VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

EVOLUČNÍ NÁVRH OBRAZŮ TVOŘENÝCH L-SYSTÉMY

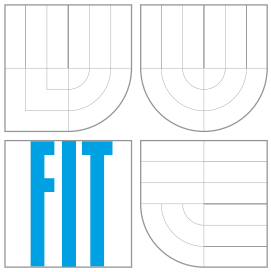
BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

ROMAN KOVAŘÍK

BRNO 2009

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

EVOLUČNÍ NÁVRH OBRAZŮ TVOŘENÝCH L-SYSTÉMY

EVOLUTIONARY DESIGN OF L-SYSTEM FRACTAL IMAGES

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — ROMAN KOVAŘÍK
AUTHOR

VEDOUCÍ PRÁCE — Ing. ZBYŠEK GAJDA
SUPERVISOR

BRNO 2009

Abstrakt

Tato práce pojednává o evolučním návrhu obrazů tvořených L-systémy. Vlastní návrh probíhá pomocí operátorů genetického programování. Ty jsou schopny pracovat s obrazem reprezentovaným ve formě syntaktického stromu. Ke komunikaci s uživatelem (návrhářem) slouží applet, který může být zobrazen na webové stránce.

Abstract

This work deals with an evolutionary design for images formed by L-systems. The design is supported by using the operators for genetic programming. This operators are able to work with the image represented in the form of syntax tree. User (designer) can use applet that can be displayed on the website.

Klíčová slova

L-systém, evoluční algoritmy, genetické programování, evoluční umění, evoluční návrh, grafické uživatelské rozhraní, applet.

Keywords

L-system, evolutionary algorithms, genetic programming, evolutionary art, evolutionary design, graphics user interface, applet.

Citace

Roman Kovařík: Evoluční návrh obrazů tvořených L-systémy, bakalářská práce, Brno, FIT VUT v Brně, 2009

Evoluční návrh obrazů tvořených L-systémy

Prohlášení

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

.....
Roman Kovařík
12. května 2009

Poděkování

Rád bych na tomto místě poděkoval svému vedoucímu bakalářské práce Ing. Zbyšku Gajdovi za jeho rady, návrhy a připomínky.

Obsah

Kapitola 1

Úvod

V posledních letech se můžeme čím dál častěji setkat s věcmi, jejichž design byl navržen samotným počítačem. Tato technika, vycházející z poznatků o biologické evoluci, se nazývá evoluční návrh. Uživatel (návrhář) určí jen pravidla pro vlastní generování a usměrňuje proces tvorby.

Tata práce se zabývá evoluční návrhem realizovaným pomocí genetického programování. Genetické programování je jednou ze skupin algoritmů využívaných k evolučnímu návrhu (tzv. evolučních algoritmů). Obecně genetické programování pracuje se syntaktickým stromem. Tento strom je vhodný k reprezentaci fraktálů. Tato práce předkládá kontrétně jeden z typů těchto útvarů – L-systémy.

Součástí zadání je i implementace programu pro interaktivní evoluční návrh s využitím rozhraní na webovou stránku, který prezentuje principy evolučního návrhu pomocí L-systémů. Hlavním cílem této práce je umožnit návrháři jednoduše vytvářet zajímavé obrazce, pomocí kombinace genetického programování a L-systémů.

Dané téma bakalářské práce jsme si vybral zéjména proto, abych čtenáře či uživatele pracujícího s vytvořeným appletem blíže seznámil s použitými metodami návrhu. Zejména s genetickým programováním, evolučním návrhem a tvorbou obrazů pomocí L-systémů. V neposlední řadě jsem měl možnost se blíže seznámit s tvorbou grafického uživatelského rozhraní v jazyce Java.

Kapitola 2 přibližuje princip evolučních algoritmů. Popisuje jednotlivé typy těchto algoritmů s důrazem na genetické programování. Kapitola 3 pojednává jednak o fraktálech obecně, tak o jednotlivých typech fraktálů. Předkládá základní pojmy s těmito útvary spojené a detailně se zaměřuje na L-systémy. V kapitole 4 se čtenář může dočíst o evolučním návrhu a používaných nástrojích k samotnému návrhu. Dozví se zde, z jakého popudu vznikl, kde brali jeho autoři inspiraci a blíže představují jedno z jeho odvětví – evoluční umění. Následující kapitola 5 se zabývá vlastní implementací programu. Popisuje vytvořený systém a pojednává o problémech, které byly v rámci implementace řešeny. Také vysvětluje symboly použité gramatiky a způsob ovládání programu. Dosažené výsledky prezentuje kapitola 6 ve formě naržených obrázků. Závěrečná kapitola 7 obsahuje zhodnocení dosažených výsledků a navrhuje další možný vývoj projektu.

Kapitola 2

Evoluční algoritmy

Evoluční algoritmy (dále jen EA) [5] jsou jednou z mnoha metod využívaných k řešení úloh. Každá metoda je úspěšná ve své třídě problémů. EA ve srovnání s jinými technikami produkují lepší výsledky v mnoha různých typech úloh. Mezi "konkurenční" techniky patří např. *hill-climbing*, *tabu-search* nebo *simulované žíhání*. EA jsou nejčastěji využívány k řešení optimalizačních problémů. Jejich podstata vychází z Darwinovy evoluční teorie. Ta spočívá v přirozeném výběru a přežívání jen těch jedinců, kteří mají největší schopnost přežít. Na rozdíl od přirozené evoluce se EA v několika ohledech liší. Protože na počítači nemůžeme simulovat vývoj trvající miliony let, je v případě EA počáteční populace náhodně generována. Stejně tak musí mít evoluční algoritmus na rozdíl od přirozené evoluce stanovenu ukončující podmínku. Tou bývá nejčastěji nalezení přijatelného jedince nebo vygenerování omezeného počtu populací. Obecně se každý EA sestává z těchto fází (viz také vývojový diagram na obr. 2):

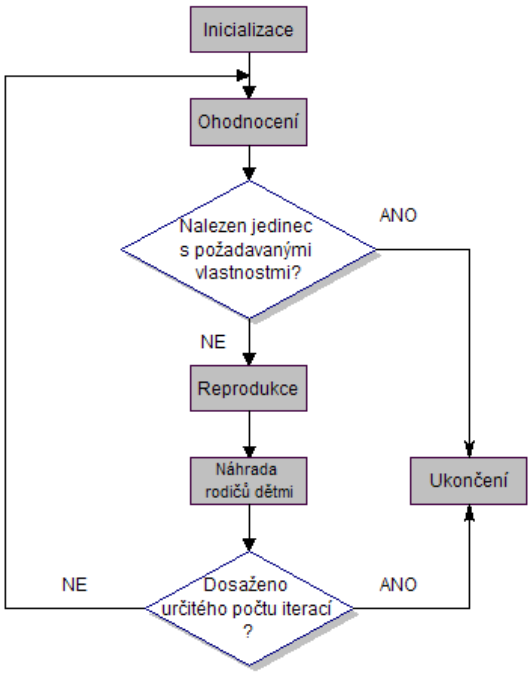
- inicializace,
- ohodnocení,
- reprodukce,
- náhrada rodičů dětmi,
- ukončení.

Počítač sám o sobě nerozumí tomu, jak funguje evoluce. Je ale řízen tak, aby zachoval nejlepší jedince v populaci a ty nevhodné odstranil. Po několika generacích produkuje jedince, kteří jsou lepší v porovnání s původní populací.

Mezi nejčastější úlohy řešené pomocí evolučních algoritmů patří data-mining, hraní her, strojové učení nebo plánování. Výhodou evolučních algoritmů je především rychlost a úspěšnost i v případě nepříznivých vstupních parametrů.

Rozlišujeme tyto čtyři základní skupiny evolučních algoritmů:

- genetické algoritmy,
- genetické programování,
- evoluční strategie,
- evoluční programování.



Obrázek 2.1: Vývojový diagram evolučního algoritmu.

2.1 Genetické algoritmy

Genetické algoritmy (dále jen GA) [9] jsou stochastické optimalizační techniky používané k řešení problémů, které jsou klasickými technikami těžko nebo pomalu řešitelné.

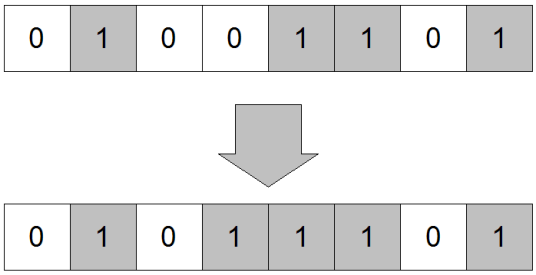
Za autora genetického algoritmu je považován John Holland, později se jimi zabýval David Goldberg. Původním cílem Johna Hollanda bylo vytvořit inteligentní systém založený na adaptivních procesech. Inspirací mu byla sama evoluce. Podstatou genetického algoritmu je napodobit vývoj organismu v komplexním prostředí. Jedince zde představuje chromozom reprezentovaný nejčastěji formou bitového řetězce. Jednotlivé bity řetězce si poté můžeme přestavit jako geny jedince. Jako jediná skupina evolučních algoritmů odděluje GA prostor hledání (*search space*) zahrnující genotypy[1] a prostor řešení (*solution space*) obsahující fenotypy[2].

Inicializace algoritmu probíhá vygenerováním populace ohodnocených jedinců.Vlastní ohodnocení probíhá aplikací tzv. *fitness funkce*[3]. Čím větší hodnota je jedinci přiřazena, tím větší šanci má poté uplatnit se při reprodukci. Reprodukce začíná výběrem jedinců (pravděpodobnost vybrání je přímo uměrná hodnotě fitness). Náhodný výběr je často realizován pomocí tzv. *rulety*. Jak už název napovídá, tato technika vychází z klasické rulety. Liší se ovšem v tom, že různí jedinci (čísla na ruletě) mají přiřazenu různě velkou výseč. Tím je docíleno náhodného výběru, ale zároveň jedinci s větší hodnotou fitness mají větší

[1] Genotyp je soubor všech genetických informací o organismu.
[2] Fenotyp je soubor všech pozorovatelných vlastností a znaků oraganismu.
[3] *Fitness funkce* určuje "vhodnost" jedince pro další vývoj.

čanci uplatnit se při procesu.

Dále reprodukce pokračuje aplikací dvou operátorů, operací *mutace* (obr. 2.2) nebo operací *křížení* (obr. 2.3). Mutace je prakticky provedena náhodným výběrem bitu (genu) v řetězci (chromozomu) , který je invertován. Pro operaci křížení jsou nejdříve vybrány dva chromozomy (dva rodiče). V každém z těchto chromozomů je náhodně vybrán bod křížení a každý z těchto chromozomů je rozdělen na dvě části. Druhé části chromozomů jsou nakonec mezi sebou zaměněny. Stálému křížení, které by již neprodukovalo vhodné jedince, má zabránit právě operátor mutace. Tento operátor tedy zajišťuje tzv. diverzitu populace (zabraňuje produkci stále stejných jedinců). Důležitým aspektem je zde prvek náhody, stejně jako v přírodě. Výsledkem reprodukce je tedy vznik nové populace (všichni původní jedinci jsou nahrazeni novými). Tento proces je aplikován, dokud nevznikne jedinec splňující určité kritérium (mající požadovanou vlastnost).



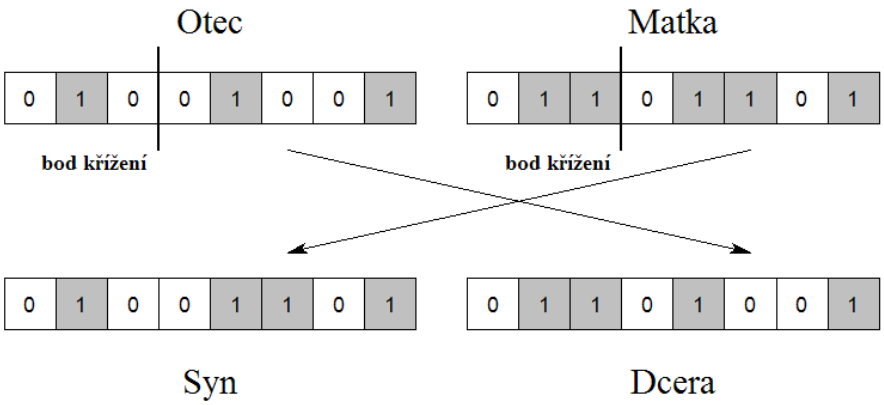
Obrázek 2.2: Mutace jedince při použití genetického algoritmu.

Existuje řada úprav genetického algoritmu poskytují lepší výsledky pro řešení určitého okruhu problémů. Mezi nejzajímavější patří [4]:

- *steady-state GA* (zde jsou jedinci generováni postupně jeden po druhém a nahrazují své předchůdce podle podobnosti nebo hodnoty *fitness*),
- *parallel GA* (využití více procesorů),
- *distributed GA* (samotatný vývoj několika populací + interakce mezi nimi),
- *messy GA* (proměnná délka chromozomů, dvoufázový evoluční proces),
- *hybrid GA* (kombinace s algoritmy pro hledání lokálního minima).

V praxi se GA uplatňují v teorii her, při hledání extrémů multimodálních funkcí, v rozpoznávacích systémech a dalších problémech řešených umělou inteligencí. Předností genetických algoritmů je možnost poskytnutí více řešení a možnost spolupracovat s uživatelem, který může do procesu zasahovat. Jejich výhodou je také to, že přestože jsou špatně provedeny, stále poskytují přijatelné výsledky (úspěšně se vyhýbají lokálnímu minimu). GA bývají ovšem často výpočetně náročné a výsledek nemusí být dosažen v konečném čase.

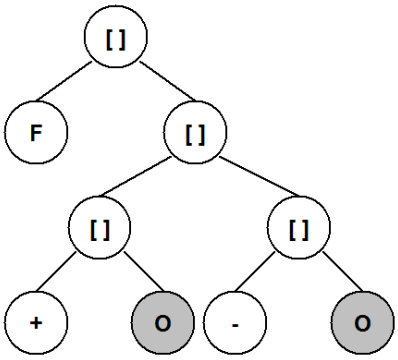
[4]Více různých variant zmiňuje Bentley ve své knize Evolutionary Design By Computers [1].



Obrázek 2.3: Křížení jedinců při použití genetického algoritmu.

2.2 Genetické programování

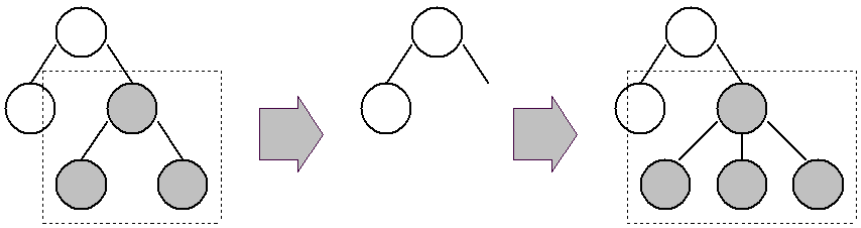
V případě Genetického programování (dále jen GP) se jedná o úpravu genetického algoritmu, kterou představil informatik John R. Koza. Zde ovšem jedince nepředstavuje pouze bitový řetězec, ale syntaktický strom. Příklad syntaktického stromu můžete vidět na obr. 2.4 Tento strom nemá pevný počet uzlů, jeho délka je proměnná. Pomocí syntaktického stromu můžeme vyjádřit složitější struktury, jako například jednoduché funkce. Listy syntaktického stromu jsou tvořeny *ternárními symboly* (vstupy programu, konstanty). Ostatní uzly stromu můžou obsahovat symboly pro operace se synovskými uzly a také uzavírají své potomky (symboly) do závorek. GP je nejčastěji používáno k učení a k tzv. symbolické regresi.



Obrázek 2.4: Jedinec reprezentovaný syntaktickým stromem.

