L – systémy

(Case Study z predmetu Evolučné algoritmy)

Richard Bartelt
4inf

21.12.2002

Úvod

Lindenmeyerove systémy (L-systémy) sú matematický formalizmus, ktorý bol predstavený v roku 1968 biológom Aristidom Lindenmeyerom. Pôvodná motivácia na ich vytvorenie bola biologického charakteru. Lindenmeyer ich použil na formálny zápis a simuláciu delenia buniek mnohobunkových organizmov, a teda aj ich biologického rastu. Neskôr sa našlo viacero možností ich využitia, najmä v oblasti počítačovej grafiky (Smith 1984; Prusinkiewicz a Hanan 1989; Prusinkiewicz a Lindenmayer 1991). Dve hlavné aplikačné oblasti sú generovanie fraktálov a realistické modelovanie rastlín.

Základná idea, na ktorej L-systémy spočívajú, je tzv. prepisovanie. Pomocou tohto formalizmu je možné definovať zložité objekty postupným nahrádzaním niektorých častí jednoduchého objektu inými jednoduchými objektami podľa určitých (tzv. produkčných alebo prepisovacích) pravidiel. Prepisovanie sa pritom môže vykonávať aj rekurzívne.

Najčastejšie používané a najviac skúmané prepisovacie systémy pracujú na reťazcoch znakov. Pozornosť na takéto prepisovacie systémy upriamila najmä Chomského práca o formálnych gramatikách (1957). Následne nastalo obdobie veľkého záujmu o syntax, gramatiky a ich využitie v informatike, čo položilo základy pre zrod odvetvia informatiky, zaoberajúceho sa formálnymi jazykmi.

Lindenmeyerove systémy využívajú trocha iný prístup k prepisovaniu reťazcov ako klasické Chomského gramatiky. Tieto pri prechode od starého reťazca k novému vykonajú transformáciu podľa niektorého produkčného pravidla iba na jednom mieste starého reťazca, to znamená, že prepisovanie nastáva sekvenčne. L-systémy prepisujú paralelne, teda na vytvorenie nového reťazca aplikujú produkčné pravidlá na všetky pozície starého reťazca naraz. Tento spôsob prepisovania vyplýva práve z vyššie uvedenej biologickej motivácie pre vznik L-systémov, z potreby simulovať delenie buniek – v určitom čase sa totiž môže deliť aj viac buniek naraz.

Z tohto je teda zrejmé, že L-systémy sú akási forma paralelnej formálnej gramatiky. Existujú aj iné typy paralelných formálnych gramatík, ktoré využívajú čiastočne aj iné formy paralelizmu. Paralené formálne gramatiky sú však časťou teórie formálnych jazykov, ktorá je veľmi málo preskúmaná.

Formálna definícia L-systémov

Budeme sa teraz venovať definovaniu L-systémov tak, ako ich definuje teória formálnych jazykov. V tejto teórii sú na definovanie gramatík potrebné štandardné 4 definície. Najskôr treba však definovať najzákladnejšie pojmy:

Definícia: Abeceda Σ je konečná množina symbolov (písmen).

Definícia: Slovo nad abecedou Σ je konečná postupnosť symbolov zo Σ.
Prázdne slovo ε je prázdna postupnosť znakov.

Definícia: Jazyk L je množina slov nad danou abecedou Σ.

Definícia: Zreťazenie jazykov je operácia nad jazykmi:
$L_1L_2 = \{uv \mid u \in L_1 \wedge v \in L_2\}$

Definícia: Iterácia jazyka (Kleeneho *) je operácia nad jazykmi:
$L^* = L^0 \cup L^1 \cup L^2 \cup ... \cup L^k \cup ...$, kde $L^0 = \{ \varepsilon \}$, $L^{i+1} = LL^i$ pre $\forall i \in \mathbf{N}$.

Definícia: Kladná iterácia jazyka (Kleeneho +) je operácia nad jazykmi:
$L^+ = L^1 \cup L^2 \cup ... \cup L^k \cup ...$, kde $L^1 = L$, $L^{i+2} = LL^{i+1}$ pre $\forall i \in \mathbf{N}$.

