v obraze), převzato z http://classes.yale.edu/Fractals/Panorama/ManuFractals/ImageCompression/.

Ve zpracování obrazu našly fraktály uplatnění i v rozpoznávání obrazu [14]. Ukázalo se, že metoda fraktální komprese je vhodnou při hledání určitých struktur v rastrovém obrazu. Metod rozpoznávání obrazu existuje celé množství, avšak až fraktální geometrie vnáší možnost hledání geometrických útvarů nezávislých na jejich měřítku a orientaci.

Velké uplatnění mají fraktály též v takzvané výtvarné počítačové grafice v odvětví *fractal art*. Jejich tvary a rozmanitost fascinují oko člověka a jsou nám líbivé. Dále je možné vytvářet animace fraktálů postupnou změnou jejich parametrů, či přidáním dalšího rozměru, kterým je čas, což je používaná metoda při animaci třídimenzionálních fraktálů.

5 Systémy iterovaných funkcí

Několik základních vlastností systému iterovaných funkcí jsem nastínil již v kapitole 3. V této kapitole se na IFS fraktály zaměříme podrobněji, jelikož jejich teorie je nutná k pochopení fraktálu se jménem „Flame fractal“, kterému se tato práce věnuje.

5.1 Matematická definice

Popis IFS systémů vychází z matematické teorie pevného bodu [1].

Nechť U je metrický prostor s metrikou d. Dále, nechť f je funkce:

$$f : A \rightarrow U \, , A \subset U \tag{5.1}$$

Jestliže má funkce f bod x_0 , pro který platí:

$$f(x_0) = x_0 \tag{5.2}$$

pak se tento bod x_0 nazývá *pevný bod*.

Jestliže je množina $A \subset U$ a f je funkce $f: A \rightarrow A$, přičemž existuje určitá konstanta δ, pro kterou platí $0<\delta<1$, potom:

$$d[f(x), f(y)] < \delta d[x, y] \tag{5.3}$$

tato funkce f se nazývá *kontrakce* a konstanta δ *kontrakčním faktorem* [14].

Symbol d zde znamená vzdálenost, z toho vyplývá, že systémy iterovaných funkcí mohou být definovány nad prostorem, ktery má definován pojem vzdálenosti.

IFS systém lze po té definovat vektorem kontrakcí $\mathbf{\Phi}=\{\varphi_1, \varphi_2, \ldots \varphi_n\}$ a k nim přidruženým vektorem pravděpodobností $\mathbf{\Pi}=\{\pi_1, \pi_2, \ldots \pi_n\}$., který při generování určuje pravděpodobnost jejich užití [3].

5.2 Transformační matice

Transformační matice jsou základní stavební kámen pro vytváření IFS fraktálů, velkou roli hrají při interaktivní tvorbě těchto fraktálů [14]. Uživatel má vzorový objekt, a může vytvářet jeho kopie, které jsou různé transformace vzoru. Právě pomocí nich jsou v paměti uchovávány jednotlivé kontrakce.

5.2.1 Afinní transformace

IFS fraktály jsou definovány jako množina kontrakcí (viz oddíl 5.1). V praxi je tato množina nejčastěji specifikována jako množina afinních transformací splňující podmínku kontrakce. Afinní transformace patří mezi lineární transformace, to znamená, že jejich aplikací např. na úsečku vznikne zase úsečka. Mezi afinní transformace patří zkosení, změna měřítka, posunutí a otočení, nebo operace vzniklé jejich skládáním.

Ve dvojrozměrném prostoru je definována vztahem [2][7]:

$$w(x)=w\begin{pmatrix}x_1\\x_2\end{pmatrix}=\begin{pmatrix}r_1\cdot\cos(\varphi)-r_2\cdot\sin(\upsilon)\\r_1\cdot\sin(\varphi)\,r_2\cdot\cos(\upsilon)\end{pmatrix}\cdot\begin{pmatrix}x_1\\x_2\end{pmatrix}+\begin{pmatrix}e\\f\end{pmatrix} \tag{5.4}$$

resp.

$$w(x)=w\begin{pmatrix}x_1\\x_2\end{pmatrix}=\begin{pmatrix}a\,b\\c\,d\end{pmatrix}\cdot\begin{pmatrix}x_1\\x_2\end{pmatrix}+\begin{pmatrix}e\\f\end{pmatrix} \tag{5.5}$$

kde úhel φ určuje otočení kolem osy x, úhel v otočení kolem osy y, parametr r_1 udává změnu měřítka ve směru osy x, parametr r_2 změnu měřítka ve směru osy y. Parametry e a f udávají posun. Ve druhém vzorci došlo ke zjednodušení, které je nutné pro vytvoření následující transformační matice:

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

Příklady základních transformačních matic:

$$T_1=\begin{bmatrix}1 & 0 & 0\\0 & 1 & 0\\0 & 0 & 1\end{bmatrix},T_2=\begin{bmatrix}1 & 0 & 0\\0 & 1 & 0\\d_x & d_y & 1\end{bmatrix},T_3=\begin{bmatrix}\cos(\alpha) & -\sin(\alpha) & 0\\\sin(\alpha) & \cos(\alpha) & 0\\0 & 0 & 1\end{bmatrix},T_4=\begin{bmatrix}S_x & 1 & 0\\1 & S_y & 0\\0 & 0 & 1\end{bmatrix}$$

Matice T_1 je identitou, matice T_2 je maticí posunu o délku d. T_3 provádí rotaci bodu kolem jeho počátku o úhel α a poslední matice T_4 je provede zkosení bodu o faktor zkosení Sx ve směru osy x a Sy ve směru osy y.

5.2.2 Výpočet transformačních matic

Při návrhu interaktivního editoru IFS systémů se programátorům nabízí dvě cesty řešení, jak uchovávat transformace v paměti.

První cesta je uchovávat si v paměti pouze transformační matice objektů a dovolit uživateli jen omezené množství operací, které jsou definovány transformačními maticemi. Výhoda je menší paměťová náročnost, která je ovšem znatelná v případě, že objekty mají velký počet vrcholů.

Druhá cesta, která se používá častěji, je uchovávání souřadnic vrcholů bázového objektu. Uživatel si zdeformuje objekt dle jeho představ (změní souřadnice řídících bodů objektu). Z těchto souřadnic lze posléze netriviálně vypočítat koeficienty transformační matice. V rovině je k popisu potřeba tří bodů, které leží na objektu. Rovnice, které je potřeba vypočítat[14]:

$$\begin{aligned} x_1 \cdot a + y_1 \cdot b + e &= x_1' \\ x_2 \cdot a + y_2 \cdot b + e &= x_2' \\ x_3 \cdot a + y_3 \cdot b + e &= x_3' \\ \\ x_1 \cdot c + y_1 \cdot d + f &= y_1' \\ x_2 \cdot c + y_2 \cdot d + f &= y_2' \\ x_3 \cdot c + y_3 \cdot d + f &= y_3' \end{aligned} \tag{5.6}$$

Je nutné tedy spočíst dvě soustavy rovnic, každou o třech neznámých. Neznámé jsou koeficienty matice a,b,c,d,e,f. Další parametry vystupující v rovnicích jsou x_i a y_i, jsou body ležící na vzorovém objektu (ty jsou známé) a body x_i', y_i' představují transformované body (též známe).

5.3 Pravděpodobnost transformací

Při generování systémů iterovaných funkcí je důležitou součastí množina pravděpodobností jednotlivých transformací $\mathbf{\Pi}=\{\pi_1, \pi_2, \ldots \pi_n\}$. Do IFS systémů byla zavedena z důvodu existence různých transformací s lišícími se kontrakčními faktory. Transformace, které jsou většími kontrakcemi, se rychleji dostávali do atraktoru systému a docházelo tak k hromadění více bodů na jednom místě [14]. Proto transformacím byly přiřazeny jednotlivé pravděpodobnosti. Transformaci s větším kontrakčním faktorem tak náleží menší pravděpodobnost užití. Tím se dosáhlo rychlejšího výsledku spolu s vyřešením problému hromadění pixelů na jednom místě.

5.3.1 Výpočet pravděpodobnosti transformační

Studiem IFS fratálů jsem zjistil, že asi nejčastějším způsobem výpočtu transformací je výpočet z poměru obsahů generovaných obrazců. Tento způsob se zejména používá v interaktivních editorech IFS fraktálů, kde sám uživatel si zvolí bázový objekt a jeho pokrytí transformacemi. Uvedu zde konkrétní příklad jednoduchého výpočtu, při použití trojúhelníku jako bázového objektu:

vzorec obsahu trojúhelníku je $S = \frac{a \cdot v_a}{2}$,

v praxi si zvolíme sami stranu a a výškou v_a, kterou je poté vzdálenost třetího bodu od této strany. Vypočteme obsahy transformovaných trojúhelníků, poté všechny obsahy sečteme. Pravděpodobnost jedné transformace se přepočítá jako poměr obsahu transformace ku součtu celkového obsahu:

$$\pi_i = \frac{S_{transformace}}{S_{součtu}} \tag{5.7}$$

Stejným postupem dopočítáme zbývající transformace. Nejedná se o jedinou metodu používanou při výpočtu pravděpodobností transformací a v odlišných případech, například při generování 3D IFS systémů, může být nepoužielná. Z toho hlediska, že samotný výpočet obsahů v prostoru je velice komplikovaný. Nicméně v případě trojúhelníku jako vzoru je tato metoda hojně užívána právě pro její nenáročnost.

5.4 Algoritmus náhodné procházky

Často se používá jen zkratka RWA z anglického random walk algorithm [14]. Tento algoritmus patří mezi nejjednoduší a nejznámější algoritmy pro generování IFS fraktálů. Jeho hlavní výhodou je jeho paměťová nenáročnost. Naopak nevýhoda spočívá v generování stejných bodů vícekrát a nutnosti generovat velké množství bodů, což vede k větší časové náročnosti algoritmu.