V případě učení můžeme funkci reprezentovanou syntaktickým stromem považovat za

program popisující chování (např. nějakého zařízení). Vstupem tohoto programu jsou informace, které ovlivňují vlastní chování. Při zpracování terminálu je programu vrácena určitá hodnota. Úkolem je tedy sestrojit syntaktický strom nejlépe vystihující chování zařízení. V případě symbolické regrese je úkolem najít syntaktický strom představující funkci, která splňuje určité vlastnosti (nejlépe aproximuje vstupní množinu). Zde jsou listy stromu tvořeny proměnnými nebo konstantami a uzly představují n-nární operace (např. uzel představující binární operaci má dva listy).



Obrázek 2.5: Při mutaci je náhodně vybraný uzel i s celým svým podstromem odstraněn a nahrazen novým.

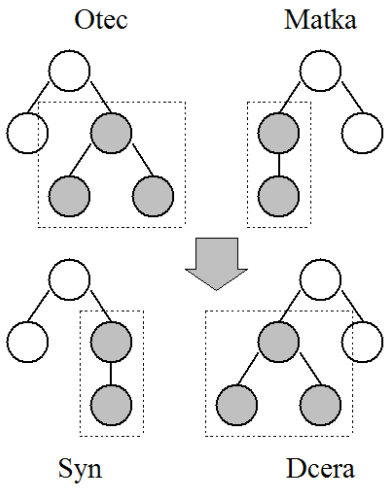
Na počátku algoritmu je vygenerována nová populace metodou *Grow*, *Full* nebo *Ramped Half-and-Half* [7]. V případě první varianty se uzly náhodně vybírají z dostupné sady terminálů a funkcí. Při výběru terminálu je větev ukončena. Druhá varianta vytváří strom do určité hloubky, kde je větev ukončena terminálem. Při použití třetí varianty jsou generovány stromy různé délky (do určitého maxima) a na polovinu tvorby stromu je použita technika *Grow*, na druhou technika *Full*.

Stejně jako v případě genetických algoritmů i zde jsou použity operátory mutace a křížení. Operátor mutace náhodně vybere uzel ve stromu a na jeho místě vygeneruje nový podstrom (znázorněno na obr. 2.5). Operátor křížení vybere dva stromy, u každého náhodně určí uzel křížení. Tyto uzly jsou poté mezi sebou prohozeny i se svými podstromy (názorněji na obr. 2.6). Výběr jedinců pro operace se nazývá *selekce*. Selekce pracuje náhodně podle určitého rozdělení pravděpodobnosti. Popřípadě může výběr určovat i člověk a tak zasahovat do procesu (usměrňovat vývoj správným směrem). Algortimus GP je ukončen po dosažení určité hodnoty *fitness* nebo po vygenerování určitého počtu populací. Prvek náhodného výběru zde hraje důležitou roli. Při vybírání jen těch jedinců s velkou hodnotou fitness, bychom dostali aplikací operátorů mutace nebo křižení pravděpodobně velmi kvalitní jedince. Problém je v tom, že množina, ze které bychom vybírali, by byla omezená. Tímto způsobem bychom dosáhli lepšího dočasného výsledku, ovšem horšího výsledku konečného.

Je to proto, že jedinec může obsahovat užitečnou informaci, přestože je jeho fitness nízká.

Původní algoritmus GP může být upraven použitím různých variací operátorů či reprezentace dat:

- *permutace* (spočívá v prohození dvou znaků ve stromě),
- *editace* (zkrácení dlouhých výrazů),
- *zapouzdření* ("zabalení" podstromu do jednoho uzlu, úspěšným příkladem jsou tzv. *ADF - Automatically Defined Functions*).



Obrázek 2.6: Při křížení jsou náhodně vybrané uzly i se svými podstromy mezi sebou prohozeny.

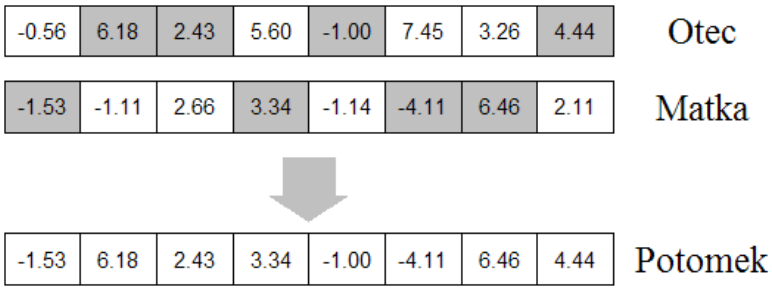
GP je s úspěchem používáno pro řešení např. nelineárních, kombinatorických úloh. V praxi je využíváno pro data mining nebo pro předpovídání počasí. Nevýhodou genetického programování je časová náročnost a velká šíře nastavitelných parametrů, různé způsoby aplikace operátorů mutace a křížení. Konkrétní implementace řeší obvykle metodou zkoušení různých variat s různými parametry a vybere se ta, která produkuje nejlepší výsledky.

2.3 Evoluční strategie

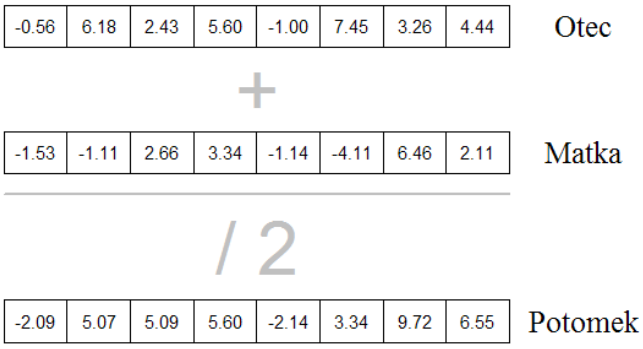
Evoluční strategii (dále jen ES) začali používat na Technické univerzitě v Berlíně studenti Peter Bienert, Hans-Paul Schwefel a Ingo Rechenberg. Jejich pole úspěšnosti však zahrnuje jen určitý typ úloh. S úspěchem se uplatňují při optimalizaci funkcí s mnoha extrémy, kde analytické řešení selhává nebo není možné.

Proměnné jsou na rozdíl od předchozích přístupů reprezentovány jako reálné hodnoty. Pro generování nových potomků jsou opět použity operátory mutace a křížení. Kvalitnější potomci nahrazují své rodiče, kteří jsou z budoucí generace odstraněni. Méně kvalitní

potomci jsou odstraněni z populace, další potomci vznikají z rodiče, na kterého byl aplikován operátor mutace. Výběr jedince určeného k mutaci se řídí normálním rozložením pravděpodobnosti. Nejpoužívanější varianty křížení jsou *diskrétní křížení* a *křížení průměrem* [6]. Diskrétní křížení tvoří vektor potomka výběrem hodnot z rodičovských vektorů (každá hodnota vektoru vznikne náhodným výběrem buď z jednoho nebo z druhého rodiče, názorněji na obr. 2.7). V případě křížení průměrem je tvořen průměrem jednotlivých hodnot obou rodičů (viz obr. 2.8).



Obrázek 2.7: Diskrétní křížení. Prvky vektoru (označené šedou barvou) jsou nahodně vybrány z "otce" nebo "matky".



Obrázek 2.8: Křížení průměrem.

Existují různé variaty ES (jak uvádí Bentley v [1]), označované jako:

- $(\mu + \lambda)$-ES,

- (μ, λ)-ES,

kde μ značí počet rodičů a λ počet potomků. Tyto varianty se v mnohém podobají genetickému programování. Ovšem rodiče nejsou vybírání náhodně, jako v případě GP, ale deterinisticky. ES navíc odděluje rodiče od potomků. První varianta vybírá nejlepší jedince z rodičů i potomků, druhá pouze z potomků. Pro funkčnost algoritmu musí být počet rodičů vždy menší než počet potomků. Jedna z variant ES rozlišuje i vliv různých proměnných, takže některé proměnné mají ve výsledku větší vliv. To je zajištěno rozdílným rozdělením pravděpodobnosti pro jednotlivé proměnné (tzv. "strategické proměnné").

2.4 Evoluční programování

Za objevitele evolučního programování (dále jen EP) je považován Dr. Lawrence Fogel (USA), dále bylo rozvíjeno jeho synem Davidem Fogelem. V mnohém se podobá evoluční strategii, i když bylo vyvinuto nezávisle. Původně bylo vyvinuto pro vytvoření stroje s inteligentním chováním. Chování bylo reprezentováno jako konečný stavový automat (FSM).

EP pracuje na principu horolezeckého algoritmu. Na rozdíl od evoluční strategie mají šanci uplatnit se i horší potomci. To je umožněno díky rozdělování jedinců do skupin. V každé skupině pak probíhá tzv. *turnaj*, tedy výběr nejlepších potomků.

Od genetického programování se od EP liší absencí operátoru křížení, používá jen operátor mutace. Je to proto, že EP se snaží napodobit vývoj jednotlivých druhů. Křížení mezi druhy v přírodě odvykle neprobíhá. Mutace probíhá u každého potomka v populaci.

Kapitola 3

Fraktály

Fraktál [10] [11] [13] je množina bodů vytvářející obraz s velkou vnitřní členitostí. Je to soběpodobný, fragmentovaný, nekonečně členitý geometrický útvar (jeho hlavní motiv se opakuje v nekonečně mnoha velikostech). Fraktál může být vyjádřen rekurzivní definicí, je příliš složitý pro vyjádření pomocí prostředků klasické Euklidovské geometrie. Fraktál se jeví na první pohled jako složitý útvar, ve skutečnosti je však tvořen opakovanou aplikací jednoduchých pravidel. Při rozšíření fraktálu o další rozměr (čas) je možné fraktály animovat.

Definice fraktálu podle B. Mandelbrota: *"Fraktál je takový útvar, jehož Hausdorfova*[1] *dimenze je větší než dimenze topologická"*.

Autorem označení fraktál (z latinského fractus – zlomený, roztříštěný) je výše zmíněný matematik Benoit B. Maldebrot, který poprvé fraktály matematicky definoval. Rozlišujeme čtyři základní typy fraktálů, které se liší metodou vytváření. Jsou to systémy iterovaných funkcí (IFS), L-systémy, dynamické systémy a stochastické systémy.

3.1 Základní pojmy

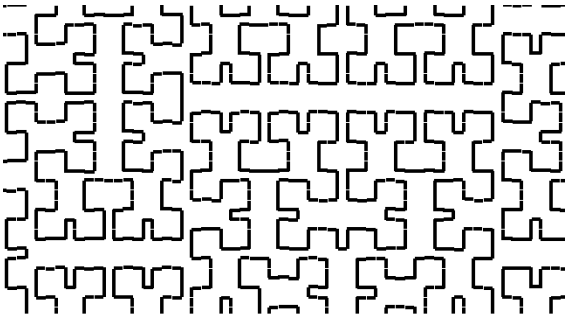
Nekonečně členité útvary

Hlavním předpokladem pro vysvětlení pojmu nekonečný členitý útvar je dělení na dimenze, které nejsou vyjádřeny celým číslem. S příkladem takového útvaru se setkáváme při měření délky hranic, pobřeží, řeky nebo výpočtu povrchu planety [10]. Při použití různých měřítek můžeme dojít k rozdílným výsledkům až v řádu násobků. Každým zmenšením měřítka totiž dostáváme stále členitější (delší) křivku. Tak se můžeme teoreticky dopracovat k nekonečné délce (povrchu). To je podstatný rozdíl od geometricky hladkého útvaru, jehož délka (povrch, objem) bude stejný při použití libovolného měřítka. Příklad nekonečně členitého útvaru můžete vidět na obrázku 3.1.

Soběpodobnost

Je vlastně invariance vůči změně měřítka. Libovolným přiblížením nebo oddálením dostáváme shodný útvar.

[1] Pojem Hausdorfova dimenze je vysvětlen v následující podkapitole 3.1.



Obrázek 3.1: Ukázka nekonečně členitého útvaru - Hilbertova křivka.

Soběpříbuznost

Vychází ze soběpodobnosti, ale je obecnější. Znamená, že libovolným přiblížením či oddálením nedostáváme úplně shodný, ovšem velmi podobný motiv (útvar). Praktickým příkladem může být struktura stromu.

Topologická dimenze

Je dimenze vyjádřená celým číslem. Obecně bod má nulovou topologickou dimenzi, křivka má dimenzi rovnu jedné, plošný útvar dvěma, těleso má topologickou dimenzi rovnu třem.

Fraktální (Hausdorfova) dimenze

Je dimenze vyjádřená reálným číslem. Např. na rozdíl od hladké křivky, která má dimenzi jedna, má geometricky nekonečně členitá křivka dimenzi větší. Ovšem nemusí být rovna dvěma (to je dimenze celé plochy), ale její hodnota je vyjádřena reálným číslem z intervalu $< 1, 2 >$. Z toho lze odvodit, že čím větší je rozdíl mezi dimenzí fraktrální a topologickou, tím je útvar členitější.

Atraktor

Je konečný (ustálený) stav systému. Například atraktorem pohybu planety je elipsa. Ovšem atraktorem pro různé objekty může být rozdílná množina bodů, či jeden jediný bod.

Rozeznáváme tyto typy atraktorů:

- bodové,
- cyklické,
- podivné (chaotické, všechny podivné atraktory představují fraktál).

Chaos

Na objevení chaosu má zásluhu americký matematik a meteorolog Edward Norton Lorenz. Ten se snažil sepsat rovnice popisující počasí. Pokoušel se zjistit proč se počasí nechová vždy podle předpovědi. Zjistil, že chování atmosféry je velmi citlivé na počáteční podmínky. Tedy i drobná změna má nečekané následky (tzv. *motýlí efekt*). Prvním popsaným chaotickým systémem bylo pravděpodobně vodní kolo se zavěšenými nádobami, které má také podivný atraktor.

Deterministický chaos

Vlastnost systémů s absencí náhodných elementů. Ta způsobuje nečekané, nepředvídatelné chování systému, které je ovšem stejné při zachování stejných počátečních podmínek.

3.2 Typy fraktálů

3.2.1 Systémy iterovaných funkcí (IFS)

IFS je generativní (deterministická nebo náhodná) metoda vytváření fraktálů. Poprvé ji v roce 1985 představil Stephen Demko, o dva roky později byla publikována také Michaelem Barnsleym. Michael Barnsley je autorem jednoho z nejzajmavějších přírodních fraktálů tvořeného pomocí IFS. Tím je obraz kapradiny, která je vlasně tvořena zmenšenými a orotovanými kopiemi sebe sama (viz obr. 3.2).

IFS vznikají aplikací souboru kontraktivních afinních zobrazení, které mění velikost hlavního motivu a natáčí jej. Kontraktivita zaručuje zmenšení původního motivu, použitím afinní transformace na libovolný tvar získáme opět stejný tvar (obvykle se využívají transformace pro změnu měřítka, posun, rotaci a zkosení). Ke tvorbě IFS fraktálu se často využívá tzv. algoritmus náhodné procházky. Ten je založen na iterativní náhodné aplikaci jednotlivých transformací na iterační bod (v každé iteraci vznikne pouze jeden nový bod). Druhým používaným algoritmem pro tvorbu IFS je tzv. deternistický algoritmus. Při jeho použití proběhne během každé iterace generování množiny bodů a všechny transformace jsou použity současně.

V praxi se IFS využívá často ke ztrátové kompresi dat (např. grafický formát FIF). Uplatnění našel i v počítačové grafice pro generování modelů přírodních útvarů, textur nebo k procedurální konstrukci objektů. V závislosti na počtu iterací lze systémy IFS využít i k simulaci růstu rostlin [2].

O evolučním návrhu obrazů pomocí IFS systémů pojednává [3].

3.2.2 Lindenmayerovy systémy (L-systémy)

L-systémy (Lindenmayerovy systémy, podle Aristida Lindenmayera, který tyto typy fraktálů zkoumal) [4] jsou vlastně fraktály definované přepisovacími gramatikami. Další z možných původců názvu L-systém je programovací jazyk LOGO, ve kterém můžeme pomocí kreslícího bodu (želvy) tvořit obrazy složené z úseček.

