VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

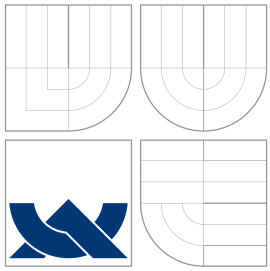
L SYSTÉMY A JEJICH APLIKACE

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

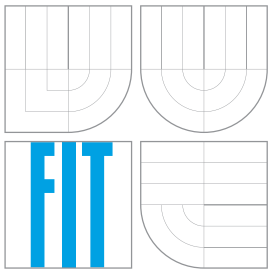
AUTOR PRÁCE
AUTHOR

Bc. JIŘÍ KOUTNÝ

BRNO 2008



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

L SYSTÉMY A JEJICH APLIKACE

L SYSTEMS AND THEIR APPLICATIONS

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — Bc. JIŘÍ KOUTNÝ
AUTHOR

VEDOUCÍ PRÁCE — Prof. RNDr. ALEXANDER MEDUNA, CSc.
SUPERVISOR

BRNO 2008

Abstrakt

Diplomová práce se zabývá deterministickými bezkontextovými L-systémy, zasazuje je do oblasti procedurálního modelování a staví je do souvislosti s fraktální geometrií. Zabývá se technikou přepisování a jejím využitím pro modelování rostlinám podobných struktur. Dále popisuje složitější typy L-systémů, zejména jejich kontextové a parametrické varianty. Ukazuje oblasti využití L-systémů zejména v oblasti počítačové grafiky a zaměřuje se na jejich využití pro procedurální modelování architektury. V závěru nastiňuje další možnosti využití procedurálního modelování pomocí L-systémů a představuje některá další rozšíření přepisovacích pravidel, která budou předmětem dalšího vývoje práce.

Klíčová slova

L-systém, modelování rostlin, modelování architektury, procedurální modelování, přepisování, větvení, želví grafika, fraktály

Abstract

This master thesis describes deterministic context-free L-systems and its context in procedural modelling, especially in fractal geometry, deals with rewriting technique and its usage for modelling structures similar to plants. Further it describes more complex types of L-systems, especially their context and parametric variations, and shows usage of L-systems in computer graphics and describes its usage for procedural modelling of architecture. At the end of this thesis there are described some other possibilities of usage procedural modelling with L-systems and introduced some extensions of rewriting rules, which will be subject of future research.

Keywords

L-system, modeling of plants, modeling of architecture, procedural modeling, rewriting, branching, turtle graphics, fractals

Citace

Jiří Koutný: L systémy a jejich aplikace, diplomová práce, Brno, FIT VUT v Brně, 2008

L systémy a jejich aplikace

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením pana Prof. RNDr. Alexandra Meduny, CSc.. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Jiří Koutný
12. května 2008

Poděkování

Děkuji prof. RNDr. Alexandru Medunovi, vedoucímu diplomové práce, za ochotu a čas, který mi při tvorbě této práce věnoval, za cenné rady a za motivaci do dalšího studia.

Obsah

Kapitola 1

Úvod

V posledních letech dochází k velkému rozvoji v oblasti generování syntetických scén. Při popisu přírodní scény se však setkáváme s poměrně velkou složitostí přírodních struktur, jejichž analytický popis by byl velmi náročný.

Ukázalo se, že část celého přírodního útvaru je podobná celku, tedy že přírodní útvary jsou *soběpodobné*, jinými slovy *invariantní ke změně měřítka*. Například větev stromu je podobná celému stromu, kámen je podobný hoře, ale také jedno patro panelového domu je podobné celé budově. Na základě této vlastnosti můžeme vytvořit řadu systémů sloužících k popisu fraktálních útvarů, mezi které patří:

- dynamické systémy s fraktální strukturou
- systémy iterovaných funkcí IFS
- stochastické fraktály
- Lindenmayerovy systémy (dále jen L-systémy)

Aristid Lindenmayer[1] v roce 1968 zavedl L-systémy jako teoretický rámec pro studium buněčných struktur. Tento rámec je však natolik obecný, že, ačkoliv generované struktury mají buněčný charakter, nemusejí být nutně rostlinné. Po přidání geometrické interpretace se modely rostlin založené na L-systémech staly natolik detailní, že umožnily použití počítačové grafiky pro realistické zobrazení rostlin. Později byly L-systémy dále rozvíjeny pro účely modelování celých rostlinných systémů. V posledních několika letech jsou pomocí L-systémů modelovány také jiné soběpodobné struktury či jejich části, zejména architektonické objekty.

1.1 Možnosti modelování

Modelování vychází z předpokladu, že organismus může být považován za množinu *diskrétních modulů*, jako jsou například kmeny, pupeny, listy či květy v případě rostlinných struktur nebo patra, schodiště, dveře či okna v případě struktur architektonických. Vývoj modulů lze předvídat a jejich vzájemné přeměny můžeme formálně zachytit v podobě *přepisovacích pravidel*. Vývoj nějaké výchozí struktury (zvané *Axiom*) sledujeme v nespojitém čase [3]: mezi každými dvěma okamžiky pozorování se struktura promění tak, že všechny její moduly jsou nahrazeny pravými stranami příslušných přepisovacích pravidel.

[1] ⋆1925–†1989, maďarský biolog

Přičemž průběh procesu nahrazení nemusí být tak přímočarý, jak plyne z předchozí věty. Při vyšším počtu opakování zřetelně vyvstává soběpodobnost výsledného tvaru a při nekonečně mnoha opakováních vznikne fraktál.

Každému modulu můžeme přiřadit předem zvolený geometrický význam, například generování objektu, rotaci či provedení akce. Lindenmayerův žák Przemyslaw Prusinkiewicz[2] zavedl metodu *interpretace symbolů želvou*, kdy si každý modul vykládáme jako příkaz pomyslné želvě, která putuje rovinou a provádí interpretaci podle geometrického významu modulu. Při konvenční notaci [19] symbol F znamená kreslení úsečky jednotkové délky, symboly + a − rotaci doleva a doprava o daný úhel. Další možnosti se naskytnou, pokud želvu obdaříme schopností pohybu v prostoru, kde je možné geometrický význam modulů definovat i pomocí polygonů, křivek, kuželoseček nebo opět pomocí L-systémů.

Zavedeme-li možnost definovat úhly a vzdálenosti reálnými čísly, jejichž hodnoty mohou být určeny vyhodnocením matematickými výrazů či funkcí, můžeme modelovat například závislost rychlosti růstu rostliny na jejím stáří. Nahodilé vlivy můžeme modelovat pomocí generátoru náhodných čísel, která ovlivňují parametry či podobu přepisovacích pravidel. Zajímavé důsledky má zavedení závislosti přepisovacích pravidel na poloze modulu v rámci struktury, tedy na kontextu. Ještě zajímavější je zavedení L-systémů s kontextovou podmínkou, kdy pravidlem předepsaný kontext modulu nemusí být těsný.

Dalším stupněm je zavedení obousměrných komunikačních modulů, směrem dovnitř je například možné L-systém informovat o množství dopadajícího světla, směrem ven může být okolí informováno například o množství CO_2 produkovaného L-systémem, který modeluje rostlinu.

1.2 Cíle práce

Cílem této práce je nastudovat vlastnosti a možnosti využití různých typů L-systémů, navrhnout systém pro jejich aplikaci a v neposlední řadě také naznačit směry dalšího možného vývoje projektu.

V kapitole Procedurální modelování jsou L-systémy zasazeny do kontextu fraktální geometrie, včetně vztahu k soběpodobnosti.

Kapitola Modelování s využitím L-systémů popisuje techniku přepisování řetězců a na jejím základě definuje deterministické bezkontextové (d0L) systémy, jejich vlastnosti a metody jejich interpretace. Dále popisuje složitější typy L-systémů, zejména jejich stochastické, kontextové a parametrické varianty.

Využití L-systémů se zaměřením na jejich aplikace pro modelování architektury je uvedeno v kapitole Využití L-systémů. Kapitola se zabývá zejména rozšířením pravidel, které dělají modelování v prostoru snazším, a využitím kontextové podmínky, která pomáhá řešit jisté problémy interpretačního charakteru.

Součástí této práce jsou dvě aplikace. První demonstruje základy procedurálního modelování v jednoduchých d0L-systémech. Druhá pak umožňuje definovat složitější typy L-systémů, počítat jejich generace a ty následně interpretovat za pomoci jazyka VRML. Návrh těchto aplikací je popsán v páté a šesté kapitole.

[2] ⋆1952, polský vědec

V kapitole Možnosti dalšího vývoje projektu jsou popsány možné směry, jimiž by se vývoj této práce mohl dále ubírat. Zmíněné směry možného dalšího vývoje jsou rozděleny do dvou oblastí. První je oblast rázu ryze teoretického, druhou pak oblast rázu ryze praktického.

Závěr shrnuje celou práci, obsahuje stručné zhodnocení projektu, krátce rekapituluje současný stav práce zejména z pohledu implementace souvisejících aplikací a nakonec se v několika větách zmiňuje o smyslu a cílech dalšího výzkumu v této oblasti.

Kapitola 2

Procedurální modelování

Trojrozměrný počítačový model objektu lze získat v zásadě třemi způsoby. První z nich je získání hrubého modelu získaného přímo z geometrie reálného objektu. Ten můžeme pořídit prostorovým scannerem, případně se můžeme pokusit o rekonstrukci z několika snímků z fotoaparátu pomocí stereografie. Těmto postupům se říká *modelování založené na obrazech* (*image based modeling*). Trojrozměrné snímání tvaru objektů má svá omezení, která jsou dána například velikostí snímaného objektu či jeho členitostí.

Druhým způsobem, jak pořídit model objektu, je jeho *interaktivní modelování*. Tento postup je velice pracný, vyžaduje talent a zkušenosti, jeho výhoda je ale v tom, že takto pořízená data jsou sémanticky naprosto přesná. To je v případě scannování v současné době problematické.

Získávání trojrozměrného modelu na základě algoritmu je třetím způsobem. Tato třída metod se nazývá *procedurální modelování* (*procedural modeling*) a je možné rozdělit ji na dvě základní podtřídy. První z nich jsou metody používané v CAD (*Computer Aided Design*) a CAGD (*Computer Aided Geometric Design*), například šablonování či generování ploch a křivek.

Druhou podtřídou je automatické generování objektů a obvykle se využívá, pokud jsou generované objekty inspirované přírodou. *Procedurální techniky jsou části kódu či algoritmy, které určují jisté charakteristiky počítačového modelu či efektu.* [27] Tuto podčást procedurálního modelování můžeme opět dělit. První skupinou jsou algoritmy, které vycházejí z gramatik – patří sem především *L-systémy* používané zejména pro generování rostlin. Druhou skupinou je *fraktální geometrie*, ta poskytuje algoritmy pro generování hor, krajin, kamenů, korálů atd. Třetí skupinou jsou *částicové systémy* a používají se především pro generování explozí, hejn ptáků, simulaci ohně atp.

2.1 Fraktální geometrie

Fraktální geometrie je intenzivně rozvíjena zhruba od šedesátých let dvacátého století a za jejího objevitele je považován Benoit B. Mandelbrot [13].

Uměle vytvořené předměty se obvykle vyznačují geometrickou přesností, avšak objekty v přírodě mají charakteristiku zcela odlišnou. Proto není efektivní popisovat některé přírodní útvary pomocí klasických geometrií. Například je obtížné popsat mrak jako množinu koulí či parametrickou plochu nebo modelovat krajinu jako množinu trojúhelníků. Fraktální geometrie je silný a relativně jednoduchý aparát, který takové modelování umožňuje a proto bývá někdy nazývána jako *morfologie amorfního* [27]. Euklidovská geometrie popisuje ob-

jekty a jejich invariance vzhledem k transformacím většinou rovnicí. Ve fraktální geometrii používáme k popisu objektů algoritmy, zejména rekurzivní, a zabýváme se invariancí vůči změně měřítka.

2.1.1 Soběpodobnost

Ústřední pojem fraktální geometrie je *soběpodobnost* (*self-similarity*), což je jen jiný název pro invarianci vůči změně měřítka. Soběpodobnou strukturu je možné rozložit na struktury, z nichž každá je zmenšenou kopií originálu. Soběpodobnost je však pouze podmínkou nutnou, nikoliv postačující k fraktálnímu charakteru objektu. Rozlišujeme dva druhy soběpodobnosti, *soběpodobnost přesnou* a *soběpodobnost statistickou*. Jejich definice podle [27] je:

Definice 1 *Přesná soběpodobnost.* Množina A je *přesně soběpodobná*, pokud je sjednocením konečného počtu transformovaných kopií sebe samé. Platí tedy

$$A = \bigcup_{i=1}^{n} \varphi_i(A)$$

kde φ_i jsou transformace posunutí nebo transformace rotace, z nichž každá je zároveň změnou měřítka $S_i \in < 0, 1 >$ nebo jsou všechny tzv. průměrně kontraktivní. Pokud by součet koeficientů S_i přesáhl hodnotu jedné, množina by prostorově divergovala do nekonečna. Podmínka průměrné kontraktivity množiny transformací $\{\varphi_i, i = 1, 2, \ldots, n\}$ má tvar

$$0 < \sum_{i=1}^{n} S_i < 1$$

Přesně soběpodobná je například Kochova vločka, viz Obrázek 3.1.

Definice 2 *Lineární transformace.* Transformace $\varphi : U \to U$ je lineární, pokud $\varphi(r_1 A + r_2 B) = r_1\varphi(A) + r_2\varphi(B)$ pro všechna $A, B \in U$ a $r_1, r_2 \in R$, kde U je vektorový prostor a R je množina reálných čísel.

Definice 3 *Statistická soběpodobnost.* Množina A je *statisticky soběpodobná*, pokud je sjednocením konečného počtu zmenšených kopií sebe samé a každá z kopií $\varphi_i(A)$ má stejné statistické charakteristiky jako množina A. Říkáme, že $\varphi_i(A)$ a A jsou statisticky nerozlišitelné. Transformace φ_i musí být zároveň změnou měřítka s koeficientem $s_i \in (0, 1)$. Jsou-li aplikované transformace lineární, resp. nelineární, je soběpodobná množina A lineární, resp. nelineární. Za zachování podmínky statistické soběpodobnosti v praxi obyčejně považujeme shodu směrodatné odchylky a průměru a ne všech statistických momentů.

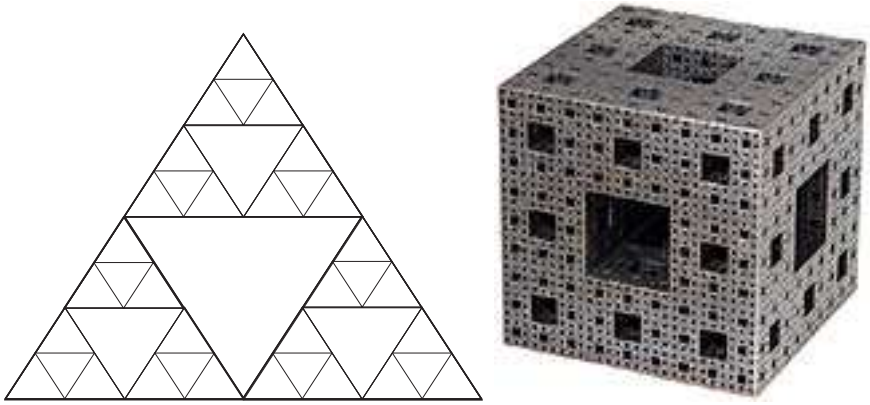
Příkladem statistické soběpodobnosti je kámen a hora. Pokud budeme porovnávat vhodně vybranou fotografii kamene a hory, bude pro nás obtížné rozhodnout, co je horou a co je kamenem. Jiným příkladem je nahrávka šumu z rádia. Když tuto nahrávku budeme přehrávat libovolnou rychlostí, bude pravděpodobně znít pořád stejně. Přičemž změna rychlosti přehrávání odpovídá změně měřítka.

Obrázek 2.1: Prvních pět iterací Cantorova diskontinua.

2.1.2 Lineární deterministické fraktály

Lineární deterministické fraktály můžeme generovat jednoduchými rekurzivními postupy s aplikací transformací otočení, posunutí a změny měřítka. Jedním z nejjednodušších fraktálů je *Cantorovo diskontinuum*. Tato množina vznikne tak, že vyjmeme z úsečky prostřední třetinu a tento proces aplikujeme rekurzivně na zbývající dvě třetiny (Obrázek 2.1).

Dalším deterministickým lineárním fraktálem je Kochova vločka (Obrázek 3.1) nebo Sierpinského trojúhelník (Sierpinského krajkoví, Sierpinského košík, Obrázek 2.2). Ten vznikne z trojúhelníku rekurzivním vyjímáním dalšího trojúhelníku, který vznikne spojením středů stran prvního trojúhelníku. W. Sierpinsky však původně hledal něco zcela jiného. Chtěl najít množinu, která obsahuje všechna možná topologická jednorozměrná větvení. Tak objevil množinu později pojmenovanou *Sierpinského koberec*. Získáme ho snadno ze čtverce rekurzivním vyjímáním dalšího čtverce, který je určen jako průsečík kolmic vztyčených nad prostřední třetinou všech stran prvního čtverce. Jejím obrazem je jedna strana *Mengerovy houby*, která vznikne rekurzivním vyjímáním části krychle (Obrázek 2.2).



Obrázek 2.2: Sierpinskeho trojúhelník a Mengerova houba.

Deterministické fraktály, jako je Kochova vločka nebo Mengerova houba, demonstrují základní vlastnosti fraktálů. Deterministické fraktály umožňují jednoduchým způsobem vytvořit dostatečně složité povrchy a objemy a často se používají pro tvorbu testovacích modelů, na kterých jsou testovány rychlosti implementace metody sledování paprsku (ray-tracing). Generování Mandelborotovy množiny, což je další z deterministických fraktálů, se používá jako testovací úloha rychlosti CPU. Peanova křivka byla zase použita pro dithering – ukázalo se, že pokud lineární paměť grafického procesoru nebudeme do obrazu mapovat po řádcích, ale ve tvaru fraktální křivky, sníží se chybné zásahy vyrovnávací paměti grafického procesoru. Tato technika se nazývá *texture swizzling* a používá se například v herní konzoli Xbox. [27]

Kapitola 3

Modelování s využitím L-systémů

L-systémy byly vytvořeny jako matematická teorie popisující vývoj jednoduchých buněčných struktur, přičemž důraz byl kladen především na popis topologie a geometrické aspekty nebyly propracovány. Z tohoto důvodu nebylo možné modelovat složitější organismy, například rostliny. Později Przemysl Prusinkiewicz s ohledem na univerzální použití při modelování rostlin zavedl metodu interpretace L-systémů želvou (*turtle graphics*). Idea teorie L-systémů a jejich interpretace želví grafikou je popsána v této kapitole.

3.1 Základní definice

Vyjdeme z následujících nezbytně nutných definic.

Definice 4 *Abeceda.* *Abeceda* je konečná, neprázdná množina elementů, které nazýváme *symboly.*

Definice 5 *Řetězec nad abecedou.* Nechť Σ je abeceda. Pak ε je řetězec nad abecedou Σ. Pokud x je řetězec nad Σ a $a \in \Sigma$, potom xa je řetězec nad abecedou Σ. ε značí *prázdný* řetězec, tj. řetězec neobsahující žádný symbol.

Definice 6 *Délka řetězce.* Nechť x je řetězec nad abecedou Σ. Pak délka řetězce x, $|x|$, je definována takto: pokud $x = \varepsilon$, pak $|x| = 0$; pokud $x = a_1 a_2 \dots a_n$, kde $a_i \in \Sigma$ pro všechna $i = 1, \dots, n$, pak $|x| = n$.

Definice 7 *Jazyk nad abecedou.* Nechť Σ^* značí množinu všech řetězců nad Σ. Každá podmnožina $L \subseteq \Sigma^*$ je *jazyk* nad Σ. Jazyk je *konečný*, pokud obsahuje *konečný* počet řetězců, jinak je nekonečný.

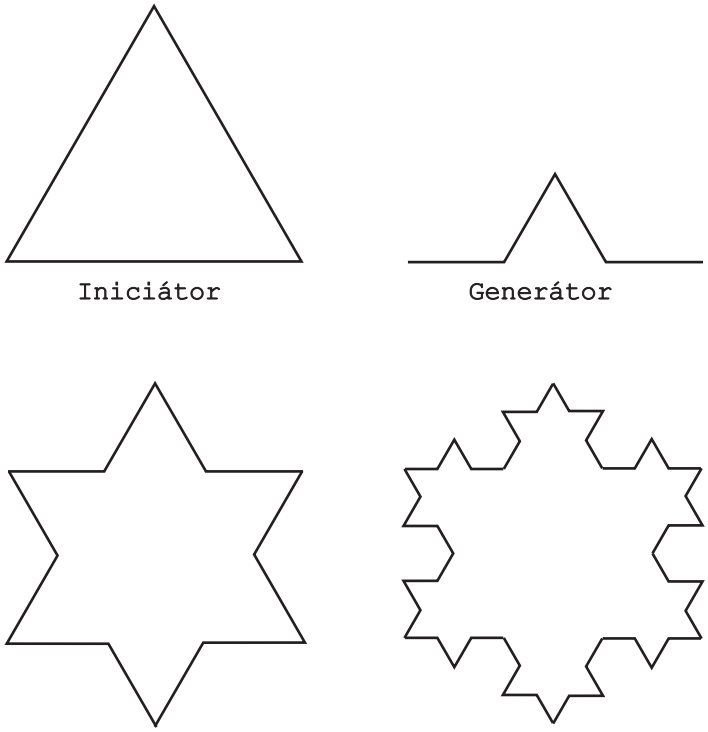
Definice 8 *Bezkontextová gramatika.* Bezkontextová gramatika je čtveřice $G = (N, T, P, S)$, kde N je abeceda *neterminálů*; T je abeceda *terminálů*, přičemž $N \cap T = \emptyset$; P je konečná množina pravidel tvaru $A \rightarrow x$, kde $A \in N$, $x \in (N \cup T)^\star$; $S \in N$ je *počáteční neterminál.*

3.2 Přepisovací systémy

Ústředním pojmem L-systémů je *přepisování*, což je technika, která postupným nahrazováním částí jednoduchého objektu a pomocí množiny přepisovacích pravidel definuje objekty složitější. Klasickým příkladem geometrického objektu definovaného technikou přepi-

sování je *sněhová vločka* (Obrázek 3.1), kterou navrhl von Koch [8] v roce 1905. Mandelbrot popsal její konstrukci takto[1] [13]:

> Vločka se skládá ze dvou útvarů - *iniciátoru* a *generátoru*. Generátor je orientovaná lomená čára, která je složena z N stejných úseků délky r. Každá část konstrukce začíná lomenou čárou – generátorem – a sestává z nahrazení každé přímé části iniciátoru kopií generátoru a jejím přesunutím tak, aby měla stejný koncový bod jako část iniciátoru, kterou nahrazuje.



Obrázek 3.1: Konstrukce von Kochovy sněhové vločky.

První formální definici systému zavedl počátkem dvacátého století Thue [21], ale velký důraz přepisovacím systémům věnoval až v padesátých letech dvacátého století Chomsky ve své práci na formálních gramatikách [2] kde, mimo jiné, rozdělil jazyky do tříd (tzv. *Chomského hierarchie*). O několik let později Backus a Naur využili přepisování a zavedli značení, aby formálně definovali programovací jazyk ALGOL-60. Brzy byla objevena ekvivalence mezi Backusovou-Naurovou formou a třídou bezkontextových gramatik v Chomského hierarchii. Vznikl nový obor počítačové vědy, jehož středem zájmu byly množiny řetězců – označované jako formální jazyky – a metody jejich generování, rozpoznávání a transformování.

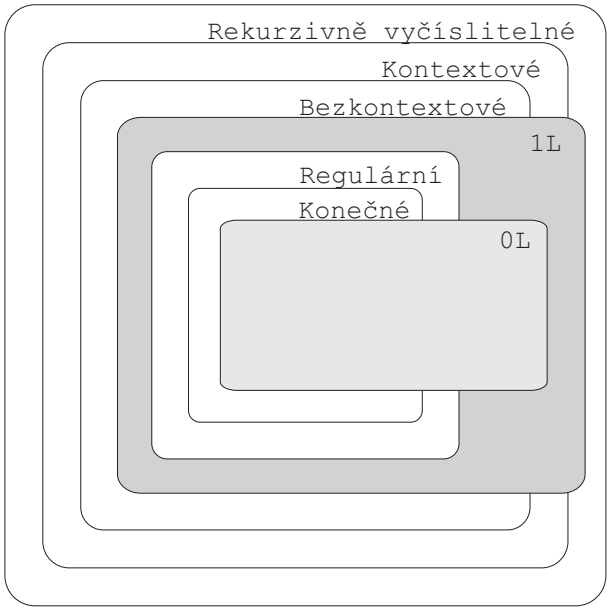
V roce 1968 biolog Aristid Lindenmayer zavedl nový mechanismus založený na paralelním přepisování řetězců, který se dnes označuje jako L-systémy. Podstatný rozdíl mezi Chomského gramatikami a L-systémy je ve způsobu použití pravidel:

Věta 1 *Chomského gramatiky a L-systémy.* V Chomského gramatikách jsou pravidla použita *sekvenčně*, kdežto v L-systémech jsou použita *paralelně* pro přepsání všech znaků v daném řetězci.