Poznámka: Symbol **N** označuje aj v ďalšom množinu prirodzených čísel.

Vyššie definované pojmy tvoria základ pre korektnú formálnu definíciu Lindenmeyerových systémov. Najskôr pomocou štyroch klasických definícií definujeme najjednoduchší variant L-systémov, tzv. 0L-systémy. Neskôr spomenieme aj ďalšie typy L-systémov, pričom uvedieme už iba ich odlišnosti od 0L-systémov.

Definícia: Nedeterministickým 0L-systémom nazývame trojicu G = (N, P, w), kde
N je abeceda
$w \in N^+$ je počiatočné slovo resp. axióm
$P \subseteq N \times N^*$ je konečná množina pravidiel taká, že $(\forall a \in N)(\exists\, a \rightarrow u \in P)$

Poznámka: Prvky P, ktoré sú usporiadané dvojice (a,u), $a \in N$, $u \in N^*$ budeme označovať $a \rightarrow u$.

Definícia: Krokom odvodenia v 0L-systéme G nazývame reláciu $\Rightarrow$ na N* definovanú takto:
$u \Rightarrow_G v \Leftrightarrow$ keď: $u = a_1a_2...a_n$, $(\forall i)\ a_i \in N$, $n \geq 1$
$v = v_1v_2...v_n$, $(\forall i)\ v_i \in N^*$
$(\forall i)\ a_i \rightarrow v_i \in P$.

Poznámka: Označenie "$u \Rightarrow^i_G v$" bude znamenať, že v danom L-systéme G zo slova u po i krokoch odvodenia môže vzniknúť slovo v, označenie "$u \Rightarrow^*_G v$" bude znamenať, že v danom L-systéme G existuje odvodenie slova v zo slova u na nejaký (nešpecifikovaný) počet krokov odvodenia (môže byť aj 0!).

Definícia: Jazyk generovaný 0L-systémom G je množina slov $L(G) = \{u \in N^* \mid w \Rightarrow^*_G u\}$.

Poznámka: Boli spomínané 4 definície, 4. by mala byť definícia vetnej formy. Tento pojem je však u L-systémov totožný s pojmom slova z L(G), preto nie je potrebné ho definovať. Ak sa niekde v texte vyskytne označenie vetná forma, je tým myslený nejaký prvok množiny L(G).

Týmto máme teda formálne definovaný 0L-systém. V ďalšom spomenieme niekoľko modifikácií tohto formalizmu.

Definícia: Deterministický 0L-systém (D0L) je 0L-systém, kde $\forall a \in N$ existuje práve jedno pravidlo $a \rightarrow u \in P$.

Definícia: Bez-ε 0L-systém (propagating, P0L) je 0L-systém, kde $P \subseteq N \times N^+$ je konečná množina pravidiel.

Poznámka: Konjunkciou predchádzajúcich dvoch definícií sú definované deterministické propagating 0L-systémy (PD0L).

Definícia: Stochastický 0L-systém je 0L-systém, pre ktorý krok odvodenia je definovaný takto:
$u \Rightarrow_G v \Leftrightarrow$ keď: $u = a_1a_2...a_n$, $(\forall i)\ a_i \in N$, $n \geq 1$
$v = v_1v_2...v_n$, $(\forall i)\ v_i \in N^*$
$(\forall i)\ a_i \rightarrow v_i \in P$
pričom ak $P_a = \{a \rightarrow u_1, a \rightarrow u_2, ...a \rightarrow u_k\} = P \cap \{a \rightarrow u \mid u \in N^*\}$, kde $\forall j \in \{1, 2, ... k\}$ $u_j \in N^*$, a ak $a_i = a$, tak pravdepodobnosť, že $v_i = u_j$, sa rovná $1/k$ pre každé $j \in \{1, 2,...k\}$.

Definícia: Stochastický 0L-systém s váhovanými pravidlami (S0L) je štvorica G = (N, P, w, h), kde $h: P \rightarrow \mathbf{N}$ je hodnotová funkcia a trojica (N, P', w) je stochastický 0L-systém, pričom P' je multimnožina taká, že každé $a \rightarrow u \in P$ sa v P' nachádza práve h(a→u)-krát.