Typickou funkcí L-systémů je přepisování řetězců představujících gramatické pravidlo na základě pravidel, které definujeme. Gramatiku (nejčastěji regulární) tvoří množina symbolů. Každý symbol představuje určitý pohyb želvy (pohyb dopředu/dozadu, natočení do-

[2] Více o využití fraktůlů se lze dočíst v seriálu o fraktálech na webu root.cz [11].

Obrázek 3.2: Barnsleyho kapradina. Obrázek převzat z [11].

leva/doprava). Želva si pamatuje svou polohu a natočení. Ve své podstatě jsou L-systémy deterministické (jednomu přepisovatelnému symbolu odpovídá pouze jedno pravidlo). Klasické L-systémy se podobně jako IFS nejčastěji používají ke generování fraktálních objektů představující přírodní útvary.

Gramatika L-Systému

Deterministický bezkontextový L-systém je tvořen uspořádanou trojicí: G = [V,P,S], kde

- V je konečná množina symbolů,
- P je konečná množina pravidel,
- S je axiom (neprázdná startovací posloupnost symbolů).

Derivaci řetězce představuje přepsání všech přepisovatelných symbolů v řetězci odpovídajícím pravidlem z množiny pravidel P. Výsledný řetězec je poté interpretován želvou. Příklad jednoduché želví gramatiky můžete vidět v tabulce 3.1, ukázku derivace potom na obr. 3.3. Příklady L-systémů s různými základními prvky jsou na obr. 3.4.

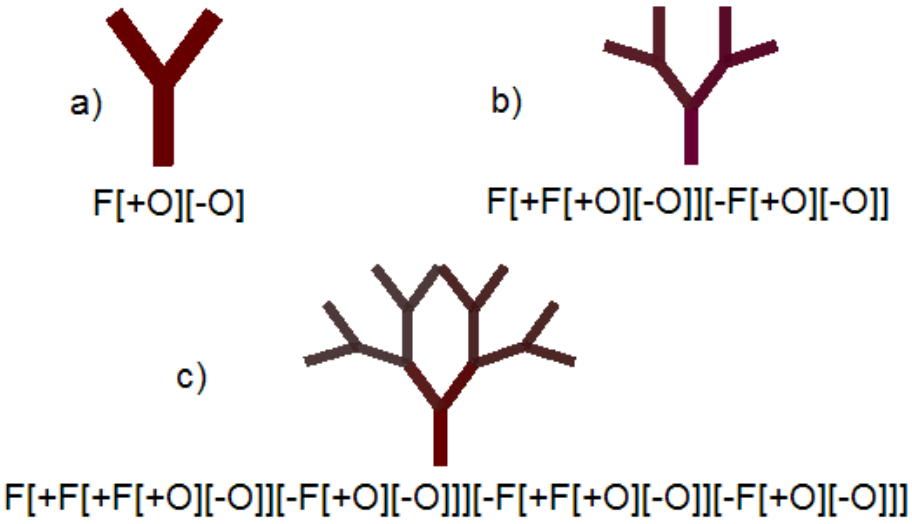
Závorkové L-systémy

Závorkové L-systémy rozšiřují základní sadu příkazů pro želvu o otvírací a uzavírací závorku ([,]). To proto, aby želva mohla tvořit větvící se struktury. K tomu musí mít paměť, do které si může uložit svůj aktuální stav (místo, kde se nachází a směr natočení). K tomu je využíván princip zásobníku. Když tedy želva narazí na otevírací závorku, uloží svůj stav na zásobník. Ten je ze zásobníku vybrán, až želva narazí na odpovídající uzavírací závorku.

Mezi nejznámější obrazce vytvářené pomocí L-systémů patří Hilbertova křivka, Peanova křivka, dračí křivka, Cantorův prach, Sierpinského trojúhelník nebo ostrov, křivka a vločka Helge von Kocha. Dva posledně jmenované obrazce, jejichž autorem je švédský matematik Niels Fabian Helge von Koch, mají některé zajímavé vlastnosti. Jsou nekonečně dlouhé, spojité, ale nemají v žádném svém bodě tečnu.

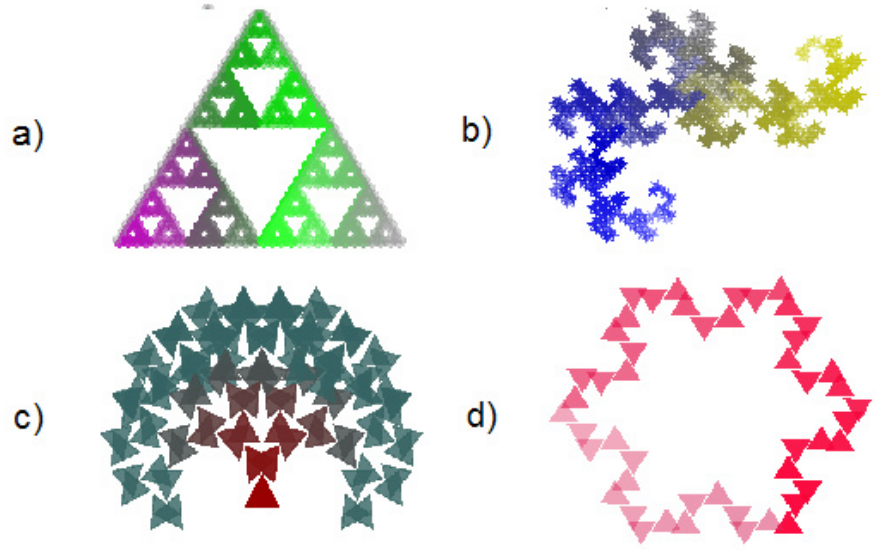
Tabulka 3.1: Příklad jednoduché želví gramatiky.

Jednoduchá želví gramatika	
F	posun želvy dopředu s nakreslením úsečky
G	posun želvy dopředu bez nakreslení úsečky
B	posun želvy dozadu s nakreslením úsečky
+	natočení želvy doleva
-	natočení želvy doprava
[	uložení stavu želvy do zásobníku
]	vyjmutí stavu želvy ze zásobníku



Obrázek 3.3: Ukázka a) axiomu, b) první a c) druhé iterace L-systému představujícího strom i s gramatickým předpisem. Je použito jediné přepisovací pravidlo, kterým je sám axiom.

V praxi mají L-systémy podobné využití jako IFS. U konstrukcí objektů poskytují navíc reprezentaci vhodnou k další úpravě objektů (využití v CAD).



Obrázek 3.4: Ukázka L-systémů s různými základními prvky: Sierspinského trojúhelník (šestiúhelník), dračí křivka (hvězda), strom (trojúhelník) a kochova vločka (trojúhelník).

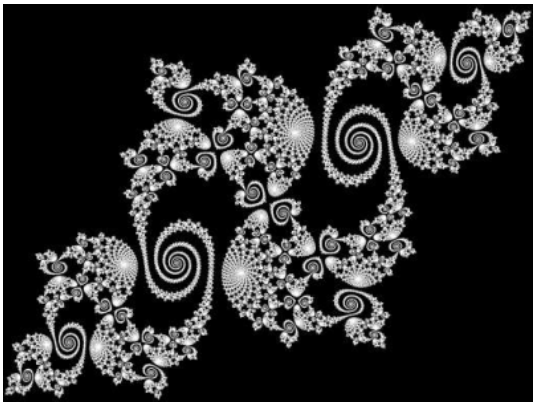
3.2.3 Dynamické systémy

Dynamické systémy se od předchozích typů fraktálů liší zajímavou vlastností. Tou je závislost na čase (nebo obecně na nějaké proměnné). Dynamický systém je tvořen stavovým prostorem a je určen počátečními podmínkami. V každém časovém okamžiku je stav systému určen stavovým vektorem. Změna v čase je nejčastěji vyjádřena soustavou diferenciálních rovnic. Nejznámějšími dynamickým systémy jsou tzv. funkce populačního růstu, Mandelbrotova množina nebo Juliovy množiny. Funkce populační růstu vychází z pozorování tohoto jevu v přírodě. Projevuje se zde závislost na minulosti, hodnota růstu v závislosti na velikosti populace a další charakteristické jevy. Zajímavostí je, že Mandelbrotova množina (pojmenovaná po "otci fraktálů") není fraktálem sama o sobě. Tím je až její hranice, jejíž rozdíl mezi topologickou a Hausdorfovou dimenzí je největší, kterého lze dosáhnout.

Iterační vyjádření Juliovy monožiny: $z_{n+1} = z_n^2 + c$.

Variace Mandelbrotovy množiny: $z_{n+1} = \sin(z_n^2) + c$, $z_{n+1} = \cos(z_n^2) + c$.

Juliovy množiny (viz obr. 3.5) jsou v podstatě Mandelbrotova množina s odlišnou počáteční podmínkou a konstantou c. Ta je u Mandelbrotovy množiny jedna jediná možná, kdežto pro Juliovu množinu můžeme použít libovolnou hodnotu konstanty. Zvláštním případem dynamického systému je tzv. deterministický chaos. Pro něj je typická nemožnost určit okamžitě stav systému v jakémkoli časovém okamžiku v budoucnosti. Podmínkou pro vznik chaotického systému je množství nestabilních počátečních podmínek. Ukázkou chaotického systému může být počasí.



Obrázek 3.5: Ukázka Juliovy množiny. Obrázek převzat z [11].

3.2.4 Stochastické systémy

Jak už z názvu této skupiny fraktálů vyplývá, tyto systémy jsou nedeterministické. Při jejich tvorbě je uplatněna náhoda. Díky tomu vznikají nepravidelné fraktály (tomuto typu fraktálů se také říká jednoduše fraktály nepravidelné). Tyto fraktály již nejsou soběpodobné, ale pouze soběpříbuzné. Ke generování tohoto typu fraktálů se používá např. spektrální syntéza, metoda přesouvání středního bodu nebo simulace Brownova pohybu (náhodný pohyb mikroskopických částic, které do sebe neustále vrážejí a tím se náhodně přemisťují) Znázornění Brownova pohybu můžete vidět na obr. 3.6.

Spektrální syntéza vychází z Fourierovy řady. Na počátku vytvoříme náhodné Fourierovy obrazy, na ty pak použijeme inverzní Fourierovu transformaci. Výsledkem mohou být objekty připomínající členitý terén. Předností spektrální syntézy je možnost rozšíření do více dimenzí, kde můžeme modelovat "neploché objekty".

Metodou přesouvání středního bodu lze generovat jednak objekty přírodní krajiny, při rozšíření o další dimenzi pak modelovat např. mraky nebo plasmu. Modely plasmy jsou použitelné jako výšková pole určená k otexturování (počítačové hry) nebo slouží k 3D reprezentaci objektů ve formě voxelů nejčastěji používané medicíně.

Obrázek 3.6: Ukázka brownova pohybu. Obrázek převzat z [2].

3.3 Nejznámější fraktály vykreslované pomocí L-systémů

Cantorovo mračno

Jeden z nejjednodužšších fraktálů. Jeho konstrukce je jednoduchá: základem je úsečka, tu rozdělíme na třetiny a prostřední třetinu vyjmeme. Se zbylými dvěma úsečkami opakujeme iterativně celý proces donekonečna (viz obr. 3.7). Takto vzniklý útvar je nutno nekonečněkrát zkopírovat vedle sebe. Tento fraktál lze zkonstruovat jak pomocí IFS, tak pomocí L-systému. Za autora tohoto fraktálu je považován německý matematik George Cantor. Topologická dimenze Cantorova mračna (také nazývaného jednoduše Cantorova množina, Cantorův prach nebo Cantorovo diskontinuum) je $D_t = 0$, fraktální dimenze je $D_f = \frac{log2}{log3} \doteq 0,6$.

Obrázek 3.7: Ukázka prvních čtyřech iterací časti Cantorova mračna.

Gramatika Cantorova mračna pro L-systém:

- $V = \{X, G\}$,
- $P = \{X \rightarrow XGX, G \rightarrow GGG\}$,
- $S = X$,
- základní tvar úsečka.

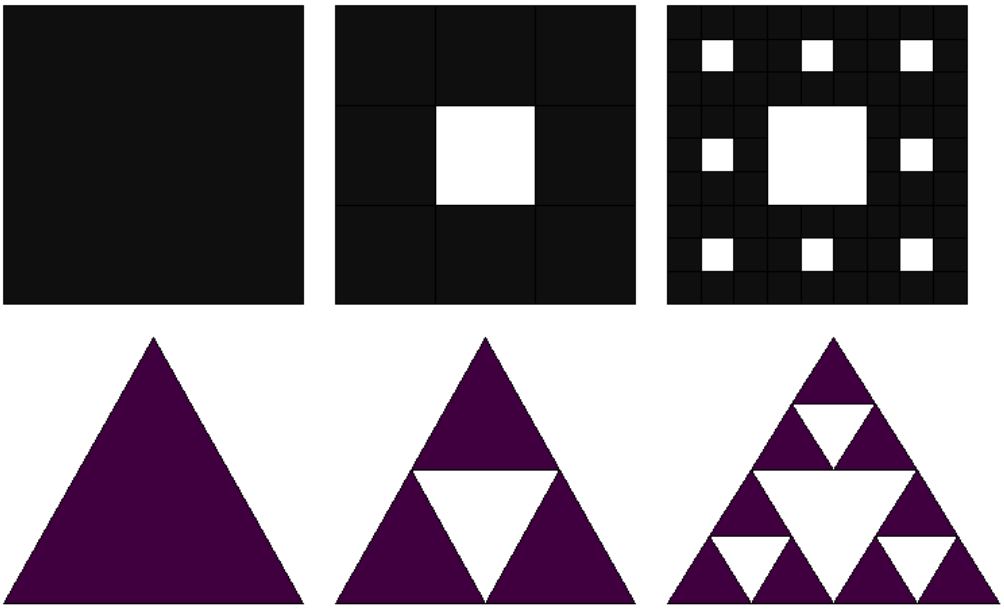
Sierpinského kobereček

Startovacím obrazcem tohoto fraktálního útvaru je čtverec. Ten je rozdělen na devět stejně velkých čtverců, prostřední z nich je vyjmut. Se zbylými osmi čtverci opakujeme iterativně celý proces donekonečna (viz obr. 3.8). Tento fraktál je typickým představitelem fraktálů

vykreslených pomocí IFS. Autorem fraktálu je polský profesor matematiky Waclaw Franciszek Sierpinski. Topologická dimenze tohoto fraktálu je $D_t = 1$, fraktální dimenze je $D_f = \frac{log8}{log3} \doteq 1,9$.

Gramatika Sierpinského koberečku pro L-systém:

- $V = \{X, G\}$,
- $P = \{X \rightarrow XGX + G + GG + +XXX - GG - GG - -XXX + G-$,
  $G \rightarrow GGG + G + GG + GG + GG + G-\}$,
- $S = X$,
- úhel $\frac{\Pi}{2}$,
- základní tvar čtverec.



Obrázek 3.8: Ukázka prvních dvou iterací Sierpinského koberečku a Sierpinského trojúhelníku.

Sierpinského trojúhelník

Další z úvarů poprvé přestavených známým polským matematikem. Konstrukce je velmi podobná konstrukci předchozího útvaru. Zde ovšem základ netvoří čtverec, ale jak už název napovídá, trojúhelník. Ten je rozdělen na čtyři shodné menší trojújelníky a prostřední je vyjmut (viz obr. 3.8). Jedná se o plošný obrazec, topologická dimenze je tedy opět je $D_t = 1$, fraktální dimenze je $D_f = \frac{log3}{log2} \doteq 1,6$.