[1] volně přeloženo

Tento zásadní rozdíl odráží biologickou motivaci L-systémů, kde například buněčné dělení v mnohobuněčném organismu může také probíhat současně, tedy paralelně. Paralelní aplikace pravidel má podstatný dopad na vlastnosti přepisovacích systémů (Obrázek 3.2):

Věta 2 *Vztah bezkontextových gramatik a bezkontextových L-systémů.* Existují jazyky, které mohou být generovány *bezkontextovými L-systémy* (nazývají se 0L-jazyky), ale nemohou být generovány *bezkontextovými jazyky v Chomského hierarchii.*

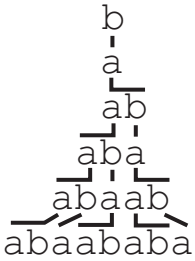


Obrázek 3.2: Vztah mezi Chomského třídami jazyků a třídami jazyků generovanými L-systémy. Symboly 0L a 1L označují třídy jazyků generované *bezkontextovými* resp. *kontextovými* L-systémy.

3.3 d0L-systémy

Nejjednodušší třídou L-systémů jsou *deterministické bezkontextové L-systémy*, které zkráceně označujeme jako d0L-systémy.

Mějme řetězce složené ze znaků a a b, které se v daných řetězcích mohou libovolně opakovat. Každému znaku je přiřazeno přepisovací pravidlo. Pravidlo $a \rightarrow ab$ znamená nahrazení každého výskytu znaku a v řetězci dvojicí znaků ab. Podobně pravidlo $b \rightarrow a$ znamená nahrazení každého výskytu znaku b v řetězci znakem a. Proces přepisování začíná na řetězci nazývaném *Axiom*. Předpokládejme, že axiom se skládá pouze z jednoho výskytu znaku b. V prvním *derivačním kroku* (v prvním kroku přepisování) bude axiom b nahrazen znakem a podle pravidla $b \rightarrow a$. V druhém kroku bude znak a nahrazen dvojicí znaků ab podle pravidla $a \rightarrow ab$. Řetězec ab se skládá ze dvou znaků, které budou oba současně nahrazeny v dalším derivačním kroku. Tedy a bude nahrazeno dvojicí znaků ab, b bude nahrazeno znakem a a vznikne tak řetězec aba. V dalším derivačním kroku vznikne z řetězce aba řetězec $abaab$, ten bude přepsán na $abaababa$, jeho derivací vznikne $abaababaabaab$ a tak dále (Obrázek 3.3).



Obrázek 3.3: Příklad derivace v d0L-systému.

3.3.1 Formální definice d0L-systému

V této podkapitole je uvedena formální definice d0L-systému, definice derivačního kroku a přehled uzávěrových vlastností 0L-jazyků podle [19].

Definice 9 *0L-systém.* Nechť V označuje abecedu, V^* množinu všech řetězců nad abecedou V a V^+ množinu všech neprázdných řetězců nad abecedou V. Pak *0L-systém* je uspořádána trojice $G = \langle V, \omega, P \rangle$, kde V je *abeceda symbolů*, $\omega \in V^+$ je neprázdná posloupnost symbolů zvaná *Axiom* a $P \subset V \times V^*$ je *konečná množina přepisovacích pravidel*. Pravidlo $(a, \chi) \in P$ píšeme jako $a \to \chi$. Znak a a řetězec χ nazýváme *předchůdce* (levá strana, predecessor) resp. *následník* (pravá strana, successor) pravidla. Předpokládáme, že pro každé $a \in V$ existuje alespoň jeden následník $\chi \in V^*$ takový, že $a \to \chi$. Pokud není pro daného předchůdce $a \in V$ žádné pravidlo explicitně definováno, předpokládá se, že *identické pravidlo* $a \to a$ náleží do množiny P.

Definice 10 *Deterministický 0L-systém.* 0L-systém je *deterministický* (označujeme d0L-systém) tehdy a jen tehdy, když pro každé $a \in V$ existuje právě jedno $\chi \in V^*$ takové, že $a \to \chi$.

Definice 11 *Propagující 0L-systém.* Speciálním případem 0L-systému je *propagující* 0L-systém (označujeme P0L, resp. Pd0L, je-li deterministický), který pro žádné $a \in V$ neobsahuje pravidlo ve tvaru $a \to \varepsilon$, což znamená, že symboly mohou být přepisovány, ale nesmějí být mazány.

Definice 12 *Derivační krok.* Nechť $\mu = a_1 \dots a_m$ je libovolný řetězec nad abecedou V. Řetězec $\nu = \chi_1 \dots \chi_m \in V^*$ je *přímo derivován* (nebo *generován*) z μ, zapisujeme $\mu \Rightarrow \nu$ tehdy a jen tehdy, když $a_i \to \chi_i$ pro všechna $i = 1, \dots, m$. Řetězec ν je generován z G derivací *délky n* pokud existuje *posloupnost řetězců* $\mu_0, \mu_1, \dots, \mu_n$ taková, že $\mu_0 = \omega$, $\mu_n = \nu$ a $\mu_0 \Rightarrow \mu_1 \Rightarrow \dots \Rightarrow \mu_n$.

Definice 13 *0L jazyky.* *Jazyk* generovaný 0L systémem G obsahuje všechny řetězce generované z axiomu v libovolném počtu derivačních kroků a označujeme ho jako $L(G)$. *Tedy* $L(G) = \{w, \omega \Rightarrow^* w\}$.

3.3.2 Základní vlastnosti 0L jazyků

V této části jsou uvedeny základní vlastnosti 0L jazyků, které vyplývají z definice 9.

Věta 3 Pro každý 0L-systém $G = \langle V, \omega, P \rangle$, pro libovolné nezáporné číslo n a pro libovolné řetězce x_1, x_2, y_1, y_2 a z patřící do Σ^* platí, že pokud $x_1 \Rightarrow^n y_1$ a $x_2 \Rightarrow^n y_2$, pak $x_1 x_2 \Rightarrow^n y_1 y_2$. Dále platí obrácené tvrzení, pokud $x_1 x_2 \Rightarrow^n z$, pak existují řetězce z_1, z_2 patřící do Σ^* takové, že $z = z_1 z_2$, $x_1 \Rightarrow^* z_1$ a $x_2 \Rightarrow^* z_2$.

Věta 4 Pro každý 0L-systém $G = \langle V, \omega, P \rangle$, pro libovolná nezáporná čísla n a m a pro libovolné řetězce x, y a z patřící do Σ^* platí, že pokud $x \Rightarrow^n y$ a $y \Rightarrow^m z$, pak $x \Rightarrow^l z$, kde $l = n + m$.

Věta 5 *Vztah mezi 0L systémy a 0L jazyky.* Jestliže L je 0L jazyk a $L \subseteq \Sigma^*$, pak existuje 0L-systém G takový, že $L(G) = L$ a Σ je abecedou v G.

Věta 6 *Délka řetězců v P0L-systému.* Je-li G P0L-systém a $x \Rightarrow y$, pak $|x| \leq |y|$.

Věta 7 *Vlastnosti derivace v 0L-systému.*

- Pokud $a \Rightarrow^* a^2$, pak i $a \Rightarrow^* a^4$.

  Důkaz: Pokud $a \Rightarrow^* a^2$, pak $a \Rightarrow^n a^2$ pro nějaké n. Pak ale $aa \Rightarrow^n a^4$ a tedy $a \Rightarrow^{2n} a^4$.

- Pokud $a^2 \Rightarrow^* a$, pak i $a^2 \Rightarrow^* \varepsilon$.

  Důkaz: Pokud $a^2 \Rightarrow^* a$, pak $a^2 \Rightarrow^n a$ pro nějaké n. Pak ale $a \Rightarrow^n a$ a $a \Rightarrow^n \varepsilon$. Pak ale $aa \Rightarrow^n \varepsilon$.

3.3.3 Uzávěrové vlastnosti 0L jazyků

Máme-li dvě množiny vygenerovaných řetězců, L_1 a L_2, můžeme z nich několika způsoby vytvořit nové množiny řetězců. Například jejich *průnik* $L_1 \cap L_2$ sestává ze všech řetězců, které náleží zároveň do množiny L_1 i do množiny L_2. Jejich *konkatenace* $L_1 L_2$ je složena z řetězců, které pokud rozdělíme na dvě části, pak jejich první část náleží do množiny L_1 a jejich druhá část náleží do množiny L_2. *Doplněk* $\neg L_1$ sestává ze všech řetězců nad danou abecedou, které nenáleží do množiny L_1. Podobně lze definovat množiny vzniklé sjednocením, iterací, pozitivní iterací atd.

Třída jazyků $\mathcal{L}$, např. třída bezkontextových jazyků, se nazývá *uzavřená vůči konkatenaci*, pokud konkatenace libovolných dvou jazyků z $\mathcal{L}$ je také v $\mathcal{L}$, podobně pro ostatní operace.

0L-jazyky nejsou uzavřeny vůči sjednocení, doplňku, konkatenaci, průniku, průniku s regulárními jazyky, homomorfismu a inverznímu homomorfismu.

Věta 8 *Uzavřenost vůči sjednocení.* Třída jazyků generovaných 0L jazyky *není* uzavřena vůči sjednocení 0L jazyků.

Důkaz: Jazyky $\{a\}$ a $\{a^2\}$ patří do třídy 0L jazyků. Ale jazyk $L = \{a\} \cup \{a^2\} = \{a, a^2\}$ do třídy 0L jazyků nepatří, což lze dokázat následovně. Pro libovolný 0L-systém $G = \langle V, \omega, P \rangle$ ukážeme, že $L \neq L(G)$. Uvažujme všechny možné hodnoty, kterých může nabývat axiom:

- $\omega = a$. Pokud $a^2 \notin L(G)$, pak $L \neq L(G)$. Pokud $a^2 \in L(G)$, pak musí platit $a \Rightarrow^* a^2$ a podle Věty 7 tedy i $a^4 \in L(G)$, pak ale $L \neq L(G)$.

- $\omega = a^2$. Pokud $a \notin L(G)$, pak $L \neq L(G)$. Pokud $a \in L(G)$, pak musí platit $a^2 \Rightarrow^* a$ a podle Věty 7 tedy i $\varepsilon \in L(G)$, pak ale $L \neq L(G)$.

- $\omega \notin L$. Zřejmě $L \neq L(G)$.

Tedy neexistuje žádný 0L-systém, který generuje konečný jazyk $\{a, a^2\}$.

Věta 9 *Uzavřenost vůči konkatenaci.* Třída jazyků generovaných 0L jazyky *není* uzavřena vůči konkatenaci 0L jazyků.

Důkaz: Jazyky $\{a\}$ a $\{\varepsilon, a\}$ patří do třídy 0L jazyků. Ale $\{a\}\{\varepsilon, a\} = \{a, a^2\}$ není 0L jazyk.

Věta 10 *Uzavřenost vůči průniku s regulárními jazyky.* Třída jazyků generovaných 0L jazyky *není* uzavřena vůči průniku s regulárními jazyky.

Důkaz: Jazyk $L_{0L} = \{a^{2^n}, n \geq 0\}$ je 0L jazyk. Jazyk $L_{RL} = \{a, a^2\}$ je regulární jazyk. Ale jazyk $L_{L_{0L} \cap L_{RL}} = L_{0L} \cap L_{RL} = \{a^{2^n}, n \geq 0\} \cap \{a, a^2\} = \{a, a^2\}$ nepatří do třídy 0L jazyků.

Věta 11 *Uzavřenost vůči homomorfismu.* Třída jazyků generovaných 0L jazyky *není* uzavřena vůči homomorfismu.

Důkaz: Nechť h je homomorfismus z $\{a\}$ do $\{a\}$ takový, že $h(a) = a^5$. Pak $h(\{\varepsilon, a, a^2\}) = \{\varepsilon, a^5, a^{10}\}$, přičemž $\{\varepsilon, a, a^2\}$ do třídy 0L jazyků patří, ale $\{\varepsilon, a^5, a^{10}\}$ nikoliv.

Důkazy dalších uzávěrových vlastností lze najít v [4] a [7].

3.4 Interpretace řetězců želví grafikou

Geometrická interpretace Kochovy sněhové vločky (Obrázek 3.1) je velmi jednoduchá. Znaky abecedy L-systému jsou interpretovány graficky jako úsečky jednotkové délky kreslené daným směrem.

3.4.1 Dřívější metody

Pro modelování složitějších organismů a rostlin potřebujeme důmyslnější metodu grafické interpretace L-systémů. První výsledky v tomto směru publikoval v roce 1974 Frijters a Lindenmayer a také Hogeweg a Hesper. V obou případech byly L-systémy použity především pro zachycení větvení modelovaných rostlin. Geometrická hlediska, jako velikost úhlů rotace či délka úsečky tvořící segment, byla zohledněna až ve fázi postprocessingu. Výsledky Hogewega a Hespera byly následně rozšířeny Smithem, který ukázal možnosti využití L-systémů pro realistické modelování a zobrazování rostlin.

Szilard a Quinton v roce 1979 navrhli jiný přístup k interpretaci L-systémů. Soustředili se na reprezentaci pomocí řetězců a ukázali, že překvapivě jednoduché d0L-systémy mohou generovat *fraktály* [13]. Jejich výsledky byly postupně rozšířeny v několika různých směrech. Siromoney a Subramanian popsali L-systémy, které generují křivky vyplňující prostor. Těmito metodami se zabývá například literatura [5, 23].

Prusinkiewicz se zaměřil na interpretaci ve stylu želvy známé z jazyka LOGO[2] a ukázal mnoho dalších příkladů fraktálů a rostlinám podobných struktur modelovaných pomocí L-systémů. Pozdější aplikace L-systémů s interpretací pomocí želvy zahrnovaly realistické modely listnatých rostlin [20] a také tzv.*kolam* vzory[3].

[2]LOGO je programovací jazyk, je založený na Lispu a byl určen pro výuku programování. Pomocí jednoduchých příkazů ovládajících pomyslnou želvu, která se pohybuje po ploše a kreslí různé obrazce

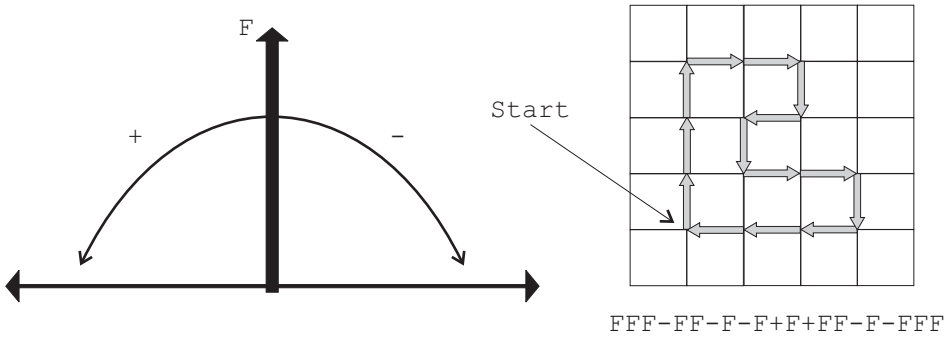
[3]Kolam vzory jsou formou umění, pocházejí z jihu Indie a zabývá se jimi například literatura [22]

3.4.2 Želví grafika

Nyní uvedeme definici stavu želvy a základních příkazů pro její ovládání podle [19].

Definice 14 *Stav a příkazy želvy v rovině.* Stav pomyslné želvy je definován jako trojice (x, y, α), kde $[x, y]$ reprezentují pozici želvy v kartézském souřadném systému a α je úhel označující směr, kterým se želva dívá, nazýváme ho *hlava* (heading). Dále je definována *velikost kroku* d a *přírůstek úhlu* δ. Želva dokáže reagovat na příkazy reprezentované těmito symboly (v konvenční notaci podle [19], Obrázek 3.4 vlevo):

F	Posun vpřed o krok délky d. Změna stavu želvy na (x', y', α), kde $x' = x + d\cos\alpha$, $y' = y + d\sin\alpha$. Vykreslení úsečky mezi body $[x, y]$ a $[x', y']$.
f	Posun vpřed o krok délky d bez kreslení úsečky.
+	Rotace doleva o úhel δ. Změna stavu želvy na $(x, y, \alpha + \delta)$.
-	Rotace doprava o úhel δ. Změna stavu želvy na $(x, y, \alpha - \delta)$.
{	Označuje start polygonu. Mezi symboly { a } mohou ležet jen symboly pro rotaci želvy nebo symboly reprezentované úsečkou.
}	Ukončuje generování polygonu. Od tohoto okamžiku lze generovat i symboly reprezentované objemovými tělesy.

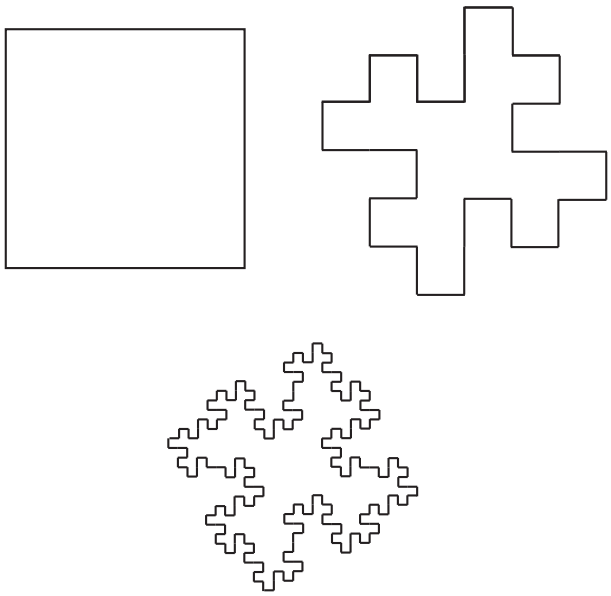


Obrázek 3.4: (vlevo) Interpretace symbolů F, $+$, $-$. (vpravo) Interpretace řetězce, přírůstek úhlu $\delta = 90°$ a želva je orientována směrem nahoru.

Mějme řetězec ν, počáteční stav želvy (x_0, y_0, α_0) a dané konstanty d a δ, příklad *želví interpretace* řetězce ν je na obrázku 3.4 vpravo. Tato metoda může být použita pro interpretaci řetězců, které jsou generovány pomocí L-systémů. Například na obrázku 3.5 jsou 3 generace *Kochova ostrova* [13], který je popsán tímto L-systémem:

$$\begin{aligned} \delta &= 90° \\ \omega &: \quad F - F - F - F \\ p &: \quad F \to F - F + F + FF - F - F + F \end{aligned}$$

Z příkladu je patrný vztah mezi konstrukcemi Kocha a L-systémy: iniciátor koresponduje k axiomu a generátor koresponduje k následníkovi (succesoru, pravé straně pravidla). Existuje mnoho dalších Kochových křivek generovaných L-systémy. Jednoduchost modifikace pravidel L-systémů je činí vhodnými pro vytváření nových typů Kochových křivek. Pravidla mohou být modifikována vkládáním, přepisováním a také vypouštěním (mazáním) některých symbolů. Některé další Kochovy křivky generované pomocí L-systémů jsou na obrázku 3.6:



Obrázek 3.5: První 3 generace Kochova ostrova.

$$
\begin{array}{lrcl}
4.generace & \delta & = & 90^\circ \\
 & \omega & = & -F \\
 & F & \rightarrow & F+F-F-F+F
\end{array}
$$

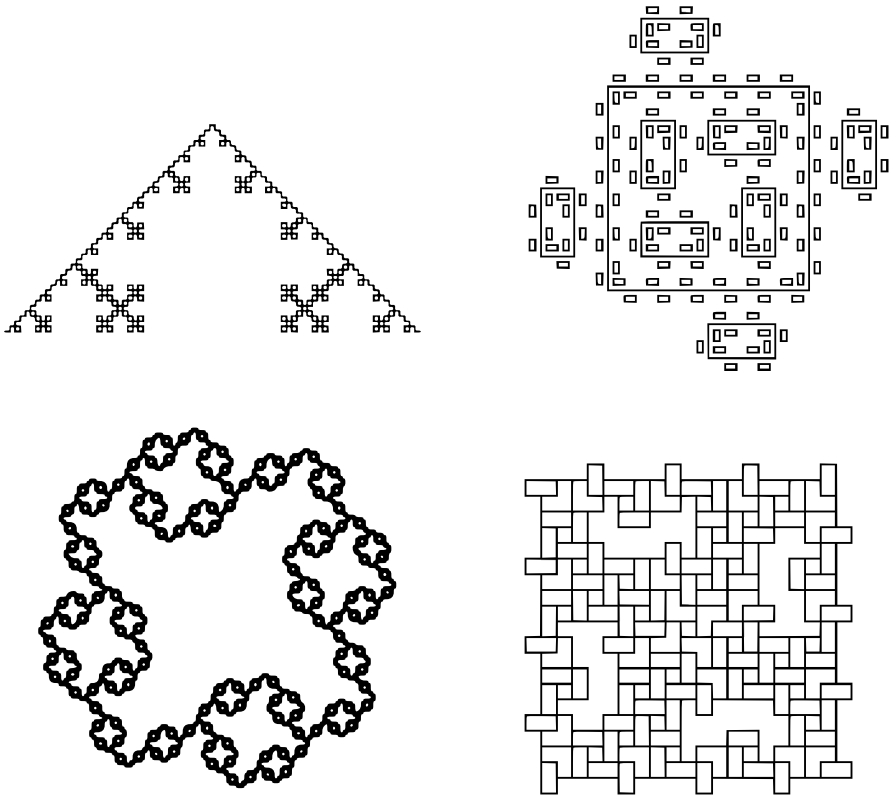
$$
\begin{array}{lrcl}
2.generace & \delta & = & 90^\circ \\
 & \omega & = & F+F+F+F \\
 & F & \rightarrow & F+f-FF+F+FF+Ff+FF-f+FF-F-FF-Ff-FFF \\
 & f & \rightarrow & ffffff
\end{array}
$$

$$
\begin{array}{lrcl}
4.generace & \delta & = & 90^\circ \\
 & \omega & = & F-F-F-F \\
 & F & \rightarrow & FF-F-F-F-F-F+F
\end{array}
$$

$$
\begin{array}{lrcl}
3.generace & \delta & = & 90^\circ \\
 & \omega & = & F-F-F-F \\
 & F & \rightarrow & FF-F+F-F-FF
\end{array}
$$

3.5 Technika přepisování

Často si přejeme vytvořit L-systém, který zachycuje strukturu popisující vývojový proces, čehož pomocí náhodné modifikace pravidel nedosáhneme. To se nazývá *odvozovací problém* (*inference problem*) v teorii L-systémů. Bylo popsáno několik algoritmů pro jeho



Obrázek 3.6: Další Kochovy křivky generované L-systémy.

řešení, nicméně v praxi byly obtížně použitelné. V následující podkapitole jsou popsány dvě metody motivované přírodou. Využívají dvou typů operací pro L-systémy s interpretací želví grafikou, podle terminologie v [6] se označují *přepisování hran* (*edge rewriting*) a *přepisování uzlů* (*node rewriting*). V případě přepisování hran pravidla nahrazují hrany mnohoúhelníku, zatímco přepisování uzlů pracuje s jeho vrcholy. Oba přístupy se opírají o rekurzivní struktury tvarů.

3.5.1 Přepisování hran

Přepisování hran můžeme chápat jako rozšíření Kochových konstrukcí. Obrázek 3.7 (vlevo) ukazuje *dračí křivku* (*dragon curve*), L-systém, který ji generuje je popsán následovně:

$$
\begin{array}{rrcl}
4.generace & \delta & = & 60^\circ \\
 & \omega & = & F_l \\
 & F_l & \rightarrow & F_l + F_r + + F_r - F_l - - F_l F_l - F_r + \\
 & F_r & \rightarrow & -F_l + F_r F_r + + F_r + F_l - - F_l - F_r
\end{array}
$$

Symboly F_l a F_r reprezentují hrany vytvořené příkazem F (viz 3.4.2) pro želvu. Index $_l$ a $_r$ značí levou, resp. pravou hranu. Tímto způsobem lze generovat mnoho zajímavých křivek, další ukázky lze najít v [19].