My sa budeme v ďalšom zameriavať na posledný definovaný typ 0L-systémov (S0L), keďže tieto je možné viacmenej presne simulovať na reálnom počítači (narozdiel od nedeterministických 0L-systémov) a takisto je aj nimi možné viacmenej presne simulovať ostatné typy 0L-systémov. Preto ak sa v ďalšom vyskytne výraz "0L-systém" resp. "L-systém", myslí sa implicitne S0L-systém.

Okrem 0L-systémov existujú aj 1L-systémy, 2L-systémy atď. Od 0L-systémov sa líšia tým, že na ľavej strane pravidiel sa nevyskytuje iba jeden symbol, ale viac symbolov, pri aplikácii pravidiel sa teda nezohľadňuje iba jeden znak, ale aj jeho okolie. Táto vlastnosť sa v teórii jazykov označuje pojmom kontextovosť. 0L-systémy sú bezkontextové, tie ostatné sú kontextové.

Korytnačia grafika

Máme teda definované L-systémy, vieme ako pracujú – ale ako sa toto dá využiť pre náš cieľ, tvorbu rastlín a simuláciu biologických entít? Je to v podstate dosť očividné: jednotlivé znaky abecedy N budú predstavovať bunky, ktoré sa rozmnožujú a vyvíjajú podľa pravidiel z P, pričom sa začína z počiatočných buniek reprezentovaných axiómom w. Toto je síce pekné, ale výsledky asi nebudú príliš názorné, budú to len akési postupnosti znakov. Bude tu síce nepochybne akási štruktúra, ktorá bude závisieť od pravidiel, ale v konečnom dôsledku to budú len nejaké reťazce, ktoré nám veľa nepovedia. Zišiel by sa nám akýsi vizualizačný aparát, pomocou ktorého by bolo možné vytvorené reťazce interpretovať a nejakým spôsobom graficky znázorňovať. A práve túto funkčnosť nájdeme v počítačovej grafike pod označením korytnačia grafika.

Korytnačiu grafiku vymyslel Seymour Papert ako systém prekladania postupností symbolov na pohyby automatu („korytnačky") po grafickom displeji. Pôvodná idea bola vytvoriť programovateľný objekt, pomocou ktorého by sa deti mohli učiť geometricky rozmýšľať. Tento systém však poskytuje aj ideálny spôsob geometrickej interpretácie L-systémov.

Základným pojmom v korytnačej grafike je korytnačka. Je to akýsi abstraktný objekt, ktorý má svoju pozíciu a orientáciu, t.j. je určený bodom a vektorom. Korytnačka sa dokáže pohybovať dopredu a dozadu, pričom za sebou zanecháva nakreslenú čiaru, a dokáže meniť svoju orientáciu. Spočiatku sa definovala 2D korytnačka, ktorá postačuje na vykreslovanie 2D fraktálov a 2D modelov rastlín. S rozvojom 3D grafiky sa definovala aj 3D korytnačka, ktorá pracuje v trojrozmenrnom priestore. Postupne sa tiež rozvíjala aj sada príkazov pre korytnačku, aby bolo umožnené realistickejšie znázornenie rastlín, prípadne z estetických dôvodov. Pribudli príkazy na pohyb korytnačky bez kreslenia čiary, zmenu farby a hrúbky kreslenej čiary, ukladanie pozície a atribútov korytnačky do zásobníka, kreslenie vyplnených oblastí atď.

Ako to všetko spojiť?

Ako teda budeme interpretovať L-systém pomocou korytnačky? Je to veľmi jednoduché. Najskôr zvolíme pre L-systém požadovaný počet iterácií (občas sa iterácie L-systému označujú pojmom generácie, najmä keď nimi modelujeme biologické entity). Nech je tento počet n. Následne necháme L-systém urobiť na axióme n krokov odvodenia. Získame nejakú vetnú formu, tzv. n-tú generáciu daného L-systému. Pre korytnačku určíme nejakú počiatočnú pozíciu, základný krok, o ktorý sa bude posúvať a základný uhol, o ktorý sa bude otáčať. Vetnú formu n-tej generácie, ktorú sme predtým získali, odovzdáme korytnačke ako akýsi „program“. Korytnačka potom ide po znakoch tejto vetnej formy a interpretuje ich ako príkazy, tieto príkazy vykonáva a pritom „kreslí L-systém“.