Princip algoritmu je velice jednoduchý, nejprve se vybere náhodný bod v ploše, na který je s ohledem na její pravděpodobnosti π_i aplikována náhodná transformace φ_i z množiny Φ**.** Po aplikaci této transformace na bod, vznikne nový transformovaný bod, na který je posléze aplikována další transformace. Tento postup se iterativně opakuje až se docílí k maximálnímu počtu iterací.

Prvních pár bodů se nevykresluje, jelikož se po prvních iterací nemusí dosáhnout atraktoru systému. IFS systémy ovšem konvergují rychle, a proto se v praxi nevykresluje přibližně jen prvních 20 iterací.

V pseudokódu vypadá algoritmus následovně:

```
(x, y) = Náhodny(bod)
iteruj {
      i = náhodné číslo s ohledem na p trans. v rozsahu 0..n-1
      (x, y) = Fi(x, y)
      iterace > 20
            Vykresli(x, y)
}
```

5.5 Barnsleyho kapradina

Vzorovým IFS fraktálem je Barnsleyho kapradina pojmenovaná po zakladateli systému iterovaných funkcí Michaelu Barnsleym. Potřebné koeficienty a pravděpodobnost jednotlivých transformací vidíme v tabulce 5.1., po dosazení těchto koeficientů do algoritmu náhodné procházky dostaneme výsledek, který je možno vidět na obrázku 5.1.

Tabulka 5.1: Koeficienty pro Barnsleho kapradinu.

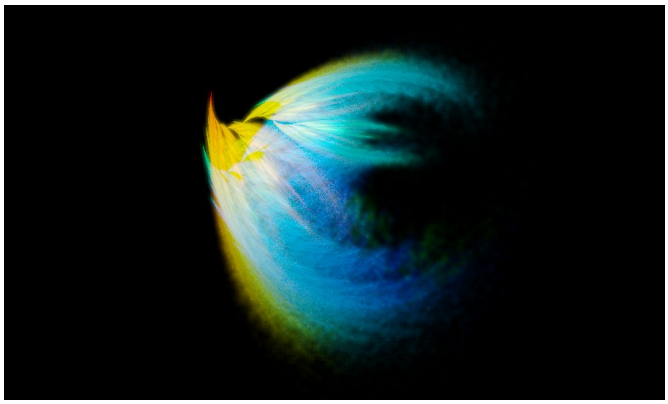
Barnsleyho kapradina (*Barnsley fern*)						
a	b	c	d	e	f	π
0	0	0	0,16	0	0	0,01
0.85	0,4	-0,4	0,85	0	1,6	0,85
0,2	-0,26	0,23	0,22	0	1,6	0,07
-0,15	0,28	0,26	0,24	0	0,44	0,07

Obrázek 5.1: Barnsleyho kapradina, pževzato z http://www.home.aone.net.au/~byzantium/ferns/fractal.html.

6 Flame fractal

Flame fractal [4], v překladu plamenný fraktál, je algoritmus, jehož základ byl navržen Scottem Dravesem v létě roku 1991. Jedná se o generalizaci systému iterovaných funkcí, která spočívá především v zavedení nových transformací (nelineárních funkcí), které navíc nemusí nutně splňovat podmínku kontrakce. Dalším vylepšením bylo zavedení logaritmické závislosti při obarvování pixelů a nový způsob výpočtu barvy. Narozdíl od IFS není kladen až takový důraz na matematickou korektnost, jako na estetický vzhled.

Algoritmus fractal flame je jedním z hlavních předmětů mé práce, jelikož jsem se rozhodl vytvořit program MyFlame určený k interaktivní tvorbě flame fraktálu.



Obrázek 6.1: Flame fraktál vytvořený v programu MyFlame (viz kapitola 9.).

6.1 Variace

Prvním zobecněním, které Scott Draves navrhl, jsou *variace* [4]. Systémy iterovaných funkcí jsou založeny na postupném aplikování afinních transformací. Algoritmus flame fractal přidává k těmto afinním transformacím nové nelineární funkce V_j z R^2 do R^2, které nazval *variace.* Matematický zápis je:

$$F_i(x, y) = V_j(a_i x + b_i y + e_i, c_i x + d_i y + f_i) \qquad (6.1)$$

Z matematického zápisu je zřejmé, že se na bod nejdříve aplikuje afinní transformace a na ni je následně aplikována nelineární funkce V_j. Tento postup změní výrazně charakter výsledného obrazu.

Variace se dělí do tří skupin [4]:

- klasické
- závislé (*dependent*)
- parametrické (*parametric*)

Klasické variace jsou nelineární funkce, které čistě přeskládají body do jiné podoby bez dalších nutných parametrů. Závislé variace je skupina funkcí, v nichž jsou jako parametry brány koeficienty transformační matice aktivní afinní transformace. Poslední skupinou jsou parametrické variace, jejichž název napovídá, že na výsledný obraz bude mít vliv externí parametr, který nikterak nesouvisí s aplikovanou transformací.

Tabulka 8:1 Příklady variací.

Označení	**Přiřazená funkce**	**Originální název**
$V_0(x,y)$	(x, y)	*Linear*
$V_1(x,y)$	$(\sin x, \sin y)$	*Sinusoidal*
$V_2(x,y)$	$\frac{1}{r^2}\cdot(x, y)$	*Spherical*
$V_3(x,y)$	$(x\cdot\sin(r^2)-y\cdot\cos(r^2), x\cdot\cos(r^2)+y\cdot\sin(r^2))$	*Swirl*
$V_{17}(x,y)$	$(x+e\cdot\sin(\tan(3\cdot y)), y+f\cdot\sin(\tan(3\cdot x)))$	*Popcorn*
$V_{24}(x,y)$	$(\sin(p1\cdot y)-\cos(p2\cdot x), \sin(p3\cdot x)-\cos(p4c\ y))$	*Pdj*

Klasickým příkladem závislé (*dependent*) funkce je V_{17} , pro jejíž výpočet jsou nutné koeficienty *e* a *f*. Příklad parametrické variace je funkce s označením V_{24},kde se vyskytují parametry *p1-p4,* které specifikují výsledný obrazec.

Další generalizací je použití více variací pro jednu afinní transformaci. Každé variaci je pak přiřazen koeficient užití. Po této generalizaci dostáváme:

$$F_i(x, y)=\sum_j v_{ij} V_j(a_i x+b_i y+e_i, c_i x+d_i y+f_i) \qquad (6.2)$$

Mezi variacemi je jako nultá funkce zahrnuta lineární funkce, která je identitou a tudíž výsledný obrazec nezmění. Proto je možné pomocí algorimu fractal flame generovat i klasické systémy iterovaných funkcí. Dokonce první verze programu Apophysis (http://www.apophysis.org/) zabývajícího se tvorbou těchto fraktálů měly v nastavení možnost odlišení IFS fraktálů a flame fraktálů.

6.1.1 Pozdní transformace

Pozdní transformace (*post transform*) [4] je dodatečnou afinní transformací aplikovanou až po přepočtu nelineární funkce. Její použití není nutné, avšak umožňuje změnu souřadnicového systému transformace. Je-li pozdní transformace definována

$$P_i(x, y) = (\alpha_i x + \beta_i y + \gamma_i, \delta_i x + \epsilon_i y + \eta_i) \quad (6.3)$$

tak je F_i předefinována následovně:

$$F_i(x, y) = P_i \sum_j v_{ij} V_j(a_i x + b_i y + c_i, d_i x + e_i y + f_i) \quad (6.4)$$

6.1.2 Konečná transfomace

Konečná transformace (*final transform*) [4] je nezávislá variace použitá ke konci iteračního cyklu. Působí jako nelineární kamera, přes kterou se jakoby díváme na vytvořený fraktál. Je nezávislá, jelikož její hodnota neovlivňuje transformaci nového bodu v algoritmu RWA a je použita př každé iteraci. Pro lepší pochopení přikládám pseudokód RWA spolu s konečnou transformací:

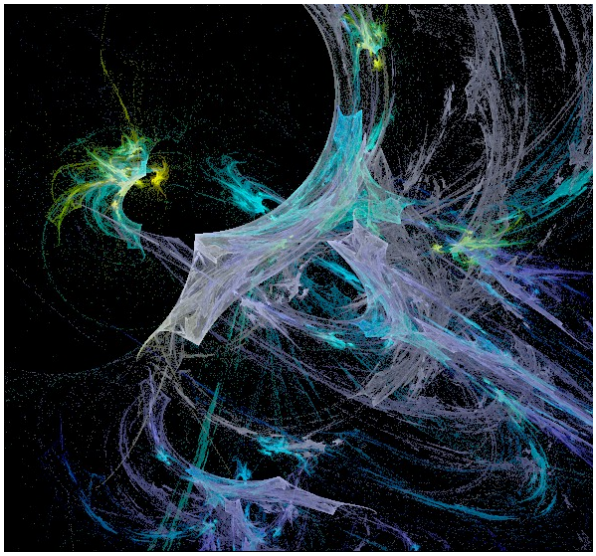
```
(x, y) = Náhodný(bod)
iteruj {
      i = náhodné číslo s ohledem na p trans. v rozsahu 0..n-1
      (x, y) = Fi(x, y)
      (xk, yk) = Fkonečná(x,y)   // na bod x,y aplikuj konečnou transf.
      iterace > 20
            Vykresli(xk, yk)
}
```

Zjednodušeně je možné říci, že konečná transformace namapuje vytvářený fraktál do jedné, či více spojených variací. Každý fraktál může mít jen jednu konečnou transformaci.

6.2 Logaritmická závislost

Chaotická hra (*chaos game*) má za následek překreslování stejných pixelů vícekrát. U IFS systémů se tomu snažilo předcházet pomocí výpočtů pravděpodobnosti transformací, nebo byl použit histogram zásahů na pixelu. Při použití histogramu byla barva odvozena podle počtu zásahů na pixelu, závislost intenzity barvy byla lineární. Scott Draves navrhl rozšíření, které spočívá v použití logaritmické závislosti namísto lineární. Rozšíření má dvě hlavní výhody, první je skutečnost, že se obraz jeví jako třídimenzionální. A druhá výhoda spočívá v zesvětlení poměrně tmavých částí fraktálu.