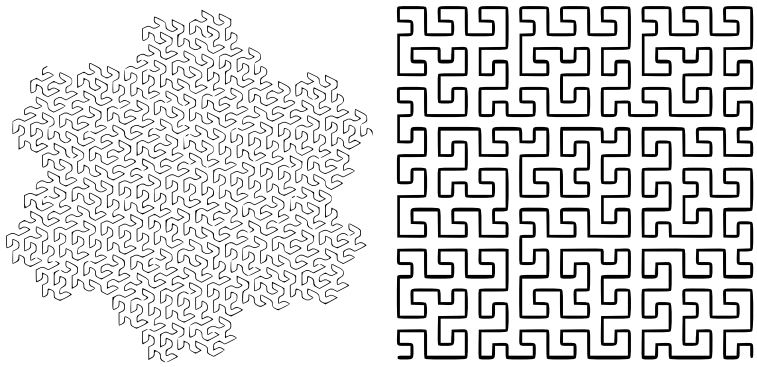
Křivky tohoto typu patří do skupiny tzv. *FASS* křivek (*space-filling, self-avoiding, simple and selfsimiliar*), tedy křivek vyplňujících prostor, neprotínajících se, jednoduchých

a soběpodobných. McKenna [19] presentoval algoritmus pro konstruování FASS křivek pomocí přepisování hran. Spočívá v tom, že symbol F_l je reprezentován polygonem, který přibližně vyplní čtverec po jeho levé straně. Podobně polygon nahrazující F_r přibližně vyplní čtverec po pravé straně této hrany. V dalších derivacích (Obrázek 3.7) je tento postup rekurzivně opakován, z čehož vyplývá, že vygenerovaná křivka vyplňuje celý prostor. V důsledku následujících dvou vlastností se křivka sama neprotíná:

- Generovaný polygon je neprotínající se.
- Nezáleží na relativní orientaci dvou sousedících polygonů, protože jejich sjednocením je neprotínající se křivka.

První vlastnost je zřejmá, druhá může být dokázána prověřením všech možných relativních orientací dvou sousedních polygonů.

McKenna také ukázal, že taková křivka je nejjednodušší FASS křivkou získanou přepisováním hran, protože všechny další takové křivky jsou tvořeny větším počtem hran.



Obrázek 3.7: (vlevo) Dračí křivka vytvořená přepisováním hran, (vpravo) Křivka vytvořená přepisováním uzlů.

3.5.2 Přepisování uzlů

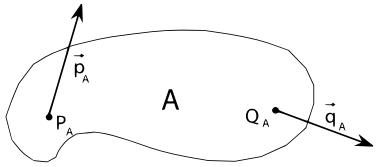
Idea přepisování uzlů spočívá v nahrazení vrcholů křivky novými polygony. Aby to bylo možné, je nutné rozšířit želví intepretaci o symboly reprezentující libovolné podobrazce. Jak je ukázáno na obrázku 3.8, každý podobrazec A z množiny podobrazců $\mathbf{A}$ je reprezentován:

- Dvojicí bodů zvaných *vstupní bod* P_A a *výstupní bod* Q_A.
- Dvěma směrovými vektory, *vstupním vektorem* $\vec{p_A}$ a *výstupním vektorem* $\vec{q_A}$.

Při interpretaci řetězce v symbol $A \in \mathbf{A}$ vloží korespondující podobrazec do obrázku. Poté je tento podobrazec posunut a otočen tak, aby jeho vstupní bod P_A a směr $\vec{p_A}$ byl stejný jako pozice a orientace želvy. Umístíme-li tímto způsobem podobrazec A, nastavíme jeho bod Q_A jako novou pozici želvy a jeho vektor $\vec{q_A}$ jako novou orientaci želvy (v rovině).

Předpokládejme, že máme definovány podobrazce L_n a R_n. Pak obrazce L_{n+1} a R_{n+1} můžeme definovat rekurzivně například takto:

$$L_{n+1} = +R_nF - L_nFL_n - FR_n +$$
$$R_{n+1} = -L_nF + R_nFR_n + FL_n -$$



Obrázek 3.8: Příklad podobrazce A.

Jestliže pak máme definovány křivky L_0 a R_0, získáme řetězec L_n a R_n, kde $n > 0$ rekurzivně v pořadí snižujícího se indexu n. V tomto příkladě:

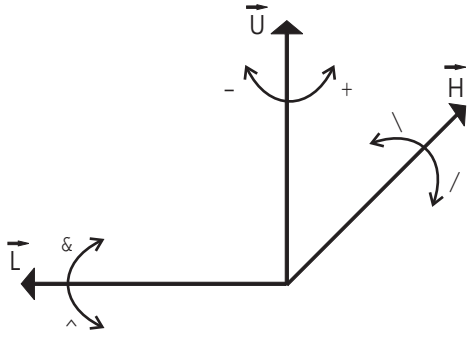
$$\begin{aligned} L_2 &= +R_1F - L_1FL_1 - FR_1+ = \\ &= +(-L_0F + R_0FR_0 + FL_0-)F - (R_0F - L_0FL_0 - FR_0+) \\ &\quad F(R_0F - L_0FL_0 - FR_0+) - F(-L_0F + R_0FR_0 + FL_0-) \end{aligned}$$

3.5.3 Vztah mezi přepisováním hran a uzlů

Třídy křivek, které mohou být generovány technikou přepisování hran a přepisování uzlů, nejsou disjunktní. Algoritmus převedení systémů s přepisováním hran na systémy s přepisováním uzlů je k nalezení v [19]. V praxi se mezi přepisováním hran a uzlů rozhodujeme podle toho, co je pro nás v daném případě výhodnější. Problematika dvou typů přepisování je důležitá zejména pro generování křivek vyplňujících prostor, které se neprotínají, což je opět motivováno biologií.

3.6 Modelování v prostoru

Ideu rozšíření želví grafiky pro modelování v prostoru přinesli Abelson a diSessa [1]. Orientace želvy v prostoru je dána třemi vektory $\vec{H}$, $\vec{U}$ a $\vec{L}$ (Obrázek 3.9) podle notace používané v knize [19]. Tyto vektory spolu s polohou želvy v prostoru tvoří její lokální souřadnicový systém. Jejich význam je:



Obrázek 3.9: Orientace želvy v prostoru.

$\vec{H}$	směr, kterým se želva dívá a kterým se pohybuje dopředu (*heading*)
$\vec{L}$	směr, ve kterém má levou přední (i zadní) nožičku (*left*)
$\vec{U}$	směr, ve kterém má krunýř (*up*)

Vektory mají jednotkovou délku a každý z nich je kolmý na ostatní, platí tedy vztah $\vec{H} \times \vec{L} = \vec{U}$. Rotace želvy v prostoru je pak vyjádřena vztahem

$$\begin{bmatrix}\vec{H'} & \vec{L'} & \vec{U'}\end{bmatrix} = \begin{bmatrix}\vec{H} & \vec{L} & \vec{U}\end{bmatrix}\mathbf{R}$$

kde $\mathbf{R}$ je 3×3 rotační matice [27]. Rotace o úhel α kolem vektorů $\vec{H}$,$\vec{L}$ a $\vec{U}$ jsou reprezentovány těmito rotačními maticemi:

$$\begin{aligned}
\mathbf{R_H}(\alpha) &= \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha \\ 0 & \sin\alpha & \cos\alpha \end{bmatrix} \\
\mathbf{R_L}(\alpha) &= \begin{bmatrix} \cos\alpha & 0 & -\sin\alpha \\ 0 & 1 & 0 \\ \sin\alpha & 0 & \cos\alpha \end{bmatrix} \\
\mathbf{R_U}(\alpha) &= \begin{bmatrix} \cos\alpha & \sin\alpha & 0 \\ -\sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix}
\end{aligned}$$

Kromě symbolů $+$, $-$ pro rotaci želvy v rovině definujeme další symboly pro rotaci v dalších dimenzích prostoru:

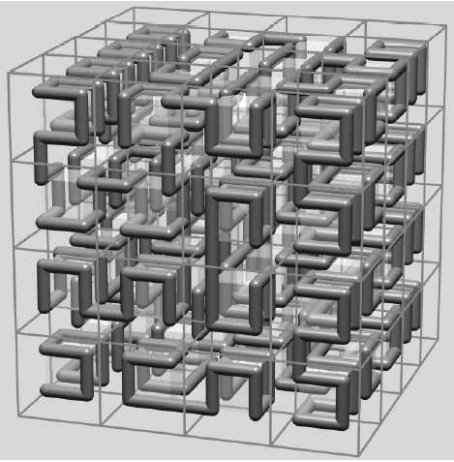
$+$	Rotace doleva o úhel δ s použitím rotační matice $\mathbf{R_U}(\delta)$
$-$	Rotace doprava o úhel δ s použitím rotační matice $\mathbf{R_U}(-\delta)$
$\&$	Rotace dolů o úhel δ s použitím rotační matice $\mathbf{R_L}(\delta)$
$\wedge$	Rotace nahoru o úhel δ s použitím rotační matice $\mathbf{R_L}(-\delta)$
$\backslash$	Rotace kolem podélné osy doleva o úhel δ s použitím rotační matice $\mathbf{R_H}(\delta)$
$/$	Rotace kolem podélné osy doprava o úhel δ s použitím rotační matice $\mathbf{R_H}(-\delta)$
$\|$	Otočení želvy *čelem vzad* s použitím rotační matice $\mathbf{R_U}(180°)$

Příklad trojrozměrného objektu vytvořeného pomocí L-systému je na obrázku 3.10. Jde o rozšíření Hilbertovy křivky [19], je vytvořen technikou přepisování uzlů a je popsán následujícím L-systémem:

$$\begin{aligned}
\delta &= 90° \\
\omega &= A \\
p : A &\rightarrow B - F + CFC + F - D\&F \wedge D - F + \&\&CFC + F + B// \\
B &\rightarrow A\&F \wedge CFB \wedge F \wedge D \wedge \wedge - F - D \wedge |F \wedge B|FC \wedge F \wedge A// \\
C &\rightarrow |D \wedge |F \wedge B - F + C \wedge F \wedge A\&\&FA\&F \wedge C + F + B \wedge F \wedge D// \\
D &\rightarrow |CFB - F + B|FA\&F \wedge A\&\&FB - F + B|FC//
\end{aligned}$$

3.7 Větvící se struktury

Dosud popsanými pravidly vygenerované řetězce želva interpretovala jako sekvenci úseček, nebo jiných geometrických primitiv. V závislosti na zvolené délce segmentu a velikosti úhlu rotace můžeme získat velmi zajímavé výsledky. Avšak v přírodě se v převážné většině



Obrázek 3.10: Trojrozměrné rozšíření Hilbertovy křivky, 2.generace. Symboly jsou reprezentovány krychlemi. Obrázek převzat z [19].

případů setkáváme s rozvětvenými strukturami, proto potřebujeme do teorie L-systémů zakomponovat matematický popis větvení. Jako základ poslouží kořenové stromy z teorie grafů, které jsou rozšířeny o botanicky motivovanou osu větve. Definice kořenového a osového stromu podle [19]:

3.7.1 Osové stromy

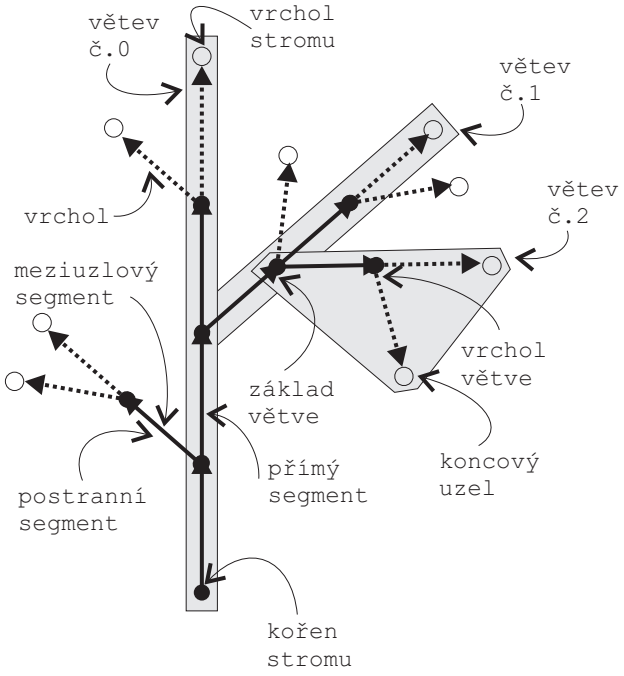
Definice 15 *Kořenový strom.* Kořenový strom je orientovaný strom, jehož hrany jsou označeny identifikátory a jeden z uzlů je označen jako *kořen* (*root*). Vrchol, který není kořenem a je obsažen pouze v jedné hraně, nazýváme *koncový vrchol*. Sled hran určuje cesty od uzlu, zvaného *kořen*, ke koncovým vrcholům. V biologickém kontextu odpovídají hrany úsekům větví (*branch segment*). Pokud je úsek následován aspoň jedním dalším úsekem, nazývá se *meziuzlový* (*vnitřní, internode*). Koncový úsek, tzn. s žádnou další navazující hranou, se nazývá *vrchol* (*apex*). Příklad takového stromu je na obrázku 3.11.

Definice 16 *Osový strom.* Osový strom je speciálním případem stromové struktury, kde z každého uzlu vychází maximálně jeden *přímý* (*straight*) úsek. Ostatní hrany označujeme jako *postranní úseky* (*lateral semgemnts*). Cesta (sled úseků) se nazývá osa, pokud:

- první úsek cesty začíná v kořeni stromu, nebo je to postranní úsek,
- každý následující úsek je přímý,
- poslední úsek cesty není následován žádným dalším přímým úsekem.

Společně se všemi svými následníky (potomky) tvoří osa *větev* (*branch*) a větev samotná je také osovým (pod)stromem.

Osy a větve jsou číslovány a to podle úrovní, na kterých začínají. Osa vycházející z kořene stromu má číslo 0. Osy začínající jako postranní úsek řádu n mají číslo $n + 1$. Číslo větve je určeno jako minimum z čísel os, které větev obsahuje.



Obrázek 3.11: Ilustrace k definicím kořenového a osového stromu.

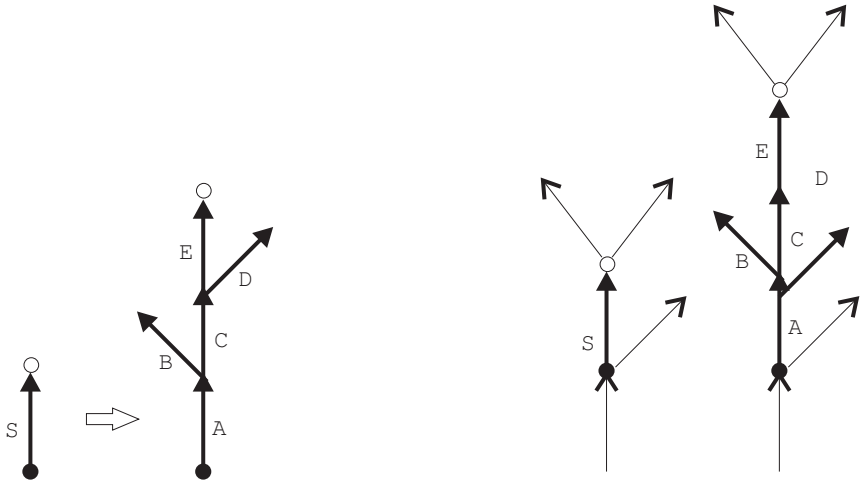
3.7.2 Stromové 0L-systémy

V modelech vývoje větvících se struktur můžeme použít techniku přepisování na osových stromech. Přepisovací pravidla těchto systémů nahrazují hranu osovým stromem tak, že kořen je připojen místo počátečního uzlu a koncový vrchol osy s číslem 0 místo koncového uzlu přepisované hrany (Obrázek 3.12).

Definice 17 *Stromový 0L-systém* Stromový 0L-systém G je určený třemi komponentami. Množinou hran V, počátečním stromem ω složeným z hran z množiny V, a množinou přepisovacích pravidel P. Osový strom T_2 je *přímo derivován* ze stromu T_1, což zapisujeme jako $T_1 \Rightarrow T_2$, pokud T_2 je z T_1 získán paralelním nahrazením každé hrany ve stromu T_1 jejím následníkem podle množiny pravidel P. Strom T je generován z G derivací délky n jestliže existuje sled stromů $T_0, T_1, \ldots T_n$ takový, že $T_0 = \omega$, $T_n = T$ a $T_0 \Rightarrow T_1 \Rightarrow \ldots \Rightarrow T_n$.

3.7.3 Závorkové L-systémy

Definice L-systémů nad osovými stromy nespecifikuje datovou strukturu pro jejich reprezentaci. Jednou z možností je použít reprezentaci rostliny stromovou topologií s využitím ukazatelů. Jinou možností reprezentace je použití *řetězců se závorkami* [10]. Rozšíření želví grafiky o řetězce se závorkami a o operace závorkových L-systémů jsou popsána v [17]. Zavádíme dva nové symboly pro vymezení větve a ty jsou želví grafikou interpretovány takto:

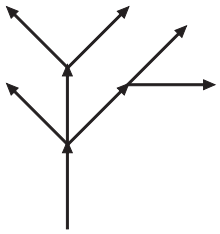


Obrázek 3.12: (vlevo) Grafické znázornění přepisovacího pravidla (vpravo) Princip přepisování ve stromovém 0L-systému.

[Uložení aktuálního stavu želvy na zásobník. Uložená informace obsahuje polohu $[X, Y, Z]$ a orientaci $(\vec{H}, \vec{U}, \vec{L})$ želvy v prostoru. Dále může obsahovat hodnoty barvy, šířky a dalších parametrů generované úsečky.

] Vyjmutí posledního uloženého stavu želvy z vrcholu zásobníku a změna polohy a orientace na hodnoty, které odpovídají situaci v okamžiku interpretace symbolu [.

Tím je želva vybavena zásobníkovou pamětí. Příklad jednoduché rozvětvené struktury a její reprezentace řetězcem se závorkami je na obrázku 3.13.



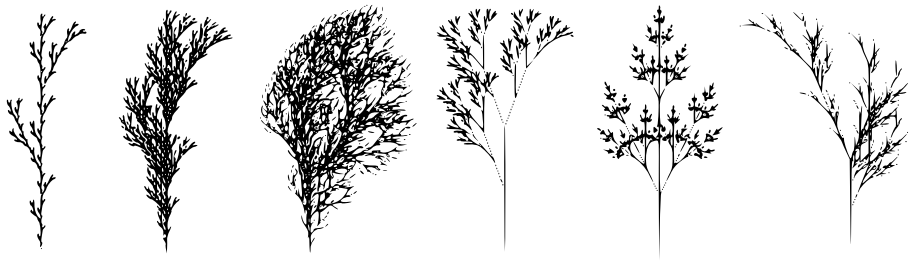
```
F[+F][-F[-F]F]F[+F][-F]
```

Obrázek 3.13: Rozvětvená struktura a její reprezentace řetězcem se závorkami.

Derivace v závorkovém 0L-systému probíhá stejným způsobem jako v 0L-systémech bez závorek. Přepis závorek nemá na princip přepisování žádný vliv, jelikož z pohledu přepisovacích pravidel se jedná pouze o další symboly. Interpretace závorek je úkolem želví grafiky. Příklady dvourozměrných závorkových struktur generovaných závorkovými 0L-systémy jsou na obrázku 3.14:

$$
\begin{array}{lrcl}
5.generace & \delta & = & 25,7^\circ \\
 & \omega & = & F \\
 & F & \rightarrow & F[+F]F[-F]F \\
5.generace & \delta & = & 20^\circ \\
 & \omega & = & F \\
 & F & \rightarrow & F[+F]F[-F][F] \\
4.generace & \delta & = & 22,5^\circ \\
 & \omega & = & F \\
 & F & \rightarrow & FF-[-F+F+F]+[+F-F-F] \\
7.generace & \delta & = & 20^\circ \\
 & \omega & = & X \\
 & X & \rightarrow & F[+X]F[-X]+X \\
 & F & \rightarrow & FF \\
7.generace & \delta & = & 25,7^\circ \\
 & \omega & = & X \\
 & X & \rightarrow & F[+X]F[-X]FX \\
 & F & \rightarrow & FF \\
5.generace & \delta & = & 22,5^\circ \\
 & \omega & = & X \\
 & X & \rightarrow & F-[[X]+X]+F[+FX]-X \\
 & F & \rightarrow & FF
\end{array}
$$

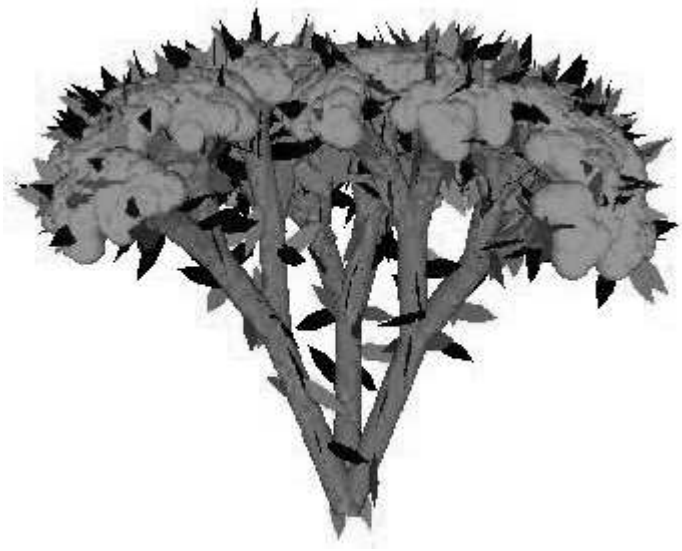
Ukázka trojrozměrné struktury generované závorkovým L-systémem je na obrázku 3.15.



Obrázek 3.14: Příklady rostlinám podobných struktur generovaných závorkovými 0L-systémy.

3.8 Stochastické L-systémy

Všechny struktury generované stejným 0L-systémem jsou identické. Pokud budeme modelovat například les pomocí jednoho 0L-systému, který popisuje strom, bude výsledkem les, ve kterém jsou všechny stromy identické. Tato *umělá pravidelnost* působí nepřirozeně,



Obrázek 3.15: Ukázka trojrozměrné struktury generované závorkovým L-systémem (popis odpovídajícího L-systému je v [9]).

protože v přírodě jsou jednotlivé prvky systému sice stejného typu, přesto však jsou mírně odlišné. Potřebujeme tedy mechanismus, který v procesu přepisování zachová obecnou podobu generovaného systému, ale změní jeho detaily.

Změny detailů generovaného systému můžeme v zásadě dosáhnout dvěma způsoby:

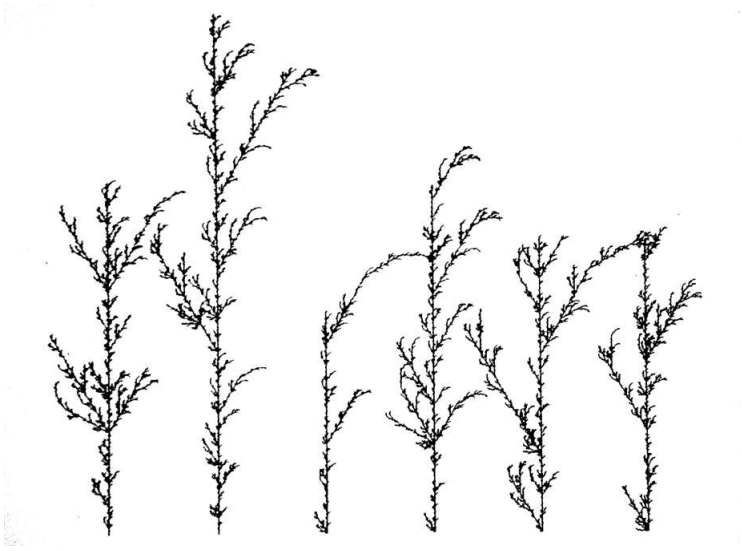
- Zavedením náhodných jevů *do interpretace* L-systému: výsledný efekt je omezený, ale zachovává topologii generovaného systému. Stačí tedy vygenerovat jen jeden řetězec symbolů.
- Zavedením náhodných jevů *do struktury* L-systému: stochastická aplikace pravidel může mít vliv jak na geometrii, tak na topologii. Každé generování řetězce může vést k jinému výsledku.

Definice 18 *Stochastický 0L-systém. Stochastický 0L-systém* je uspořádaná čtveřice $G_\pi = (V, \omega, P, \pi)$. Kde abeceda V, axiom ω a množina přepisovacích pravidel P jsou definovány stejně jako v 0L-systému (viz definice 9). Distribuční funkce pravděpodobnosti $\pi : P \rightarrow (0, 1 >$ mapuje množinu pravidel do množiny *pravděpodobností pravidel*. Pro každý symbol $a \in V$ se předpokládá, že součet všech pravděpodobností pravidel, které mají na levé straně a, je roven jedné.