Je jasné, že ak chceme, aby korytnačka niečo kreslila, L-systém musí byť špeciálne definovaný. Najskôr treba všetky príkazy korytnačky označiť nejakými znakmi (podľa možnosti rôznymi). Následne by sme mali tieto znaky použiť aj v L-systéme tak, aby sa dostali do výslednej vetnej formy. V L-systéme sa samozrejme môžu vyskytovať aj znaky, ktoré neoznačujú príkazy. Tieto znaky korytnačka pri vykonávaní programu jednoducho ignoruje.

Existuje štandardné označenie základných korytnačích príkazov, ktorého sa pridŕža prakticky každý program na vizualizáciu L-systémov. Táto konvencia rezervuje niekoľko symbolov, ktoré sa umiestnia do abecedy L-systému, na špeciálne použitie. Symboly sú takéto:

- **F** : Tento znak označuje, že korytnačka sa má posunúť dopredu o predtým zadaný krok a pritom za sebou nakresliť čiaru, zo starej pozície do pozície, na ktorú sa posunie. Korytnačka sa posúva v smere svojej aktuálnej orientácie.
- **+** : Tento znak znamená otočenie korytnačky (zmenu jej aktuálnej orientácie) smerom doľava o predtým zadaný uhol. Pri tomto príkaze sa nič nekreslí.
- **–** : Tento znak znamená otočenie korytnačky (zmenu jej aktuálnej orientácie) smerom doprava o predtým zadaný uhol. Pri tomto príkaze sa nič nekreslí.

Pri 3D korytnačke sa príkazy **+** a **–** chápu ako otočenie okolo osi y a existujú ešte ďalšie 4 príkazy otáčania okolo ostatných osí (samozrejme, že sa berie do úvahy lokálna súradicová sústava korytnačky).

Teraz sme už schopní definovať jeden z najzákladnejších a najznámejších 2D fraktálov, Kochovej krivku. Uvedieme ju ako názorný príklad pre L-systém interpretovaný korytnačkou.

L-systém pre Kochovej krivku vyzerá veľmi jednoducho:
N = {F, +, -}, w = F, P = {F→F+F--F+F}.
Máme teda abecedu iba z korytnačích príkazov, axióm označuje jednoduchú čiaru a máme jedno produkčné pravidlo. Pre korytnačku definujeme nejaký krok (môže byť v podstate ľubovoľný) a uhol otáčania bude 60°. Tu je niekoľko prvých generácií a príslušné vetné formy:

0. gen.: F

1. gen.: F+F--F+F


2. gen.: F+F--F+F+F+F--F+F--F+F--F+F+F+F--F+F 3. gen.

Dĺžka vetnej formy zrejme narastá exponenciálne. Aby sa zachovala priestorová veľkosť krivky, delí sa základný krok pre k-tu generáciu číslom 3^k.

Uzátvorkované L-systémy

Máme teda metódu ako interpretovať a graficky znázorňovať L-systémy. Vyššie popísaný spôsob je úplne postačujúci na vykreslovanie krivkových fraktálov, ktoré pozostávajú z jednej lomenej čiary. My však chceme L-systémy využívať všeobecnejšie, na generovanie rastlín. A keďže rastliny bohužiaľ väčšinou nemajú tú vlastnosť, že by boli nakresliteľné lomenou čiarou "jedným ťahom", budeme musieť naše doterajšie formalizmy nejakým spôsobom rozšíriť.

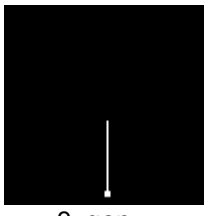
Toto rozšírenie prezentoval vo svojej práci už aj sám Lindenmeyer. Navrhol notáciu stromov z teórie grafov pomocou reťazcov so zátvorkami. Jeho idea bola založená na formálnom popise vetvenia štruktúr, ktoré môžeme nájsť u väčšiny rastlín, pomocou L-systémov. Tento prístup sa neskôr tiež zahrnul do formalizmu na geometrické interpretovanie L-systémov, a síce ako dva nové príkazy pre korytnačku.