Gramatika Sierpinského trojúhelníku pro L-systém:

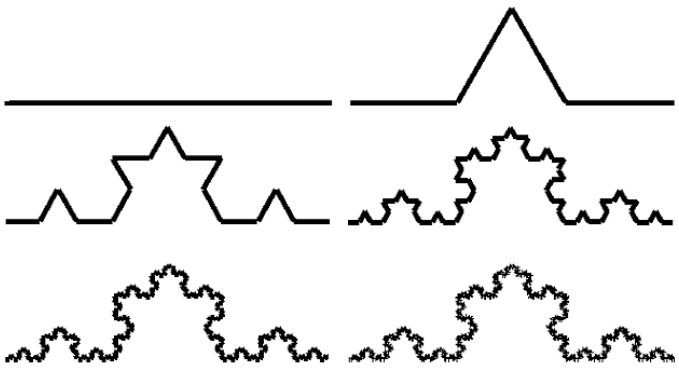
- $V = \{X, G\}$,
- $P = \{X \to XX, M \to M - X + M + X - M\}$,
- $S = M - X - X$,
- úhel $\frac{2\Pi}{3}$,
- základní tvar úsečka.

Kochova křivka a Kochova vločka

Tvůrcem obrazce je švédský matematik Helge von Koch. Tento obrazec je typickým fraktálem, který je tvořen pomocí L-systému. To proto, že základním objektem je úsečka, tedy standardní prvek L-Systému. Křivku získáme z úsečky, jejíž prostřední třetinu změníme na rovnostranný trojúhelník bez základny. Tento postup opakujeme se všemi vzniklými úsečkami donekonečna (viz obr. 3.9). Tato křivka je nekonečná a nemá v žádném svém bodě tečnu. Podobně získáme Kochovu vločku, výchozím tvarem bude ovšem trojúhelník (viz obr. 3.10). Topologická dimenze křivky je $D_t = 1$, fraktální dimenze je $D_f = \frac{log4}{log3} \doteq 1,3$.

Gramatika Kochovy křivky pro L-systém:

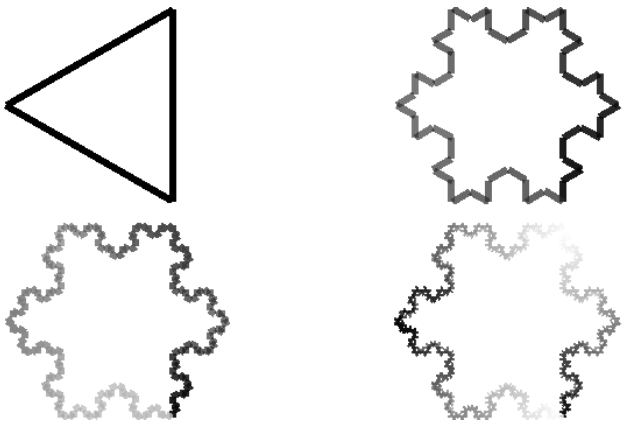
- $V = \{X, +, -\}$,
- $P = \{X \to X - X + +X - X, M \to M - X + M + X - M\}$,
- $S = X$,
- úhel $\frac{\Pi}{3}$,
- základní tvar úsečka.



Obrázek 3.9: Ukázka pěti iterací Kochovy křivky.

Gramatika Kochovy vločky pro L-systém:

- $V = \{X, +, -\}$,
- $P = \{X \to X + X - -X + X\}$,
- $S = X - -X - -X$,
- úhel $\frac{\Pi}{3}$,
- základní tvar úsečka.



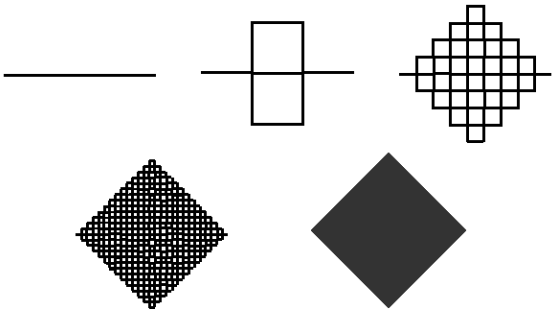
Obrázek 3.10: Ukázka tří iterací Kochovy vločky.

Peanova křivka

Základem je opět úsečka rozdělená na třetiny. Prostřední z nich je nahrazena sedmi úsečkami stejné délky, které tvoří dva čtverce s jednou totožnou stranou (viz obr. 3.11). Zajímavostí je, že Peanova křivka vyplňuje celou rovinu. Její fraktální dimenze je tedy $D_f = \frac{log9}{log3} \doteq 2$, tedy stejná hodnota jako její dimenze topologická. "Otcem" křivky je italský matematik a logik Giuseppe Peano.

Gramatika Peanovy křivky pro L-systém:

- $V = \{X, M, +, -\}$,
- $P = \{X \to MXM + MXM + MXM + MXMMXM + MXM + MXM-, M \to MXM\}$,
- $S = MXM$,
- úhel $\frac{\Pi}{2}$,
- základní tvar úsečka.



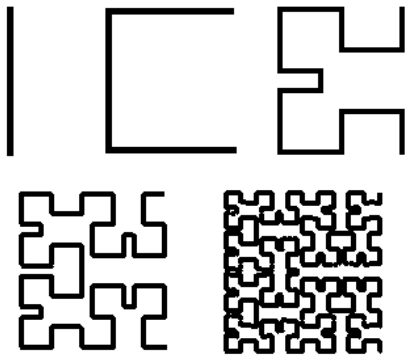
Obrázek 3.11: Iterace Peanovy křivky.

Hilbertova křivka

Každá další iterace Hilbertovy křivky se tvoří ze čtyř Hilbertových křivek získaných v předchozí iteraci. Každá z nich je ovšem zmenšena a různě natočena či zrcadlově převrácena (názorněji na obr. 3.12). Hilbertova křivka, stejně jako křivka Peanova, pokrývá celou rovinu. Její fraktální dimenze je teda rovna dvěma.

Gramatika Hilbetovy křivky pro L-systém:

- $V = \{X, M, +, -\}$,
- $P = \{X \rightarrow -MF + XFX + FM-, M \rightarrow +XF - MFM - FX+\}$,
- $S = MXM$,
- úhel $\frac{\Pi}{2}$,
- základní tvar úsečka.



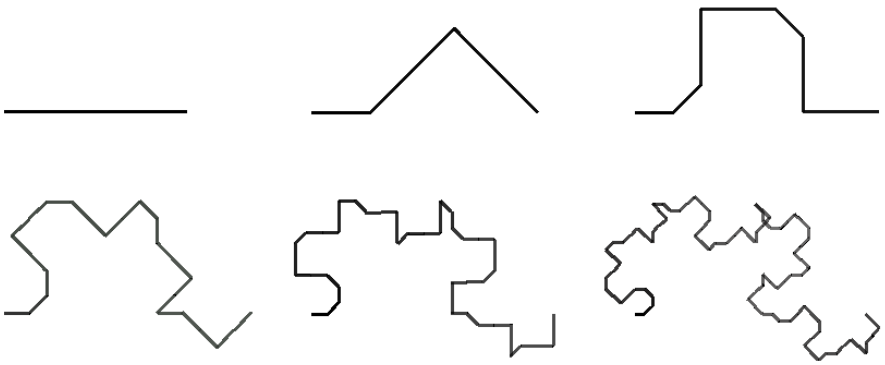
Obrázek 3.12: První čtyři iterace Hilbertovy křivky.

Dračí křivka

Existují různé variaty dračích křivek. Jedna z nich, tvořena níže uvedenou gramatikou, je znázorněna na obr. 3.13. Její fraktální dimenze je rovna dvěma.

Gramatika dračí křivky pro L-systém:

- $V = \{F, M, N, +, -\}$,
- $P = \{M \rightarrow -FM + +FN-, N \rightarrow +FM - -FN+\}$,
- $S = FN$,
- úhel $\frac{\Pi}{4}$,
- základní tvar úsečka.



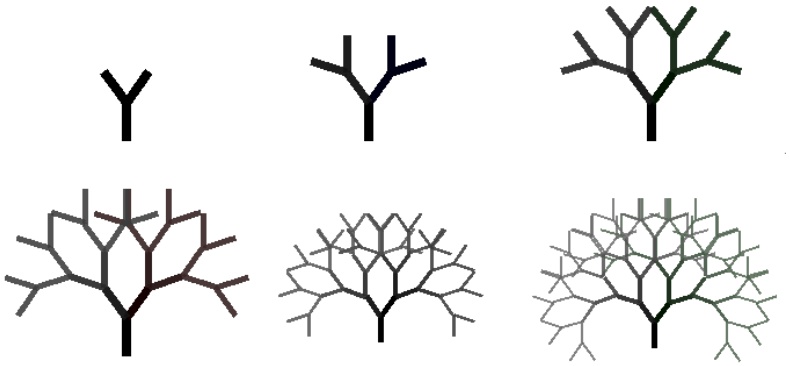
Obrázek 3.13: Ukázka tvorby dračí křivky.

Strom

U vykreslení stromové struktury pomocí L-systému jsou poprvé využity symboly otevírací a uzavírací závorky. Využívá se tedy zásobníku pro návrat na vrchol předchozí větve. Pro realističtější zobrazení stromu či keře se může využít rovněž zkrácení a zúžení větve v každé další iteraci. Vhodná je také postupná změna barevné složky (např. přechod od hnědého knene po zelené listy). Takto ovšem vznikne velmi pravidelný a na pohled pěkný strom (viz obr. 3.14). Pro simulaci přírodních objektů se proto využívá ještě náhodná změna úhlu či délky větve.

Gramatika pro tvorbu stromu pomocí L-systému:

- $V = \{F, O, +, -, [,]\}$,
- $P = \{O \rightarrow [CTSF[+O][-O]]\}$,
- $S = F[+O][-O]$,
- úhel $\frac{\Pi}{5}$,
- základní tvar úsečka.



Obrázek 3.14: Několik iterací stromové struktury.

Trojrozměrná Peanova křivka

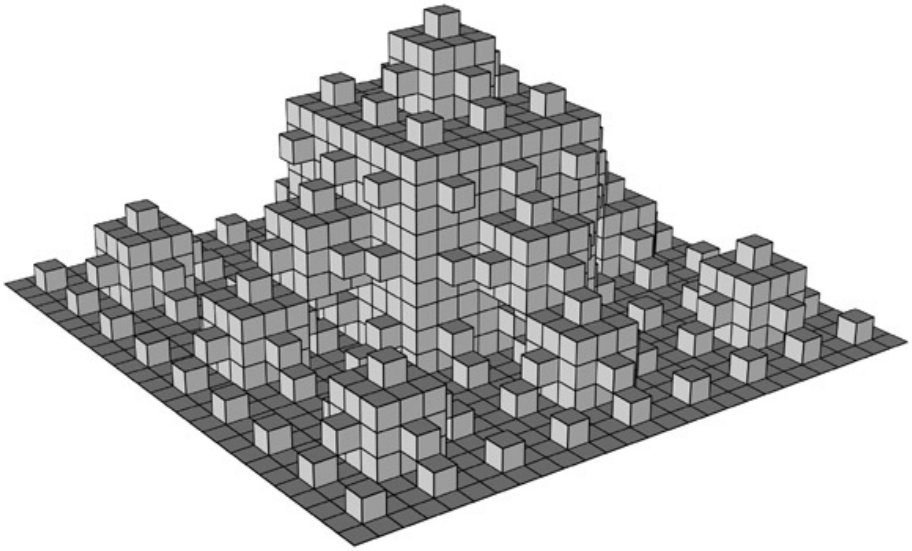
Při tvorbě se vychází ze čtverce. Ten je rozdělen na devět stejných čtverců, prostřední z nich je nahrazen krychlí bez spodní podstavy. Takto jsme získali třináct shodných čverců tvořící povrch obrazce. s těmito čtverci opakujeme zmíněný postup. Výsledek můžete vidět na obr. 3.15. Topologická dimenze tohoto trojrozměrného obrazce je je rovna dvěma, fraktální potom $D_f = \frac{log13}{log3} \doteq 2,3$. Protože každé přidané krychli chybí spodní podstava, při pohledu zespodu obrazce můžeme vidět Sierpinského koberec.

Gramatika pro tvorbu trojrozměrné Peanovy křivky pomocí L-systému:

- $V = \{X, G, B, +, -, \hat{}, \&\}$[3],
- $P = \{X \rightarrow X\hat{}X+\&XB--\hat{}\hat{}XB+X\&X\hat{}X+G-BBBXXX-GG+XXX+G-\}$,

[3] ^ a & jsou symboly gramatiky pro pohyb v trojrozměrném prostoru. "^" značí naklonění želvy přední částí směrem nahoru a "&" naklonění želvy přední částí směrem dolů. Více o trojrozměrných L-systémech v [11].

- $S = X$,
- úhel $\frac{\Pi}{2}$,
- základní tvar čtverec.



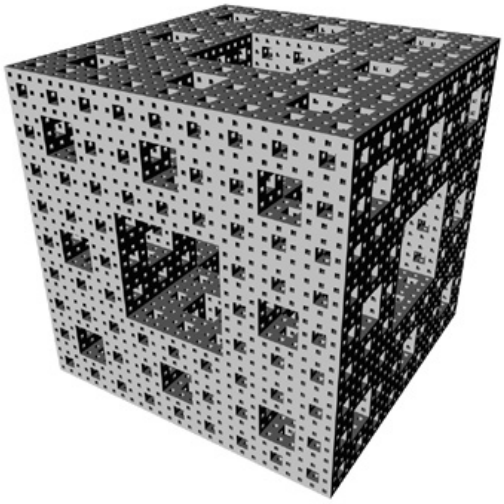
Obrázek 3.15: Trojrozměrná Peanova křivka. Obrázek převzat z [12].

Mengerova houba

Při konstrukci Mengerovy houby, pojmenované podle rakouského matematika Carla Mengera, se vychází z vyplněné krychle. Tato krychle je rozdělena na 3x3x3 stejných krychlí s třetinovou délkou hrany. Krychle umístěné ve středech stěn celkové krychle a krychle v samotném středu jsou vyjmuty (celkem je tedy vyjmuto sedm krychlí). Se zbylými dvaceti krychlemi je opakován zmíněný postup. Vzhled obrazce po čtyřech iteracích je vidět na obr. 3.16. Topologická dimenze Mengerovy houby se rovná dvěma, fraktální dimenze je potom $D_f = \frac{log20}{log3} \doteq 2,7$. Podobně jako u trojrozměrné Peanovy křivky můžeme vidět při pohledu zespodu Sierpinského koberec, zde jej můžeme vidět ze všech šesti stran obrazce.

Gramatika pro tvorbu Mengerovy houby pomocí L-systému:

- $V = \{X, G, +, -, \hat{}, \&\}$,
- $P = \{X \rightarrow +X-G-X-G-X-G-X\hat{}G\&XXX-XX-XX-XG-\&GX\hat{}XXX-XX-XX-X-G-G\hat{}G\&\}$,
- $S = X$,
- úhel $\frac{\Pi}{2}$,



Obrázek 3.16: Mengerova houba. Obrázek převzat z [12].

- základní tvar krychle.

Kapitola 4

Evoluční návrh

Počítače byly v minulosti chápány jako "pouzí" pomocníci neschopní kreativního myšlení. Našli se ovšem i tací uživatelé, kteří chtěli dokázat, že tomu tak nemusí být. Díky tomu byly vyvinuty programy téměř dosahující kvalit člověka v oblasti návrhářství. Zde se nejedná už o počítačem podporovaný návrh (CAD), ale o návrh realizovaný samotným počítačem.

Klíčem k úspěchu v této oblasti se ukázalo využití principů biologické evoluce. Touto technikou byly úspěšně navrženy umělecké obrazy, sochy, počítačové sítě, budovy, mosty nebo analogové obvody.

Evoluční nástroje

Jsou pokročilé softwarové nástroje určené pro zlepšení produktivity a kvality navrhování. Umožňují vytvářet fotorealistické obrazy nebo vytvářet animace. Evoluční návrh je založen na použití těchto nástrojů. Jednou z mnoha předností tohoto přístupu je možnost porovnání mnoha potencionálních řešení a vybrat to nejlepší. Přesto, že lze pomocí evolučních nástrojů dosáhnou velmi dobrých výsledků, jejich činnost musí být kontrolována člověkem.