Definice 19 *Stochastická derivace (podle [25])* Derivaci $\mu \Rightarrow \nu$ nazveme *stochastickou derivací* právě tehdy, když pro každý *výskyt* symbolu a ve slově μ je pravděpodobnost aplikace pravidla p s levou stranou a rovna $\pi(p)$.

Příklad stochastického závorkového L-systému:

$$\begin{aligned} \omega &: F \\ p1 &: F \overset{1/3}{\rightarrow} F[+F]F[-F]F \\ p2 &: F \overset{1/3}{\rightarrow} F[+F]F \\ p3 &: F \overset{1/3}{\rightarrow} F[+F]F \end{aligned}$$



Obrázek 3.16: Ukázka struktury generované stochastickým závorkovým L-systémem (specifikace viz 3.8).

3.9 Kontextové L-systémy

V 0L-systémech probíhá přepisovací proces bez ohledu na kontext, ve kterém levá strana pravidla (předchůdce) leží. Později byly zavedeny různé rozšíření L-systémů, které při přepisovacím procesu berou ohled na kontext (viz [12]).

- *1L-systémy*: Přepisovací pravidla mají jednostranný kontext, jejich formát je $a_l < a \rightarrow \chi$ nebo $a > a_r \rightarrow \chi$, kde symbol a může být přepsán na χ tehdy a jen tehdy, když symbol a je předcházen symbolem a_l, resp. je následován symbolem a_r.
- *2L-systémy*: Přepisovací pravidla mají oboustranný kontext a jsou ve formátu $a_l < a > a_r \rightarrow \chi$, kde symbol a může být přepsán na χ tehdy a jen tehdy, když symbol a je předcházen symbolem a_l a následován symbolem a_r.

Dosud popsané typy, tj. 0L-systémy, 1L-systémy a 2L-systémy, patří do třídy tzv. *(k,l)-systémů*, kde levý kontext je slovo délky k a pravý kontext je slovo délky l, přičemž celá tato třída je také označována jako *IL-systémy*. Dále se předpokládá, že pokud v množině P jsou dvě a více pravidel se stejnou levou stranou, pak mají kontextová pravidla vyšší prioritu, než pravidla bezkontextová.

Levá strana kontextového pravidla p se skládá ze tří komponent: *cesty* l udávající levý kontext, *hrany* S (tj. předchůdce) a *osového stromu* r udávajícího pravý kontext. Je zřejmé, že v takovém systému je jen jediná cesta od kořene stromu k přepisované hraně (což je levý kontext), zatímco může existovat mnoho cest od přepisované hrany ke koncovým uzlům. Pravidlo p odpovídá výskytu hrany S ve stromě T, pokud l je cesta ve stromě T končící v počátečním uzlu hrany S a r je podstrom stromu T začínající v koncovém uzlu hrany S. Aplikace tohoto pravidla pak znamená nahrazení hrany S osovým stromem, kterým je pravá strana pravidla.

V případě závorkových kontextových L-systémů není při řetězcové reprezentaci osových stromů zachována sousednost jednotlivých segmentů. Například aplikace pravidla s levou stranou $BC < S > G[H]M$ na symbol S v řetězci $ABC[DE][SG[HI[JK]L]MNO$ musí přeskočit symboly $[DE]$ při hledání levého kontextu a symboly $I[JK]L$ při hledání pravého kontextu.

Příklad jednoduchého 1L-systému, který modeluje propagaci signálu skrz řetězec symbolů:

$$\begin{aligned} \omega &: baaaaaaaa \\ p1 &: b < a \rightarrow b \\ p2 &: b \rightarrow a \end{aligned}$$

Prvních několik řetězců generovaných tímto L-systémem (v závorce je délka derivace):

$$\begin{aligned} &baaaaaaaa \quad (0) \\ &abaaaaaaa \quad (1) \\ &aabaaaaaa \quad (2) \\ &aaabaaaaa \quad (3) \\ &aaaabaaaa \quad (4) \\ &aaaaabaaa \quad (5) \\ &\ldots \quad (\ldots) \end{aligned}$$

3.10 Parametrické L-systémy

L-systémy umožňují generovat různé zajímavé objekty, ale jejich modelovací síla je omezená, protože číselné hodnoty jsou v L-systému zadány pevně. Číselnými hodnotami zde máme na mysli například hodnoty udávající délku či šířku generované úsečky nebo velikost úhlu rotace. Z toho důvodu je jediným způsobem zadání iracionálních hodnot jejich racionální aproximace (např. hodnota délky přepony v rovnoramenném pravoúhlém trojúhelníku s odvěsnami délky 1, tj. hodnota $\sqrt{2}$).

Řešením by bylo zavést speciální moduly, které by se v mnoha případech vzájemně lišily jen velikostí generovaného objektu. Vznikl by tak L-systém se stovkami modulů a pravidel, čímž by se ale specifikace modelů stala obtížnou a matematická krása L-systémů by byla ztracena. A.Lindenmayer v [11] navrhuje jiné řešení, kdy k symbolům L-systému přidává numerické parametry a L-systémy, které takto vzniknou, označuje jako *parametrické*. Příslušné definice je možné nalézt např. v [18].

Definice 20 *Parametrické slovo. Parametrická slova* jsou řetězce modulů, které se skládají se *symbolů* a *parametrů*. Symboly jsou z *abecedy* V, parametry jsou z množiny *reálných čísel*

$\Re$. Modul se symbolem $V \in A$ a parametry $a_1, a_2, \ldots a_n$ označujeme jako $A(a_1, a_2, \ldots a_n)$. Každý modul náleží do množiny $M = V \times \Re^*$, kde $\Re^*$ je množina všech konečných sekvencí parametrů. Množinu všech řetězců modulů označujeme jako $M^* = (V \times \Re^*)^*$ a množinu všech neprázdných řetězců modulů označujeme jako $M^+ = (V \times \Re^*)^+$.

Hodnoty *skutečných parametrů* korespondují k *formálním parametrům* uvedeným ve specifikaci L-systému. Nad parametry je možné tvořit *logické* a *aritmetické* výrazy. V těchto výrazech je možné požívat aritmetické operátory $+$, $-$, $*$, $/$, operátor mocniny $\hat{}$, relační operátory $<$, $>$, $=$, logické operátory not !, and &, or | a závorky (), přičemž se předpokládá standardní priorita operátorů. Dále se předpokládá, že relační a logické výrazy vyhodnocené jako pravdivé mají hodnotu 1, jinak mají hodnotu 0. Množinu všech správně vytvořených logických výrazů s parametry ze Σ označujeme jako $\mathcal{C}(\Sigma)$. Množinu všech správně vytvořených aritmetických výrazů s parametry ze Σ označujeme jako $\mathcal{E}(\Sigma)$

Definice 21 *Parametrický 0L-systém. Parametrický 0L-systém* je definován jako uspořádaná čtveřice $G = (V, \Sigma, \omega, P)$, kde

- V je *abeceda* systému
- Σ je množina *formálních parametrů*
- $\omega \in (V \times \Re^*)^+$ je neprázdné parametrické slovo, které označujeme *Axiom*
- $P \subset (V \times \Sigma^*) \times \mathcal{C}(\Sigma) \times (V \times \mathcal{E}(\Sigma))^*$ je konečná *množina pravidel*

Pro oddělení *předchůdce*, *podmínky* a *následníka* jsou používány symboly : a $\rightarrow$. Pravidlo vybrané pro přepsání modulu musí splňovat následující tři podmínky

- Symbol modulu a symbol předchůdce ve vybraném pravidle musí být stejné.
- Počet skutečných parametrů modulu je stejný, jako počet formálních parametrů předchůdce.
- Podmínka musí být vyhodnocena jako pravdivá (po nahrazení formálních parametrů skutečnými hodnotami parametrů).

Definice 22 *Derivace v parametrickém 0L-systému.* Pokud modul a produkuje parametrické slovo χ jako výsledek přepisovacího procesu v L-systému G, pak píšeme $a \rightarrow \chi$. Mějme parametrické slovo $\mu = a_1 a_1 \ldots a_m$, pak řekneme, že slovo $\nu = \chi_1 \chi_1 \ldots \chi_m$ je *přímo derivováno* (nebo *generováno*) z μ, píšeme $\mu \Rightarrow \nu$, tehdy a jen tehdy, když $a_i \rightarrow \chi_i$ pro všechna $i = 1, 2, \ldots, m$. Parametrické slovo ν je generováno L-systémem G *derivací délky* n, pokud existuje sekvence slov $\mu_0, \mu_1, \ldots, \mu_n$ taková, že $\mu_0 = \omega$, $\mu_n = \nu$ a $\mu_0 \Rightarrow \mu_1 \Rightarrow \cdots \Rightarrow \mu_n$.

Pokud v množině pravidel není nalezeno žádné odpovídající pravidlo, je modul včetně parametrů přepsán sám na sebe. Pokud se k symbolu váže jeden nebo více parametrů, pak se předpokládá, že hodnota prvního parametru udává stav želvy při interpretaci.

V případě parametrických kontextových 2L-systémů je každá komponenta levé strany pravidla (levý kontext, předchůdce, pravý kontext) parametrickým slovem se symboly z abecedy V a formálními parametry z množiny Σ a každý formální parametr se může objevit také v podmínce pro přepsání pravidla.

Kontextové pravidlo v 2L-systému může vypadat například takto:

$$A(x) < B(y) > C(z) : x + y + z > 10 \rightarrow E((x + y)/2)F((y + z)/2)$$

Toto pravidlo pak může být v parametrickém slově $..A(4)B(5)C(6)...$ použito na modul $B(5)$, protože podle podmínek v 3.10 sekvence symbolů A,B,C v přepisovacím pravidle a parametrickém slově jsou stejné; shoduje se počet skutečných a formálních parametrů; podmínka $4 + 5 + 6 > 10$ je pravdivá.

Výsledkem aplikace uvedeného pravidla na uvedené parametrické slovo bude nahrazení modulu $B(5)$ dvojicí modulů $E(4.5)F(5.5)$ (samozřejmě ve stejném derivačním kroku budou podle příslušných pravidel přepsány také moduly $A(4)$ a $C(6)$).

3.10.1 Otevřené L-systémy

Otevřené L-systémy, přesněji řečeno nedeterministické kontextové parametrické L-systémy, byly navrženy především pro potřeby simulace růstu rostlin. Jejich mechanismus umožňuje propagaci biologických signálů od kořenů k listům a zpět. Otevřenost spočívá v jejich možnosti interagovat s prostředím. Tato interakce je obousměrná, jak směrem do L-systému, tak ven. Směrem dovnitř je možno například informovat přepisovací proces o detekci kolizí s překážkami, nebo o množství dopadajícího světla, o kyselosti půdy, či o přítomnosti hmyzu. L-systém může informovat okolí o svém rozložení v prostoru, o množství látek, které z něj vycházejí atp.

Otevřené L-systémy jsou parametrické bezkontextové stochastické L-systémy rozšířené o tzv. *komunikační moduly* tvaru:

$$?E(x_1, \ldots, x_m)$$

které slouží k přenášení informací mezi posloupností modulů a okolím objektu, který L-systém reprezentuje.

Před vlastním procesem přepsání modulů se provede jakýsi mezikrok, ve kterém se získávají hodnoty skutečných parametrů komunikačních modulů $?E(x_1, \ldots, x_m)$. Na začátku tohoto kroku jsou skutečné hodnoty parametrů komunikačních modulů neznámé. Posloupnost modulů je nejprve prohlížena zleva doprava a vypočítává se stav želvy v prostoru s tím, že není vytvářen geometrický model. Při nalezení komunikačního modulu je vytvořena zpráva a ta je následně předána prostředí. V této zprávě je obsažen dotaz na hodnoty parametrů. Okolí tyto parametry nastaví (stav želvy je znám) a L-systém pokračuje v prohlížení posloupnosti modulů. Poté, kdy jsou nastaveny parametry všech komunikačních modulů, začíná fáze přepisování modulů, která je shodná jako u parametrických L-systémů.

Příklad otevřeného L-systému, jehož pravidla zabraňují tomu, aby objekt rostl ve směru y dále, než do vzdálenosti dva:

$$\begin{array}{rcll} \omega & : & F(0,0)A?E(0) & \\ p1 & : & A >?E(y) & : y < 2 \rightarrow F(x, y + 1)A \\ p2 & : & A >?E(y) & : y \geq 2 \rightarrow \varepsilon \end{array}$$

Modul $F(x, y)$ vygeneruje úsečku z aktuální pozice do bodu o souřadnicích $[x, y]$. První pravidlo je aplikováno, pokud vrchol A nepřekročil hranici $y = 2$. Pokud k tomu došlo, je

aplikováno tzv. epsilon pravidlo $p2$, které vrchol A smaže z posloupnosti modulů. Derivace tedy mají tvar:

$$\begin{aligned} F(0,0)A?E(0) &\Rightarrow F(0,0)F(0,1)A?E(1) \Rightarrow \\ &\Rightarrow F(0,0)F(0,1)F(0,2)A?E(2) \Rightarrow \\ &\Rightarrow F(0,0)F(0,1)F(0,2)?E(3) \end{aligned}$$

3.11 L-systémy s kontextovou podmínkou

Rozšířením kontextových L-systémů jsou *L-systémy s kontextovou podmínkou*[15]. Jejich hlavní přínos je v tom, že závislost přepisovacích pravidel na kontextu *nemusí být těsná*. Tedy požadovaný kontext může ležet v libovolné vzdálenosti před nebo za přepisovaným symbolem. Přepisovací pravidla mají následující tvar:

$$A[?ContexCondition][: LogicalExpression] \rightarrow B$$

kde:

- A symbol s parametry, tzv. předchůdce, tj. *levá strana* přepisovacího pravidla.
- $ContexCondition$ je seznam kontextových podmínek oddělených čárkou. Každá kontextová podmínka je řetězec symbolů s formálními parametry.
- $LogicalExpression$ je libovolný výraz nad parametry A a reálnými čísly, který tvoří logickou podmínku pro použití přepisovacího pravidla. Pokud výraz $LogicalExpression$ není specifikován, pak se předpokládá pravdivost logické podmínky.
- B je seznam symbolů s parametry určenými pomocí aritmetických nebo logických výrazů.

Příkladem jednoduchého přepisovacího pravidla s kontextovou podmínkou může být

$$A(x)?B(y), C(r,z) : x < y + r \rightarrow D(x)E(y + r)$$

Toto pravidlo může být v parametrickém slově $..C(3,8)D(-1)B(5)H(0,0)A(1)F(3)..$ použito na symbol $A(1)$, protože existuje $C(3,8)$ a $B(5)$ takové, že platí $1 < 5 + 3$. Pokud by kontextové podmínce vyhovovalo více symbolů s parametry, pak může být použit libovolný z nich.

V L-systémech pro reprezentaci mnohonásobně rozvětvených struktur generujeme řetězec, což je lineární struktura. Je tedy běžné, že vzdálenost popisující dvě sousední větve, které vycházejí z jednoho uzlu, je značná. Právě s využitím kontextové podmínky se však můžeme podívat vpřed či vzad do libovolné vzdálenosti, což přináší nové možnosti v modelování.

Kapitola 4

Využití L-systémů

L-systémy se uplatňují především při simulaci rostlin, můžeme je však použít i pro modelování říčních toků, mořských mušlí, křivek definovaných dělícími schématy, ale i pro generování budov a silniční sítě ve virtuálním městě. Umožňují též vytváření klasických lineárních deterministických fraktálů.

4.1 Výhody modelování pomocí L-systémů

- Obecným principem L-systémů je na základě velmi malého množství vstupních dat vytvářet relativně složité struktury. Můžeme tedy generovat rozsáhlé přírodní scény, například lesy, pouze na základě několika málo vstupních dat uložených v paměti počítače.
- Možnost zvolit si počet iterací při aplikování přepisovacích pravidel. To může znamenat možnost zvolit si konkrétní stupně detailů například modelu města na základě našich požadavků či technického vybavení.
- Paralelní přepisování řetězců může simulovat paralelní vývoj objektu, kdy se všechny jeho části vyvíjejí současně. Modelovat paralelní vývoj pomocí sekvenčního přepisování by bylo problematické.
- Každá větná forma generovaná L-systémem představuje stav objektu v určitém nespojitém čase.
- Stav objektu v následujícím časovém okamžiku $t+1$ získáme v jednom kroku pomocí derivace věty v_t, která popisuje objekt v čase t, a následnou interpretací věty v_{t+1}. Modelování s využitím L-systémů je tak vhodné k vytváření animace vývoje objektu.

4.2 Aplikace L-systémů

- Modelování rostlin je oblast, pro kterou jsou L-systémy využívány nejčastěji a byly pro ni do jisté míry i navrženy. Protože modelování rostlin provází všechny kapitoly této práce, nebudeme zde principy znova opakovat.
- Modelování interakcí mezi rostlinami a prostředím. Pro modelování vývoje rostlin či celých ekosystémů je důležitá jejich interakce s prostředím nebo i mezi sebou navzájem. Začlenění interakce do teorie L-systémů je důležité pro modelování predikce

úrody nebo těžby dřeva, ale také pro počítačem podporovanou zahradní architekturu. Interakci dělíme do tří skupin:

- Globální vlastnosti prostředí, např. délka dne ovlivňuje rozkvétání, teplota zase ovlivňuje rychlost růstu.
- Lokální vlastnosti prostředí, např. překážky ovlivňují směr růstu kořenů, opora zase určuje tvar popínavých rostlin.
- Vzájemná interakce prostředí a rostliny, např. detekce kolizí či přístup ke světlu.

Obrázek 4.1: Model kořenů rostliny, jejichž růst je ovlivňován koncentrací vody. Převzato z [27].

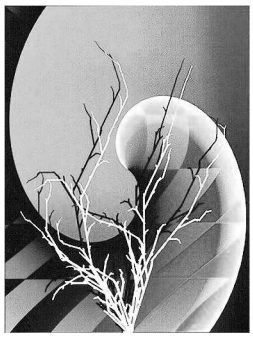
- Modelování ekosystémů. Ekosystém je samoregulující se sdružení rostlin a živočichů, které se vzájemně ovlivňují (neživým) prostředím, kde probíhá látková a energetická výměna. Důležitou roli hraje analýza systému s cílem pochopit jeho základní vlastnosti za účelem předpovědi jeho chování a vývoje v čase. Modely ekosystémů se používají v zahradní architektuře, pro predikci a zobrazení dopadu těžby dřeva v daném regionu, ale také pro tvorbu scén v počítačové animaci či leteckých simulátorech.

- Zpracování přirozeného jazyka. Pří analýze sémantiky přirozeného jazyka, ať už ve formě psané nebo mluvené, potřebuje identifikovat jednotlivé větné členy jako jsou podmět, přísudek, předmět atd. Přirozený jazyk je v podstatě kontextový jazyk, přičemž je běžné, že kontext není těsný. S výhodou zde lze použít L-systémy s kontextovou podmínkou. Představme si například anglickou větu *"I have a car."*, kde *"I"* je podmět, *"have"* je přísudek a *"a car"* je předmět. Je patrné, že člen *"a"* a podstatné jméno *"car"* jsou v těsném kontextu. Pokud ale větu upravíme na *"I have a new car."*, pak už člen a podstatné jméno nejsou v těsném kontextu, přesto však k sobě patří, a pro test můžeme použít pravidlo s kontextovou podmínkou.

- Počítačové umění. L-systémy se používají i v počítačovém umění, kde v kombinaci většinou s fraktály tvoří zajímavé kompozice. Nejznámějším představitelem tohoto umění je Alvy Ray Smith.

4.3 Modelování architektury pomocí L-systémů

V poslední době hrají L-systémy důležitou roli i při modelování ulic, budov a celých měst. Pro tyto účely byla vytvořena modifikace L-systémů, která se nazývá *self sensitive L-*



Obrázek 4.2: Příklady počítačového umění. K nalezení na `http://alvyray.com/Art/`. Autor: Alvy Ray Smith.

systems. Důvodem pro vznik této modifikace bylo umožnit práci především s prostorovým ohraničením bez nutnosti zvýšení složitosti přepisovacích pravidel. Tato modifikace je založena na parametrických stochastických L-systémech.

Obrázek 4.3: Prvních 5 iterací vývoje budovy a příklad virtuálního města. Obrázky převzaty z [16].

Modelování architektury s vysokou vizuální kvalitou a velkým množstvím geometrických detailů je činnost velmi časově náročná a také nároky na výkon počítače jsou poměrně vysoké. Tradiční modelování pomocí technik CAD produkuje modely geometricky i sémanticky naprosto přesné, ale prakticky neumožňuje modelovat různé *úrovně detailu* (*level of detail*, zkráceně *LOD*) stejného objektu, případně umožňuje, ale až na úrovni post-processingu před samotným zobrazením modelovaného objektu. Pomocí L-systémů můžeme modelovat úrovně detailu už na úrovni topologie objektu, kde každá úroveň detailu gene-

rovaného objektu odpovídá řetězci, který je generován jedním derivačním krokem v L-systému. To je velice výhodné například pro počítačové hry a filmy. V případě jednodušších objektů můžeme úrovně detailu objektu generovat a interpretovat až za běhu, pokud jsou objekty složitější (tj. detailnější), je možné řetězce reprezentující různé úrovně detailů předgenerovat a v čase běhu provádět jen jejich interpretaci. Případně lze také předem provést interpretaci a výsledek uložit jako model ve vhodném grafickém formátu pro reprezentaci částí 3D scény.

Z předchozího odstavce plyne další výhoda. Představme si scénu s dlouho ulicí, ve které je řada budov generovaných jedním L-systémem. To však neznamená, že budovy v této ulici musí být nutně všechny stejné, v případě použití stochastických L-systémů (viz 3.8) se mohou jednotlivé budovy značně lišit, přesto bude pro jejich generování použit jediný L-systém. Nyní uvažujme průchod takovou virtuální ulicí. Je výhodné bližší objekty zobrazovat s větším množstvím detailů, než objekty vzdálené. Nejbližší budovu vygenerujeme s nejvyšší úrovní detailu, tzn. pro generování řetězcové reprezentace budovy použijeme derivaci délky n, kde n je počet budov ve scéně. Řetězcovou reprezentaci druhé nejbližší budovy můžeme získat derivací délky $n - k$, kde $k = 1$. Obecně každou další budovu pak můžeme generovat derivací délky $n - k$, kde $k = 1 \dots n$, až nejvzdálenější budovu budeme reprezentovat pouze axiomem. Tento příklad je pouze ilustrativní, ve skutečnosti úroveň detailu generovaných objektů nemusí se vzdáleností klesat lineárně. Případně některé objekty mohou být v dané scéně významnější na základě jiných aspektů a proto mohou být generovány s větší úrovní detailu, i v případě, že jsou vzdálenější. Zásadní ideou tohoto odstavce však je použití *jediného L-systému pro generování celé množiny objektů.*

V oblasti modelování architektury najdou uplatnění všechny typy L-systémů uvedené v kapitole 3. V kontextu modelování architektury přepisovací pravidla nejprve vytvoří hrubý objemový 3D model, pak pokračují vytvořením struktury fasády a nakonec přidají detaily jako dveře a okna. Hierarchická struktura modelu je tedy specifikována přímo v modelovacím procesu.

4.3.1 Koncepty pro rozšíření přepisovacích pravidel

Idea modelování architektury pomocí gramatik byla zavedena v [16], kde autoři používají techniky vycházející ze *sekvenční aplikace přepisovacích pravidel* jako v Chomského gramatikách a zavádějí některá rozšíření (viz 4.3.1). Tato práce z jejich díla částečně vychází, ale zaměřuje se na *paralelní aplikaci přepisovacích pravidel* ve stylu L-systémů a také ukazuje některé nové možnosti využití L-systémů s kontextovou podmínkou.

Je výhodné zavést některé další modifikace pravidel, které přináší zjednodušení některých prvků při modelování v prostoru. Jde o zavedení konceptů *prostorového dělení objektů*, *relativních rozměrů* a *cyklického dělení* do specifikace L-systému, zejména do přepisovacích pravidel.

- *Prostorové dělení objektů.* Tento typ pravidel umožňuje rozdělení modulu na levé straně přepisovacího pravidla (předchůdce) podél dané osy. Pravidla jsou pak ve tvaru

  $$A \rightarrow div(axe, sizes)\{modules\}$$

  kde

  - A je *předchůdce*
  - *div* je klíčové slovo *funkce prostorového dělení*

- *axe* specifikuje *osu*, podle které bude dělení provedeno. Může nabývat hodnot "X", "Y" a "Z". Teoreticky je možné i dělení podél více os, pak k možným hodnotám tohoto parametru patří také "XY", "XZ", "YZ" a "XYZ".
- *sizes* jsou hodnoty oddělené čárkou a udávají *poměry* jednotlivých částí.
- *modules* je *seznam modulů* oddělených čárkou, které vzniknou jako výsledek přepisovacího procesu. Jejich počet musí být stejný jako počet hodnot v *sizes*.