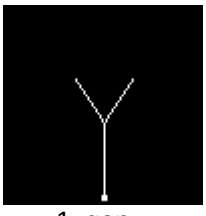
Do abecedy L-systémov pribudli dva nové symboly "[" a "]", ktoré vymedzujú „vetvy". Tieto symboly sa začali používať aj ako štandardizované príkazy pre korytnačku. Korytnačke sa priradila klasická dátová štruktúra zásobník so štandardnými operáciami vkladania a vyberania z vrchu. Pomocou tohto zásobníka potom korytnačka interpretuje dva nové príkazy:

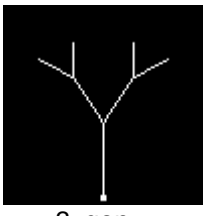
- **[** : Tento symbol znamená, že na vrch zásobníka sa uložia všetky aktuálne parametre korytnačky, t.j. pozícia, orientácia, dĺžka kroku, uhol otočenia, farba a hrúbka kreslených čiar a iné.
- **]** : Tento symbol znamená, že sa z vrchu zásobníka vezmú všetky parametre, na ktoré sa potom korytnačka nastaví. Korytnačka sa presunie na uloženú pozíciu (pri zmene pozície sa nekreslí čiara), nastaví sa na príslušnú orientáciu, nastavia sa dĺžka kroku, uhol otočenia, farba a hrúbka kreslených čiar a pod.

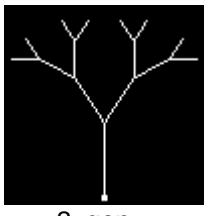
Vďaka tejto úprave je možné vykreslovať už prakticky hocičo, vrátane napríklad binárnych stromov danej výšky podľa definície v teórii grafov. Uvedieme si túto konštrukciu ako názorný príklad.

L-systém pre binárne stromy je opäť veľmi jednoduchý:
N = {F, +, -}, w = F, P = {F→F[+F][-F]}
Máme teda abecedu iba z korytnačích príkazov, axióm označuje jednoduchú čiaru a máme jedno produkčné pravidlo. Pre korytnačku definujeme nejaký krok (môže byť v podstate ľubovoľný) a uhol otáčania bude 60°. Číslo generácie L-systému v tomto prípade zodpovedá výške binárneho stromu (pod výškou rozumieme výšku ako ju chápe teória grafov). Tu je niekoľko generácií:


0. gen.


1. gen.


2. gen.


3. gen.

Aby sa stromy lepšie zobrazili, použilo sa tu aj skracovanie kroku - pri každom vnorení sa krok vynásobil číslom 0.7.

L-System Laboratory

Teraz už máme vlastne všetko, čo sme z teórie potrebovali, aby sme mohli začať modelovať rastliny a ich vývin. Existuje niekoľko programov, ktoré interpretujú a graficky znázorňujú L-systémy. Program, ktorý je súčasť tejto práce, sa nazýva L-System Laboratory. Tento program poskytuje veľkú paletu možností na definovanie, vizualizáciu a dokonca vytváranie mutácií 0L-systémov v podstate ľubovoľného druhu. Program pracuje v zásade podľa popísanej schémy, na zadanom L-systéme vykoná daný počet krokov odvodenia, vygenerované slovo následne podá 3D korytnačke, ktorá ho použije ako svoj program.

Rozdielom oproti predchádzajúcemu teoretickému popisu je, že korytnačka počas interpretácie vetnej formy nekreslí na obrazovku, ale svoje pozície ukladá do akejsi pomocnej pamäte ako postupnosť bodov, ktorá sa následne vykresluje na obrazovku pomocou grafického systému OpenGL. Tento prístup je zvolený preto, aby sa následné transformácie pohľadu na vykreslený L-systém vykonávali rýchlejšie. Je totiž rýchlejšie vykresliť postupnosť bodov ako znovu interpretovať reťazec, prípadne ho znovu generovať.