Efekt 3D vysvětluje následující případ. Ve fraktálu se kříží dva různé motivy, první motiv má v histogramu 1000 zásahů a druhý 100 zásahů se světelností 30 a 20. Při překřížení je hustota rovna 1100, což odpovídá hodnotě 30.4, která je pro uživatele jen těžce rozeznatelná od 30. To znamená, že užívanější motiv jakoby uzavře méně užívaný motiv, jelikož jeho hustota k celku je bezvýznamná.



Obrázek: Logaritmická závislost pixelů způspbující efekt 3D, obrázek byl vytvořen v programu MyFlame.

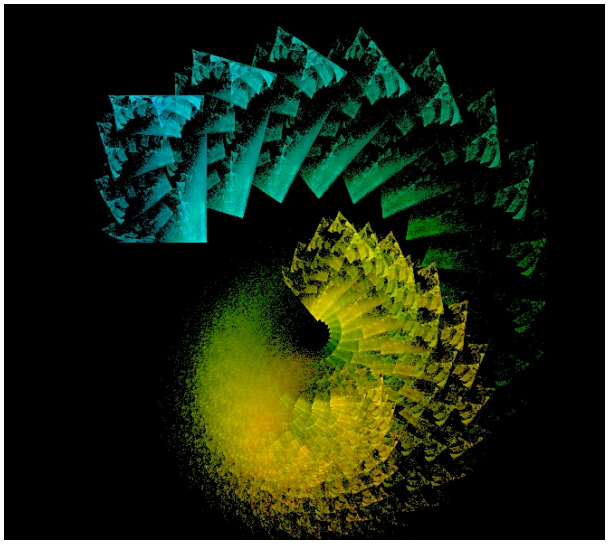
6.3 Způsob obarvení pixelů

Algoritmus fractal flames disponuje dalším rozšířením oproti původním IFS systémům. Nové rozšíření spočívá ve způsobu obarvení pixelů. IFS systémy jsou obarvovány na základě počtu zásahů pixelu ve fraktálu, nebo pevně danou barvou podle zvolené transformace. Naopak v algoritmu fractal flames byla k souřadnicím x a y přidána nová souřadnice c, kterou je právě barva. Nová souřadnice vkládá do fraktálu nový rozměr. Barva se mění na základě použité funkce F_i. Rozsah této souřadnice je od 0.0 do 1.0. a udává buď intenzitu barvy, ale častěji pozici v barevné paletě. Barevná paleta je definována jako funkce z [0,1] do (r,g,b). V praxi se užívá homogenní trojrozměrné pole o délce 256 prvků.

Abychom toho dosáhly je nutné přiřadit každé funkci F_i číslo c_i v rozsahu [0..1] a vložit do RWA nezávislou souřadnici *c* následně:

```
(x, y) = Náhodný(bod)
c = Náhodné(0..1)
iteruj {
      i = náhodné číslo s ohledem na p trans. v rozsahu 0..n-1
      (x, y) = Fi(x, y)
      c = (c + ci) / 2.0
      (xk, yk) = Fkonečná(x,y)    // na bod x,y aplikuj konečnou transf.
      iterace > 20
            Vykresli(xk, yk, c)
}
```

Tento postup zajišťuje, že barvy ve výsledném obrazu tvoří plynulé přechody z jedné barvy do druhé dle užitých funkcích. Dále také to, že nejvíce používanější funkce F_i (funkce s největší váhou) nejvíce ovlivňuje barvu vygenerovaného fraktálu.

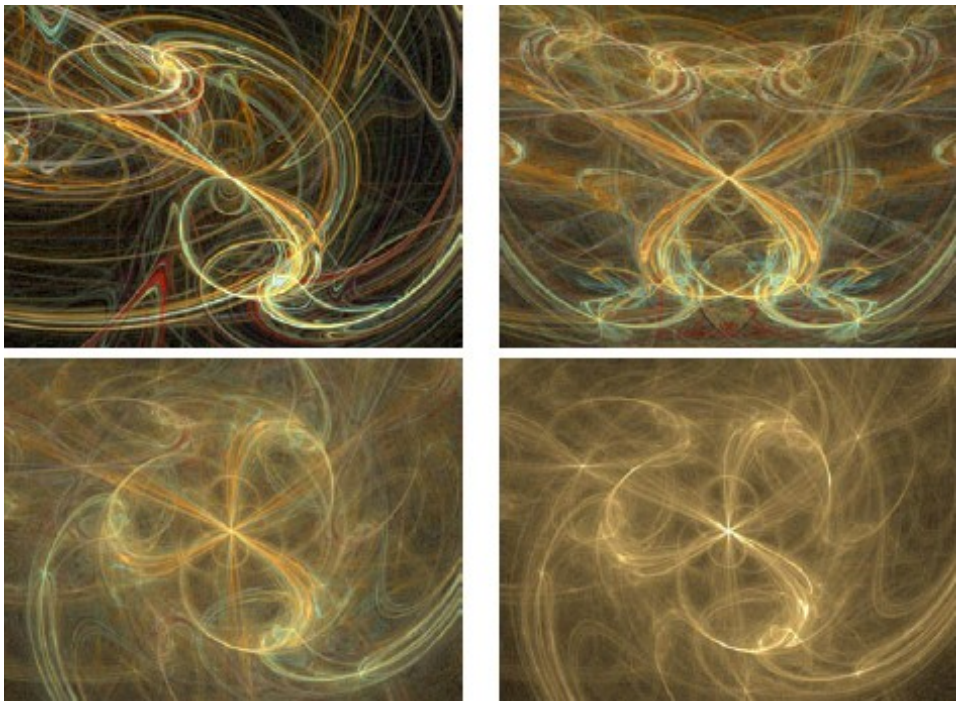


Obrázek: Postupné přechody ve fraktálu,obrázek byl vytvořen v programu MyFlame.

6.4 Symetrie

V algoritmu fractal flames se nabízí dva typy symetrií: rotační a zrcadlová. Rotační symetrie se docílí přidáním nové funkce F_i , která rotuje o 180 stupňů, tak se vytvoří dvojná rotační symetrie. Nová funkce by měla mít váhu stejnou, jako součet ostatních funkcí. Proto je její pravděpodobnost nastavena na 50% (každá druhá transformace provede symetrii). Trojné rotační symetrie se docílí přidáním rotační funkce o 120 stupňů, navíc je nutné přidat další funkci rotující o 240 stupňů. Důvodem přidáním další funkce je čistě matematický: do větve 120 stupňů se dostáváme s 50% pravděpodobností, ale do větve 240 stupňů je pravděpodobnost jen poloviční (25%), proto se přidává z pravidla další funkce, která řeší tento problém. Obecně je tedy nutné k n-násobné symetrii přidat n-1 nových funkcí, jejichž pravděpodobnost je nastavena na součet všech ostatních funkcí.

Zrcadlové symetrie se docílí funkcí, která změní znaménka osy x, nebo osy y (transformace změny měřítka). Též pro ni platí stejné nastavení pravděpodobnosti jako u symetrie rotační. Vícenásobné zradlové symetrie dosáhneme kombinací se symetrií rotační.



Obrázek: Příklady symetrie. V pravém rohu vidíme nevhodné použití funkce symetrie s následkem ztráty barevnosti fraktálu, převzato z [4] .

Při používání symetrie je nutné, aby funkce, které ji sprotředkovávají neměly vliv na výslednou barvu *c*. Jinak by fraktál ztrácel svoji barevnost. Tím se dosáhne nejen symetrie tvarové, ale i barvové.Samozřejmě je možné symetrii tvořit klasicky změnou (rotační, nebo zracdlovou) souřadnic pixelů v každém iteračním cyklu a vykreslovat tak více pixelů současně. Výsledný fraktál ale ztrácí svou přirozenost a vypadá uměle.

7 Metody vylepšení obrazu

Chaotická hra se hodí k různému použití anti-aliasingových technik. Aliasing nebudu rozebírat podrobně, bude postačovat, když alias budeme chápat jako nechtěný detail v obraze. Podrobně o aliasingu najdete například v [7].

Nebude následovat celý výčet technik, které se v praxi používají. Pouze se zaměřím na techniky, které jsem si vybral pro praktické použití v mojí aplikaci.

7.1 Supersampling

Též znám jako vzorkování s vyšší frekvencí [17]. Jedná se o metodu, která se aliasu plně nezbaví, avšak jej posune do vyšších frekvencí. Touto cestou dochází k potlačení aliasingu. V praxi v počítačové grafice spočívá supersampling ve vygenerování obrazu ve větším rozlišením a jeho filtrací do rozlišení menšího. Při filtraci se používá následujícího vzorce:

$$I(i,j)=\sum_{k=1}^{n}\eta_k\omega_k \qquad (7.1)$$

Plocha výsledného pixelu se vzorkuje n vzorky η_k, ω_k je koeficient, který určuje váhu každého vzorku. Je nutné splnit podmínku, aby spočtená suma ke každému vzorku byla rovna jedné, též je nutné, aby váhové koeficenty ω_k byly nezáporná čísla větší jak nula. Ve většině praktických případů mají ovšem vzorky stejnou váhu, čímž je možné vše zredukovat na pouhý výpočet průměru.

Podle způsobu rozmístění vzorků dělíme vzorkování na [17]:

- pravidelné vzorkování (*regular supersampling*)
- náhodné vzorkování (*stochastic supersampling*, *random supersampling*)

Každá technika má své výhody a nevýhody. Hlavní nevýhodou supersamplingu je nutnost většího výpočetního výkonu, což je způsobeno potřebou vypočtení více informací. Například při použití supersamplingu 2x2 potřebujeme na výsledný obraz 4krát více informací.