Příkladem takového pravidla může být $A \rightarrow div("Y", 0.1, 0.3, 0.3, 0.3)\{B, C, C, C\}$, které modul A nahradí podél osy Y modulem B s atributem výšky 0.1 a třemi moduly C s atributy výšky 0.3. Pro ilustraci, takové pravidlo může být použito například pro rozdělení budovy na přízemí (modul B) a tři patra (moduly C).

- *Relativní rozměry.* Jde o rozšíření předchozího konceptu tak, aby mohl pracovat s relativními rozměry. Dohoda je, že standardně jsou hodnoty v *sizes* považovány za absolutní, pokud potřebujeme spíše relativní hodnoty, přidáme k nim písmeno r. Příkladem by pak bylo pravidlo $C \rightarrow div("X", 2, 1r, 1r, 2)\{D, E, E, D\}$, ve kterém budou relativní hodnoty r_i nahrazeny hodnotami

$$r_i \cdot (C.sx - \sum abs_i) / \sum r_i$$

, kde $C.sx$ je velikost modulu C ve směru osy x. Opět pro ilustraci, takové pravidlo může být použito například pro rozdělení jednoho patra budovy z předchozího ilustračního příkladu na čtyři části, z nichž ty krajní budou mít velikost 2 a velikost vnitřních bude určena v závislosti na velikosti patra ve směru osy x.

- *Cyklické dělení.* Předchozí dva koncepty jsou vhodné pro jednodušší případy dělení, pro složitější případy je vhodnější následující koncept. Pokud bychom například chtěli v příkladu na prostorové dělení objektů rozdělit budovu na mnohem větší počet pater, bylo by příslušné pravidlo velmi dlouhé a jeho případná modifikace velmi obtížná. Koncept cyklického dělení zavádí pravidla ve tvaru

$$A \rightarrow repeat(axe, size)\{module\}$$

kde

- A je *předchůdce*
- *repeat* je klíčové slovo *funkce cyklického dělení*
- *axe* specifikuje *osu*, podle které bude dělení provedeno. Může nabývat hodnot "X", "Y" a "Z".
- *size* udává *velikost* jednotlivých modulů.
- *module* je *použitý modul.*

Počet výsledných objektů bude roven hodnotě $A.sAxe/size$, kde $A.sAxe$ je velikost předchůdce ve směru osy *axe*. Pokud bychom tedy pro ilustraci chtěli rozdělit budovu do pater o výšce 0.3, můžeme použít pravidlo $B \rightarrow repeat("Y", 0.3)\{C\}$.

Příklad L-systému využívajícího těchto rozšíření i jeho možnou interpretaci lze nalézt v dodatku A.

Možnosti využití stochastických, kontextových a parametrických L-systémů jsou poměrně obsáhle popsány v kapitole 3. K výše uvedeným rozšířením tato práce nově přidává využití L-systémů s kontextovou podmínkou.

4.3.2 Využití L-systémů s kontextovou podmínkou

Kontextová podmínka nám umožňuje používat pravidla, jejichž kontextová závislost není těsná. Jinými slovy, specifikovaný kontext nemusí ležet bezprostředně vlevo nebo bezprostředně vpravo od přepisovaného symbolu, ale jeho vzdálenost od přepisovaného symbolu může být libovolná.

Uvažujme situaci z oblasti modelování architektury. Mějme dvě navzájem kolmé moduly stěny a řekněme, že jednu z nich chceme pokrýt několika moduly s okny. Na základě rozšíření z 4.3.1 můžeme s výhodou použít například pravidlo

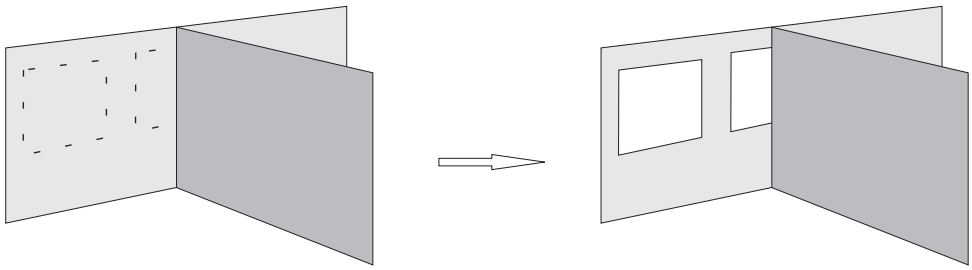
$$wall \rightarrow div("X", 0.3, 0.3, 0.3)\{window, window, window\}$$

Bez použití rozšíření můžeme toto pravidlo přepsat na

$$\begin{aligned} wall(x) &\rightarrow winplace(x * 0.3)winplace(x * 0.3)winplace(x * 0.3) \\ winplace &\rightarrow window \end{aligned}$$

První pravidlo rozdělí stěnu horizontálně na tři stejné části *winplace*. Druhé pravidlo do každé z těchto částí vloží modul s oknem *window*.

Problém může nastat v místě, kde se naše dvě zdi stýkají. Je možné, že část modulu s oknem může ležet přímo v místě styku dvou modulů stěny. Třebaže je to topologicky naprosto korektní, při modelování skutečné architektury by takové objekty působily nereálně (viz obrázek 4.4).



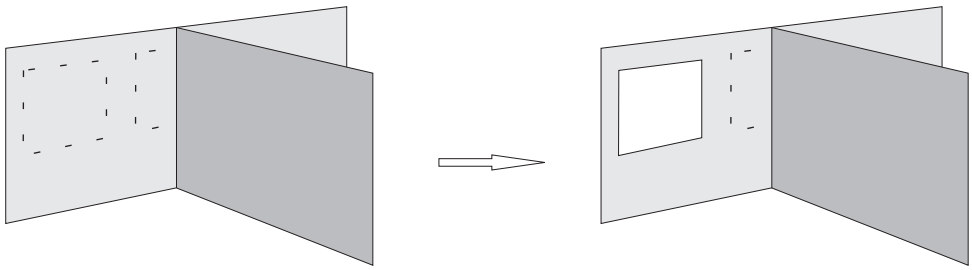
Obrázek 4.4: Modelování bez kontextové podmínky. Ilustrace příkladu z 4.3.2.

Tento problém je možné řešit pomocí L-systémů s kontextovou podmínkou, kde v podmínce můžeme testovat vzdálenost dvou modulů. Zaveďme klíčové slovo *distance*, které bude volat funkci *distance*(). Tato funkce má parametry $distance(axe, expression)$, kde *axe* udává osu v prostoru a *expression* je logický výraz. Výsledkem tohoto volání je pravdivostní hodnota udávající podmínku, která musí být splněna, aby dané pravidlo mohlo být aplikováno. V příkladě z předchozího odstavce můžeme do pravidla přidat kontextovou podmínku a získat tak pravidlo

$$\begin{aligned} wall(x) \rightarrow winplace(x * 0.3)winplace(x * 0.3)winplace(x * 0.3) \\ winplace?distance("x", wall > winplace.sx) \rightarrow window \end{aligned}$$

kde výraz $win > winplace.sx$ s parametrem x vyjadřuje výrok: "vzdálenost předchůdce od modulu *wall* je větší, než velikost modulu *winplace* ve směru osy x."

První pravidlo opět stěnu horizontálně rozdělí na tři stejné části *winplace*. Druhé pravidlo však každou z těchto částí naplní oknem pouze v případě, že horizontální vzdálenost od dalšího modulu stěny je větší, než horizontální velikost aktuálně přepisovaného modulu *winplace*. Situaci ilustruje obrázek 4.5.



Obrázek 4.5: Modelování s kontextovou podmínkou. Ilustrace příkladu z 4.3.2.

Kapitola 5

Návrh interpretu d0L-systémů

První navržená aplikace využívající L-systémů je *geometrický interpret d0L-systémů* a byla implementována ve vývojovém prostředí Microsoft Visual Studio 2005 (verze 8.0.50727-4200) v programovacím jazyce C#. Skládá se z tříd popisujících, kromě formuláře aplikace, především lexikální a syntaktický analyzátor, generátor řetězců, interpret a třídy pro práci s maticemi. V následujících podkapitolách budou principy těchto tříd popsány, konkrétní popis metod je uveden v programové dokumentaci.

5.1 Formát textového souboru pro popis L-systému

Struktura popisující d0L-systém v textovém souboru obsahuje:

- Definici použitých symbolů a popis jejich geometrické interpretace. Následuje za klíčovým slovem `modules:` a je složena z povinného *názvu* modulu a volitelného popisu *geometrické reprezentace*. `Název` je reprezentován jedním velkým nebo malým písmenem anglické abecedy. Každý ze symbolů může být reprezentován jedním z těchto grafických primitiv:
  - $C(x, y)$ pro reprezentaci válcem, kde x je průměr a y je výška válce v pixelech.
  - $P(x)$ pro reprezentaci úsečkou, kde x definuje její délku v pixelech.
  - $S(x)$ pro reprezentaci koulí, kde x definuje její průměr v pixelech.

  Symboly s nedefinovaným geometrickým významem jsou želvou ignorovány a nemají pro interpretaci žádný význam.
- Definici axiomu. Je uvozena klíčovým slovem `axiom:`, za nímž následuje řetězec znaků, který je složen ze symbolů definovaných v `modules` a případně ze symbolů pro ovládání želvy (viz 3.6, 3.7.3).
- Definici velikosti úhlů pro rotaci želvy ve směru vektorů $\vec{H}$, $\vec{U}$ a $\vec{L}$ (viz 3.6). Ta sestává z identifikátoru úhlu a příslušné hodnoty v úhlových stupních.
- Definice přepisovacích pravidel následuje za klíčovým slovem `rules:`. Každé přepisovací pravidlo je definováno levou a pravou stranou. Symbol na levé straně musí náležet do množiny symbolů definovaných v části `modules`. Pravá strana pravidla je řetězec libovolné délky, skládá se ze symbolů definovaných v části `modules` a dále z příkazů pro ovládání želvy (viz3.6, 3.7.3).

- Klíčové slovo `END`, za nímž může následovat libovolná posloupnost znaků. Mohou zde být například uvedeny poznámky k popsanému L-systému.
- Klíčové slovo \\ označuje počátek komentáře, všechny následující znaky na stejném řádku jsou ignorovány.

Příklad d0L-systému uloženého v textovém souboru (Obrázek 3.15) je v dodatku B.2.

5.2 Lexikální a syntaktická analýza

Lexikální analyzátor je založen na konečném automatu a poskytuje několik typů tokenů. Rozlišuje klíčová slova, identifikátory, symboly interpretované želvou, číselné konstanty a syntakticky nepřípustné znaky.

Vzhledem k velice jednoduché struktuře textového souboru je i konečný automat přijímající tento jazyk relativně jednoduchý. Využívá zásobníku, ale není to klasický zásobníkový konečný automat. Zásobník je objektový a je tedy do něj možné uložit hodnoty různých typů. Je využit například pro uložení stavu, v němž se má konečný automat nacházet v příštím kroku, čímž je redukován počet potřebných stavů.

Pokud jsme například přijali identifikátor proměnné, pak podle syntaxe jazyka má následovat znak = a po něm celočíselná hodnota pro přiřazení. Uložíme si tedy na zásobník číslo toho stavu, který reprezentuje načítání celočíselné hodnoty a do proměnné reprezentující stav konečného automatu uložíme číslo stavu reprezentujícího načtení znaku =. Následně ve stavu pro přijetí = vyjmeme ze zásobníků číslo stavu a vložíme ho do proměnné reprezentující stav konečného automatu. Tím se po přijetí dalšího tokenu dostaneme do správného stavu, v tomto případě do stavu pro načtení celočíselné hodnoty.

Dále je zásobník využit i pro počet parametrů, které chceme načíst. To se týká případu načítání geometrického významu modulů, kde parametry udávají rozměry grafického primitiva. Je zřejmé, že pro definici rozměrů koule potřebujeme jiný počet parametrů než pro definici rozměrů válce.

5.2.1 Princip generování řetězců

Pro každý symbol axiomu je nalezeno korespondující pravidlo z množiny přepisovacích pravidel. Jeho pravou stranou je nahrazen výskyt tohoto symbolu. Pokud pravidlo nalezeno není, pak se předpokládá identické pravidlo (viz Definice 9). V další iteraci se postup opakuje a výsledný řetězec je následně interpretován.

Délka generovaného řetězce na počtu iterací může (v závislosti na délce pravých stran přepisovacích pravidel) záviset i *exponenciálně*. Proto může být generování i relativně malého počtu iterací časově náročné. Například první iterace relativně jednoduchého L-systému z Dodatku B.2, jehož interpretace je znázorněna na obrázku 3.15, má délku 36 znaků. Druhá iterace čítá 216 znaků, ve třetí iteraci už generujeme znaků 777, ve čtvrté 2556 a v páté 8010 znaků. Je zřejmé, že zmíněná exponenciální závislost bude mít podstatný vliv i na rychlost interpretace vyšších iterací L-systému z tohoto příkladu. Částečným řešením urychlení interpretace je využití displaylistů (viz následující podkapitola).

5.3 Geometrická interpretace

L-systém popsaný a uložený v textovém souboru je graficky interpretován s využitím knihovny OpenGL, konkrétně jejího klonu, který pracuje v prostředí .NET frameworku, ve

verzi 1.4.1.

Princip interpretace spočívá v generování základního geometrického primitiva pro každý symbol vygenerovaný L-systémem. Následně je z těchto primitiv sestavena scéna a ta je uložena do tzv. displaylistu, což je seznam objektů, které jsou předem připravené pro zobrazení ve scéně. Do tohoto seznamu můžeme umístit libovolný objekt nebo skupinu objektů, můžeme s nimi libovolně manipulovat nebo vykreslovat a později je z displaylistu smazat. Tento postup je mnohem efektivnější, než neustálé opakované počítání, generování a sestavování geometrických primitiv.

V prostředí OpenGL je takto sestavený displaylist pouze ve smyčce zobrazován, přičemž je umožněno měnit směr pohledu na scénu. Dále je umožněno zobrazení na celou obrazovku a ve stavovém řádku jsou vypisovány údaje o aktuálním, minimálním a maximálním počtu snímků zobrazených za sekundu. Klávesou F1 lze vyvolat nápovědu. Po stisku klávesy F5 je pořízen záznam scény (screenshot), který je uložen do adresáře aplikace v grafickém formátu JPG (Joint Photographic Expert Group).

Podrobnější návod na ovládání aplikace je uveden v dodatku B.1.

Kapitola 6

Návrh interpretu parametrických, stochastických 2L-systémů s kontextovou podmínkou

K této práci byla dále navržena a implementována aplikace pro demonstraci možností modelování pomocí složitějších typů L-systémů. Aplikace byla implementována v .NET Frameworku verze 3.0 ve vývojovém prostředí Microsoft Visual C# 2008.

Jejím vstupem je definice L-systému s podporou parametrických modulů, stochastických pravidel, levého a pravého kontextu, logické a kontextové podmínky a definice geometrické interpretace jednotlivých modulů. Výstupem pak jsou jednotlivé generace získané derivací axiomu L-systému, jež je možné transformovat do popisu v jazyce VRML a zobrazit vhodným grafickým nástrojem.

Na následujících stránkách popíšeme datové struktury vhodné pro uložení L-systému, na jejich základě navrhneme aplikaci a způsob implementace derivace v L-systému a probereme také možnosti geometrické interpretace a vizualizace.

6.1 Datové struktury pro L-systémy

V aplikaci potřebujeme pracovat s L-systémem, proto je důležité navrhnout datovou strukturu pro jeho uložení a soubor metod pro práci s jeho jednotlivými komponentami jako je přidávání, odstraňování a editace modulů a jejich parametrů, vytváření logických a aritmetických výrazů pro použití v logické a kontextové podmínce a také pro definici hodnot parametrů na pravé straně pravidla.

6.1.1 Moduly

Základem L-systému je *abeceda*, nad níž je pak vystaven *axiom* a jednotlivá *přepisovací pravidla*. Pro účely aplikace funkci jednotlivých symbolů abecedy plní objekty *Module*, které k symbolům navíc přidávají pojmenované parametry a datový typ jejich hodnoty.

Objekt *Module* má atributy a metody:

- Atribut *Name* typu *String*, což je řetězec udávající název modulu. Název modulu musí být unikátní v rámci jednoho L-systému.
- Atribut *Parameters*, což je kolekce objektů typu *Parameter*. Jeden modul může obsahovat libovolný počet parametrů (objektů *Parameter*).

- Metodu $ToString()$ a atribut $Print$ sloužící pro získání textové reprezentace objektu.

Jednotlivé moduly L-systému je pak potřeba uchovávat v kolekci, ta má v aplikaci název $Modules$ a obsahuje tyto atributy a metody:

- Atribut $Items$ slouží pro přístup k jednotlivým objektům $Module$ v kolekci.
- Metody Add a $Contains$ sloužící pro přidání objektu $Module$ do kolekce a pro kontrolu unikátnosti názvu modulu (atributu $Name$ objektu $Module$).
- Přetížený selektor [] tak, aby bylo možné objekty v kolekci indexovat názvem (atributem $Name$ objektu $Module$).
- Metodu $ToString()$ sloužící pro získání textové reprezentace objektu.

V obecném L-systému jsou symboly resp. moduly obvykle ve skupinách, samostatný jediný symbol resp. modul se vyskytuje jen na levé straně pravidla. Z objektů $Module$ je proto možné vytvářet kolekce modulů, jejich název je $ModSeq$ a obsahuje tyto atributy:

- Atribut $Items$ slouží pro přístup k jednotlivým objektům $Module$ v kolekci. Zde už samozřejmě název modulu v rámci dané kolekce nemusí být unikátní.
- Atribut $Elements$ obsahující seznam názvů objektů $Module$ v kolekci.
- Metodu Add sloužící pro přidání objektu $Module$ do kolekce.
- Metodu $Contains$ vracející pravdivostní hodnotu $True$, pokud kolekce obsahuje modul s názvem předaným jako parametr této metody, a pravdivostní hodnotu $False$ v opačném případě.
- Metodu $IndexOf()$ vracející index modulu s názvem předaným jako parametr metody v rámci kolekce.
- Přetížený selektor [] tak, aby bylo možné objekty v kolekci indexovat názvem (atributem $Name$ objektu $Module$).
- Metodu $ToString()$ a atributy $Print$ a $PrintWithParams$ sloužící pro získání textové reprezentace objektu.

6.1.2 Parametry

V parametrickém L-systému si s sebou jednotlivé moduly nesou parametry, které později mohou sloužit například pro geometrickou interpretaci. Objekty typu $Parameter$ mají atributy:

- Atribut $Name$ typu $String$ udávající název parametru. Existují klíčové názvy parametrů vyhrazené pro geometrickou interpretaci, viz 6.3.2.
- Atribut $Value$, což je ukazatel na libovolný objekt udávající hodnotu parametru. Obvykle jde o celočíselnou hodnotu, ale může se jednat také o hodnotu, která je výsledkem vyhodnocení aritmetického nebo logického výrazu.
- Atribut $IsGeom$ nabývající pravdivostní hodnoty, která určuje, zda se jedná o parametr obecný či o parametr určený pro geometrickou interpretaci.

a lze je také sdružovat do kolekcí. Kolekce objektů *Parameter* jsou objekty *Parameters* a mají následující atributy a metody:

- Atribut *Items* slouží pro přístup k jednotlivým objektům *Parameter* v kolekci.
- Metody *Add* a *Contains* sloužící pro přidání objektu *Parameter* do kolekce a pro kontrolu unikátnosti názvu parametru (atributu *Name* objektu *Parameter*).
- Přetížený selektor [] tak, aby bylo možné objekty v kolekci indexovat názvem (atributem *Name* objektu *Parameter*).
- Metodu *IndexOf*() vracející index parametru s názvem předaným jako parametr metody v rámci kolekce.
- Metodu *ToString*() sloužící pro získání textové reprezentace objektu.

6.1.3 Logická podmínka

Logická podmínka je logický výraz, který musí být vyhodnocen jako pravdivý, aby mohlo být přepisovací pravidlo použito. Objekt *LogCondition* obsahuje tyto atributy a metody

- Atribut *Values* slouží pro přístup k jednotlivým objektům *Parameter* v kolekci.
- Metodu *ToString*() a atribut *Print* sloužící pro získání textové reprezentace objektu.

a je kolekcí následujících objektů:

- *LogValueInt* je objekt s celočíselným atributem *Value*.
- *LogValueString* je objekt s atributem *Value* typu *String*.
- *LogOperator* je aritmetický či logický operátor s atributem *Value*, který udává aritmetickou či logickou operaci.
- *LogElement* je objekt dědící z objektu *Module*, k němuž přidává celočíselný atribut *ParameterIndex* udávající index parametru modulu, který do logické podmínky zahrnujeme.

6.1.4 Kontextová podmínka

Kontextovou podmínku lze chápat jako kolekci objektů *ModSeq*, tedy kolekci kolekcí objektů *Module*. V navrhované aplikaci se objekt uchovávající kontextovou podmínku jmenuje *ContextCondition* a má tyto atributy a metody:

- Atribut *Items* slouží pro přístup k jednotlivým objektům *ModSeq* v kolekci.
- Atribut *UsedCCItems* pro uchování použitých modulů vyhovujících kontextové podmínce. Účel a použití tohoto atributu budou vysvětleny v částech 6.2.2 a 6.2.3.
- Metody *Add* a *Contains* sloužící pro přidání objektu.
- Metodu *Evaluate*() sloužící k vyhodnocení kontextové podmínky. Podrobné vysvětlení práce této metody bude popsáno v části 6.2.2.
- Metodu *ToString*() sloužící pro získání textové reprezentace objektu.

6.1.5 Přepisovací pravidla

Máme-li na základě výše uvedených datových struktur definovány jednotlivé moduly spolu s jejich parametry, můžeme z nich skládat přepisovací pravidla L-systému, což jsou objekty *Rule*. Každé pravidlo je složeno z následujících částí:

- *LefContext* je objekt typu *ModSeq* a tvoří levý kontext pravidla.
- *LefSide* je objekt typu *Module* a tvoří levou stranu pravidla.
- *RightContext* je objekt typu *ModSeq* a tvoří pravý kontext pravidla.
- *LogCondition* je objekt typu *LogCondition* a tvoří logickou podmínku pravidla.
- *ContextCondition* je objekt typu *ContextCondition* a tvoří kontextovou podmínku pravidla.
- *RightSide* je objekt typu *ModSeq* a tvoří pravou stranu pravidla.
- *Probability* je hodnota z intervalu $< 0; 1 >$ a udává pravděpodobnost použití pravidla, přičemž suma všech pravidel se stejnou pravou stranou může mít hodnotu maximálně rovnu 1.

Kromě výše zmíněných částí má pravidlo metodu *ToString*() a atribut *Print* sloužící pro získání textové reprezentace objektu.

Pravidla jsou v rámci L-systému opět uchovány v kolekci *Rules* s následujícími atributy a metodami:

- Atribut *Items* slouží pro přístup k jednotlivým objektům *Rule* v kolekci.
- Metody *Add* a *Contains* sloužící pro přidání objektu *Rule* do kolekce.
- Metoda *GetSumm*() vrací číselnou hodnotu udávající sumu všech pravidel levou stranou předanou jako parametr metody a slouží pro kontrolu při přidávání pravidla do kolekce.
- Metodu *ToString*() sloužící pro získání textové reprezentace objektu.

6.1.6 L-systém

Na základě všech v této podkapitole zmíněných datových struktur můžeme definovat datovou strukturu *LSystem* pro uložení celého L-systému. Základní atributy kopírují definici 9 (definice d0L-systému):

- Atribut *Modules* je objekt *ModSeq* a slouží pro uložení modulů definovaných v L-systému. V podstatě implementuje uložení abecedy V z definice 9 v paměti počítače.
- Atribut *Axiom* je objekt *ModSeq* a slouží pro uložení startovacího řetězce resp. startovací sekvence modulů. Jde o uložení axiomu ω z definice 9 v paměti počítače.
- Atribut *Rules* je objekt *Rules* a slouží pro uložení přepisovacích pravidel L-systému. Jde o uložení množiny pravidel P z definice 9 v paměti počítače.

Objekt *LSystem*, kromě výše uvedených základních atributů, obsahuje následující atributy a metody:

- Atribut *Print* sloužící pro získání textové reprezentace objektu.
- Atribut *GeomInt* definující přípustné typy geometrické interpretace modulů.
- Atribut *SentForms* pro uchování kolekce již vypočtených generací L-systému.
- Metodu *Derive*() pro provedení další derivace v L-systému, tj. pro výpočet další generace. Tato metoda je podrobně popsána v části 6.2.
- Metodu *Interpret*() pro vygenerování geometrické interpretace generace předané jako parametr metody ve formátu VRML. Podrobný popis této metody je uveden v části 6.3.2 a popis použitého výstupního formátu je uveden v části 6.3.1.