Ďalší rozdiel je v možnostiach korytnačky – pre korytnačku je totiž definovaných oveľa viac príkazov ako bolo doteraz uvedených. Popis príkazov pre korytnačku sa nachádza v ďalšej časti.

Korytnačie príkazy

Korytnačka v programe podporuje dosť veľkú sadu príkazov. Na tomto mieste sú uvedené všetky príkazy aj s podrobným popisom ich činnosti.

Korytnačka v sebe zahŕňa niekoľko parametrov a dátových štruktúr. Sú to:

- *krok* – číslo označujúce dĺžku, o ktorú sa korytnačka pohne pri príkazoch pohybu
- *uhol* – číslo (v stupňoch), ktoré označuje uhol, o ktorý sa korytnačka otáča
- *farba* – index farby (0-19), ktorou korytnačka kreslí čiary
- *hrúbka* – hrúbka čiary kreslenej korytnačkou, v bodoch-1 (0 označuje 1-bodovú čiaru atď.)
- *stack* – zásobník používaný pri uzátvorkovaných L-systémoch (pozri vyššie)
- *pstack* – zásobník používaný pri kreslení polygónov (pozri nižšie)

Tu sú korytnačie príkazy:

F : Štandardný príkaz pohybu vpred s kreslením, korytnačka sa pohne o *krok* dopredu (ak je *krok* záporný, tak dozadu) a stará a nová pozícia sa spoja čiarou.

f : Príkaz pohybu vpred bez kreslenia, korytnačka sa pohne o *krok* dopredu (ak je *krok* záporný, tak dozadu), ale nekreslí čiaru.

H : To isté ako príkaz F, ale hýbe sa o (*krok*/2).

h : To isté ako príkaz f, ale hýbe sa o (*krok*/2).

G : Tento príkaz spôsobí, že sa zaznamená aktuálna pozícia korytnačky do pomocnej pamäte, korytnačka sa však nepohne z miesta. Ak predtým zaznamenaná pozícia korytnačky bola iná ako práve zaznamenaná pozícia, spoja sa tieto dve pozície čiarou.

g : Príkaz pohybu vpred o *krok*, čiara sa nekreslí a nová pozícia sa nezaznamená do pomocnej pamäte (rozdiel oproti f).

+ : Štandardný príkaz otočenia vľavo, korytnačka sa otočí vľavo o *uhol* (ak je *uhol* záporný, tak vpravo). Nič sa nekreslí. Otočenie nastáva okolo osi y lokálneho súradnicového systému korytnačky.

- : Štandardný príkaz otočenia vpravo, korytnačka sa otočí vpravo o *uhol* (ak je *uhol* záporný, tak vľavo). Nič sa nekreslí. Otočenie nastáva okolo osi y lokálneho súradnicového systému korytnačky.

^ : Príkaz otočenia hore, korytnačka sa otočí hore o *uhol* (ak je *uhol* záporný, tak dole). Nič sa nekreslí. Otočenie nastáva okolo osi z lokálneho súradnicového systému korytnačky.

& : Príkaz otočenia dole, korytnačka sa otočí dole o *uhol* (ak je *uhol* záporný, tak hore). Nič sa nekreslí. Otočenie nastáva okolo osi z lokálneho súradnicového systému korytnačky.

< : Príkaz rolovania vľavo, korytnačka roluje vľavo o *uhol* (ak je *uhol* záporný, tak vpravo). Nič sa nekreslí. Otočenie nastáva okolo osi x lokálneho súradnicového systému korytnačky (v smere pohybu vpred).

> : Príkaz rolovania vpravo, korytnačka roluje vpravo o *uhol* (ak je *uhol* záporný, tak vľavo). Nič sa nekreslí. Otočenie nastáva okolo osi x lokálneho súradnicového systému korytnačky (v smere pohybu vpred).

| : Príkaz otočenia o 180°. Nič sa nekreslí. Otočenie nastáva okolo osi y lokálneho súradnicového systému korytnačky.