7.2 Maticové konvoluční filtry

Konvoluční filtry v počítačové grafice jsou často užívány v programech na úpravu obrazů, jako jsou Adobe Photoshop, Corel Photopaint či GIMP.

Na úpravy fraktálů typu „*flame fractal*“ se nejvíce hodí filtry způsobující rozmazání, jelikož se tak zahladí nerovnosti v obraze. Ovšem i při použití jiných filtrů s kombinací rozmazání můžeme vytvořit zajímavé efekty (viz kapitola 8).

7.2.1 Konvoluce

Důležitým pojmem v oblasti teorie konvolučních filtrů je konvoluce [7][17]. Konvoluce dvou funkcí $I(x)$ a $h(x)$ se značí operátorem * a je definována jako

$$I(x)*h(x)=\int_{-\infty}^{\infty} I(x-\alpha)\,h(\alpha)\,d\alpha \tag{7.2}$$

Funkcí $h(x)$ se značí konvoluční jádro (*kernel*). Jádro si můžeme představit jako okno, kterým posouváme po obraze. Hodnoty v jádře určují výpočet nového pixelu v obraze.

Při práci s digitálním obrazem se používá *diskrétního konvoluce* [7][17], která je diskrétní podobou integrálu (7.2), její dvojrozměrna podoba je

$$I'_{(i,j)}=I_{(i,j)}*h_{(i,j)}=\sum_{x=-k}^{k}\sum_{y=-k}^{k} I_{(i-x,j-y)}h_{(ij)} \tag{7.3}$$

Základní postup aplikace konvolučních filtrů je následující. Konvoluční jádro můžeme popsat jako tabulku s rozměry <-*k*,*k*> na <-*k*,*k*>. Výsledný obraz $I'_{i,j}$ získáme tím, že na každý bod funkce $I_{i,j}$ položíme konvoluční jádro $h_{i,j}$ a vypočítáme součet (7.3).

Obrázek 7.1: Aplikace Gaussova filtru na rastrový obraz. Obrázek převzat z http://www.gamedev.net.

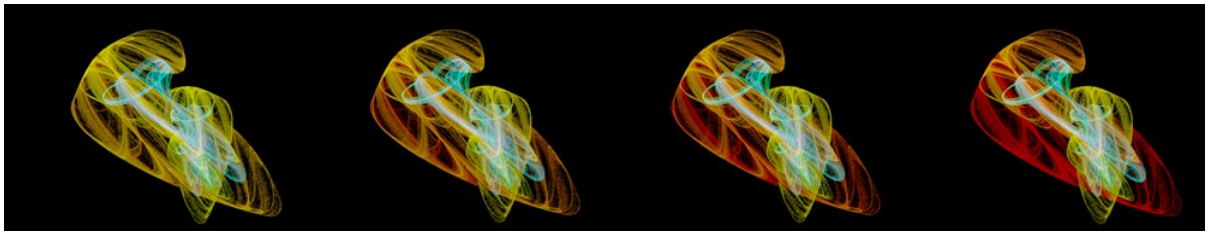
8 Navrhnutá rozšíření

Hledání rozšíření mě provázelo celou dobu bakalářské práce. Zvolil jsem si fraktál nazývaný flame fractal, nastudoval jsem jeho vlastnosti z originální práce Scotta Dravese [4]. Prvotní impuls, ale pocházel ze seriálu Pavla Tišnovského [14], který na serveru root.cz publikoval základní vlastnosti algoritmu fractal flame v českém jazyce. Prozkoumal jsem dosavadní vytvořené programy, které nabízely tvorbu flame fraktálu, zejména jsem se zaměřil na program Apophysis, který pravděpodobně nemá ve tvorbě flame fractal konkurenci. První idea byla implementovat nové variace popsané Scottem Dravesem v jeho práci, které jsem v programu Apophysis nenašel. Bohužel jsem přehlédl beta verzi programu, v níž byly nové variace implementovány. Nabízela se tedy možnost vymyslet vlastní variace, ale z důvodu stále rostoucího počtu variací a možnosti nového pluginu v programu Apophysis pro vytváření uživatelských variací jsem se vydal odlišnou cestou a navrhl následující rozšíření.

8.1 Odlišné chápání barvy

Barva použité transformace je v programech na interaktivní tvorbu fraktálů typu „flame fractal“ odvozována podle indexu v barevné paletě. Barevná paleta se ovšem může skládat z více barev, a tudíž je nezávislá na počtu transformací a je nezávislá i nad výsledným fraktálem. Rozhodl jsem se proto barevnou paletu více přizpůsobit použitým transformacím, čímž se na jednu stranu mírně odkláním od původní idei Scotta Dravese, který klade důraz na estetičnost fraktálu, ale na druhou stranu se tím více přikláním k jejich fraktálové podstatě. Barva je tedy pevně přiřazena transformacím, mezi kterými jsou programově tvořeny lineární přechody barev. Čím více transformací fraktál obsahuje, tím větší je počet barev ve fraktálu a vygenerovaný fraktál tak obsahuje i větší množství přechodů.

Dalším rozšířením je přidání nového koeficientu do algoritmu náhodné procházky, který určuje sílu barvy. Každé transformaci tedy přiřadím váhu barvy. Váha udává jak moc bude barva transformace ovlivňovat výsledný fraktál.

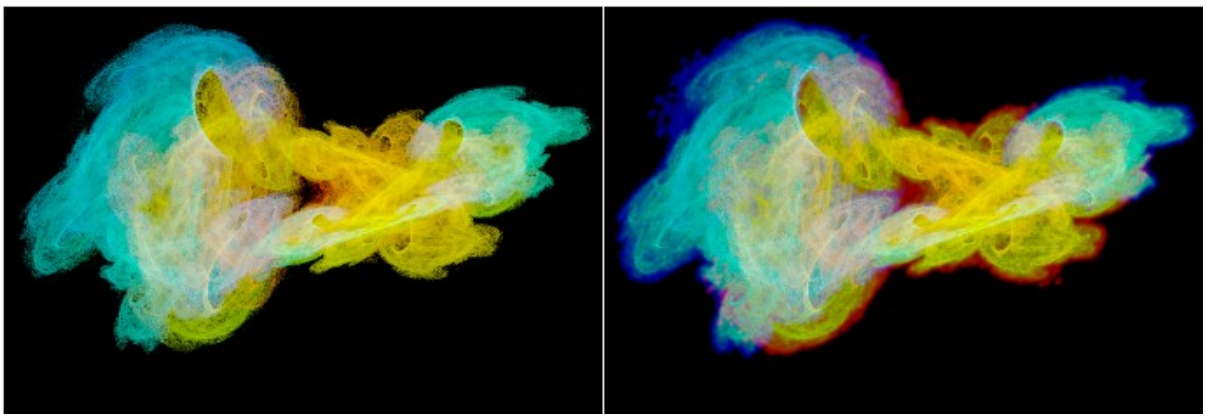


Obrázek 8.1: Změna flame fraktálu se zvyšujícím se koeficientem váhy červené transformace.

8.2 Použití konvolučních filtrů

V předchozí kapitole jsem se věnoval různým metodám pro úpravu obrazu. Na konvoluční filtry jsem narazil náhodou, při studiu OpenGL, které konvoluční filtry přímo podporuje [13]. Velký význam při tvorbě flame fraktálu má filtr způsobující rozmazání. Jelikož při nedostatečně dlouhé době generování se v obraze nacházejí samostatné pixely, které tvoří šum. Scott Draves navhrl vlastní rozmazávací filtr, který se mi bohužel nepodařilo nikde najít. Jediné, co o filtru publikoval je, že má proměnou délku, která je inversně závislá na počtu zásahů v histogramu hustoty fraktálu. Navrhl jsem zjednodušenou verzi o několika úrovních filtru, který filtruje pixely s hustotou zásahu menší jak určitá prahová mez. Dále jsem implementoval i jiné filtry, které rozmažou obraz i jinými způsoby.

Po té mě napadlo, proč nezkusit i jiné konvoluční filtry a neomezovat se na jednoduché rozmazávání? Experimentoval jsem s různými filtry, které ve většině případů nevedly k zajímavým obrazovým motivům. Důvodem byla obvykle velká rozmanitost fraktálních útvarů. K zajímavému motivu jsem dospěl až kombinací více filtrů. Přesně filtru, který se používá při ostření obrazu následujícího filtrem způsobující rozmazání. První filtr se použije na necelou barevnou složku obrazu, například pouze na kanál modré a červené barvy. Vznikne tak barevný šum okolo pixelů s malou hustotou zásahu a aplikací druhého filtru se vzniklý šum rozmaže a vytvoří záři. Pro bližší pochopení přikládám demonstrační obrázky. Na prvním obrázku je neupravený flame fractal. Druhý ukazuje stav po aplikaci filtru.



Obrázek 8.2: Před aplikováním filtru záře (vlevo) a po aplikování filtru (vpravo).

Podobného efektu bychom v programech určených na úpravu rastrových obrazů dosahovali s větší obtíží, jelikož obraz sám o sobě neuchovává hodnoty počtu zásahů v pixelech fraktálu.

9 Interaktivní editor MyFlame

Byl navrhnut interaktivní editor, který vytváří fraktály typu „flame fractal“ a implementuje mé vlastní rozšíření. Název MyFlame (čti anglicky) odráží skutečnost, že si každý uživatel může vygenerovat svůj vlastní fraktál. Editor poskytuje prakticky neomezené možnosti vytváření fraktálů různých tvarů a variací.

9.1 Použité technologie

Z hlediska programátoské efektivity jsem se rozhodl aplikaci psát ve Visual Studiu 2008, které poskytuje mnoho funkcí při návrhu okenních programů pro systém Microsoft Windows. V následujících řádcích se zaměřím na knihovny a jazyk C#, ve kterém je aplikace implementována.