6.2 Implementace jednoho derivačního kroku

Následující řádky se zabývají popisem implementace jednoho derivačního kroku v L-systému, jehož specifikace je uložena v objektech zmíněných v 6.1.

Implementace jednoho derivačního v 0L-systému kroku je velmi přímočará, stačí číst aktuální derivaci zleva doprava, pro každý symbol najít vhodné přepisovací pravidlo (pokud existuje) a daný symbol nahradit pravou stranou vybraného pravidla.

Derivace ve stochastickém L-systému je mírně komplikovanější. Pro každý přepisovaný symbol je navíc potřeba najít všechna možná přepisovací pravidla, jež je možné aplikovat, a mezi nimi zvolit náhodně jedno a jeho pravou stranou nahradit aktuálně přepisovaný symbol.

Situace se dále komplikuje zavedením parametrického L-systému, kde je nutné navíc:

- Při výběru pravidla vyhodnotit logickou podmínku.
- Při aplikaci vybraného přepisovacího pravidla vyhodnotit případné aritmetické nebo logické výrazy v definicích hodnot parametrů modulů na pravé straně přepisovacího pravidla.

6.2.1 Kontrola platnosti logické podmínky a vyhodnocení výrazů

Kontrola platnosti logické podmínky je nutná, abychom mohli určit, zda je možno dané přepisovací pravidlo aplikovat. Potřebujeme tedy vyhodnotit výraz, v tomto případě logický. Hodnoty parametrů modulů na pravé straně přepisovacího pravidla mohou být určeny hodnotou výrazu, tentokrát aritmetického či logického, které opět potřebujeme vyhodnotit.

Pro vyhodnocení výrazů, logických i aritmetických, je v této práci využita *precedenční syntaktická analýza* nad symboly $!, +, -, *, /, \&, |, =, <, >, <=, >=, (,), i, \$$, z nichž symbol i označuje identifikátor, symbol \$ označuje koncový symbol vstupu, symbol ! označuje operátor negace a ostatní symboly mají běžně používaný význam a označují operátory a závorky. Precedenční syntaktická analýza pracuje nad *precedenční tabulkou*, která definuje vzájemné priority operátorů a symbolů \$ a i, a nad *redukčními pravidly*. Princip konstrukce této tabulky je podrobně popsán např. v [14], zde uvedeme jen její výsledný tvar:

	!	+	−	∗	/	&	|	=	<	>	<=	>=	(	)	*i*	$
!	<	>	>	>	>	>	>	>	>	>	>	>	<	>	<	>
+	<	>	>	<	<	<	<	>	>	>	>	>	<	>	<	>
−	<	>	>	<	<	<	<	>	>	>	>	>	<	>	<	>
∗	<	>	>	>	>	<	<	>	>	>	>	>	<	>	<	>
/	<	>	>	>	>	<	<	>	>	>	>	>	<	>	<	>
&	<	>	>	>	>	>	>	<	<	<	<	<	<	>	<	>
|	<	>	>	>	>	>	>	<	<	<	<	<	<	>	<	>
=	<	<	<	<	>	>	<	>	>	>	>	>	<	>	<	>
<	<	<	<	<	>	>	<	>	>	>	>	>	<	>	<	>
>	<	<	<	<	>	>	<	>	>	>	>	>	<	>	<	>
<=	<	<	<	<	>	>	<	>	>	>	>	>	<	>	<	>
>=	<	<	<	<	>	>	<	>	>	>	>	>	<	>	<	>
(	<	<	<	<	<	<	<	<	<	<	<	<	<	=	<	
)		>	>	>	>	>	>	>	>	>	>	>		>		>
i		>	>	>	>	>	>	>	>	>	>	>		>		>
$	<	<	<	<	<	<	<	<	<	<	<	<	<		<	

Přesný princip precedenční syntaktické analýzy, stejně jako informace o souvisejících sémantických akcích použitých s redukčními pravidly, zde nebudeme uvádět, tyto informace lze nalézt např. v [14]. Výstupem této analýzy je *pravý rozbor vstupu*, pokud jde o aritmetický či logický výraz definovaný nad výše uvedenou tabulkou, nebo chyba, pokud tomu tak není.

Pokud uvažujeme kontextové L-systémy, pak levý a pravý kontext nepředstavují velké implementační komplikace. Pouze při ověřování aplikovatelnosti přepisovacího pravidla je potřeba zkontrolovat, zda se vlevo resp. vpravo od aktuálně přepisovaného symbolu nachází symbol korespondující s hodnotou levého resp. pravého kontextu v definici přepisovacího pravidla.

Relativně velké implementační komplikace však nastanou, vezmeme-li do úvahy L-systém s kontextovou podmínkou, kdy je nutné ověřit platnost kontextové podmínky a to, jak je popsáno v následující podkapitole, je implementačně poměrně velmi náročné.

6.2.2 Kontrola platnosti kontextové podmínky

V kontextovém, parametrickém nebo stochastickém L-systému při výběru vhodného přepisovacího pravidla je z definice zřejmé nejen *co* musíme zkontrolovat, ale také *kde* potřebné informace najdeme. V případě L-systému s kontextovou podmínkou sice víme *co* potřebujeme ověřit, ale nemáme žádné znalosti o tom, *kde* dané informace hledat. Do úvahy připadá libovolné umístění v dané generaci, kromě aktuálně přepisovaného symbolu.

Chceme-li tedy vyhodnotit platnost kontextové podmínky, je možno postupovat následovně.

- Pro každou položku v seznamu kontextových podmínek, označme ji i.
- Najít a zapamatovat si všechny výskyty i v aktuální generaci L-systému. Pokud se v aktuální generaci i nevyskytuje, pak kontextová podmínka neplatí a přepisovací pravidlo nemůže být použito.

- Pokud jsme v aktuální generaci našli všechny i a zároveň logická podmínka přepisovacího pravidla je prázdná, pak kontextová podmínka platí a přepisovací pravidlo může být použito. Pokud logická podmínka přepisovacího pravidla prázdná není, pak pokračujeme dalším krokem.
- Pokud jsme našli pouze jeden výskyt každého i v aktuální generaci L-systému, je potřeba vyhodnotit jedinou logickou podmínku přepisovacího pravidla. Pokud jsme našli více než jeden výskyt alespoň jednoho i, pak musíme vytvořit všechny možné kombinace nalezených výskytů i a následně vyhodnotit všechny logické podmínky vzniklé z těchto kombinací.
- V krajním případě tedy dostaneme pole logických podmínek. Pro každou z těchto logických podmínek je nutné nahradit formální parametry hodnotami skutečných parametrů. Formální parametry se definují ve tvaru {*location, module_name, parameter_name*} kde:
  - *location* udává část pravidla, ve které se požadovaný parametr nachází. Může nabývat hodnot LC resp. RC pro umístění v levém resp. pravém kontextu aktuálně přepisovaného symbolu, LS pro umístění na levé straně přepisovacího pravidla a CC pro umístění v kontextové podmínce přepisovacího pravidla.
  - *module_name* určuje v případě, že *location* obsahuje více než jeden modul, název modulu, ve kterém se požadovaný parametr nachází.
  - *parameter_name* udává název parametru, jehož hodnotu hledáme.
- Máme-li takto vytvořené pole logických podmínek se skutečnými hodnotami parametrů, pak bychom je měli všechny vyhodnotit tak, jak je uvedeno v 6.2.1, následně vybrat libovolnou jednu z těch, které byly vyhodnoceny jako pravdivé, a zapamatovat si moduly použité pro získání hodnot parametrů pro vybranou logickou podmínku. Protože platných logických podmínek může být z důvodu velkého množství kombinací mnoho a z nich vybíráme jednu libovolně, můžeme použít první logickou podmínku, která bude vyhodnocena jako pravdivá, a nemusíme tedy vyhodnocovat všechny.

Z předchozího je tedy zřejmé, že vyhodnocení kontextové podmínky může být značně časově náročné.

Uvážíme-li parametrický, stochastický, kontextový L-systém s kontextovou podmínkou, pak je nutné vzít také do úvahy všechny výše uvedené implementační komplikace zároveň, což dělá proces výběru vhodného přepisovacího pravidla z hlediska implementace poměrně složitým.

V parametrickém L-systému potřebujeme pro každý modul vyhodnotit maximálně n aritmetických výrazů a 1 výraz logický, tedy celkem $n + 1$ výrazů.

V parametrickém, stochastickém L-systému je nutno vyhodnotit maximálně $\sum_{i=1}^{m}(n + 1)_i$ výrazů, kde m je počet definovaných pravidel se stejnou levou stranou.

V parametrickém, stochastickém, kontextovém L-systému musíme navíc zkontrolovat x modulů vlevo od aktuálně přepisovaného modulu a maximálně y modulů vpravo od aktuálně přepisovaného modulu. Tedy $\sum_{i=1}^{m}(n + 1)_i$ výrazů a $x + y$ modulů.

V parametrickém, stochastickém, kontextovém L-systému s kontextovou podmínkou je třeba dále vyhodnotit platnost kontextové podmínky. Maximální počet logických podmínek, které je potřeba vyhodnotit, stoupne z hodnoty 1 na hodnotu všech možných kombinací výskytů položek kontextové podmínky v aktuální generaci L-systému.

Zavedení L-systémů s kontextovou podmínkou, stejně jako parametrických, stochastických a kontextových L-systémů, přináší nové možnosti v procedurálním modelování. Každý ze zmíněných typů L-systémů však na druhou stranu zvyšuje náročnost implementace derivace, přičemž největší zvýšení implementační náročnosti představují L-systémy s kontextovou podmínkou.

6.2.3 Vložení modulů z pravé strany pravidla

Máme-li vybráno vhodné přepisovací pravidlo, pak je dalším krokem nahrazení aktuálně přepisovaného symbolu resp. modulu těmi symboly resp. moduly, které se nacházejí v pravé straně zvoleného přepisovacího pravidla. Postup je poměrně přímočarý:

- Pro každý modul na pravé straně přepisovacího pravidla, označme jej i.
- Pro každý parametr modulu i, označme jej j.
- Nahradit formálně definovanou hodnotu parametru j skutečnou hodnotou. Zde je postup obdobný jako v 6.2.2 s tím rozdílem, že výrazy nemusí být jen logické, ale mohou být také aritmetické. Výrazy jsou vyhodnocovány postupem uvedeným v 6.2.1.
- Přidat modul i do nové generace L-systému.

Tím je proces přepsání jednoho modulu kompletní. Aplikace uvedeného postupu na všechny moduly v aktuální generaci L-systému pak je *jeden derivační krok*, čímž získáme novou generaci L-systému a tu můžeme vzápětí interpretovat.

6.3 Geometrická interpretace

Řetězce generované L-systémem chceme nějakým vhodným způsobem využít, jednou z možností je jejich geometrická interpretace. Protože L-systémy jsou vhodné pro modelování paralelního vývoje nejen v rovině, ale i v prostoru, chceme, aby geometrická interpretace byla také prostorová. Chceme tedy generovat trojrozměrnou počítačovou scénu, nějakým vhodným způsobem ji popsat a nakonec také zobrazit. Nabízí se dva přístupy:

- Generovat scénu a přímo ji zobrazit. Scénu budeme generovat pomocí vhodné grafické knihovny, např. OpenGL či DirectX, a aplikace ji bude přímo zobrazovat. Výhodou je větší kontrola nad generovanými grafickými primitivy, nevýhodou pak je závislost na aplikaci a nutnost zvolit ideálně jednu grafickou knihovnu. Tento přístup využívá např. aplikace popsaná v kapitole 5.
- Generovat pouze popis scény ve vhodném jazyce a na základě tohoto popisu pak scénu vhodným nástrojem zobrazit. Popis scény budeme generovat ve vhodném jazyce pro popis trojrozměrné scény, např. STL, WRML či X3D, a její zobrazení provedeme načtením vygenerovaného popisu ve vhodné aplikaci. Výhodou je menší závislost na aplikaci, nezávislost na konkrétní knihovně pro zobrazení scény, nevýhodou je menší kontrola nad generovanými grafickými primitivy.

Navrhovaná aplikace využívá druhý z výše zmíněných možných přístupů, přičemž jako jazyk pro popis trojrozměrné scény byl zvolen jazyk VRML (Virtual Reality Modeling Language). Pro samotné zobrazení scény poslouží libovolná aplikace pro prohlížení scény popsané tímto jazykem. V následující kapitole je na základě informací z [24] a [26] stručně popsán jazyk VRML.

6.3.1 Jazyk VRML

Jazyk VRML je jedním z jazyků určených pro popis prostorových scén. K dalším jazykům a potažmo datovým formátům určeným ke stejnému účelu patří například STL, X3D a do určité míry i DXF. Protože ve VRML lze popisovat aktivní i pasivní geometrické objekty, řadíme ho výše, než DXF, který je založen na reprezentaci výhradně pomocí polygonů. Nástupcem VRML pak je X3D, který je založen na dnes stále více populárním formátu XML.

Jazyk VRML je definován normou *ISO/IEC DIS 14772-1* a slouží jednak pro popis prostorových těles i rozsáhlých geometrických scén, ale také jako prostředek pro přenos dat popisujících trojrozměrné modely. VRML popisuje prostorová tělesa pomocí seznamu souřadnic jejich vrcholů a plochami určenými indexy svých vrcholů v seznamu vrcholů. Pro základní geometrická primitiva, jako je krychle, kužel či koule, však definuje klíčová slova, takže pomalý a prostorově náročný rozklad na jednotlivé trojúhelníky není v tomto případě nutný. Zmíněný rozklad na jednotlivé trojúhelníky je pak úkolem pro prohlížeč VRML, který nějakým způsobem musí spolupracovat s grafickým akcelerátorem, což se může dít např. pomocí API grafické knihovny OpenGL.

Idea VRML je inspirována formátem použitým v *Open Inventoru*. Myšlenkou je vytvářet hierarchické stromové struktury geometrických těles a jednotlivé vlastnosti, jako je transformace, barva či textura, měnit pro každou větev stromu zvlášť. Stromová reprezentace je výhodná, protože umožňuje jednoduše manipulovat jak s jednotlivými objekty, tak i s celými skupinami objektů. Soubory typu VRML mají koncovku *.wrl* a jsou textové, je tedy možné je upravovat se všemi výhodami i nevýhodami v libovolném textovém editoru.

Zápis prostorové scény ve VRML

Od VRML 2.0 se texty zapisují v kódování UTF-8 či ASCII, přičemž na prvním řádku textového souboru typu VRML je hlavička začínající znakem #, za kterým následuje použitá verze VRML a použitý způsob kódování. Za prvním řádkem následuje stromová struktura trojrozměrné scény, ta je reprezentována uzly, jejichž textový formát má tvar *jméno_uzlu*{*vnitřní obsah*}. Vnitřním obsahem uzlu se rozumí další uzly a jejich atributy, což jsou číselné, řetězcové, pravdivostní a další hodnoty. Pomocí speciálních uzlů nazvaných *Separator* a *Group* lze uzly sdružovat.

Následuje jednoduchý příklad popisu prostorové scény v jazyce VRML (převzato z [24]):

```
#VRML V1.0 ascii
Separator {
    DirectionalLight {  # nastavení osvětlení
        direction 0 0 -1
    }
    PerspectiveCamera { # nastavení pozorovatele (kamery)
        position       -8.6 2.1 5.6
        orientation    -0.1352 -0.9831 -0.1233  1.1417
        focalDistance  10.84
    }
    Separator {   # červená koule
        Material {
            diffuseColor 1 0 0
        }
```

```
        Translation {
            translation 3 0 1
        }
        Sphere {
            radius 2.3
        }
    }
    Separator {  # zelená krychle
        Material {
            diffuseColor 0 0 1
        }
        Transform {
            translation -2.4 .2 1
            rotation 0 1 1  .9
        }
        Cube {}
    }
}
```

V tomto příkladu je nastaveno osvětlení, nastavena pozice pozorovatele a následně jsou přidány dva uzly typu *Separator*, z nichž:

- První zobrazí červenou kouli o poloměru 2.3 prostorových jednotek a posunutou o 3 prostorové jednotky ve směru osy x a o 1 prostorovou jednotku ve směru osy z.
- Druhý zobrazí zelenou krychli s implicitními rozměry posunutou o -2.4 prostorových jednotek ve směru osy x, o 0.2 prostorových jednotek ve směru osy y a o 1 prostorovou jednotku ve směru osy z a otočenou o 0.9 radiánu ve směru osy y a z.

Datové typy a typy uzlů ve VRML

Ve VRML verze 1.0 je možno používat 16 datových typů, přičemž každý typ atributu vyžaduje jiný datový typ. V následující tabulce je přehled všech 16 datových typů a jejich přípustných hodnot, vyšší verze VRML zavádí další datové typy, které zde uvedeny nejsou.

SFBitMask	pouze hodnoty 0 a 1
SFBool	hodnoty true a false
SFColor	zápis barvy ve formátu RGB
SFEnum	interně se jedná o celé číslo
SFFloat	IEEE float
SFImage	pixmapa
SFLong	celočíselná hodnota
SFMatrix	matice (většinou transformační)
SFRotation	rotace zadaná čtyřmi hodnotami
SFString	ve VRML 1.0 většinou ASCII
SFVec2f	2D vektor
SFVec3f	3D vektor
MFColor	vektor více barev
MFLong	vektor více celočíselných hodnot
MFVec2f	vektor 2D vektorů
MFVec3f	vektor 3D vektorů

VRML 1.0 dále definuje 36 typů uzlů v několika skupinách, z nichž pro aplikaci v této práci jsou důležité zejména tyto:

- Popis geometrie objektů. Tato skupina 8 typů uzlů, z nichž využity jsou uzly typu *Cone*, *Cube*, *Cylinder* a *Sphere*.
- Vlastnosti objektů je skupina obsahující 12 typů uzlů, z nichž využit je zejména uzel typu *Material*.
- Transformace definuje 5 možných typů transformací, využity v této práci jsou *Rotation* a *Translation*.
- Vytvoření pozorovatele umožňuje vytvořit *perspektivní* resp. *ortogonální* pohled do scény, k čemuž slouží typy *PerspectiveCamera* resp. *OrthographicCamera*.
- Vytvoření světelného zdroje definuje tři typy světelných zdrojů, z nichž využit je jen typ *DirectionalLight*.
- Definice hierarchie objektů je ve VRML možno určit pěti typy uzlů, využity je typ *Separator*.

Zobrazení scény popsané ve VRML

Popis prostorové scény ve VRML lze zobrazit příslušnou aplikací pro vizualizaci. Vhodných aplikací je hned několik, přičemž rozlišujeme dva základní typy:

- Prvním z nich jsou samostatné aplikace, jejich výhodou je rychlost zobrazení scény a rychlost a plynulost pohybu v ní. Příkladem takové aplikace je *MYRIAD 3D Reader*, kterou lze nalézt na CD přiloženém k této práci, odkud je možné ji nainstalovat do systému.
- Druhým typem jsou aplikace ve formě doplňků do internetového prohlížeče, které umožňují vestavět VRML scénu do HTML stránek. VRML doplňkem pro internetové prohlížeče je např. *Cortona VRML Client*a lze jej opět nalézt a do systému nainstalovat z přiloženého CD.

6.3.2 Interpretace jedné generace L-systému

V této podkapitole je popsána samotná transformace z L-systému uloženého v datových strukturách v paměti počítače, resp. jeho jednotlivých generací, do jazyka VRML.

Vyhrazené názvy parametrů

Typ objektu, který má být pro daný modul při geometrické interpretaci vygenerován, je určen pomocí vyhrazených klíčových slov použitých jako název parametru modulu. S každým tímto klíčovým slovem se pojí seznam dalších vyžadovaných parametrů, které pak udávají parametry generovaného geometrického primitiva. Úplný výčet podporovaných typů geometrických primitiv a jejich parametrů následuje:

- *_typeCone* pro generování kuželu vyžaduje parametry:
  - *_bottomRadius*, udává poloměr kuželu.
  - *_height*, udává výšku kuželu.
  - *_color*, udává barvu koule.
- *_typeCube* pro generování kvádru vyžaduje parametry:
  - *_width*, udává šířku kvádru.
  - *_height*, udává výšku kvádru.
  - *_depth*, udává hloubku kvádru.
  - *_color*, udává barvu koule.
- *_typeCylinder* pro generování válce vyžaduje parametry:
  - *_radius*, udává poloměr válce.
  - *_height*, udává výšku válce.
  - *_color*, udává barvu koule.
- *_typeSphere* pro generování koule vyžaduje parametry:
  - *_radius*, udává poloměr koule.
  - *_color*, udává barvu koule.
- *_typeRotation* pro generování rotace vyžaduje parametry:
  - *_x*, udává rotaci ve směru osy x.
  - *_y*, udává rotaci ve směru osy y.
  - *_z*, udává rotaci ve směru osy z.
- *_typeTranslation* pro generování translace vyžaduje parametry:
  - *_x*, udává posun ve směru osy x.
  - *_y*, udává posun ve směru osy y.
  - *_z*, udává posun ve směru osy z.

Zmíněná transformace je relativně snadná a přímočará. Nejprve je vytvořena obálka, zahrnující definici použité verze VRML, použité kódování, dále pak vytvoření světelného zdroje a nastavení perspektivního pohledu do scény.

Pro každý modul v generaci L-systému, která má být interpretován, je v závislosti na jeho typu a hodnotách požadovaných parametrů vygenerován příslušný zápis ve VRML a ten je následně obalen uzlem typu *Separator* a přidán do scény.

Nakonec je celý vygenerovaný zápis ve VRML zapsán do souboru a uložen do adresáře, ze kterého je aplikace spuštěna.

Kapitola 7

Možnosti dalšího vývoje projektu

Tato práce má poměrně velký potenciál dalšího pokračování a nabízí několik různých oblastí, kterými by se další vývoj mohl ubírat. Na následujících stránkách se na jednotlivé možnosti dalšího pokračování podíváme podrobněji a v hrubých rysech si představíme i základní ideje navrhovaných řešení.

7.1 Rozšíření aparátu L-systémů

Při modelování pomocí L-systémů vyvstanou další požadavky, které by přinesly praktické výhody, ať už v podobě snadnější a intuitivnější definice pravidel, tak i v podobě ryze technických aspektů jako je rychlost a kvalita generování scény či omezení počtu elementů ve scéně za účelem snížení značné zátěže výpočetního systému. Z těchto důvodů by bylo výhodné v dalším pokračování projektu pracovat na následujících tématech:

- *Systémy L-systémů.* V teoretické rovině si lze představit, že hodnoty parametrů L-systému je možné definovat nejen jako konstantní hodnoty či jako výsledek vyhodnocení aritmetického, případně logického, výrazu, ale také jako řetězce definované derivací v jiném L-systému. Tím by bylo dosaženo určité formy komunikace mezi jednotlivými L-systémy a bylo by možné vytvářet hierarchické, případně i rekurzivní, struktury vzájemně komunikujících L-systémů.

- *Závislost přepisovacích pravidel na pohledu.* Díváme-li se na interpretaci generovaného řetězce z jistého pohledu (např. z přední strany), pak je zbytečné generovat příliš velké množství detailů na odvrácené straně. V této úvaze můžeme jít ještě dále, je zbytečné generovat příliš velké množství detailů na straně objektu, která je v aktuálním pohledu, v případě, že tuto stranu nevidíme celou. Řešením by mohlo být zavedení závislosti přepisovacích pravidel na pozici kamery, k tomu by bylo možné využít podobných funkčních volání jako v 4.3.1. Tyto funkční volání otevírají L-systém okolnímu světu, díky čemuž by bylo možné z vnějšku do L-systému předat informace o aktuální pozici kamery resp. o tom, co kamera v daném okamžiku snímá.

- *Kombinace s výhodami CAD.* Procedurální modelování pomocí L-systémů má ve srovnání s tradičním přístupem modelování v CAD své značné výhody, ale také slabiny. Bylo by vhodné nějakým způsobem výhody obou přístupů zkombinovat. Zmiňované kombinace by bylo možné dosáhnout například přídavným modulem do konkrétních CAD systémů, který by umožňoval modelování pomocí L-systémů. V tomto modulu

by uživatel mohl na základě definovaného L-systému procedurálně vymodelovat objekt v hrubých obrysech a jemu následně domodelovat detaily tradičním přístupem. Zajímavé by také bylo spojení s moderními technologiemi jako *subdivision modeling* a *T-splines*[1], což jsou nové přístupy v oblasti vizualizace, které do jisté míry automaticky dokončují hrubý model systému.