Biologická evoluce

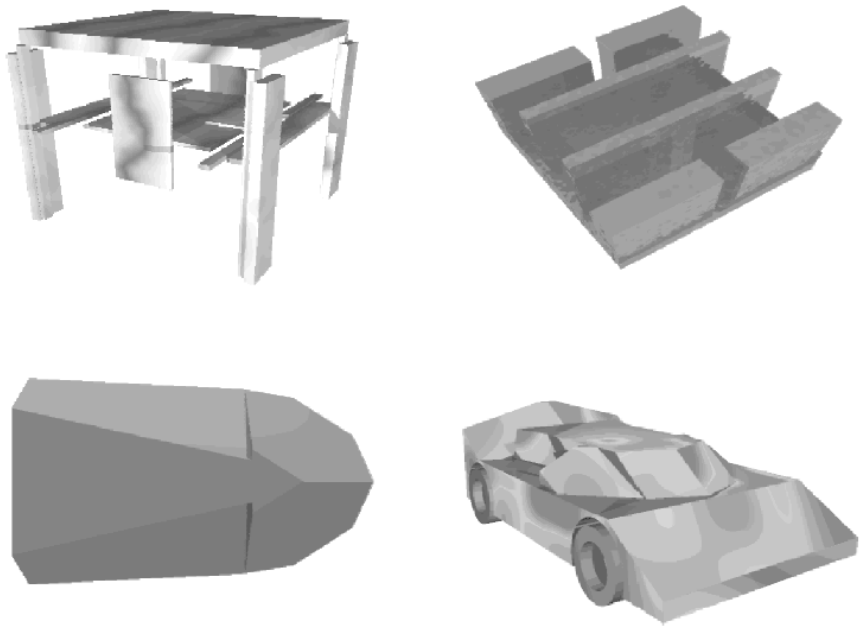
Je inspirací pro evoluční návrh pomocí počítačů. V přírodě dochází k vývoji už stovky milionů let a mnoho "lidských" vynálezů si bralo příklad ať už vědomě či nevědomě v přírodě (křídlo, čerpadlo atp.).

Dnes se evoluční návrh používá v těchto oblastech:

- kreativní evoluční návrh (creative evolutionary design),
- evoluční umění (evolutionary art),
- evoluční formy umělého života (evolutionary artificial life forms),
- evoluční optimalizace návrhu (evolutionary design optimisation).

Evoluční umění

Je dnes nejrozšířenější oblast evolučního návrhu. Typické je pro něj určení hodnoty *fitness funkce* člověkem. Vývoj je tedy ovlivněn estetickým cítěním daného člověka. Velikost populace je obvykle malá (kolem deseti jedinců), aby jejich *fitness* byl člověk schopen určit v krátkém čase. Výsledkem bývá často věc s atraktivním vzhledem, ovšem s omezenou funkčností (což může být problém například při navrhování nábytku). Dalším problémem je různorodost navržených objektů (problém navržení sady k sobě patřících objektů).

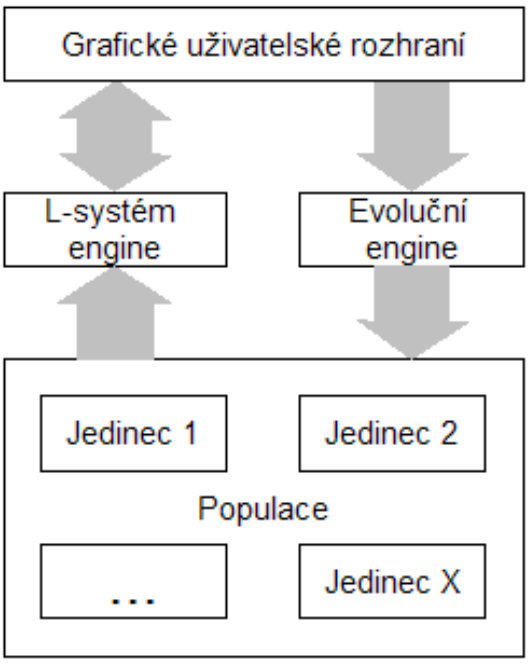


Obrázek 4.1: Ukázka kreativního evolučního umění. Obrázek převzat z [1].

Kapitola 5

Implementace

Součástí zadání této bakalářské práce bylo implementovat rozhraní na webovou stránku. Proto byl zvolen programovací jazyk Java, který umožňuje implemetaci appletu. Tento applet umožňuje uživateli interaktivní tvorbu obrazů tvořených pomocí operátorů genetického programování. Listy stromu jsou tvořeny prvky gramatiky L-systému. Každý jedinec (jednotlivé obrazy - *fenotypy*) je reprezentován syntaktickým stromem (ten představuje *genotyp*). Blokové schéma programu je vidět na obr. 5.1.



Obrázek 5.1: Blokové schéma programu. Ke komunikaci s uživatelem slouží *Grafické uživatelské rozhraní*, k vykreslení jedinců slouží *L-Systém engine*. Operace nad syntaktickými stromy (operace genetického programování, změna pravidel, ...) jsou prováděny *Evolučním enginem*.

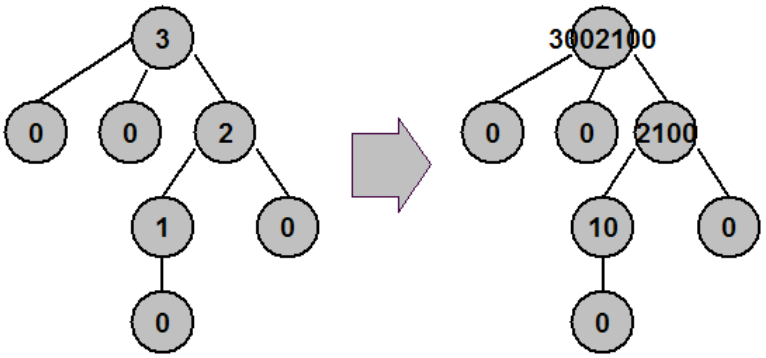
5.1 Popis systému

Při spuštění systému jsou vygenerováni jedinci populace. Uživatel může na vybrané jedince aplikovat operátory mutace nebo křížení. Pro vytvoření nového jedince může použít předdefinované klasické obrazce (dva druhy stromu, dvě variace křivky Helge von Kocha, dva druhy dračí křivky, Peanova křivka, Hilbertova křivka, dvě variace vločky Helge von Kocha, Kochův ostrov dva druhy Sierpinského trojúhelníku, dračí křivka) nebo si nechá obrazec náhodně vygenerovat. Systém nevyužívá jako základní tvar pouze standardní úsečku, ale nabízí možnost jiné volby (bod, trojúhelník, čtverec, šestiúhelník, hvězda, kružnice). Výsledné obrazce je možné barevně doladit (tlačítka *barva obrysů, barva pozadí, vykreslení obrysů, výplň tvaru*), měnit množství iterací a zadat startovací velikost úhlu. Pro uživatele, kteří chtějí experimentovat se syntaktickým stromem, je umožněno zasahovat do gramatického předpisu (tlačítka *Axiom, Pravidlo X, Pravidlo M, Pravidlo N*).

5.2 Způsob implementace

Readův lineární kód

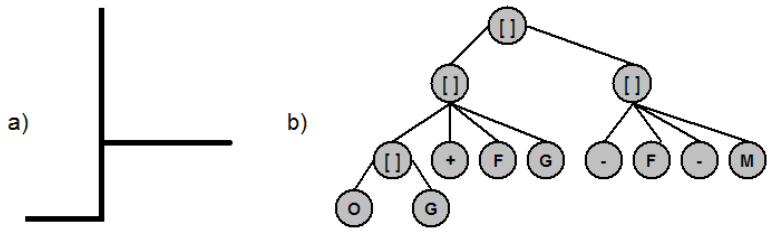
Jak se uvádí v Knize Evolučné algoritmy [5], v počátcích bylo GP implementováno pomocí jazyka LISP. Přestože má tento jazyk přímou podporu pro práci se stromovými strukturami, tato implementace je neefektivní. Proto se v praxi často používá tzv. Readův lineární kód. Ten kóduje strom do řetězce celých čísel. Každý vrchol je ohodnocen valencií (počet synovských uzlů). Výsledný kód získáme postupným spojováním valencií od listů ke kořenu (názorněji viz obr. 5.2). V této práci byl Readův kód využit pro generování náhodného stromu podle algoritmu, který uvádí Kvasnička v [5]. Implementovaná funkce vrátí Readův kód zadané délky. Ten je pak interpretován do podoby syntaktického stromu, jehož listy tvoří náhodně vybrané prvky gramatiky L-systému.



Obrázek 5.2: Ukázka tvorby Readova kódu.

Operace nad syntaktickými stromy (*Evoluční engine)*

Původní gramatický předpis obrazce a přepisovacího pravidla jsou reprezentovány syntaktickým stromem (použita třída *TreeNode* z knihovny jawa.swing.tree [8]). Příklad vygenerovaného axiomu spolu s jeho syntaktickým stromem je na obr. 5.3. Při vytváření náhodného obrazce je tento strom náhodně generován. Nad stromem jsou pak prováděny operace mutace a křížení. V případě mutace je náhodně vybrán startovací řetězec nebo jedno z pravidel, nad kterým je provedena operace mutace podrobněji popsaná v kapitole 2.2. Podobně při operaci křížení dojde k výměně odpovídajících částí vybraného stromu.



Obrázek 5.3: Ukázka a) vygenerovaného axiomu a b) jeho syntaktického stromu.

Rozšíření L-gramatiky

Základní gramatika L-Systému (uvedená v kapitole 3.2.2) byla rozšířena o další symboly. Díky tomu bylo dosaženo větší variability při generování obrázků, větších možností při editaci gramatiky. Například byla přidána možnost změny barvy či barevného alfa kanálu. Více viz tabulka. 5.1.

Iterace

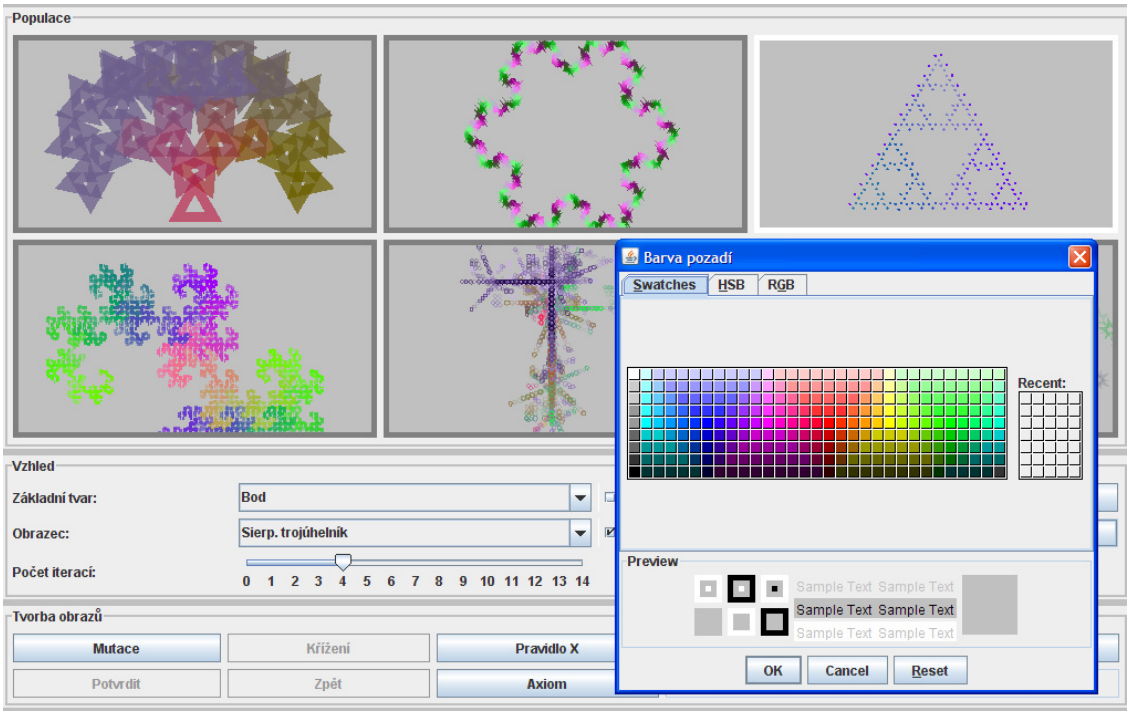
Iterace gramatického předpisu (přepis pomocí gramatický pravidel) probíhá v řetězcové reprezentaci. K tomu byla využita standardní třída *StringBuilder*. Ta na rozdíl od třídy *String* nevytváří při změně řetězce zcela nový objekt, ale zvětšuje objekt stávající. To je v tomto případě, kde pobíhá mnoho změn v řetězci, výhodné.

Kreslící engine (*L-systém engine)*

Pro vykreslení obrazce vyjádřeného gramatickým předpisem jsou použity funkce třídy *Graphics2D* knihovny java.awt [8]. Stav "želvy" je popsán její polohou, úhlem natočení,

Tabulka 5.1: Rozšíření želví gramatiky.

Rozšíření želví gramatiky	
Nepřepisovatelné symboly	
T	změna tloušťky vykreslovací čáry
A	změna velikosti úhlu natočení
s	změna velikosti základního tvaru
C	změna barvy
H	změna barevného alfa kanálu
|	otočení o 180°
Přepisovatelné symboly	
O, X, M, N	nakreslení tvaru směrem dopředu



Obrázek 5.4: Screenshot grafického uživatelského rozhraní.

typem základního tvaru, velikostí základního prvku, tloušťkou čáry, barvou a velikostí úhlu, o který se bude "želva" implicitně natáčet. Tyto informace jsou uchovávány v objektu třídy *Item*. Při přečtení symbolu v gramatice je provedena příslušná akce (natočení, změna barvy apod.). V případě symbolu pro nakreslení obrazce jsou nejprve vypočítány body nutné pro vykreslení obrazce a ty jsou uloženy do polygonální reprezentace (třída *Polygon*).

Při reprezentaci souřadnic celými čisly docházelo ke kumulaci chyby, která se ve výsledku projevila nenavazujícími častmi vykresleného obrazce. Proto je pozice želvy v souřadném

systému reprezentována reálnými čisly. Až po vlastním výpočtu je provedeno zaokrouhlení hodnot. Samotné vykreslení obrazce reprezentovaného jednotlivými body je provedeno pomocí elementárních funkcí knihovny *Graphics2D*.

Pro možnost interpretace závorkových symbolů (a tedy možnost vytváření větvících se struktur) je při kreslení použit zásobník. Při zpracování symbolu "[" je současný stav želvy (objekt třídy *Item*) uložen na zásobník. Naopak při dosažení symbolu "]" je stav ze zásobníku vyjmut.