9.1.1 OpenGL

Grafický systém OpenGL (*Open Grafic Library*) je softwarové rozhraní pro grafický hardware. Umožnuje vytvářet interaktivní aplikace sloužící k vizualizaci grafické informace v počítači [13]. Nejedná se o programovací jazyk jako je C nebo C++, spíše se OpenGL dá přirovnat ke knihovně, která zajišťuje určité funkce. OpenGL neobsahuje žádné příkazy pro práci s okny, proto je nutné použít jiný systém pracující s okny, závislý na platformě, kterou používáte. Z programátorského hlediska se OpenGL chová jako stavový automat, který je založen na povolování a zakazování různých příznaků, které určují výsledný ráz scény. Aktuální stav automatu je tak dán množinou aktivních příznaků. Vykreslování scény se provádí procedurálně, to znamená, že voláním OpenGL funkcí vykreslíme rastrový obraz. OpenGL vykresluje pouze *grafická primitiva* (bod, úsečka, polygon, pixmapa), to znamená, že neobsahuje integrované žádné vyšší modely.

Knihovnu OpenGL jsem si vybral zejména pro její komplexnost, jelikož nejenže poskytuje funkce pro tvorbu 3D grafiky, ale má implementované i funkce pro práci s bitmapami, které jak jsem doufal by mi mohly urychlit moji práci.

9.1.2 Programovací jazyk C#

Jazyk C# (čti „sí šárp“) je poměrně mladý programovací jazyk, jehož první verze byla představena v lednu roku 2002 společností Microsoft spolu s rámcem .NET, pro který byl programovací jazyk vytvořen. Momentálně aktuální verzí je C# 3.0 vydaná spolu s .NET Frameworkem 3.5 koncem roku 2007. Jedná se o jednoduchý, moderní, typově orientovaný jazyk [13] odvozený z C a C++, který se

vyznačuje syntaktickými podobnostmi s jazyky C++ a java. Z důvodu jeho robustnosti se ovšem nemůže rychlostí měřit s programy napsané například v jazyce C.

Před psaním této práce jsem se s tímto programovacím jazykem nesetkal. Proto byla potřeba nastudovat jeho základní vlastnosti a odlišnosti od jiných programovacích jazyků. Nevýhodou jazyka C# je jeho omezenost na programy psané pro operační systém Microsoft Windows, ovšem obsahuje mnoho možností, které urychlují a usnadňují vývoj aplikací.

9.1.3 CsGL

CsGL (*C sharp Graphics Library*) [5] je knihovna implementující obálku pro základní funkce OpenGL v jazyce C#. Podporuje funkce OpenGL do verze 1.4 a poskytuje třídy pro zpracování událostí myši a klávesnice. Projekt byl ovšem v roce 2002 uzavřen. Prakticky k této knihovně neexistuje žádná uživatelská podpora. Tato skutečnost velice komplikovala moji práci a některé funkce se mi nepodařilo implementovat. Později jsem při bližším hledání objevil Tao (http://www.taoframework.com/), původně navržené Randy Ridgem, který přepsal a vylepšil CsGL. Tao je multiplatformní a poskytuje více funkcí (neomezuje se jen na OpenGL). V dnešní době se na vývoji projektu Tao podílí již více programátorů.

9.2 Postup řešení

K realizaci programu bylo potřeba detailně nastudovat fraktální geometrii IFS systémů, která tvoří základ pro pochopení interaktivní tvorby fraktálů typu „flame fractal“. Dále bylo nutné nastudovat základy programovacího jazyka C# a OpenGL.

Můj první funkční program vygeneroval pár základních fraktálů, ale nereagoval na uživatelské vstupy. Z mnou odzkoušených programů jsem zjistil, že k této potřebě se velmi často používá trojúhelníková reprezentace transformací. Proto byl navržen editor, ve kterém bylo možné měnit rozměry a polohu trojúhelníků. Z těchto trojúhelníků a jejich vztahu k bázovému objektu se pomocí rovnic (5.5) vypočítaly potřebné transformace. To dalo základ editoru. Při vykreslování fraktálů jsem se setkal s implementačními problémy v knihovně CsGL, která neumožňovala použít pole bitmapy jako vícerozměrné pole (pozice a barva ve formátu složek RGB). Bylo zapotřebí použít jednorozměrné pole, což vedlo k znepřehlednění jistých částí kódu. V případech, kde byla nepřehlednost neúnosná bylo zvoleno pomocné pole, na úkor menší rychlosti a paměťové náročnosti programu.

Další etapou postupu bylo hledání rozšíření. Které má konečnou podobu popsanou v předchozí kapitole. Ze zájmu jsem zkoušel navrhnout i vlastní transformace, které ovšem nebyly nikterak

zvláštní, proto editor zůstal u originálů popsaných v práci Scotta Dravese [4]. Uživatelské rozhraní se formulovalo do nynější podoby od začátku až po konec mojí práce.

9.3 Realizace aplikace

Celý program je rozdělen do několika modulů. Každý modul zajišťuje určitou činnost. Dosáhlo se tak většího zpřehlednění programu a oddělení jednotlivých částí na logické celky.

- *Program.cs* - hlavní vstupní bod do aplikace, inicializuje program.
- *Form1.cs* - modul, který obsahuje definici pro chování okenní části hlavního okna programu.
- *Form1.designer.cs* - definice vzhledu hlavního okna, seznam ovládacích prvků a jejich umístění.
- *FlameData.cs* - popis fraktálu, data jeho transformací, variací a barvy, práce s nimi.
- *FractDraw.cs* - vykreslovací okno, obsahuje algoritmus pro generování fraktálu a použitých konvolučních filtrů.
- *Triangles.cs* - základní práce s trojúhelniky a jejich vlastnosti, přepočet jejich souřadnic na transformační matice a zpět.
- *IFSeditor.cs* - okno editoru, obsahuje reakce na události od uživatele, vykreslení trojúhelníků a tvorbu náhledu na fraktál.
- *Basecode* - složka obsahující moduly pro základní práci s CsGL, distribuovaná spolu s knihovnou CsGL. Zde bylo nutné upravit pár vlastností, aby bylo možné propojit práci editoru s knihovnou CsGL.

Fraktál v programu popisuje třída *Flame,* která obsahuje kolekce typu *List:*

- *IFSList* - kolekce objektů *IFS,* které obsahují jednotlivé transformace a metody pro práci s nimi, jako je nahrání identické matice, rotace, změna měřítka a posun. Každý objekt *IFS* též nese kolekci k němu příslušných variací *VariationList*.
- *TriangleList* - je složen z objektů typu *Ttriangle,* uchovávající pozici trojúhelníku. Obsahuje metody, které převádějí pozici na plátně do transformačních matic a zpět. Dále má implementovány metody pro přepočet obsahu trojúhelníku, které se používají při přepočtu pravděpodobnosti transformace (viz oddíl 5.3).
- *ColorList* - kolekce barev všech transformací.
- *FilterList* - kolekce s dvourozměrnými poly s jádry konvolučních filtrů.

Výběr kolekce byl zvolen zejména proto, že umožňuje shromažďovat a uspořádávat objekty stejného typu a pro jednoduchou práci s nimi. Kolekce *IFSList* a *VariationList* jsou kolekce typu *BindingList*, což je vylepšená třída *List* o možnost rychlého navázání dat s ovládacími prvky aplikace,

která tak umožňuje intertaktivitu editoru transformací s hlavním oknem. Stejnou indexací kolekcí se provedlo provázání dat, což vedlo k usnadnění práce s transformacemi.

Zobrazení okna editoru transformací sprotředkovává třída *IFSeditor*, která je odvozena od třídy *Model* knihovny CsGl Basecode (podporuje práci s OpenGL). Náhled editoru je implementován tak, že prvních pár set iterací se provádí výpočet středu fraktálu a teprve až po té dochází k samotnému vykreslovacího algoritmu, který je založen na algoritmu RWA (viz oddíl 5.4). Podobným způsobem je realizováno i zobrazení fraktálu ve vlastním okně, s tím rozdílem, že pro výpočet středu je použito více iterací, tím se dosáhne vyšší přesnosti. Editor transformací okamžitě reaguje na změny provedené v ovládacím panelu. Dosáhlo se tak vylepšené interaktivity s uživatelem. Data bodů fraktálu jsou uchovávany v instanci třídy *Tpixmap,* která obsahuje metody pro práci s těmito body (supersampling, aplikace filtrů, vložení nového bodu, logaritmickou konverzi). Samotný fraktál v paměti představuje trojrozměrné pole s údaji o poloze bodu na obrazovce a s barevnými složkami (červená, zelená a modrá). Vložení nového bodu vygenerovaného chaotickou hrou je realizováno jako vložení barevných přírustků (odpovídajících barevné hodnotě v gradientu, dle třetí proměnné *c*) do pole obsahujícího body fraktálu. Před vlastním vykreslením se provede logaritmická konverze (viz oddíl 9.3.3) a též supersampling, je-li zapnut. Generování fraktálu se provádí ve vlastním vlákně, aby nezatěžovalo program při zpracování událostí. Za pomoci vláken je realizována i konvoluční filtrace fraktálu.

9.3.1 Výpočet barevných přechodů

Jelikož moje chápání barvy se mírně odlišuje od základní idei, považuji za potřebné vysvětlit implementaci přístupu popsané v kapitole 8. Barevná paleta je v programu implementována jako trojrozměrné pole o délce 256 prvků. Počet přechodů v barevné paletě je roven počtu n-1 transformací použitých ve fraktálu. Přechod z jedné barvy do druhé je lineární, dále jsou přechody rovnoměrně rozděleny do barevné palety a stejně jsou i rozděleny koeficienty z intervalu [0..1], které určují zásah do barevné palety, korespondují tak s hranicemi jednotlivých přechodů.