7.2 Grafické zpracování

Složitější typy L-systémů dovolují modelovat mnohem reálněji vypadající přírodní struktury. Dalším stupněm k věrnosti zobrazení je grafické zpracování L-systémem vygenerovaných struktur. Pokud jsme dosud například kmen stromu modelovali pomocí na sobě stojících a vzájemně pootočených válců, můžeme zobrazení vylepšit nanesením různých typů textur. Listy, dosud modelované pomocí polygonů, lze věrněji vytvořit s využitím křivek. Celkové grafické ztvárnění vygenerované struktury bylo dosud prováděno analytickými útvary, jinou možností může být využití implicitních ploch. V následujících kapitolách je nastíněna základní problematika výše popsaných technik.

7.2.1 Textury

Textura je popisem vlastností povrchu a je důležitá pro vnímání struktury, barvy a kvality povrchu objektu. Je to vzorek, který může být jak pravidelný, tak i nepravidelný. Prvek textury se jmenuje *texel*. Textura je spjata s materiálem, který by měl povrch objektu jednoznačně popisovat.

Aplikací textury můžeme dosáhnout podstatného zvýšení vizuální kvality objektu za cenu relativně malých nákladů, proto je textura intenzivně používána zejména v časově kritických aplikacích. Ukázalo se, že je často efektivnější definovat jednoduchou geometrii a složité textury, nežli definovat složité geometrické detaily. Výsledek je od složitých objektů k nerozeznání, obzvlášť u objektů, které jsou zobrazeny jen krátce nebo z velké vzdálenosti. Typickým příkladem jsou billboardy nebo modelování příšer v počítačových hrách.

Textury se rozdělují na základě toho, jakou vlastnost povrchu popisují:

- *Barva povrchu* je určena koeficientem difúzního odrazu. Mapování difúzní složky materiálu je nejčastěji používaným způsobem aplikace textury.
- *Odraz světla* se může měnit s místem povrchu a simuluje se jako změna zrcadlové složky materiálu. Projevem této vlastnosti je odrážející se okolí objektu na jeho povrchu.
- *Změna normálového vektoru* opticky mění tvar povrchu, aniž by měnila geometrii objektu. Výsledkem je povrch, který vypadá zprohýbaný, či jinak geometricky změněný. Typickým reprezentantem této techniky je hrbolatá textura (*bump mapping*). Využití hrbolaté textury v případě geometrické interpretace L-systémů je vhodné například pro nanesení jemných geometrických nepravidelností na povrch kmene, který je reprezentován válcem.
- Textura může určovat *průhlednost* povrchu, čímž se také docílí dojem změny geometrie povrchu. Tohoto typu textur by bylo možné využít pro realistické modelování určitých typů listů nebo květů.

[1] Podrobnější informace o těchto nových přístupech jsou mimo rozsah a téma této práce, lze je však najít na internetu na adresách http://www.subdivisionmodeling.com/ a http://www.tsplines.com/

- *Hypertextura* určuje optické vlastnosti nad povrchem objektu a hodí se zejména pro modelování vlasů, ohně nebo trávy.

Podrobnější informace o technikách texturování lze nalézt například v [27]

7.2.2 Reprezentace křivkami

Křivky a plochy se používají v počítačové grafice na mnoha různých místech. Setkáváme se s nimi při modelování ve třech i dvou dimenzích, při definici fontů, při určování dráhy pohybujících se objektů v animaci, při definici objektů pro šablonování aj.

V roce 1959 používal P. de Castelajau u firmy Citroën matematický model křivek a ploch, jenž mu je umožňoval jednoduše zadávat. Podobně v 60. letech vedl P. Béziere vývoj programového systému UNISURF pro návrh křivek a ploch u firmy Renault. Metody tvarování křivek a ploch se postupem času vyvíjely a zdokonalovaly a v současné době je k dispozici velmi silný nástroj. Podstatou tohoto rozvoje je kvalitní matematický aparát a v neposlední řadě i zřetelný komerční efekt, který se projevil zejména v oblasti průmyslového designu.

Výrazný pokrok a především sjednocení dříve používaných různorodých přístupů přineslo používání racionálních B-spline křivek a ploch s neuniformní parametrizací – NURBS (Non-Uniform Rational B-Spline). Tyto metody umožňují generovat klasické geometrické prvky, jako jsou například úsečky, kružnice, koule či válce, za pomoci stejných metod, které umožňují vytvořit křivky a plochy se složitými průběhy a tvary.

Problematice modelování pomocí křivek se věnuje například literatura [27].

7.2.3 Implicitní plochy

Přestože geometrické modely objektů používaných v počítačové grafice jsou založeny především na parametrickém vyjádření, praktické aplikace používají i modelování založené na implicitním vyjádření objektů.

Blinn v roce 1982 navrhl chápat izoplochy vzniklé při modelování elektrického potenciálu elementárních částic jako objekty. Postupně byla tato technika rozšiřována a tyto objekty dostávaly různé názvy, podle toho, jaké směšovací funkce používaly. J. Bloomenthal upozornil, že všechny tyto objekty patří do stejné kategorie a mohou proto být označovány jediným názvem *implicitní plochy* (*implicitní tělesa, implicit surfaces*).

Implicitní plocha je množina bodů P v prostoru, pro které platí $Q(P) = konst$. Příkladem takové množiny je například kulová plocha, která je zadána implicitní rovnicí $Q(P) = x^2 + y^2 + z^2 = r^2$, kde volbou konstanty r^2 (průměru koule) volíme v prostoru jednotlivé izoplochy, tj. místa, ve kterých nabývá implicitní funkce zvolené konstantní hodnoty. Modelovací možnosti jednoduchých implicitních ploch jsou omezené, jejich kombinací však lze modelovat tvarově bohaté objekty. Základem modelování je tzv. *kostra* (skeleton). Ta se skládá z *generátorů*[2], což jsou prvky kostry.

Pro všechny body P v dosahu generátoru je definována funkce $d = d(P)$, která určuje vzdálenost bodu P od generátoru. Dále je pro každý generátor definována *potenciálová funkce* $F(d)$, která určuje vliv generátoru na místa, která jsou ve shodné vzdálenosti d. Modelování implicitních ploch spočívá v tom, že používáme jednoduché prvky kostry (body, úsečky, polygony, části křivek), vyhodnotíme jejich potenciálové funkce a výsledná izoplocha pak vznikne kombinací dílčích příspěvků od generátorů.

[2] zde je význam slova generátor jiný než v 3.2

Modelování pomocí kostry má mnoho výhod. Kostra je poměrně intuitivní aproximací mnoha objektů (např. lidské tělo či rostliny mají kostru). Kostra je snadno zobrazitelná, modelování je tedy názorné, relativně snadné a přehledné. Kostra obsahuje zhuštěnou informaci o celém objektu. Kostru je také možné hierarchizovat, tj. sdružovat jednotlivé části do bloků a určovat jejich vzájemné působení.

Podrobnější teorie modelování s využitím implicitních ploch a techniky jejich zobrazování jsou k nalezení v [27].

Kapitola 8

Závěr

Tato diplomová práce na základě vlastností fraktálů, Chomského gramatik, techniky paralelního přepisování a grafické interpretace tzv. želví grafikou představuje d0L-systémy a jejich interpretaci, dále pak jejich vlastnosti včetně vybraných důkazů a naznačuje jejich aplikace zejména v oblasti modelování rostlinám podobných struktur.

Výše uvedený nejjednodušší typ L-systémů spolu s využitím stromových a větvících se struktur slouží jako základ pro pokročilejší typy L-systémů. Jde především o stochastické L-systémy, jednostranně kontextové L-systémy (tzv. 1L-systémy), oboustranně kontextové L-systémy (tzv. 2L-systémy), parametrické L-systémy a také tzv. otevřené L-systémy a v neposlední řadě také L-systémy s kontextovou podmínkou. Důraz je také kladen na možnosti aplikace představených typů L-systémů zejména v oblasti počítačové grafiky.

Důležitou částí práce bylo představení konceptů pro rozšíření pravidel, které je vhodné využívat nejen pro zjednodušení notace specifikace L-systému při modelování v prostoru, ale zejména pro otevření brány komunikace mezi L-systémem a okolním světem. Dále pak bylo prezentováno možné využití kontextové podmínky za tím účelem, aby pomáhala předcházet pozdějším potencionálním problémům při interpretaci jednotlivých derivací L-systému.

Na tomto teoretickém základě byly v rámci této práce implementovány dvě aplikace. První z nich slouží k demonstraci základních principů paralelní derivace a věnuje se výhradně d0L-systémům a jejich přímé interpretaci s využitím grafické knihovny OpenGL. Druhá aplikace pak demonstruje i principy pokročilejších typů L-systémů, avšak z tohoto důvodu už je aplikace natolik komplexní, že základní podstata už není tak zřejmá jako v případě první aplikace. Druhá aplikace se při interpretaci zaměřuje na překlad do jazyka VRML a jeho následné zobrazení ve vhodném softwarovém nástroji. Obě aplikace v současném stavu implementace poskytují prostor pro další vývoj, zejména v oblasti optimalizací za účelem snížením paměťových nároků aplikace, zvýšení rychlosti generování dalších derivací L-systémů a zajištění vyšší vizuální kvality generovaných grafických výstupů.

Možnosti dalšího vývoje této práce byly podrobně popsány výše v textu. V teoretické rovině jde o výzkum zejména tzv. systémů L-systémů, které mohou nějakou formou vzájemně komunikovat. Tento výzkum povede až k dvourozměrným jazykům a jeho cílem bude konstrukce nových systémů adaptovaných tak, aby jejich komponenty pracovaly paralelně a adekvátním způsobem odrážely potřeby moderního paralelního zpracování informace. Důraz bude kladen také na aplikace těchto systémů, které se nebudou soustředit jen na geometrickou interpretaci, jak tomu bylo v rámci této práce, ale také na překlad jazyků, matematickou lingvistiku a bioinformatiku.

Literatura

[1] Abelson, H.; diSessa, A. A.: *Turtle Geometry*. Cambridge: M.I.T. Press, 1980.

[2] Chomsky, N.: Three models for the description of language. *IRE Transactions on Information Theory*, ročník 2, č. 3, 1956: s. 113–124.

[3] Cvrčková, F.: *Přírodovědný časopis Vesmír: Procházka virtuální zahradou*. Praha, 1999, iSSN 1214-4029.

[4] Doucet, P.: *Studies on L-systems*. Dizertační práce, Univ. of Utrecht, 1976.

[5] Frijters, D.; Lindenmayer, A.: A Model for the Growth and Flowering of Aster Novae-Angliae on the Basis of Table $< 1, 0 >$ L-Systems. In *L Systems*, 1974, s. 24–52.

[6] Gunning, B. E. S.: *Cytomorphogenesis in plants*. Wien: Springer-Verlag, 1981.

[7] Herman, G. T.; Rozenberg, G.: *Developmental Systems and Languages*. Amsterdam: North-Holland, 1975.

[8] von Koch, H.: Une méthode géométrique élémentaire pour l'étude de certaines questions de la théorie de courbes planes. *Acta Mathematica*, ročník 30, 1905: s. 145–174.

[9] Koutný, J.: *L-systémy a jejich použití v praxi, bakalářská práce*. FIT VUT v Brně, 2006.

[10] Lindenmayer, A.: Mathematic models for cellular interactions in development. *Journal of Theoretical Biology*, ročník 18, 1968: s. 280–315.

[11] Lindenmayer, A.: *Adding continuous components to L-systems*, *LNCS*, ročník 15. Springer, 1974, s. 53–68.

[12] Lindenmayer, A.; Rozenberg, G. (editoři): *Automata, Languages, Development*, Amsterdam: North-Holland, 1976,
Early collection of papers on L-Systems.

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

[14] Meduna, A.: *Elements of Compiler Design*. Taylor & Francis Informa plc, 2008, ISBN 978-1-4200-6323-3, 304 s.
URL `http://www.fit.vutbr.cz/research/view_pub.php?id=8538`

[15] Meduna, A.; Švec, M.: *Grammars with Context Conditions and Their Applications [Wiley, 2004]*. John Wiley & Sons, 2005, ISBN 0-471-71831-9, 1–225 s.
URL `http://www.fit.vutbr.cz/research/view_pub.php?id=7638`

[16] Parish, Y. I. H.; Müller, P.: Procedural modeling of cities. In *SIGGRAPH*, 2001, s. 301–308.
URL `http://portal.acm.org/citation.cfm?id=383259.383292`

[17] Prusinkiewicz, P.: Graphical applications of L-systems. In *Graphics Interface/Vision Interface*, 1986.

[18] Prusinkiewicz, P.; Hanan, J.: Visualization of botanical structures and processes using parametric L-systems. *Scientific Visualization and Graphics Simulation,Wiley & Sons*, 1990: s. 183–201.

[19] Prusinkiewicz, P.; Lindenmayer, A.: *The Algorithmic Beauty of Plants*. New York: Springer, 1990, ISBN 0-387-97297-8.

[20] Prusinkiewicz, P.; Lindenmayer, A.; Hanan, J.: Developmental models of herbaceous plants for computer imagery purposes. *Computer Graphics*, ročník 22, č. 4, Srpen 1988: s. 141–150.

[21] Salomaa, A. K.: *Formal Languages*. Academic Press, 1973.

[22] Siromoney, G.; Siromoney, R.: Rosenfeld's cycle grammars and kolam. In *Proceedings of the 3rd International Workshop on Graph-Grammars and Their Application to Computer Science*, *LNCS*, ročník 291, editace A. R. H. Ehrig, M. Nagl, G. Rozenberg, Warrenton, VA: Springer, Prosinec 1986, ISBN 3-540-18771-5, s. 564–579.

[23] Siromoney, R.; Subramanian, K. G.: Space-filling curves and infinite graphs. In *Graph grammars and their application to computer science*, *LNCS*, ročník 153, editace H. Ehrig; M. Nagl; G. Rozenberg, 1983, s. 380–391.

[24] Tišnovský, P.: VRML: jazyk pro popis virtuální reality. 2007.
URL `http://www.root.cz/clanky/vrml-jazyk-pro-popis-virtualni-reality/`

[25] Yokomori, T.: Stochastic Characterizations of EOL Languages. *Information and Control*, ročník 45, č. 1, Duben 1980: s. 26–33.

[26] Zara, J.: An Introduction to Virtual Reality Modeling Language. *Lecture Notes in Computer Science*, ročník 1338, 1997: s. 331–??, ISSN 0302-9743.

[27] Žára, J.; Beneš, B.; Felkel, P.: *Modern Computer Graphics (Moderni pocitacova grafika)*. Computer Press, Brno, Czech Republic, 2004, in Czech.

Seznam obrázků

Dodatek A

L-systém s rozšiřujícími koncepty

Řekněme, že úkolem je specifikovat L-systém, který bude procedurálně modelovat panelový dům. Pro jednoduchost předpokládejme použití jen deterministických bezkontextových pravidel a interpretaci všech symbolů pomocí krychle.

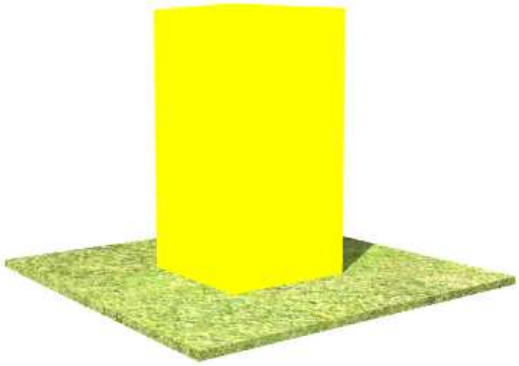
Protože účelem tohoto příkladu není podat detailní specifikaci L-systému, ale spíše demonstrovat princip modelování s využitím konceptů z 4.3.1, bude následující specifikace podána sice formálně, ale ne detailně. Parametry modulů budou vynechány a ke každému pravidlu bude uveden komentář vysvětlující jeho princip a pro lepší představu bude vždy uveden i ilustrační obrázek. Definujme tedy L-systém takto:

- $V = \{pozemek, panelak, prizemi, patra, patro, byt, pokoj, balkon, okno, dvere, zabradli\}$
- $\omega = pozemek$
- $P = \{\}$ ...jednotlivá pravidla představíme dále v textu.

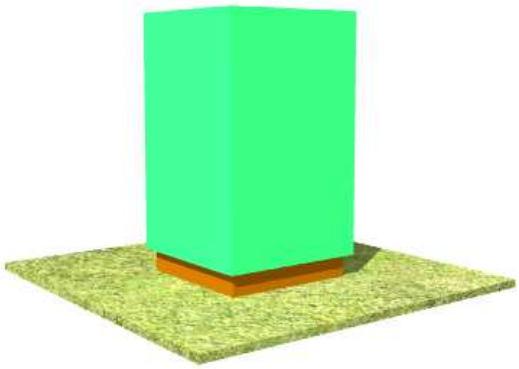
Axiom *pozemek* tedy interpretujeme jako kvádr, viz obrázek A.1.
Pravidla pak budou vypadat například následovně:

- $0 : pozemek \rightarrow I(panelak)$. Funkce I bude znamenat vložení objektu (insert), tedy neformálně "do pozemku vlož panelák". Viz obrázek A.2.
- $1 : panelak \rightarrow div(y, \dots)\{prizemi, patra\}$. Neformálně "panelák rozděl na přízemí a patra ve směru osy y", viz obrázek A.3.
- $2 : patra \rightarrow repeat(y, \dots)\{patro\}$. Neformálně "rozdělíme panelák na jednotlivá patra", viz obrázek A.4.
- $3 : patro \rightarrow div(xz, \dots)\{byt\}$. Neformálně "rozdělíme patro ve směru osy x a z na byty", viz obrázek A.5.
- $4 : byt \rightarrow div(x, \dots)\{pokoj, balkon\}$. Neformálně "rozdělí každý byt ve směru osy x na pokoj a balkon", viz obrázek A.6.
- $5 : pokoj \rightarrow I(okno)$ a $6 : balkon \rightarrow I(zabradli)$. Neformálně "do každého pokoje vložíme okno a do každého balkonu zábradlí", viz obrázek A.7.
- $7 : prizemi \rightarrow Split(x, \dots)\{sklep, dvere, sklep\}$, $8 : sklep \rightarrow I(okno)$ a $9 : dvere \rightarrow I(vstup)$. Neformálně "rozdělíme přízemí na dva sklepy a vstup, do každého sklepa přidáme okno a do vstupu přidáme dveře", viz A.8.
- Pokračovat bychom mohli dalšími pravidly, které by přidávaly další a další detaily...

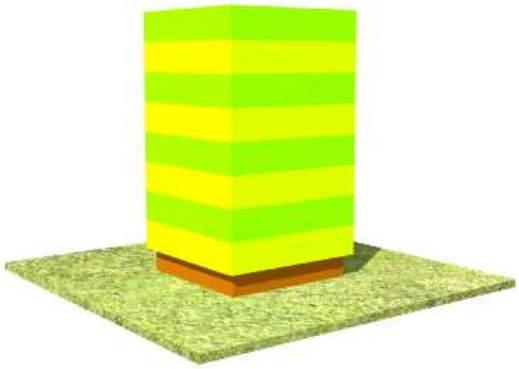
Obrázek A.1: $Axiom = pozemek$



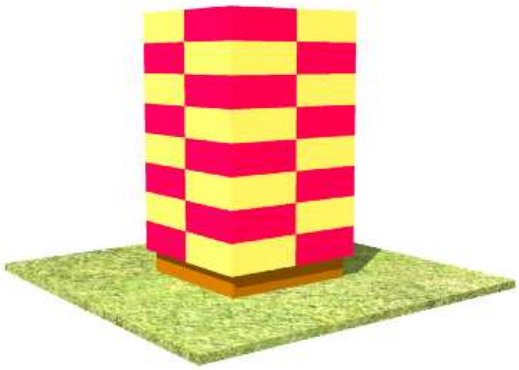
Obrázek A.2: $0 : pozemek \rightarrow I(panelak)$



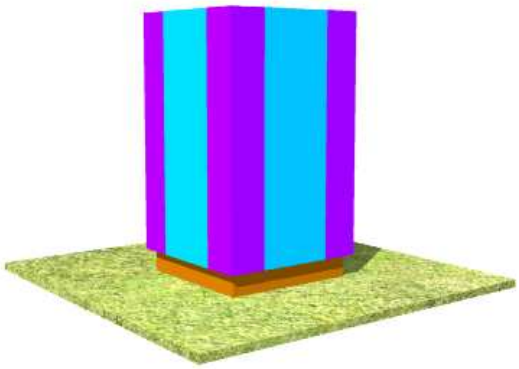
Obrázek A.3: $1 : panelak \rightarrow div(y, \dots)\{prizemi, patra\}$



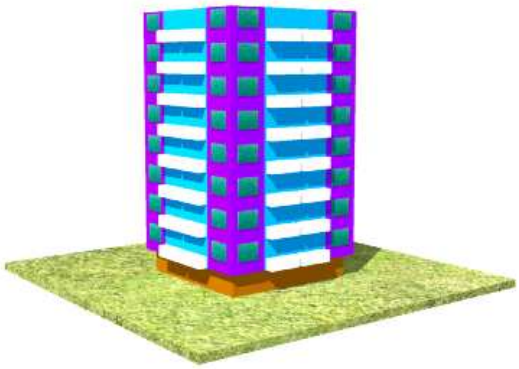
Obrázek A.4: $2 : patra \rightarrow repeat(y, \dots)\{patro\}$



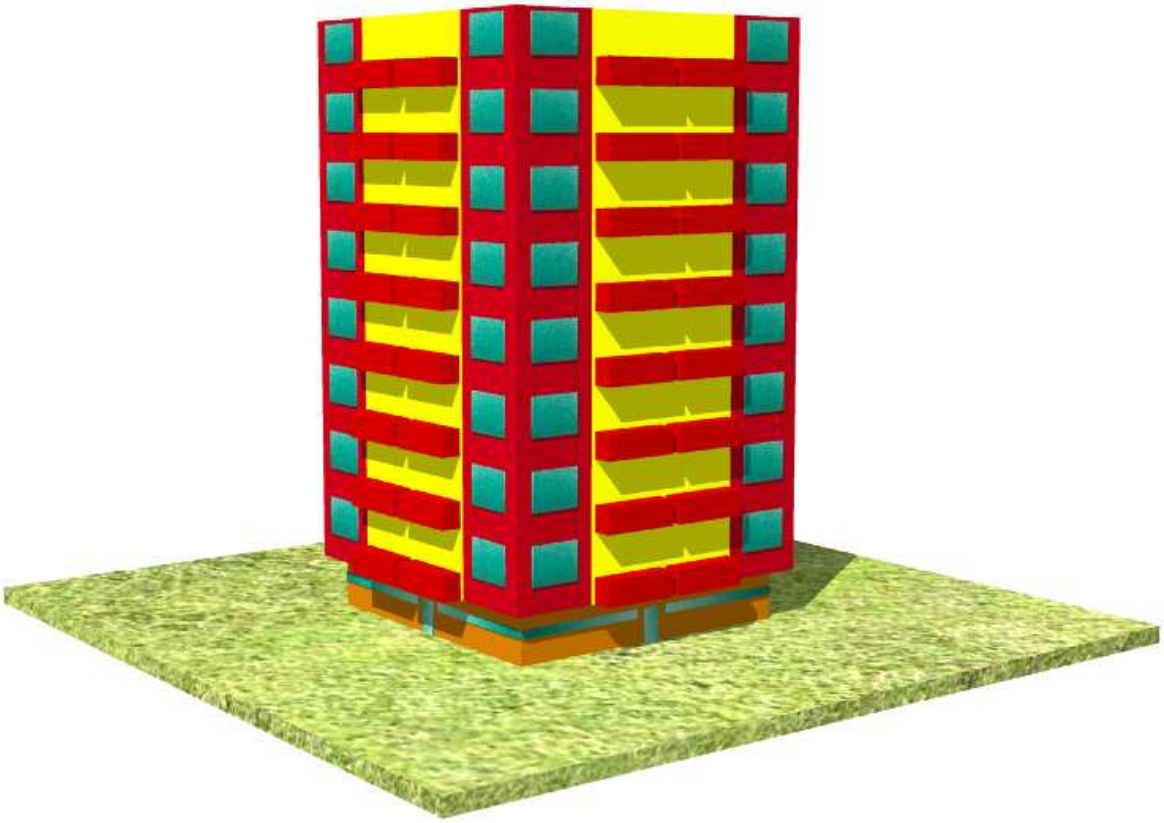
Obrázek A.5: $3 : patro \rightarrow div(xz, \dots)\{byt\}$



Obrázek A.6: $4 : byt \rightarrow div(x, \dots)\{pokoj, balkon\}$



Obrázek A.7: $5 : pokoj \rightarrow I(okno)$ a $6 : balkon \rightarrow I(zabradli)$



Obrázek A.8: $7 : prizemi \rightarrow Split(x, \dots)\{sklep, dvere, sklep\}$, $8 : sklep \rightarrow I(okno)$ a $9 : dvere \rightarrow I(vstup)$

Dodatek B

Interpret d0L-systémů

B.1 Uživatelské rozhraní

Implementovaný interpret deterministických bezkontextových L-systémů se skládá ze dvou oken - editačního a interpretačního.