Interaktivní přepis gramatiky

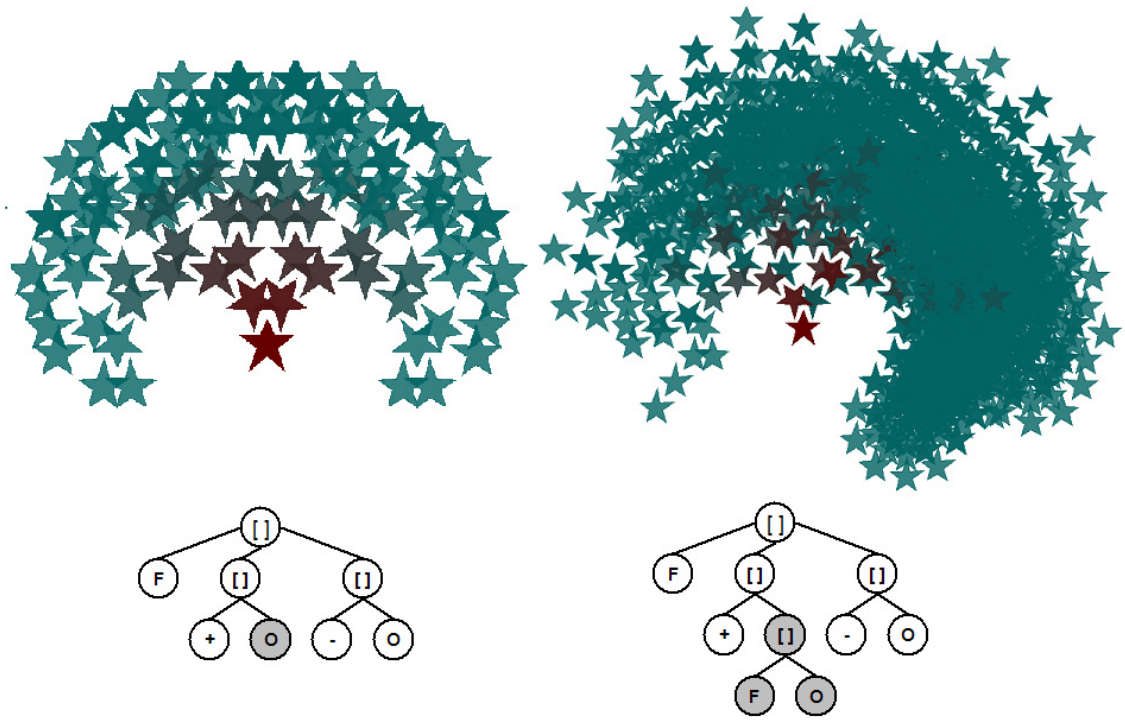
Vlastní gramatika byla navržena co nejjednodušším způsobem (každé akci odpovídá právě jeden symbol – jeden znak). Díky tomu je umožněno i uživateli, který nezná detailně principy L-systému či genetického programování, interaktivně měnit předpis nebo přepisovací pravidla.

Kapitola 6

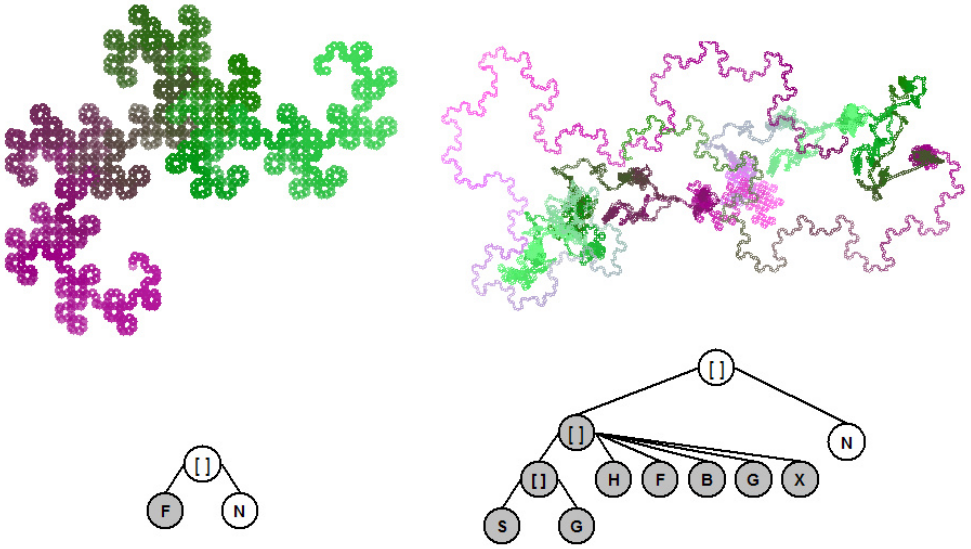
Výsledky

Tato kapitola obsahuje ukázky obrázků (obr. 6.1 - obr. 6.9) vzniklé pomocí implementovaného systému. V každém obrázku lze vidět vykreslené obrazce (nahoře) a syntaktické stromy mutovaného pravidla či axiomu (dole). Body mutace jsou zvýrazněny na původním stromu šedou barvou. Na zmutovaném stromu je šedá barva použita ke zvýraznění nově vygenerovaného podstromu.

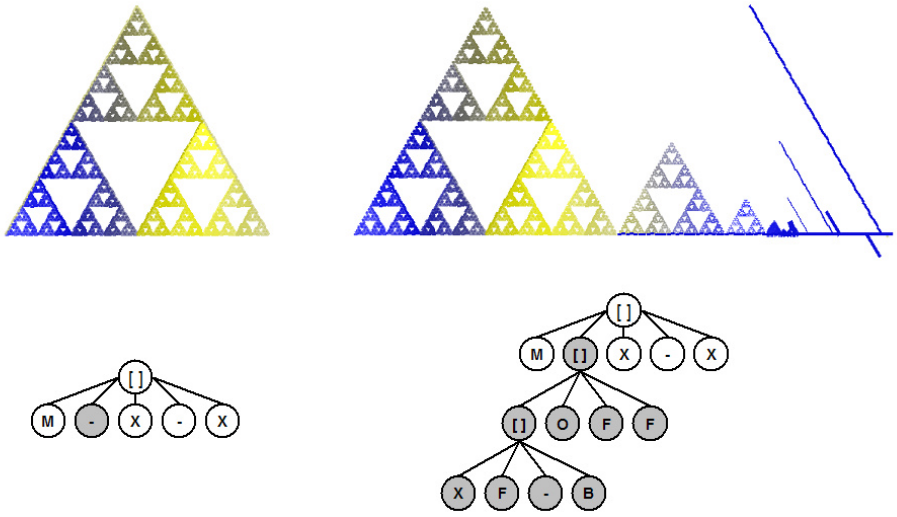
6.1 Aplikace operátoru mutace



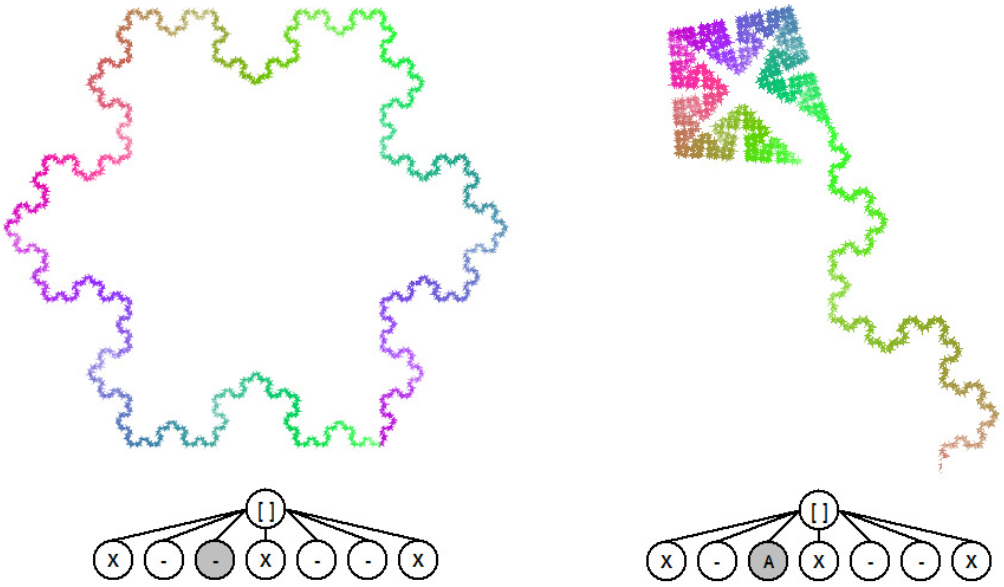
Obrázek 6.1: Mutace stromu.



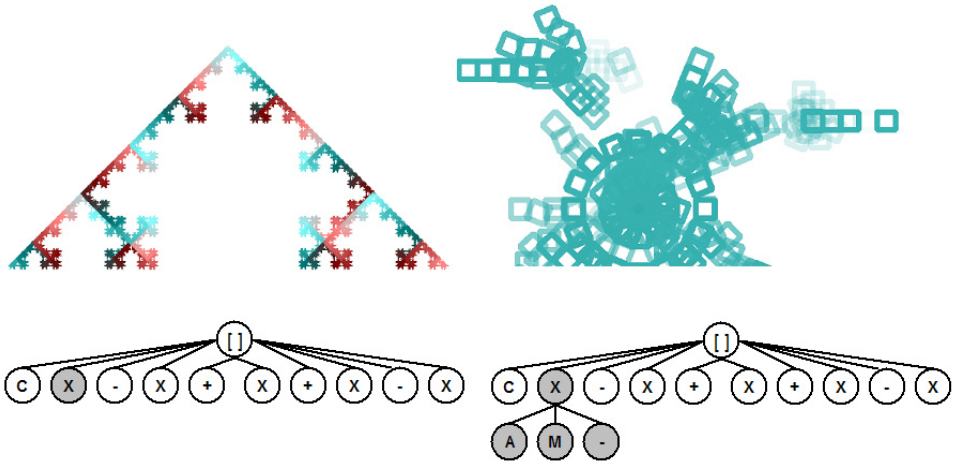
Obrázek 6.2: Mutace axiomu dračí křivky.



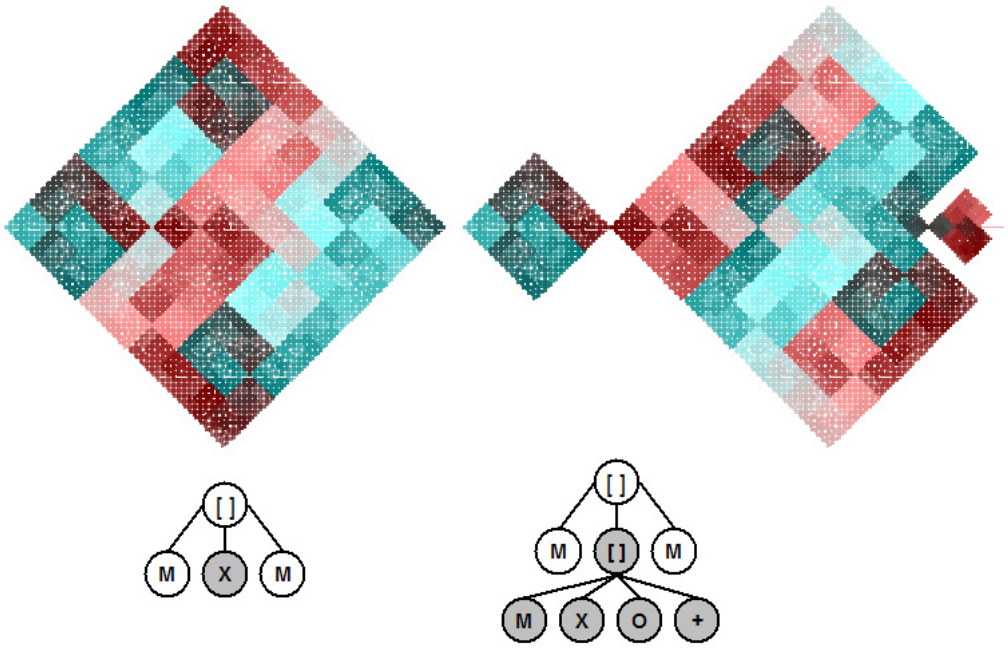
Obrázek 6.3: Mutace axiomu sierpinského trojúhelníku.



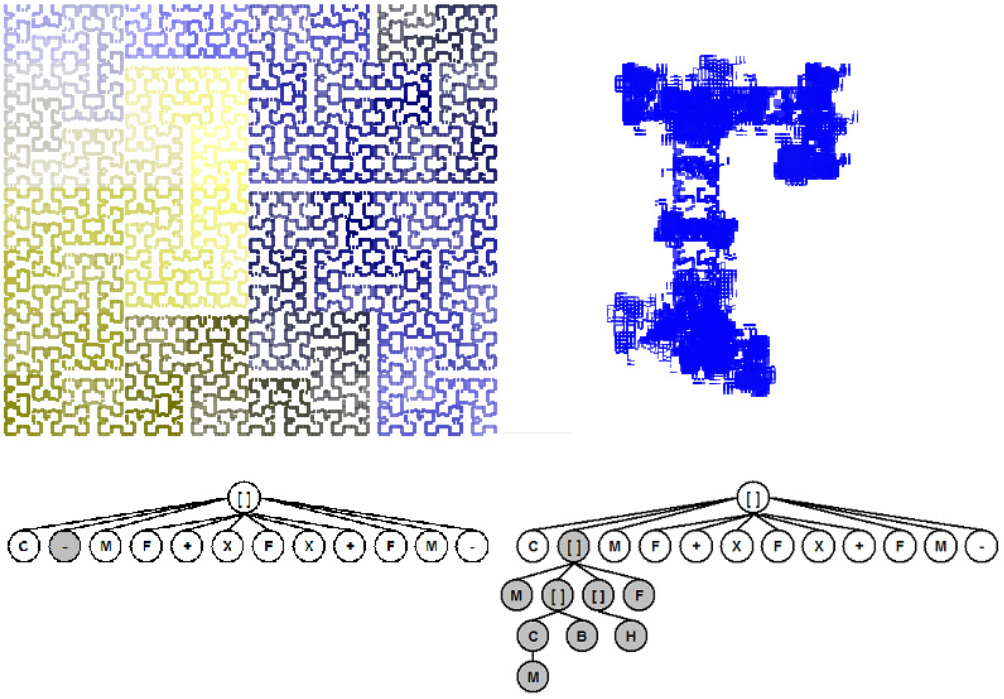
Obrázek 6.4: Mutace axiomu Kochovy vločky.



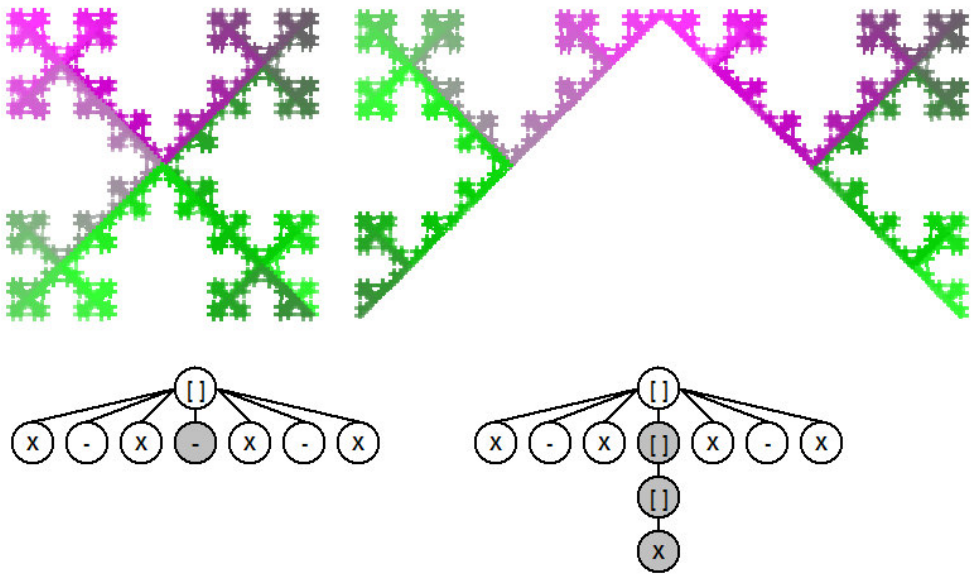
Obrázek 6.5: Mutace pravidla X Kochovy křivky.