% : Príkaz rolovania o 180°, korytnačka sa v podstate "otočí na chrbát". Nič sa nekreslí. Otočenie nastáva okolo osi x lokálneho súradnicového systému korytnačky (v smere pohybu vpred).

[: Začiatok vetvy. Do záslobníka *stack* sa uloží aktuálna pozícia, orientácia, *krok*, *uhol*, *farba* a *hrúbka* korytnačky. Inak sa nič nestane.

] : Koniec vetvy. Zo záslobníka *stack* sa vytiahne pozícia, orientácia, *krok*, *uhol*, *farba* a *hrúbka* korytnačky. Následne korytnačka prevezme všetky tieto atribúty. Ak nastane zmena pozície, čiara sa nekreslí.

{ : Začiatok polygónu. Program umožňuje do L-systémov zakomponovať vyplnené mnohouholníky (napr. na modelovanie listov, kvetov a pod.). Pri tomto príkaze sa do zásobníka *pstack* uloží presne to isté ako pri príkaze [do zásobníka *stack*. Taktiež sa zmenia pravidlá aplikovania niektorých príkazov, konkrétne:

- Po príkaze] sa nezaznamená nová pozícia do pomocnej pamäte. Ak pozíciu chceme napriek tomu zaznamenať, musíme použiť príkaz G.
- Príkazy bez kreslenia f a h majú rovnaký efekt ako príkazy F a H. Ak chceme korytnačku posúvať bez toho aby sa definovali nové vrcholy polygónu, treba používať príkaz g.
- Zmeny *farby* a *hrúbky* (pozri nižšie) nemajú žiadny efekt.

} : Koniec polygónu. Zo zásobníka *pstack* sa vyberú atribúty korytnačky zo začiatku polygónu a vykoná sa to isté ako pri príkaze]. Polygón vznikne spojením posledného definovaného vrcholu vnútri zátvoriek { a } a prvého bodu polygónu (pozície korytnačky pri interpretácii príkazu {). Tento polygón sa vyplní danou *farbou*, ktorá bola aktuálna počas interpretácie príkazu {.

Poznámka: Príkazy F, f, H, h, g, +, -, ^, &, <, > môžu mať argument. V takom prípade sa ignoruje *krok* resp. *uhol*, a namiesto toho sa použije zadaný argument. Argument môže byť ľubovoľné reálne číslo, používa sa desatinná bodka (.) a argument treba uviesť bezprostredne za príkazom (napr. F(20.5), +(40), <(-10.778) atď.).
Nasledovné príkazy tiež všetky umožňujú zadanie argumentu (väčšina bez argumentu nemá efekt).

! : Definovanie *kroku*. Nastavuje *krok* korytnačky na hodnotu argumentu. Ak argument nie je uvedený, nemá efekt.

? : Definovanie *uhla*. Nastavuje *uhol* korytnačky na hodnotu argumentu. Ak argument nie je uvedený, nemá efekt.

. : Definovanie *farby*. Nastavuje *farbu* korytnačky na hodnotu argumentu. Ak argument nie je uvedený, index farby sa zvýši o 1 (ak je potom viac ako 19, vráti sa na 0).

$: Definovanie *hrúbky*. Nastavuje *hrúbku* korytnačky na hodnotu argumentu. Ak argument nie je uvedený, nemá efekt.

***** : Vynásobenie *kroku*. Vynásobí *krok* korytnačky hodnotou argumentu a výsledok uloží ako nový *krok*. Ak argument nie je uvedený, nemá efekt.

: Pripočítanie ku *kroku*. Pripočíta ku *kroku* korytnačky hodnotu argumentu a výsledok uloží ako nový *krok*. Ak argument nie je uvedený, nemá efekt.

@ : Vynásobenie *uhla*. Vynásobí *uhol* korytnačky hodnotou argumentu a výsledok uloží ako nový *uhol*. Ak argument nie je uvedený, nemá efekt.

/ : Pripočítanie k *uhlu*. Pripočíta k *uhlu* korytnačky hodnotu argumentu a výsledok uloží ako nový *uhol*. Ak argument nie je uvedený, nemá efekt.