Editace L-systému

Editační okno obsahuje prostor pro tvorbu a editaci L-systémů v textovém režimu. Popis struktury textového souboru je v 5.1 a příklad textového souboru je v dodatku B.2.

Toto okno dále obsahuje tlačítko `Analyze`, po jeho stisknutí proběhne lexikální a syntaktická analýza vytvořeného L-systému. Pokud jsou analýzy úspěšné, je povoleno generování řetězců (viz dále), nejsou-li analýzy úspěšné, je vypsáno chybové hlášení.

Generování řetězců probíhá v levé části editačního okna. Je možné zde nastavit počet požadovaných iterací, přičemž průběžný výsledek je v podobě vygenerovaného řetězce zobrazován. Tlačítkem `Clear` lze smazat vygenerovaný řetězec a tím se vrátit zpět k axiomu. Tlačítko `Draw` slouží pro spuštění interpretace (viz B.1).

Další součástí tohoto okna je menu a stavový řádek. V menu lze nalézt položky pro otevírání a ukládání vytvořených L-systémů. Dále se zde nastavují dva parametry pro interpret. Prvním je volba požadovaného rozlišení a druhým je požadovaná barevná hloubka. Ve stavovém řádku jsou zobrazovány nastavené parametry interpretu, případné syntaktické chyby a informace o délce řetězce a o průběhu jeho generování.

Interpretace L-systému

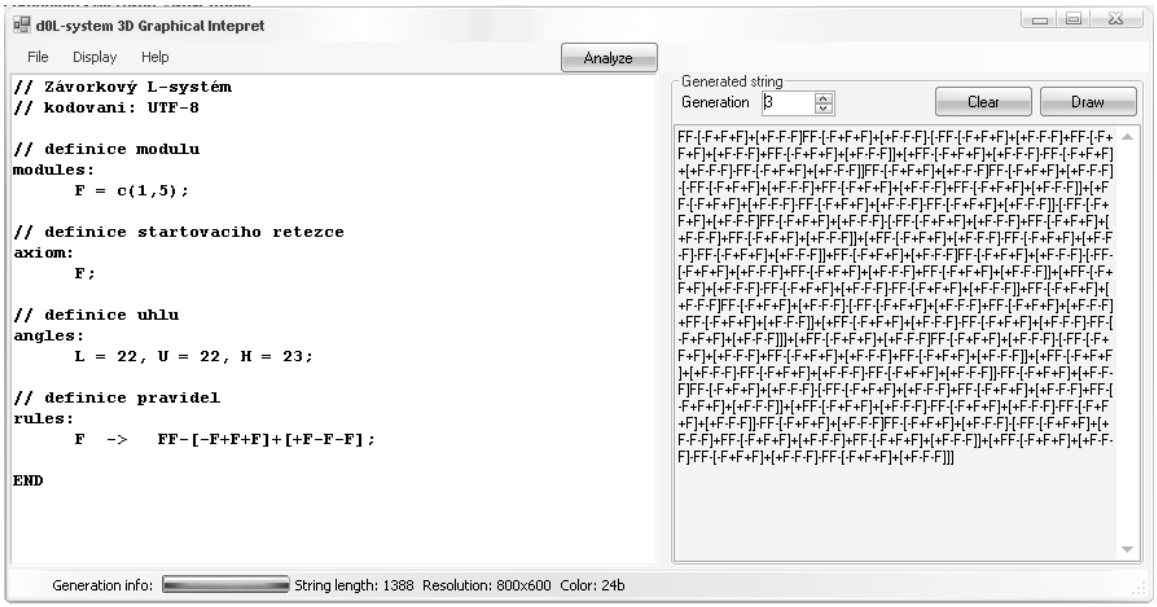
Jde o okno obsluhované knihovnou OpenGL a slouží ke zobrazení výsledku interpretace vygenerovaného řetězce. Ve stavovém řádku jsou vypsány informace o rozlišení okna (v pixelech), o barevné hloubce (v bitech) a také o minimálním, maximálním a aktuálním počtu snímků za sekundu (FPS). Stiskem klávesy `F1` je možné zobrazit nápovědu. Mezi zobrazením v okně a zobrazením na celou obrazovku lze přepínat klávesou `F2`. Klávesa `F3` slouží pro inicializaci čítačů počtu snímků za sekundu. Stiskem klávesy `F4` je možné skrýt nebo zobrazit stavový řádek. A klávesa `F5` slouží pro pořízení snímku obrazovky (screenshotu), ten bude uložen do adresáře aplikace.

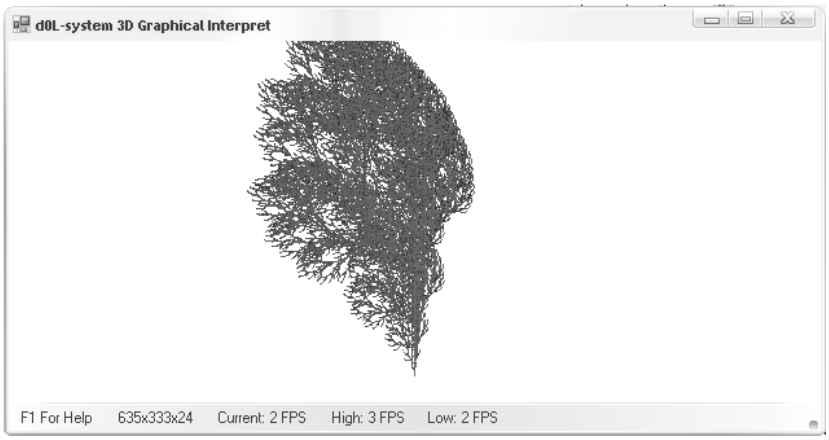
B.2 Příklad d0L-systému v textovém souboru

```
// Závorkový 3D L-systém

// definice modulu
modules:
        A, F = c(5,5), f = p(8), S, L, K= s(6), Q;

// definice startovaciho retezce
axiom:
        A;

// definice uhlu
angles:
        L = 22, U= 22, H = 23;

// definice pravidel
rules:
        A  ->   [\&FLK!A]/////[\&FLK!A]///////[\&FLK!A],
        F  ->   S/////F,
        S  ->   FL,
        K  ->   Q,
        L  ->   [^^{-f+f+f-|-f+f+f\}];
END
```

B.3 Přehled interpretace symbolů želvou

F	Posun vpřed o krok délky d. Změna stavu želvy na (x', y', α), kde $x' = x + d\cos\alpha$, $y' = y + d\sin\alpha$. Vykreslení úsečky mezi body $[x, y]$ a $[x', y']$.
f	Posun vpřed o krok délky d bez kreslení úsečky.
+	Rotace doleva o úhel δ. Změna stavu želvy na $(x, y, \alpha + \delta)$.
-	Rotace doprava o úhel δ. Změna stavu želvy na $(x, y, \alpha - \delta)$.
{	Označuje start polygonu. Mezi symboly { a } můžou ležet jen symboly pro rotaci želvy nebo symboly reprezentované úsečkou (viz 5.1).
}	Ukončuje generování polygonu. Od tohoto okamžiku lze generovat i symboly reprezentované objemovými tělesy.





Obrázek B.1: Příklad oken aplikace.

Dodatek C

Interpret parametrických, stochastických 2L-systémů s kontextovou podmínkou

C.1 Uživatelské rozhraní

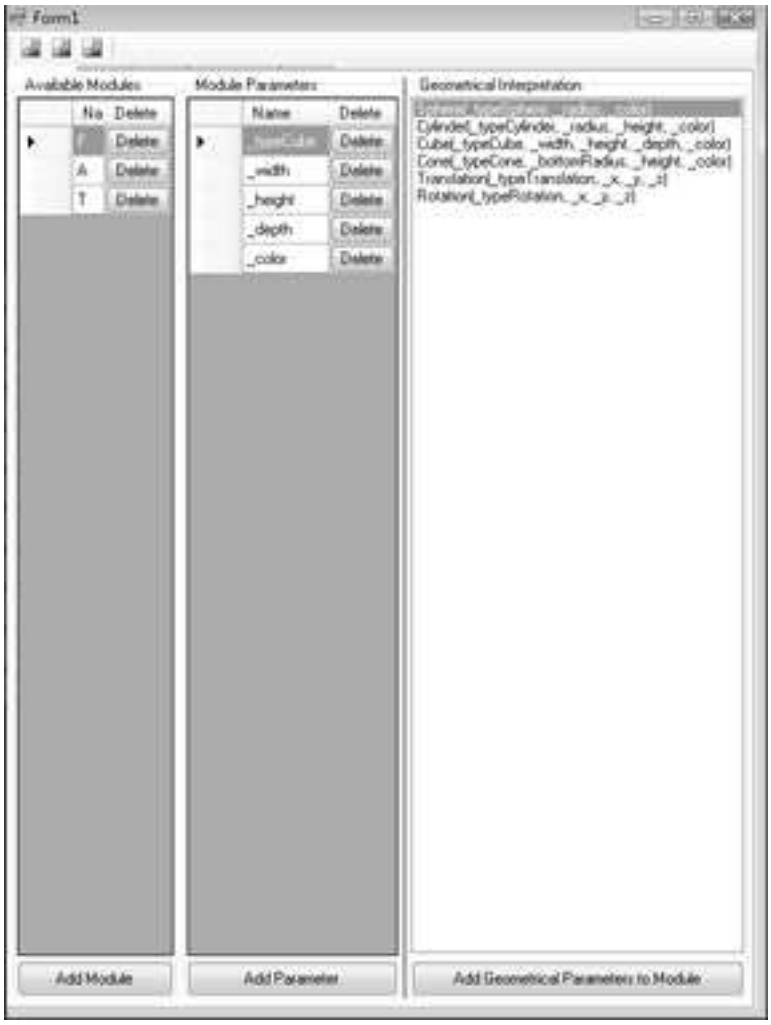
Uživatelské rozhraní navržené aplikace je přísně účelné a snaží se usnadnit definici L-systému a v neposlední řadě také slouží pro spuštění derivace v L-systému a následnému zobrazení výsledku interpretace. Grafické uživatelské rozhraní je složeno z následujících prvků.

- *Definice modulů a parametrů.* Slouží pro definici modulů dostupných v L-systému. Ve vztahu k definici 9 jde o abecedu V. Ke každému modulu lze přidávat dva typy parametrů:
  - *Geometrické parametry.* Lze zvolit z podporovaných typů geometrických primitiv, přičemž s každým z nich se pojí několik dalších parametrů, které udávají hodnoty potřebné pro geometrickou interpretaci (např. výška, šířka, hloubka, poloměr atd.).
  - *Obecné parametry.* Obecné parametry mohou sloužit k libovolnému účelu, zejména však slouží pro použití v kontextových a logických podmínkách.
- *Definice axiomu.* Slouží k definici axiomu L-systému, což odpovídá ω v definici 9. Jde o sestavení sekvence modulů a přiřazení hodnot jejich parametrům.
- *Definice přepisovacích pravidel.* Jde o okno aplikace obsahující nejvíce ovládacích prvků sloužících k definici přepisovacích pravidel L-systému, tj. P v definici 9. Okno je rozděleno na tyto části:
  - *Seznam definovaných pravidel.* Obsahuje seznam již definovaných pravidel a tlačítko pro vytvoření pravidla nového.
  - *Levý kontext vybraného pravidla.* Obsahuje seznam dostupných modulů a tlačítko pro přidání vybraného modulu do levého kontextu pravidla.
  - *Pravý kontext vybraného pravidla.* Obsahuje seznam dostupných modulů a tlačítko pro přidání vybraného modulu do pravého kontextu pravidla.
  - *Levá strana vybraného pravidla.* Obsahuje seznam dostupných modulů a tlačítko pro nastavení vybraného modulu jako levé strany pravidla.

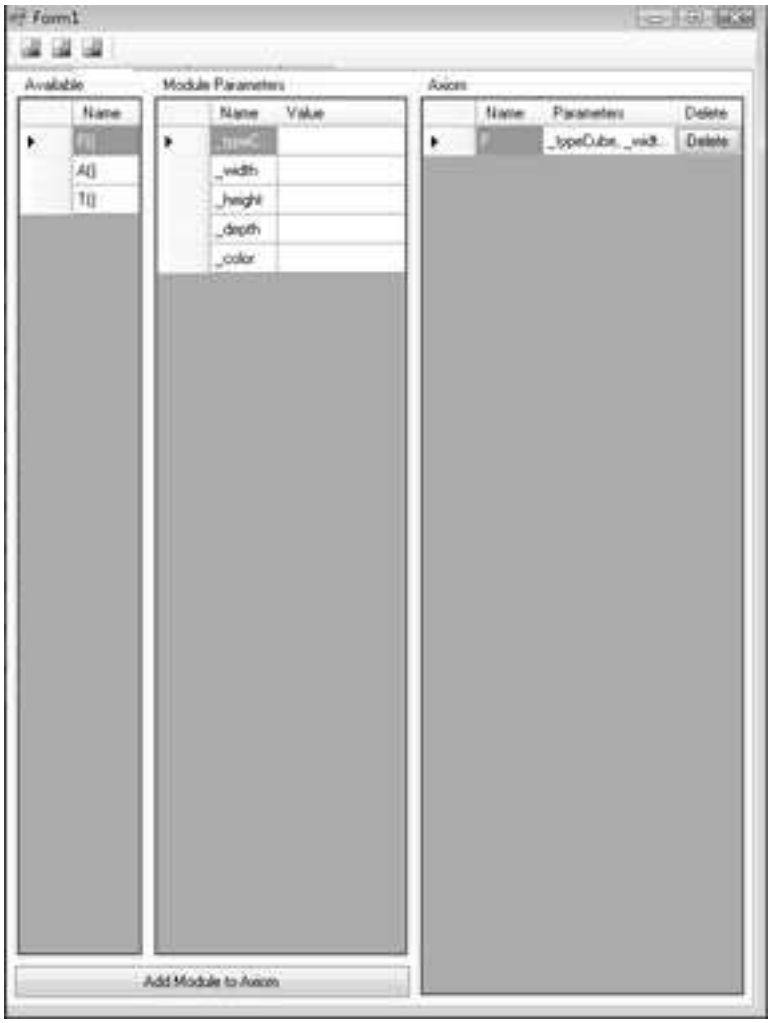
  - *Logická podmínka vybraného pravidla.* Obsahuje pole zobrazující logickou podmínku pravidla.
  - *Kontextová podmínka vybraného pravidla.* Obsahuje seznam všech položek kontextové podmínky pravidla.
  - *Pravá strana vybraného pravidla.* Obsahuje seznam modulů definovaných jako pravá strana pravidla.
  - *Pravděpodobnost vybraného pravidla.* Obsahuje pole pro nastavení pravděpodobnosti použití pravidla.
  - *Záložky pro definici podmínek a výrazů.* Pro větší přehlednost jsou pro definici logické a kontextové podmínky a pro definici výrazů určujících hodnoty parametrů modulů na pravé straně pravidla dostupné samostatné záložky:
    * *Logická podmínka.* Obsahuje prvky pro zadávání operátorů a operandů logického výrazu. Dostupné operátory jsou popsány v 6.2.1 a operandy lze zadávat buď jako konstantní hodnotu nebo jako formální parametr tak, jak je uvedeno v 6.2.2.
    * *Kontextová podmínka.* Obsahuje seznam dostupných modulů, z nichž lze skládat položky kontextové podmínky.
    * *Pravá strana pravidla.* Podobně jako záložka pro logickou podmínku obsahuje prvky pro zadávání výrazů, tentokrát však aritmetických, a dále prvky pro přiřazení těchto výrazů k parametrům dostupných modulů.

- *Generování řetězců a jejich interpretace.* Okno pro generování řetězců obsahuje popis definovaného L-systému, tlačítko, které provede výpočet následující generace podle popisu v 6.2. Dále obsahuje seznam již vypočtených generací, z nichž každou lze interpretovat stiskem tlačítka.

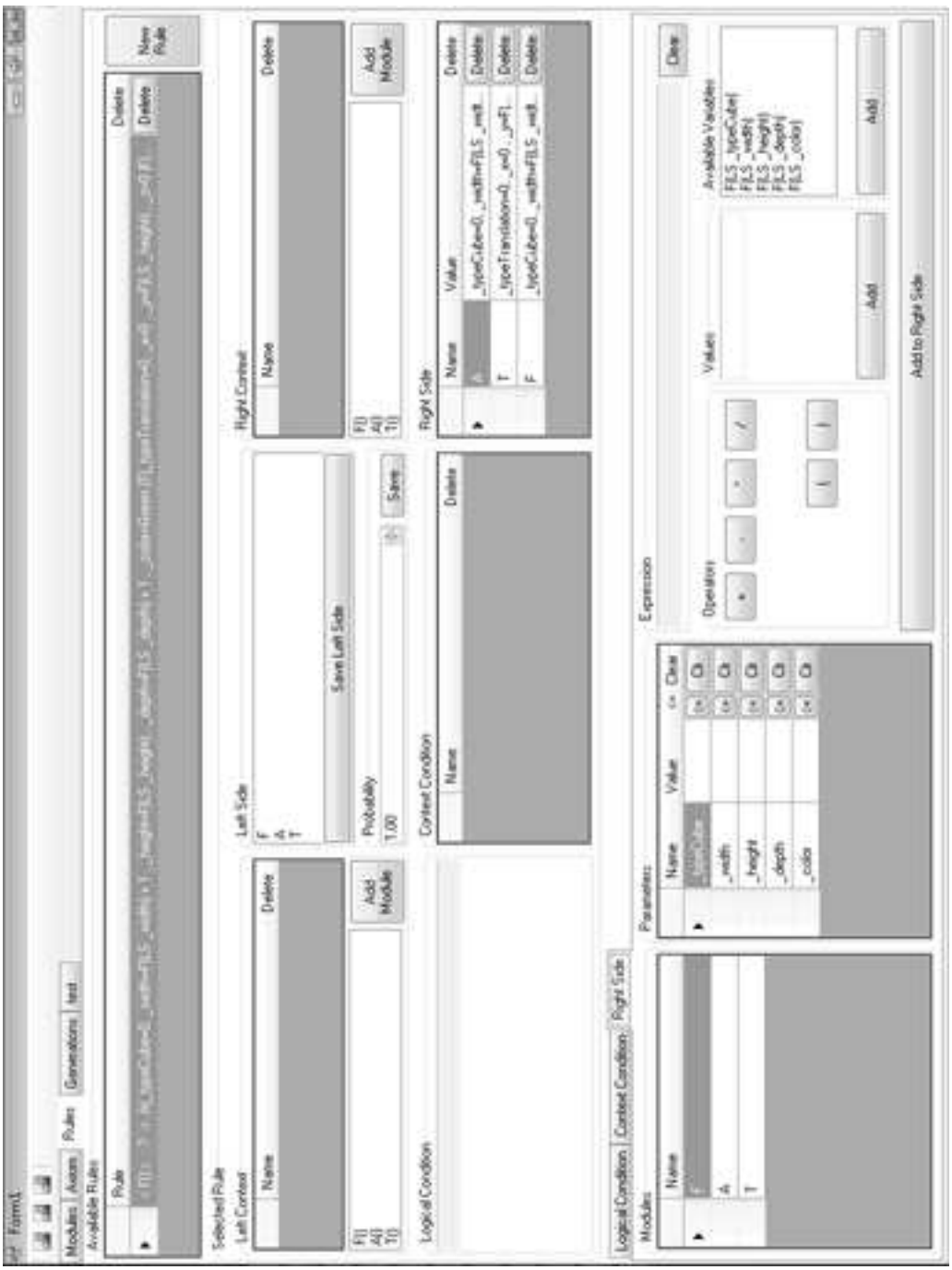
- *Zobrazení 3D scény.* Při interpretaci je vygenerován VRML popis prostorové scény tak, jak je uvedeno v 6.3.2. Následně je vygenerovaný soubor otevřen v aplikaci asociované se soubory s koncovkou *.wrl*.



Obrázek C.1: Okno pro definici modulů L-systému, jejich parametrů a jejich geometrické interpretace. Tento L-systém obsahuje moduly F a A interpretované jako krychle a modul T interpretovaný jako posun v souřadném systému.

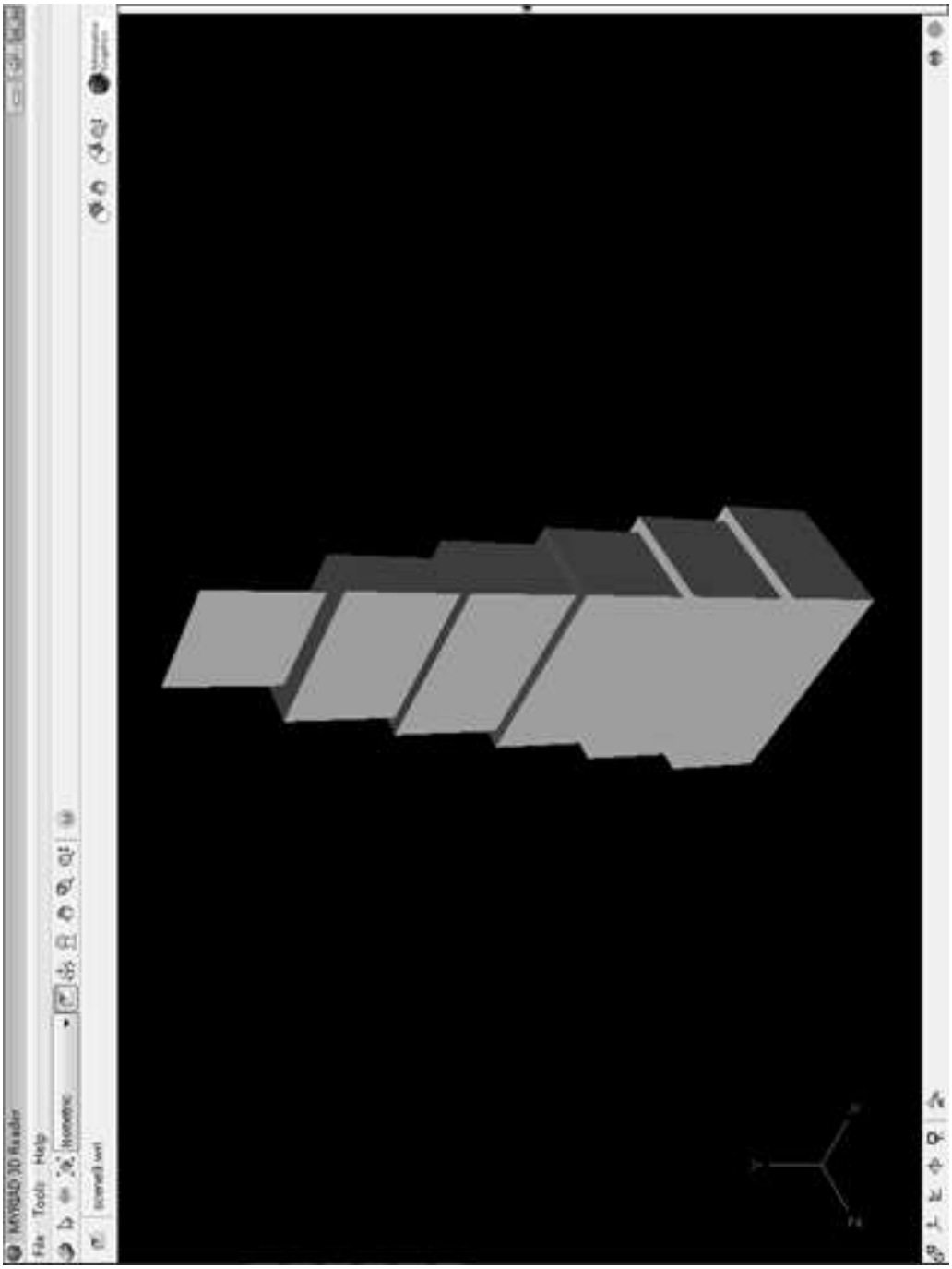


Obrázek C.2: Okno pro definici axiomu L-systému. Axiomem tohoto L-systému je modul F s parametry *width=10*, *height=5* a *depth=5*.



Obrázek C.3: Okno pro definici pravidel L-systému. Ve horní třetině seznam definovaných pravidel, uprostřed jednotlivé části vybraného pravidlo, ve spodní třetině záložka pro definici aritmetických výrazů. Jediné pravidlo v tomto L-systému je ve tvaru $F() \rightarrow A()T()F()$, hodnoty parametrů zde neuvádíme.

Obrázek C.4: Okno pro derivace v L-systému. V horní části definovaný L-systém, v dolní části seznam prvních pěti generací.



Obrázek C.5: Příklad interpretace páté generace.