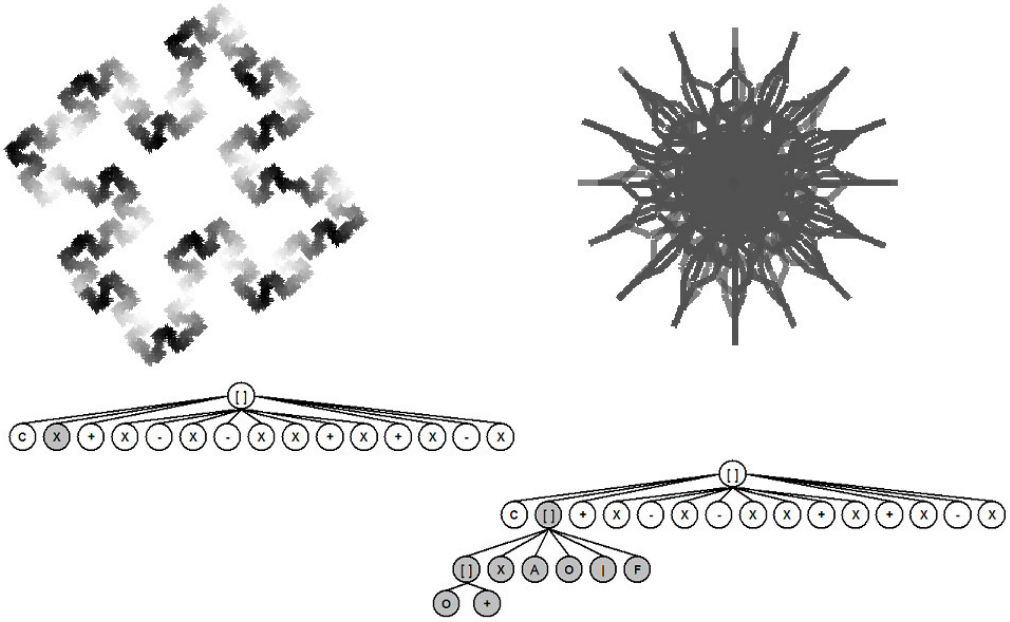
Obrázek 6.6: Mutace axiomu Peanovy křivky.



Obrázek 6.7: Mutace pravidla X Hilbertovy křivky.



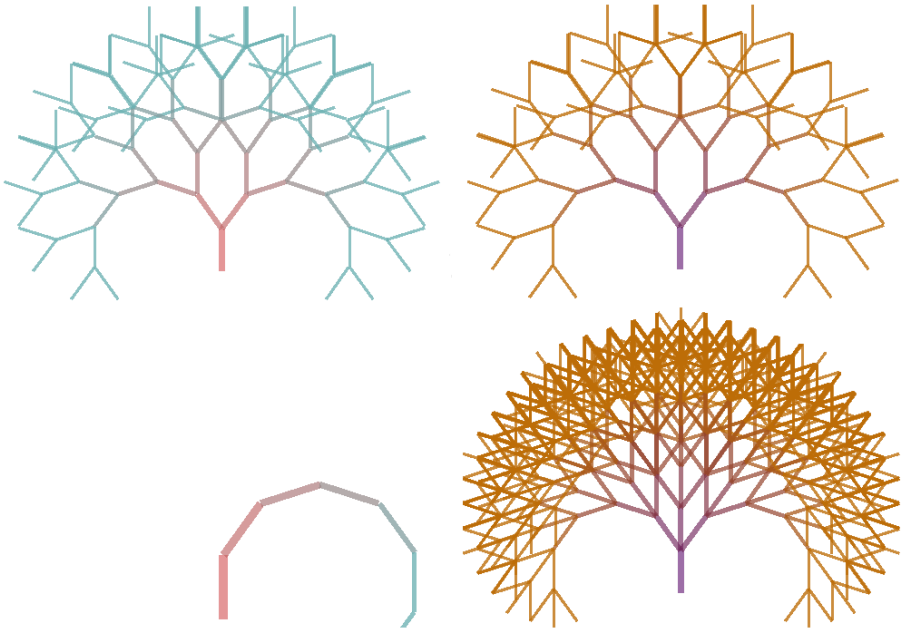
Obrázek 6.8: Mutace axiomu jedné z variací Kochovy křivky.



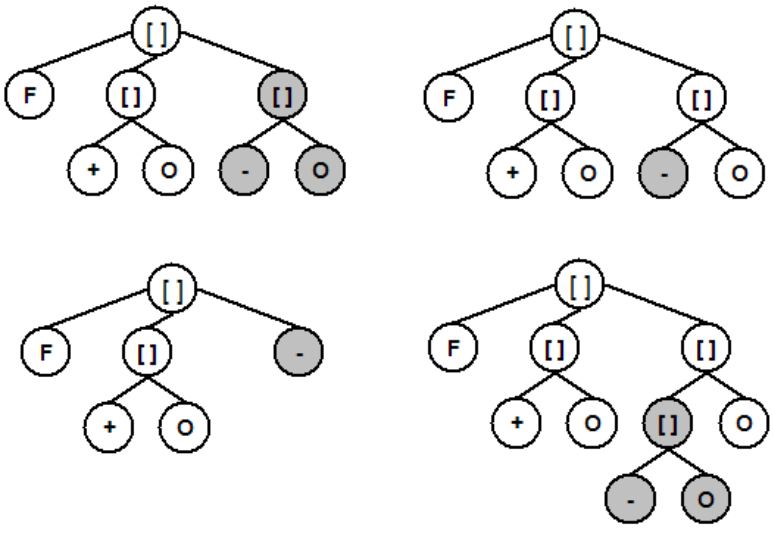
Obrázek 6.9: Mutace pravidla X Kochova ostrova.

6.2 Aplikace operátoru křížení

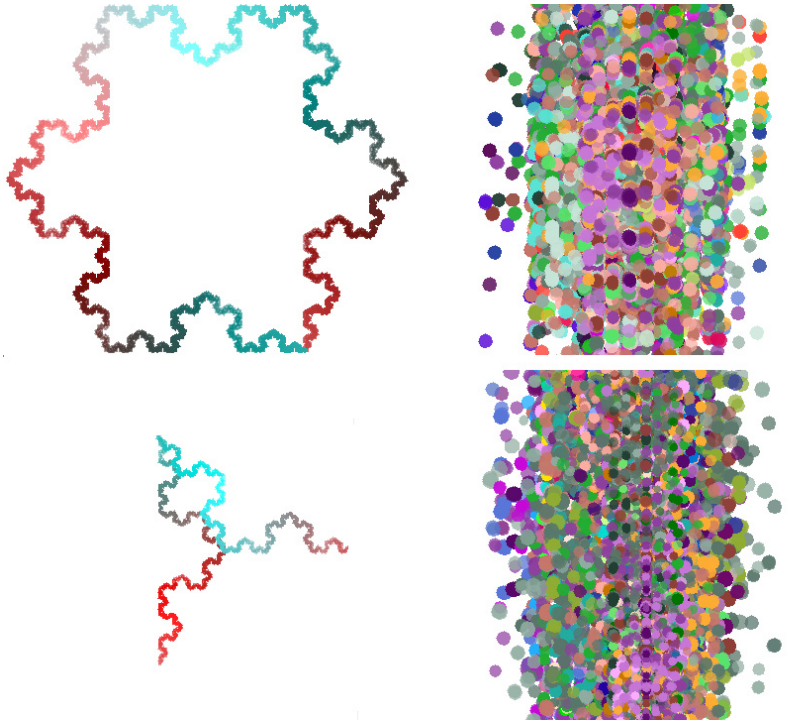
Následující obrázky 6.10 - 6.16 ukazují aplikaci operátoru křížení. Každou čtveřici obrázků tvoří vždy rodiče (nahoře) a potomci (dole).



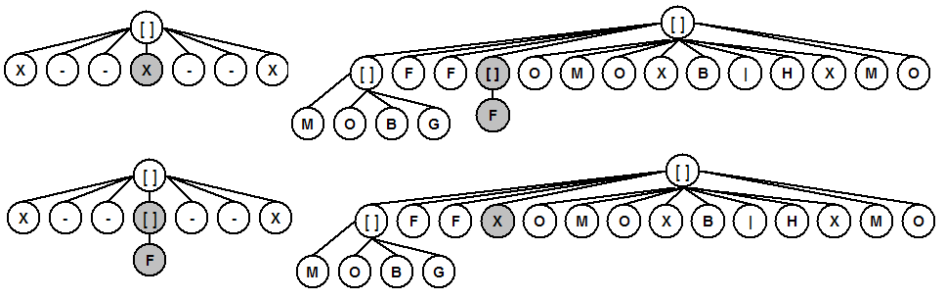
Obrázek 6.10: Křížení dvou stromů. Syntaktické stromy těchto obrazů můžete vidět na obr. 6.11.



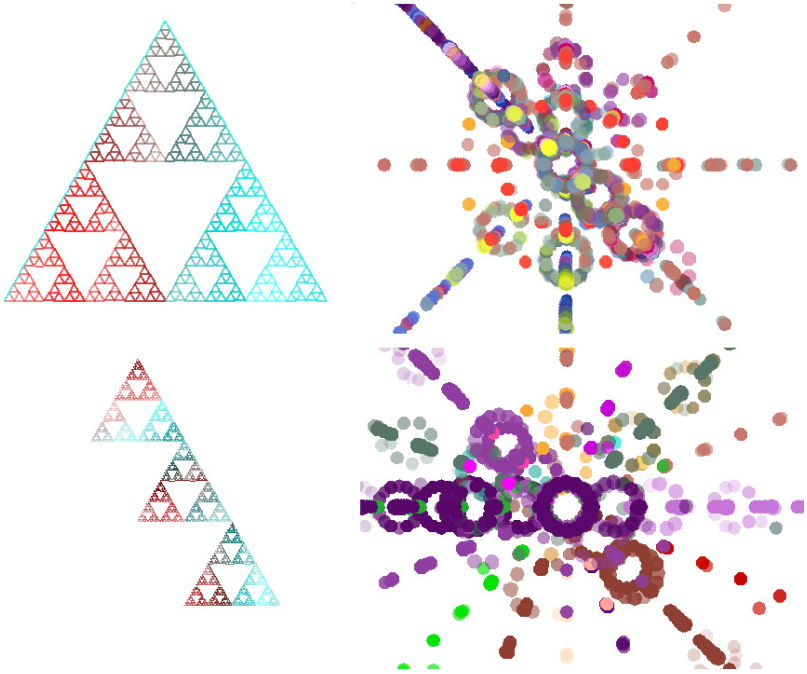
Obrázek 6.11: Křížení dvou stromů - syntaktické stromy. Obrázky vykreslené podle těchto syntaktických stromů můžete vidět na 6.10.



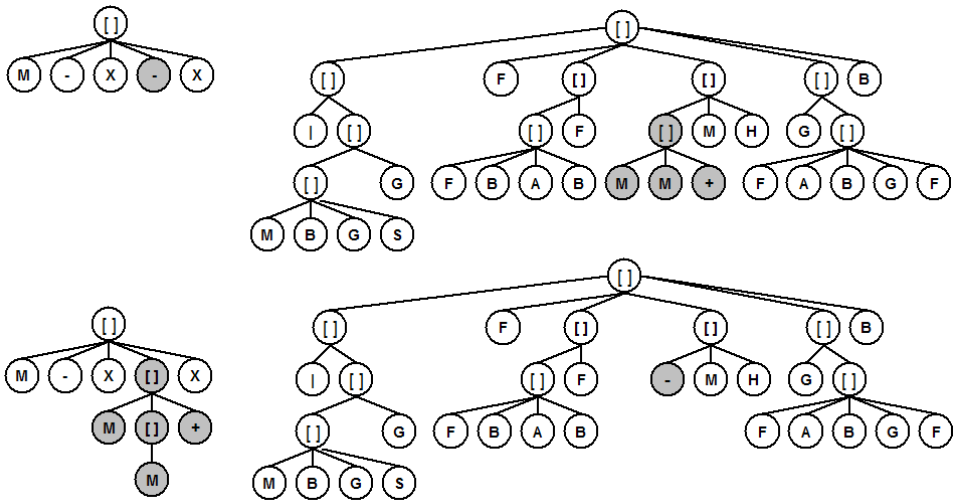
Obrázek 6.12: Křížení Kochovy vločky a náhodně vygenerovaného obrazce. Syntaktické stromy těchto obrazů můžete vidět na obr. 6.13.



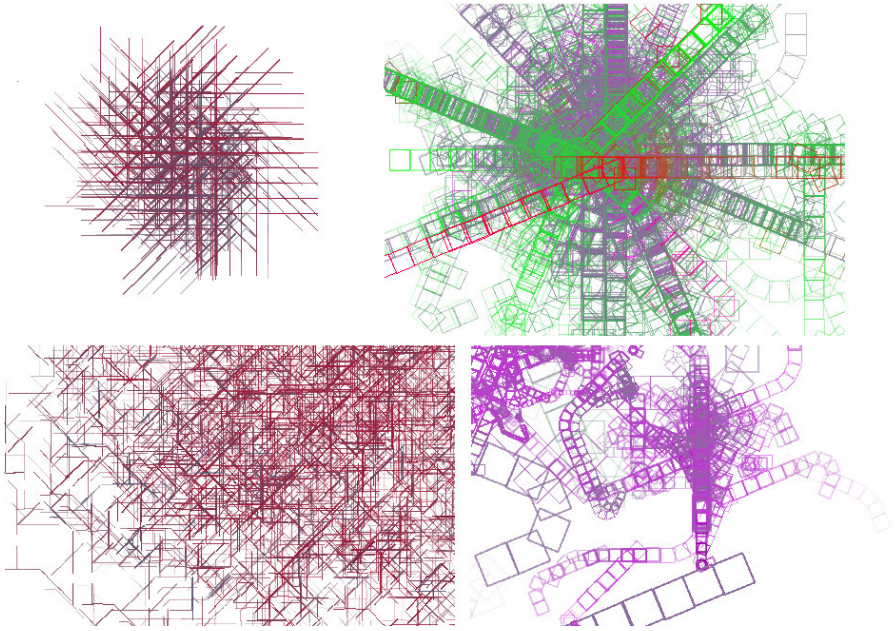
Obrázek 6.13: Křížení Kochovy vločky a náhodně vygenerovaného obrazce - syntaktické stromy. Obrázky vykreslené podle těchto syntaktických stromů můžete vidět na 6.12.



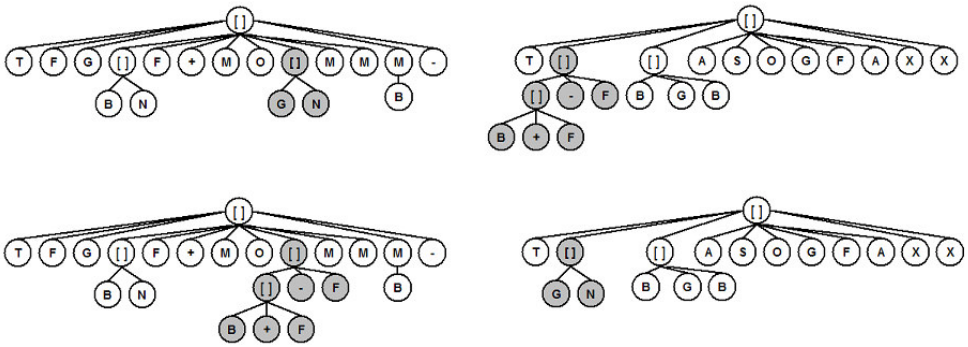
Obrázek 6.14: Křížení Sierpinského trojúhelníku a náhodně vygenerovaného obrazce. Syntaktické stromy těchto obrazů můžete vidět na obr. 6.12.



Obrázek 6.15: Křížení Sierpinského trojúhelníku a náhodně vygenerovaného obrazce - syntaktické stromy. Obrázky vykreslené podle těchto syntaktických stromů můžete vidět na 6.14.