9.3.2 Váha barvy transformace

Dalším rozšířením je váha barvy transformace, která určuje vliv barvy transformace na výsledný fraktál. Každé transformaci je tak přiřazena nová celočíselná hodnota, která může být i nulová (transformace tak nebude mít vliv na obarvení fraktálu). Do algoritmu na tvorbu flame fraktálů v místě přepočtu barvy pak přidáme nový úsek kódu:

```
iteruj N-krát{        // N je koeficient váhy barvy
   c = (c + ci) / 2.0
}
```

Implicitně je *N* nastaveno na 1, avšak v grafické uživatelské rozhraní obsahuje ovládací prvek, který může váhu barvy změnit.

9.3.3 Logaritmická závislost pixelů

V paměti je trojrozměrné pole, které uchovává údaje o pozici pixelu v a jeho barvě, barva se skládá ze tří složek (r,g,b), kde každá složka nabývá hodnot od 0 do 255. Logaritmickou závislost pixelů poté docílíme najitím maxima jednotlivých složek v celém poli, jejichž logaritmus následně vynásobíme převrácenou hodnotou maximální délky složky (255), tak dostaneme faktor konverze. Po té budeme procházet celé trojrozměrné pole a jejich logaritmus násobit faktorem konverze. Touto cestou je implementován algoritmus v programu.

9.3.4 Supersampling

Supersampling byl implementován vykreslením fraktálu ve 2x větším rozlišení a jeho následným zmenšením. Dosáhli jsme tak supersamplingu 4x. Jádro supersamplingového filtru je:

$$kernel = \begin{bmatrix} 1/4 & 1/4 \\ 1/4 & 1/4 \end{bmatrix}$$

9.3.5 Konvoluční maticové filtry

Princip aplikace konvolučních filtrů byl popsán v sedmé kapitole. Jelikož konvoluční filtry nemusí být aplikovány na všechny body ve fraktálu, ale jen na body s určitou hustotou, proto bylo zavedené pole hustoty zásahů *DensityMap.* Nové pole bylo vytvořeno také proto, že v poli fraktálu jsou pouze uchovány přírustky barevných složek, ze kterých nejde odvodit počet zásahů daného bodu.

Speciální rozmazávací filtr

Konvoluční filtr průměrného rozmazání, který má délku závislou na hustotě pixelu. Má 3 úrovně:

- Hustota < 2 filtr o velikosti 9x9
- Hustota 2-5 filtr o velikosti 5x5
- Hustota 5-8 filtr o velikosti 3x3

Slabá záře

Způsobí slabé rozmazání barevné složky bodů s hustotou zásahů menší než je prahová mez. Prahová mez je implicitně nastavena na 8, uživatel si ji ovšem může změnit. Jedná se vlastně o filtr rozmazání použitý jen na určitý kanál barvy. Filtr rozmazání je použit ze seznamu implementovaných filtrů.

Silná záře

Silná záře je způsobena aplikací filtru ostření na daný barevný kanál, stejným způsobem jak slabá záře. Filtr ostření tak způsobí barevný šum, který následně rozmaže speciální rozmazávací filtr.

Implementované konvoluční filtry

$$F_1=\frac{1}{3}\begin{bmatrix}1&0&0\\0&1&0\\0&0&1\end{bmatrix}, F_2=\frac{1}{3}\begin{bmatrix}0&0&1\\0&1&0\\1&0&0\end{bmatrix}, F_3=\frac{1}{5}\begin{bmatrix}1&0&1\\0&1&0\\1&0&1\end{bmatrix}, F_4=\frac{1}{3}\begin{bmatrix}0&0&0\\1&1&1\\0&0&0\end{bmatrix}, F_5=\frac{1}{3}\begin{bmatrix}0&1&0\\0&1&0\\0&1&0\end{bmatrix},$$
$$F_6=\frac{1}{5}\begin{bmatrix}0&1&0\\1&1&1\\0&1&0\end{bmatrix}, F_7=\frac{1}{9}\begin{bmatrix}1&1&1\\1&1&1\\1&1&1\end{bmatrix}, F_8=\frac{1}{16}\begin{bmatrix}1&2&1\\2&4&2\\1&2&1\end{bmatrix}, F_9=\begin{bmatrix}-1&-1&-1\\-1&9&-1\\-1&-1&-1\end{bmatrix}$$

F_1-F_8 je skupina filtrů způsobující rozmazání, F_1, F_2 diagonální, F_4 horizontální, F_5 vertikální,F_3 a F_6 křížové, F_7 průměrné F_8 Gaussovo. F_9 je filtr ostření.

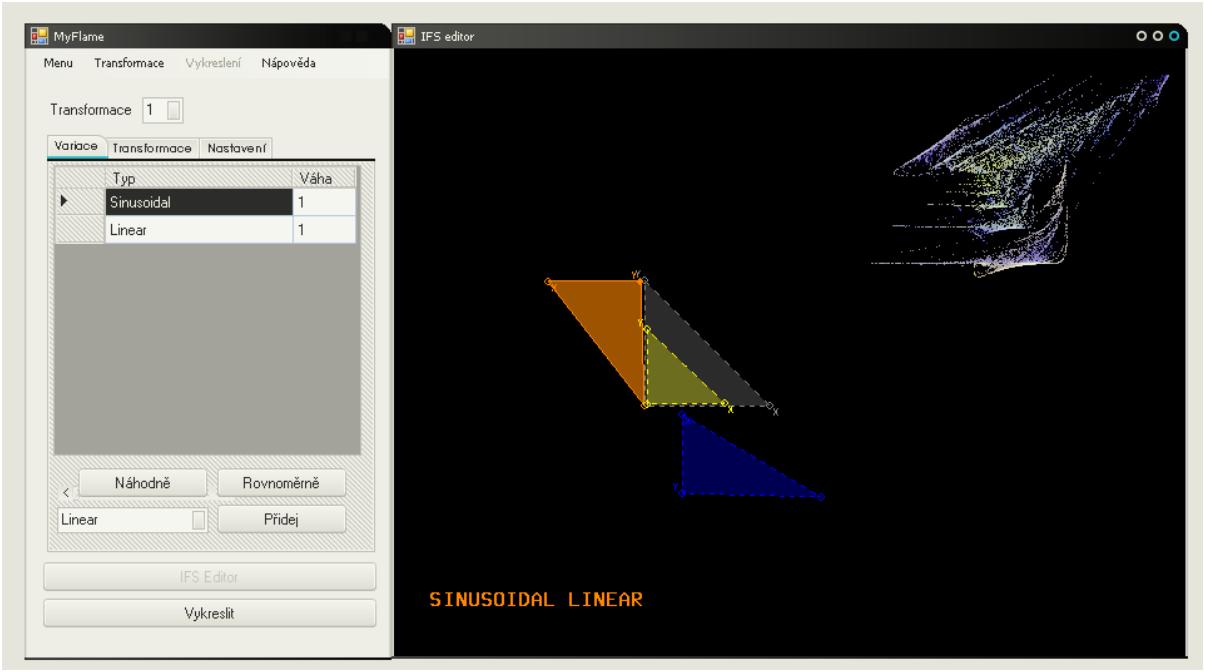
9.3.6 České uživatelské rozhraní

České uživatelské rozhraní jsem zvolil z důvodu neexistence žádného českého programu pro tvorbu fraktálů typu „flame fractal“ i z důvodu, že na zahraniční programy neexistuje český překlad. Touto cestou jsem chtěl dosáhnout větší přívětivosti pro české uživatele, a též snadnějšímu pochopení vlastní tvorby fraktálů.

9.4 Popis aplikace

Filosofie editoru je založena na jednom hlavním oknu (ovládací panel) a dvou vykreslovacích oknech. Program je navržen pro systém Microsoft Windows a obsahuje ovládací prvky, které jsou v tomto operačním systému běžně používané (funkční tlačítka, různé zašrtávací boxy, taby)

Ovládací panel umožňuje měnit různé nastavení fraktálu, transformačních matic, variací a výsledné barvy. Též obsahuje ovládací prvky, které mění a nastavují vlastnosti vykreslení, polohu počátku souřadnicového systému a přiblížení, či oddálení fraktálu. Mezi další nastavení patří volba konvolučních filtrů. Aplikace též podporuje i návrh vlastního konvolučního filtru. Podrobný popis aplikace najdete v uživatelské příručce (viz Příloha B).



Obrázek 9.1: Screenshot z programu MyFlame, práce s editorem transformací.

Vykreslovací okno editoru transformací obsahuje trojúhelníky, které je možné přidávat, umazávat, hýbat s nimi a měnit jejich tvar. Na základě jejich transformací je formován výsledný fraktál. Vpravo nahoře vykreslovacího okna editoru transformací se nachází náhled fraktálu. Existence náhledu urychluje práci a zpříjemňuje tvorbu nových fraktálu.

Druhé okno zprostředkovává hlavní vykreslení fraktálu. Bohužel knihovna CsGL byla navržena jen pro běh jednoho vykreslovacího okna podporující OpenGL, z toho důvodu není možné mít obě vykreslovací okna spuštěné naráz.

9.5 Shrnutí programu

Byl vytvořen interaktivní program pro tvorbu fraktálu typu „*flame fractal*“. Práce s programem je založena na transformacích bázového objektu, kterým byl pro jednoduchost zvolen trojúhelník. Do programu bylo implementováno vlastní rozšíření v podobě aplikaci různých konvolučních filtrů v závislosti na hustotě počtu zásahů pixelu a přidání nového koeficientu ovlivňujícího barevnost fraktálu. Program umožňuje uživateli výběr filtru a rozsah jeho aplikace, též i nadefinování vlastního filtru. Navíc program obsahuje filtr *záře*, který způsobí barevnou záři kolem fraktálu. Pro vylepšení kvality obrazu byl implementován supersampling 4x, který uživatel může buď povolit, nebo zakázat. Do programu nebyly zařazeny pozdní transformace a symetrie, které originální algoritmy implementují.

V oblasti generovaní výsledného fraktálu program z větší části nemůže konkurovat rozsáhlému projektu Apophysis, nejedním z důvodů je například neexistence kvalitního vykreslovacího jádra, najdou se ovšem vyjímky, podle kterých některé obrazce dosahují podobné kvality.

Obrázek 9.2: Flame fraktál obsahující binární strom. Vytvořeno v programu MyFlame.

10 Závěr

Závěrem bych chtěl shrnout obsah celé práce. Prvním úkolem bakalářské práce bylo seznámit se s teorií fraktálů. Sama fraktální teorie je ovšem velmi široký pojem, který jde na pár stránkách jen těžko vyjádřit, proto jsou v práci uvedeny pouze základní vlastnosti fraktálů a u většiny jsou uvedeny příklady, které dané vlastnosti demonstrují. Práce se též zaměřila na průřez využití fraktálů v počítačové grafice, která fraktálové geometrii vdechla nový život.

Jelikož i uplatnění fraktálů v počítačové grafice je velmi obsáhlé, byla v práci zvolena konkrétní oblast jejich využití. Touto oblastí se stala interaktivní tvorba fraktálů, zaměřená na generování poměrně nového fraktálu typu „*flame fractal*“. Pro jejich tvorbu bylo ovšem nutné nastudovat matematické vlastnosti IFS systémů, ze kterých daný typ fraktálu vychází a proto jsou detailně v práci popsány.

Práce se zaměřila i na hledání rozšíření v oblasti zvoleného fraktálu a jeho generování. Jedním rozšířením se stalo aplikování různých konvolučních filtrů v závislosti na hustotě zásahu pixelu ve fraktálu, která tak dává uživateli možnost mírně upravit vygenerovaný fraktál a v jistých případech dosáhnout i zajímavých efektů. Vlastní pojetí barvy též vnáší změnu, která usnadňuje pochopení užití barev při generování *flame* fraktálu. Přidáním nového koeficientu do algoritmu je dosáhnuto rychlé změny barvy ve výsledném fraktálu.

Dalším úkolem práce bylo implementovat navrhnutá rozšíření. Z toho důvodu vznikl nový program MyFlame, který implementuje základní možnosti algoritmu pro tvorbu flame fraktálu. Práce s ním ovšem není úplně intuitivní, jelikož k pochopení práce s programem je nutná i menší dávka abstrakce. Program se zaměřil i na kvalitu grafického výstupu, proto byly v práci popsány metody jako je supersampling a konvoluční filtry, které s tvorbou fraktálů úzce nesouvisí.

Možný vývoj programu vidím v optimalizaci vykreslovacího algoritmu, na který není dbán velký důraz, pro optimalizaci bych zvolil jazyk symbolických adres. Pro pohodlnější použití OpenGL bych navrhnul jinou knihovnu, než je zvolená CsGL, jejíž vývoj byl zastaven, například framework Tao, který je v této práci zmíněn. Budoucnost projektu vidím i v doimplementování vlastností původního algoritmu, které v programu MyFlame nebyly realizovány, mezi než patří symetrie, pozdní transformace (*post transformation*) a též i nové variace, které stále přibývají. Co v programu opravdu chybí je možnost exportu fraktálu do formátu XML, která by zajistila uchovatelnost a přenositelnost fraktálu pro jeho pozdější editaci. Práce s filtry by též mohla být vylepšena, což ovšem úzce souvisí s použitím jiné knihovny pro OpenGL.

Literatura

[1] Agarwal, Ravi P.; Meehan, Maria; O'Regan, Donal: *Fixed Point Theory and Applications*. Cambridge University Press, *2001*. ISBN 0-521-80250-4

[2] Barnsley F. M.: *Fractals everywhere*. San Diego: Academic Press, 1993.

[3] Dayton P., Albahari B., Neward T.: *C# v kostce Pohotová referenční příručka*. Praha: Grada Publishing a.s, 2003. ISBN 80-247-0443-9

[4] Draves S., Eric Reckase: *The Fractal Flame Algorithm* [online]. 20.12. 2008 http://flam3.com/flame.pdf [cit. 3.5. 2009]

[5] Dupont L.: *CsGL – C# graphic library* [online]. http://csgl.sourceforge.net/index.html [cit. 3.5. 2009]

[6] Elert G.: *The Chaos Hypertextbook* [online] http://hypertextbook.com/chaos/ [cit 3.5. 2009]

[7] Foley J.D., Andries van Dam, Feiner S. K. , Hughes F. J.: *Computer Graphics principles and practise*. Addison-Wesley Company, Inc. 2003. ISBN 0-201-84840-6

[8] Fournier, Fussel A. D., Carpenter L. : *Computer Rendering of Stochastic Models*. CACM, 1982

[9] Kiminek J.: *Introduction to Fractal compression* [online]. http://www.faqs.org/faqs/compression-faq/part2/index.html, [cit 3.5. 2009].

[10] Mandelbrot, B.: *Fraktály: Tvar, náhoda a dimenze*. Praha: Mladá Fronta, 2003. ISBN 80-204-1009-0

[11] Moser, J. K., Ed: *Dynamical Systems Theory and Applications*. *Springer Lecture Notes in Physics*. Berlin: Springer-Verlag, 1975.

[12] Prusinkiewicz P. And Lindenmayer A: *The Algorhitmic Beauty of plants*. Springer-Verlag, 1990.

[13] Shreiner D., Woo M., Neider J., Tom Davis: *OpenGL Průvodce programátora*. Brno: Computer press, a.s, 2006. ISBN 80-251-1275-6

[14] Tišnovský P.: *Seriál fraktály v počítačové grafice* [online]. http://www.root.cz/serialy/fraktaly-v-pocitacove-grafice/, [cit. 3.5. 2009].

[15] The US Patent and Trademark Office [online] http://patft.uspto.gov/ [cit 3.5. 2009]

[16] Zelinka, I.: *Fraktální geometrie: principy a aplikace.* Praha: BEN, 2006. ISBN 80-7300-191-8

[17] Žára J., Beneš B., Sochor J., Ferkel P.: *Moderní počítačová grafika*. Brno: Computer Press, a.s., 2004. ISBN 80-251-0454-0

Seznam příloh

Příloha A: Tabulka implementovaných variací
Příloha B: Uživatelská příručka
Příloha C: Ukázky vytvořených fraktálů
Příloha D: CD obsahuje:

- databázi vytvořených fraktálů
- zdrojové texty programu s přeloženým binárním souborem
- plakát

Příloha A: Tabulka použitých variací

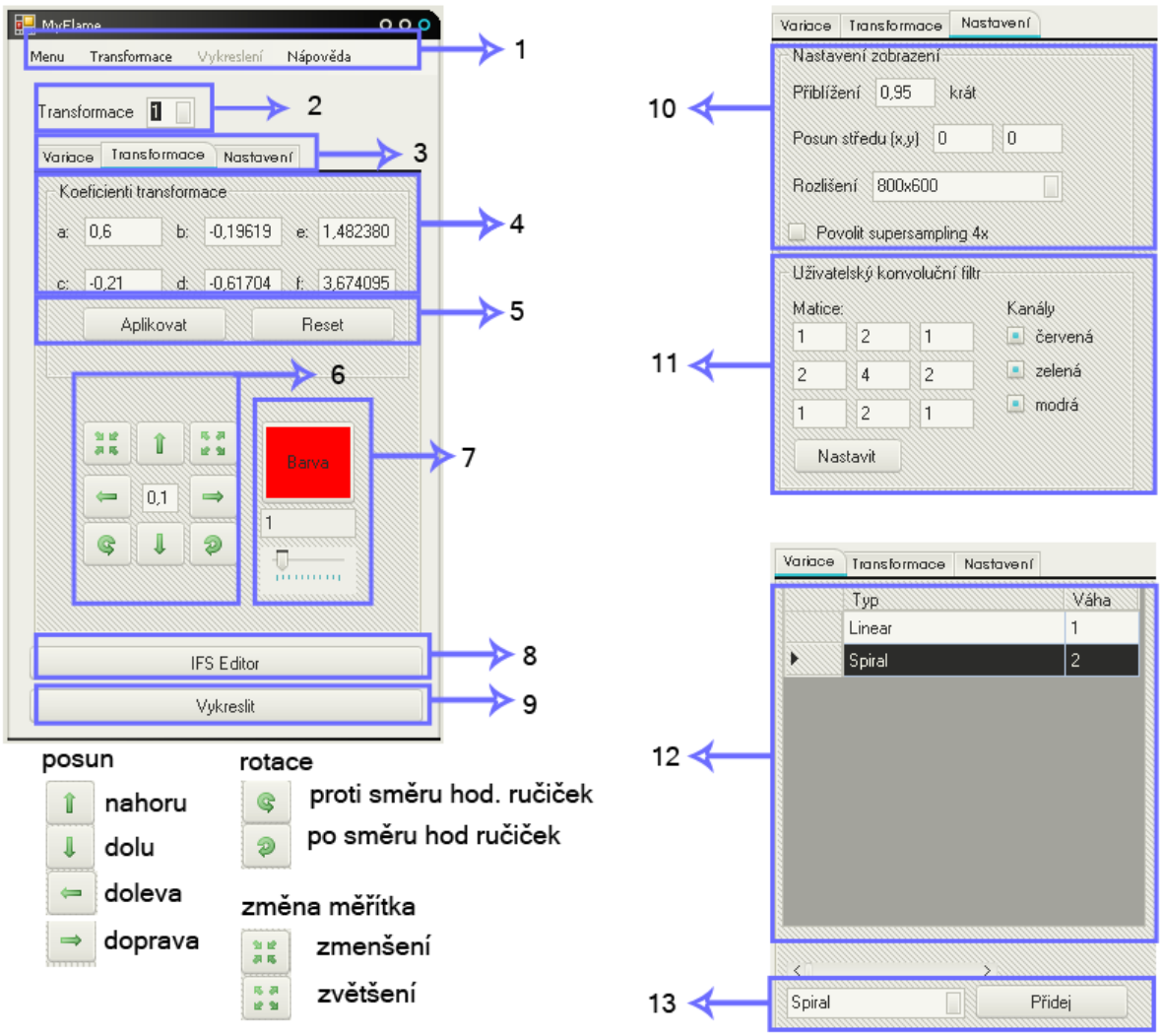
Označení	Přiřazená funkce	Originální název
$V_0(x,y)$	(x, y)	*Linear*
$V_1(x,y)$	$(\sin x, \sin y)$	*Sinusoidal*
$V_2(x,y)$	$\frac{1}{r^2}\cdot(x, y)$	*Spherical*
$V_3(x,y)$	$(x\cdot\sin(r^2)-y\cdot\cos(r^2), x\cdot\cos(r^2)+y\cdot\sin(r^2))$	*Swirl*
$V_4(x,y)$	$(r\cdot\cos(2\theta), r\cdot\sin(2\theta))$	*Horseshoue*
$V_5(x,y)$	$(\theta/\pi, r-1)$	*Polar*
$V_6(x,y)$	$(r\cdot\sin(\theta+r), r\cos(\theta-r))$	*Handkerchief*
$V_7(x,y)$	$(r\cdot\sin(\theta\cdot r), -r\cos(\theta\cdot r))$	*Heart*
$V_8(x,y)$	$((\theta\cdot\sin(\pi\cdot r))/\pi, (\theta\cdot\cos(\pi\cdot r))/\pi)$	*Disc*
$V_9(x,y)$	$((\cos\theta+\sin r)/r), ((\sin\theta-\cos r)/r)$	*Spiral*
$V_{10}(x,y)$	$((\sin\theta)/r, (\cos\theta)/r)$	*Hyperbolic*
$V_{11}(x,y)$	$(\sin\theta\cdot\cos r, \cos\theta\cdot\sin r)$	*Diamond*
$V_{12}(x,y)$	$(r\sin^3(\theta+r), r\cos^3(\theta-r))$	*Ex*
$V_{13}(x,y)$	$(\sqrt{r}\cos(\theta/2+\omega), \sqrt{r}\sin(\theta/2+\omega))$	*Julia*