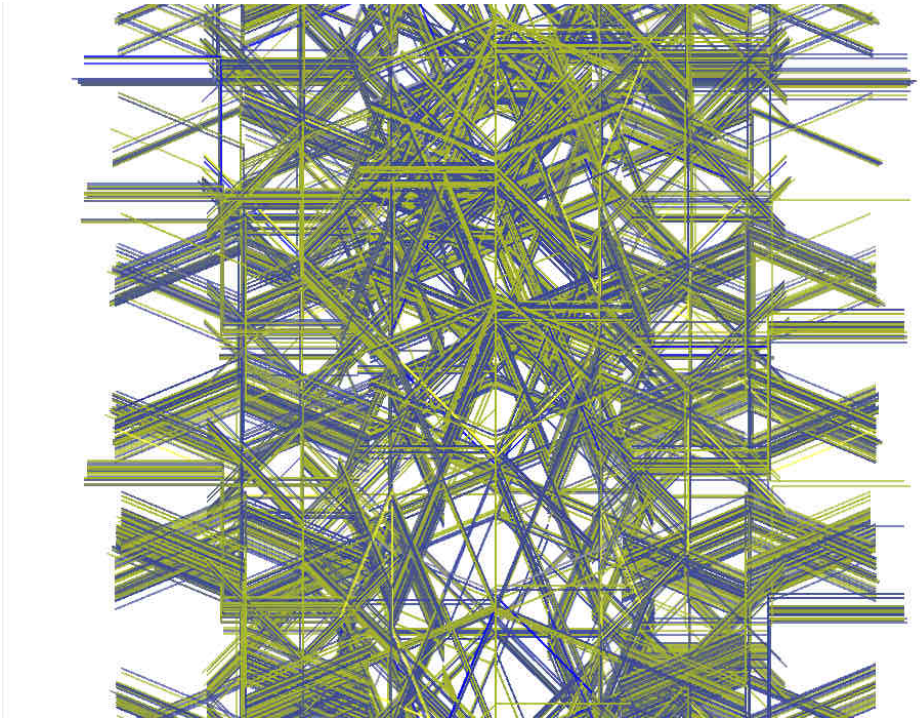
Obrázek 6.16: Křížení dvou náhodně vygenerovaných obrazců. Syntaktické stromy těchto obrazů můžete vidět na obr. 6.17.



Obrázek 6.17: Křížení dvou náhodně vygenerovaných obrazců - syntaktické stromy. Obrázky vykreslené podle těchto syntaktických stromů můžete vidět na 6.16.

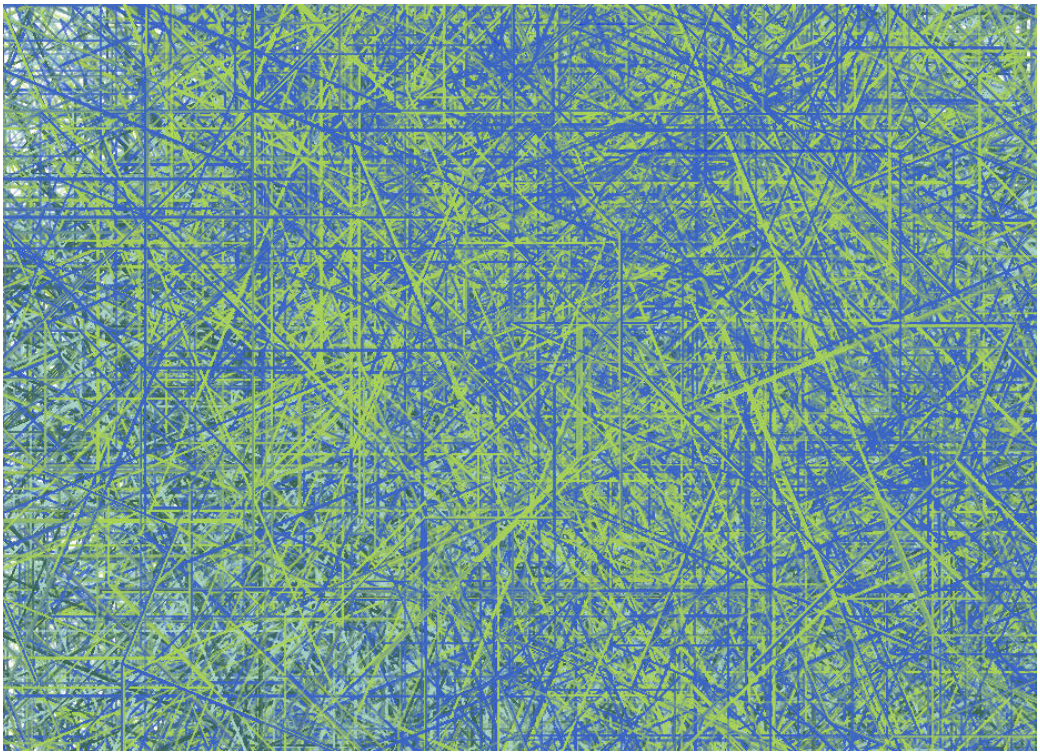
6.3 Navržené obrazce

Tato kapitola ukazuje několik navržených obrazů (obr. 6.18 - obr. 6.21) pomocí genetických operátorů.

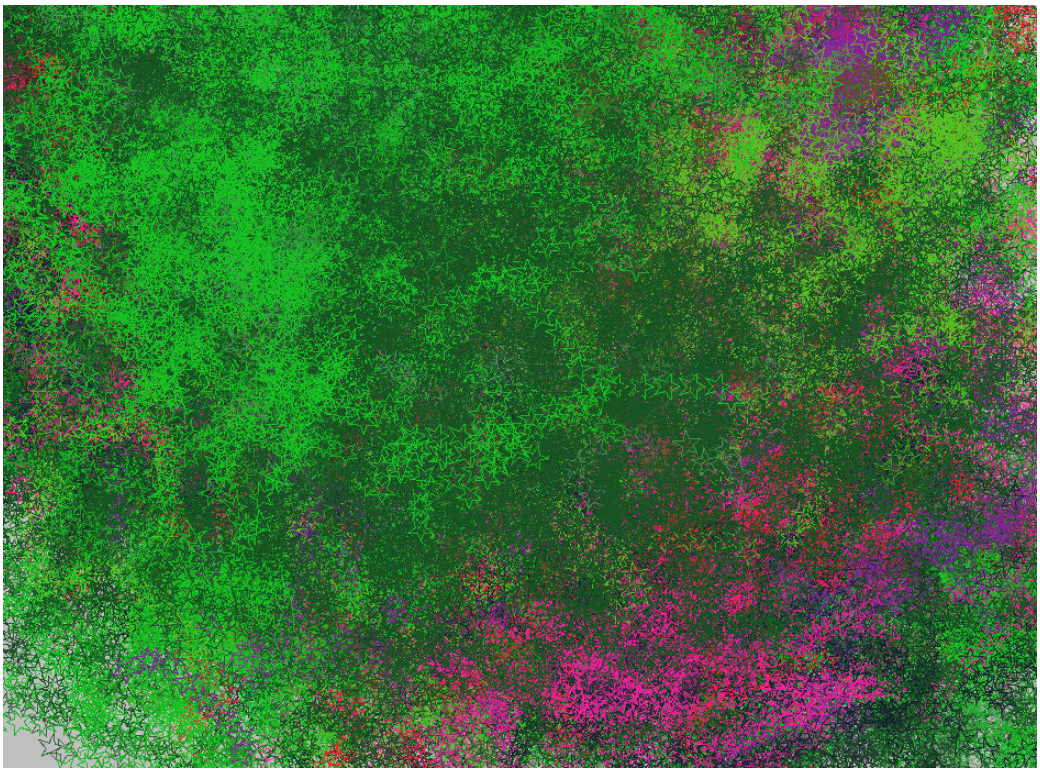


Obrázek 6.18: *"Futuristická konstrukce"*

Obrázek 6.19: *"Malba (technika tečkování)"*



Obrázek 6.20: *“Vlákna”*



Obrázek 6.21: *“Mořské řasy”*

Kapitola 7

Závěr

Vytvořený program je schopen ve spolupráci s uživatelem tvořit různorodé obrazce nejčastěji připomínající abstraktní malbu. Uživatel může pomocí operátorů genetického programování usměrňovat vývoj populace. V případě degenerace populace je umožněno uživateli zasahovat do gramatických předpisů a tím docílit požadovaného výsledku. Mutací či křížením předdefinovaných obrazů lze získat jejich zajímavé variace či vzhledově úplně odlišné jedince. Program může sloužit také uživateli dosud neseznámeného s prvky evolučního návrhu. A to především k pochopení principů genetického programování či fungování gramatiky L-systémů. K tomu může využít sadu předdefinovaných obrazců, na kterých lze názorně vyzkoušet operátory mutace nebo křížení.

Samotné vykreslování výsledného obrazce může být poměrně náročné na výkon počítače. Náročnost roste v závislosti na počtu vygenerovaných přepisovacích symbolů v řetězci a počtu iterací L-systému. Pro složité obrazce je vhodné pro urychlení vykreslování provádět operace při malém počtu iterací a až výsledný obrazec zobrazit v požadovaném počtu iterací.

Rozšířením tohoto projektu může být například umožnění generování i samotného základního tvaru systému. Ten by pak nemusel být tvořen pouhým geometrickým útvarem, ale mohl by být tvořen rovněž fraktálem (ať už by se jednalo rovněž o L-systém nebo systém IFS). Překážkou zde může být harwarová náročnost, která roste při použití více iterací často i exponenciálně. Jako jiný způsob rozšíření se nabízí implementace L-systémů ve 3D prostoru. Rozšířením o třetí rozměr by bylo možno generovat nejen dvourozměrné obrazy, ale také např. prostorové skulptury. Zajímavé by bylo i rozšíření implementace o použití různých složitějších barevných map.

Literatura

[1] Bentley, P.: *Evolutionary Design By Computers*. Morgan Kaufmann Publishers, 1999.

[2] Frame M., N. N., Mandelbrot B.: Fractal Geometry [online]. `http://classes.yale.edu/fractals/`, 2009 [cite 2009-05-04].

[3] Gajda, Z.: Evoluční návrh fraktálních obrazů [online]. `https://www.fit.vutbr.cz/~gajda/projects/gefra/report.pdf`, 2004 [cite 2009-05-04].

[4] Kolka, M.: Spojování želv v jazyce založeném na L-systémech [online]. `http://www.fit.vutbr.cz/~tisnovpa/fract/uvod.html`, 2001 [cite 2009-05-04].

[5] Kvasnička, V.; Pospíchal, J.; Tiňo, P.: *Evolučné algoritmy*. STU Bratislava, 2000.

[6] Matoušek, V.: Evoluční algoritmy a umělý život [online]. `http://www.kiv.zcu.cz/studies/predmety/uir/gen_alg2/E_alg.htm`, 2009 [cite 2009-05-04].

[7] Schwarz, J.; Sekanina, L.: *Aplikované evoluční algoritmy [Studijní materiály]*. FIT VUT v Brně, 2004.

[8] Sun Microsystems, I.: Java 2 SDK, Standard Edition, Documentation. [online]. `http://java.sun.com/j2se/1.4.2/docs/index.html`, 2003.

[9] Teda, J.: Genetické algoritmy a jejich aplikace v praxi [online]. `http://programujte.com/index.php?akce=clanek&cl=2005072601-geneticke-algoritmy-a-jejich-aplikace-v-praxi`, 2005 [cite 2009-05-04].

[10] Tišnovský, P.: Fraktály [online]. `http://www.fit.vutbr.cz/~tisnovpa/fract/uvod.html`, 2000 [cite 2009-05-04].

[11] Tišnovský, P.: Seriál Fraktály v počítačové grafice [online]. `http://www.root.cz/serialy/fraktaly-v-pocitacove-grafice/`, 2004-2007 [cite 2009-05-04].

[12] Vančura, J.: Fraktály [online]. `http://www.fractals.webz.cz/index.htm`, 2009 [cite 2009-05-04].

[13] Wegner, T.; Tyler, B.: *Fractal Creations*. The Waite Group Press, 1993.

Příloha A

Manuál

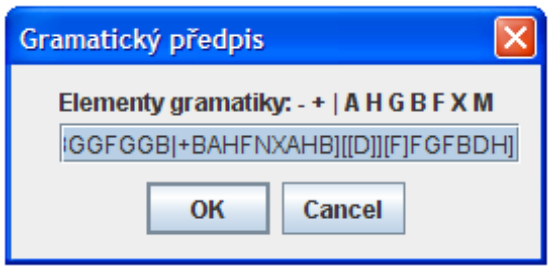
Applet byl vyvořen pro webové prohlížeče s pluginem Java2SE verze 1.6.0 (plugin pro webový prohlížeč lze stáhnout z `http://java.com/`). Applet tvořen šesti kreslícími plátny (obrázky). Ty zde představují jedince populace. Uživatel může vybrat libovolné kreslící plátno *levým tlačítkem myši*. Takto vybraný obrázek je označen bílým rámem. Na takto označeném plátně (obraze) může uživatel provést tyto akce:

- vybrat *„Základní tvar“* pomocí listboxu,
- vybrat *„Základní obrazec“* pomocí listboxu,
- změnit počet iterací L-systému pomocí posuvníku *„Počet iterací“*,
- povolit vykreslování výplně základních obrazců (checkbox *„Výplň tvaru“*),
- povolit vykreslování obrysů základních obrazců (checkbox *„Obrys tvaru“*),
- změnit velikost startovacího úhlu v rozsahu $0 - \pi$ (posuvník *„Velikost úhlu“*),
- vybrat startovací barvu obrazu tlačítkem *„Barva tvarů“*,
- vybrat barvu pozadí obrazu tlačítkem *„Barva pozadí“*,
- přiblížit či oddálit vybraný obraz *kolečkem myši*,
- provést mutaci tlačítkem *„Mutace“*. Mutace poté může být potvrzena (*„Potvrdit“*) nebo zrušena (*„Zpět“*),
- editovat axiom nebo gramatická pravidla (tlačítka *„Pravidlo X“*, *„Pravidlo M“*, *„Pravidlo N“*, *„Axiom“*). Po kliknutí na příslušné tlačítko se objeví dialogbox s textovým řádkem zobrazující příslušné pravidlo (viz obr. A.1). Nad dialogboxem jsou zobrazeny všechny použitelné elementy gramatiky. Po potvrzení ediditace (*„OK“*) je zkontrolována syntaktická správnost předpisu a výsledek je zobrazen ve stavovém řádku. Přehled použitelným elementů je zobrazen v tabulce A.1.

Pro křížení je nutné vybrat dva jedince (obrazy pomocí *pravého tlačítka myši*). Takto označené obrazy jsou zvýrazněny černým rámem. Po provedení křížení lze operaci potvrdit (*Potvrdit*) nebo zrušit (*Zpět*).

Tabulka A.1: Přehled použitelných gramatických symbolů

Význam symbolů gramatiky	
Nepřepisovatelné symboly	
F	posun želvy dopředu s nakreslením úsečky
G	posun želvy dopředu bez nakreslení úsečky
B	posun želvy dozadu s nakreslením úsečky
+	natočení želvy doleva
-	natočení želvy doprava
[	uložení stavu želvy do zásobníku
]	vyjmutí stavu želvy ze zásobníku
T	změna tloušťky vykreslovací čáry
A	změna velikosti úhlu natočení
S	změna velikosti základního tvaru
C	změna barvy
H	změna barevného alfa kanálu
|	otočení o 180°
D	změna základního tvaru
Přepisovatelné symboly	
O, X, M, N	nakreslení tvaru směrem dopředu



Obrázek A.1: Screenshot dialogboxu pro editaci gramatických pravidel.

Příloha B

Obsah CD

Struktura adresářů:

- technicka_zprava
  - pdf (technická zpráva ve formátu pdf)
  - tex (zdrojový kód systému LaTeXtechnické zprávy)
- applet
  - kod (zdrojový kód programu)
  - web (ukázka webové stránky s appletem)
- obrazky
  - ea (evoluční algoritmy)
  - preddefinovane (ukázky předdefinovaných obrazů)
  - mutace (ukázky mutace obrazů)
  - krizeni (ukázky krizeni obrazů)
  - ruzne (ukázky předdefinovaných a navržených obrazů)
  - gui (screenshoty grafického uživatelského rozhraní)