kde

$$r=\sqrt{(x^2+y^2)}$$
$$\theta=atan(\frac{y}{x})$$

a ω je náhodné číslo z dvouprvkové množiny {0, π}.

Příloha B: Uživatelská příručka



Legenda k obrázku:

(1) hlavní menu programu

(2) ukazatel aktuální transformace

(3) záložky panelu (variace, transformace, nastavení)

(4) koeficienti transformační matice aktuální transformace

(5) potvrzení koeficientů (při ručním přepisu) a pro reset (nastavení identity)

(6) práce s transformacemi

(7) barva aktuální transformace, změna její váhy

(8) zobrazení editoru transformací

(9) vykreslení fraktálu

(10) nastavení zobrazení

(11) uživatelské nastavení konvolučního filtru

(12) oblast aktivních variací příslušné transformace

(13) rozbalovací menu s výběrem nové variace

Menu programu

Menu

- **Nový fraktál** - smaže všechny transformace a vytvoří nové pro jejich následnou editaci
- **Uložit obrázek***- uloží vygenerovaný obrázek ve formátu png do aktuální složky

Transformace

- **Přidat novou** - vytvoří novou transformaci a nastaví ji jako aktivní
- **Odstranit aktivní** - smaže aktuální transformaci, aktivní bude předešlá
- **Přepočítat pravděpodobnosti** - přepočet pravděpodobností transformací podle obsahů trojúhelníků
- **Rovnoměrná pravděpodobnost** - pravděpodobnost všech transformací bude stejná

Vykreslení*

- **Iterovat -** spustí iterační proces
- **Zastavit iteraci -** pozastaví iterační proces vykreslovacího okna
- **Aplikovat filtr -** aplikuje zvolený filtr na fraktál, aplikace může trvat delší dobu
- **Výběr filtru**
- **Výběr záře**
- **Síla filtru -** určuje kolikrát se má filtr aplikovat
- **Hustota -** určuje pro které pixely se má filtr aplikovat, tedy výběr hustoty zásahů

**položky, které jsou aktivní pouze při běhu vykreslovacího okna*

Záložka variace

V oblasti (12) se zobrazují aktivní variace k aktuálně editované transformaci (2). Přidání nové variace provedete výběrem variace z oblasti (13), tlačítkem přidat potvrdíte vaši volbu. Přidaná variace se ihned zobrazí v tabulce (12).

U aktivních variací je možné editovat jejich váhu, která je přednastavena u všech variací na 1. Váha variace prakticky udává kolikrát bude daná transformace použita. Váhou může být i necelé číslo jako například 1,4. Variaci si představte jako funkci, kterou tímto číslem násobíte.

Odstranění variace se provádí levým kliknutím myši na buňku, která je označena šipkou v oblasti (12), tím dojde k označení celého řádku s variací a následným stiskem klávesy *delete.*

Speciální variacemi jsou variace nulté transformace, které nenáleží finální transformaci. Jedná se o zvláštní případ, jelikož variace jsou použity při každé transformaci. Finální transformaci si lze představit jako nelineární kameru, kterou se díváme na fraktál.

Záložka transformace

Koeficienty transformační matice najdeme v oblasti (4). Jejich značení vychází z následujícího vzorce afinní transformace:

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

Rozumíte-li, co koeficienty představují, můžete je ručně změnit a aktualizovat tlačítkem *aplikovat* z oblasti (5). Pokud nechápete co znamenají, nevadí, z toho důvodu program obsahuje editor afinních transformací, který spustíte tlačítkem v oblasti (8), funkce editoru transformací bude probrána později.

Barvu aktuální transformace změníte kliknutím na barevné tlačítko *barva* v oblasti (7), otevře se systémový dialog pro výběr barvy.

Oblast (6) obsahuje tlačítka pro změnu transformační matice. Je možné je použít i bez spouštění editoru transformací, avšak při otevřeném editoru je práce s tlačítky intuitivnější.

Záložka nastavení

Záložka nastavení je jako jediná ze záložek nezávislá na zvolené transformaci. Slouží k nastavení vykreslovacího okna a uživatelského konvolučního filtru.

Nastavení zobrazení

Oblast (10) nám poskytuje základní nastavení vykreslovacího okna fraktálu.

- **Přiblížení** - udává koeficient přiblížení fraktálu, akceptuje desetinná čísla (1,1 znamená 110x zoom)
- **Posun středu** - posune střed fraktálu o zadaný počet pixelů
- **Rozlišení** - výběr rozlišení vykreslovacího okna
- **Zapnout supersampling 4x** - zapne metodu vylepšení obrazu, za následek má delší dobu generování fraktálu

Uživatelský konvoluční filtr

Zde je možné nastavit vlastní konvoluční matici. Akceptuje pouze celá čísla, jiný vstup překonvertuje na číslo 0 (to znamená, že buňka nebude mít vliv na výsledek filtru). Použití filtru můžete

zaškrtávacími boxy omezit jen na určité barvy. Je nutné dbát na to, aby při použití uživatelského konvolučního filtru byla v hlavním menu (1) *vykreslení → filtr* zvolena položka *uživatelský filtr.*

Editor transformací

Základní filosofie editoru transformací

Pro pochopení práce s editorem afinních transformací je nutná menší dávka abstrakce. Editor je založen na práci s trojúhelníky. Bázový objekt je objekt, který budeme transformovat. V editoru je naznačen šedou barvou a v ovládacím panelu mu je přiřazena transformace 0. Pro vytvoření fraktálu je nutné na bázový objekt aplikovat transformace (minimální počet transformací je roven dvoum). Tyto transformace v editoru představují ostatní trojúhelníky a jsou též provázány s ovládacím panelem. To znamená, že aplikace transformací se neprovádí přímo na bázový objekt, ale na jeho kopie, to vše aby bylo možné vizuálně oddělit jednotlivé transformace. Příkladem transformace je například rotace. Trojúhelník první transformace otočíte kolem počátku a dáte tím základ spirálového fraktálu.

Práce s editorem

Editor transformací spustíme kliknutím na tlačítko (8). Otevře se nové okno editoru, které obsahuje trojúhelníky, jejichž význam jsme si vysvětlily výše. Práci s nimi můžete provádět dvěma způsoby a nebo nejlépe jejich kombinací.

Způsob první spočívá v ruční deformaci trojúhelníků na základě práce s body. Levým kliknutím myši na jeden ze tří bodů dojde k aktivaci trojúhelníku. V ovládacím panelu se nastaví aktuální transformace korespondující s daným trojúhelníkem. Držením levého tačítka myši a jejím tahem změníte souřadnici aktivního bodu. Tím se změní i koeficienty aktuální transformace a i vzhled výsledného fraktálu, který v editoru můžete pozorovat v pravém rohu okna.

Druhá varianta je použít přednastavené transformace, které se nachází v oblasti (6). Jejich aplikací se změní transformace trojúhelníku. Číslo uprostřed udává krok transformace, jedná se o necelé číslo. Krok s hodnotou 1,0 značí velikost délky odvěsny bázového objektu.

Barvy trojúhelníků jsou určeny dle transformace ke které náleží.

Vykreslovací okno fraktálu

Spuštění provedeme tlačítem (9). Objeví se před námi okno v rozlišení, které jsme navolili v oblasti nastavení (10). V menu (1) se povolí zakázané položky. Zejména položka *vykreslení.* Pokud jsme

správně nastavili fraktál, měl by se před námi zobrazit výsledný fraktál. Implicitně se spustí nekonečná iterace, kterou můžeme zastavit v menu (1) *vykreslení → zastavit iteraci.*

Aplikaci filtrů provádíme položkou v menu (1) *vykreslení → aplikovat filtr.* Obrázek uložíme kliknutím na položku *menu → uložit obrázek.*

Při práci s vykreslovacím oknem je zakázáno měnit parametry fraktálu. Jediné co je možné změnit jsou hodnoty v matici uživatelského filtru (11).

Příloha C:

Ukázky vytvořených fraktálů