‘ : Vynásobenie *hrúbky*. Vynásobí *hrúbku* korytnačky hodnotou argumentu a výsledok uloží ako novú *hrúbku*. Ak argument nie je uvedený, nemá efekt.

“ : Pripočítanie ku *hrúbke*. Pripočíta ku *hrúbke* korytnačky hodnotu argumentu a výsledok uloží ako novú *hrúbku*. Ak argument nie je uvedený, nemá efekt.

Posledná skupina príkazov sú príkazy na generovanie náhodných čísel. Tieto príkazy je možné uvádzať na všetkých miestach, kde je možné uvádzať argumenty. V takom prípade sa do zátvorky nepíše konštantné číslo, ale niektorý z týchto príkazov. Korytnačka si tiež pamätá posledné náhodne vygenerované číslo. Toto číslo sa dá tiež použiť ako argument (pozri nižšie).

_ : Generovanie celého nezáporného náhodného čísla s rovnomerným rozdelením. Táto funkcia môže mať dva argumenty, hornú a dolnú hranicu (v takomto poradí!). Ak sa neuvedie dolná hranica, považuje sa za 0. Ak sa neuvedie ani horná hranica, vezme sa za ňu posledné náhodne vygenerované číslo. Príklad: **_**(10)(5) vygeneruje číslo s hodnotou od 5 do 9.

~ : Generovanie reálneho náhodného čísla s Gaussovským rozdelením. Táto funkcia môže mať dva argumenty, stred a rozptyl (v takomto poradí!). Ak sa neuvedie rozptyl, považuje sa za 1. Ak sa neuvedie ani stred, považuje sa za 0. Príklad: ~(10)(5) vygeneruje číslo s Gaussovským rozdelením pravdepodobsti so stredom 10 a rozptylom 5.

: : Ak sa v argumentovej zátvorke vyskytne tento symbol, ako argument sa použije náhodné číslo, ktoré bolo naposledy vygenerované príkazom _ alebo ~.

Poznámka: Argumenty príkazov na generovanie náhodných čísel môžu byť znova príkazy na generovanie náhodných čísel! Predlžuje to však výpočtový čas, takže je dobré sa takýmto konštrukciám vyhýbať, pokiaľ je to možné.

Písanie L-systémov

Teraz prejdeme k tomu, ako vlastne písať L-systémy v programe L-System Laboratory. Na to slúži textové pole na pravej strane hlavného okna programu. Konvencia je taká, že sa v definícii L-systému **nesmú vyskytovať prázdne riadky, prípadne medzery na začiatku riadkov s pravidlami** (do axiómu je možné dať aj medzery, tie sa však berú do úvahy ako obyčajné znaky).

Prvý riadok je rezervovaný pre axióm. **Axióm je ľubovoľný reťazec** z ľubovoľných znakov.
Ďalšie riadky musia byť tvaru: **a=u**, pričom toto symbolizuje definíciu pravidla a→u. **a musí byť práve jeden znak, u je ľubovoľný reťazec. = nie je možné nahradiť iným znakom.**

Z konvencie, že na ľavej strane pravidiel môže byť len jeden znak, existujú 2 výnimky:

1. Prvý znak pred pravidlom môže byť +. Týmto sa označí koniec pravidiel, ktoré sa mutujú (pozri ďalej).

2. Pred pravidlom (za prípadným +, ale pred znakom a) môže byť uvedené nezáporné celé číslo v zátvorkách typu ( ). Toto číslo označuje tzv. váhu pravidla – program pracuje nad S0L-systémami (definícia je uvedená vyššie). Váhované pravidlá sa interpretujú tak, ako je uvedené v definícii S0L-systému. Implicitne sú všetky pravidlá váhované 1.

Vizualizácia L-systémov

Pozrime sa teraz na to, ako sa L-systémy v programe L-System Laboratory vlastne prezerajú. Štandardný postup vytvorenia L-systému je takýto:

1. Napíšeme axióm a pravidlá L-systému (pozri vyššie).
2. Nastavíme počet iteračných krokov (krokov odvodenia resp. generácií L-systému).
3. Klikneme na tačidlo Draw. Vpravo dole sa zobrazuje napredovanie výpočtu L-systému – najskôr samotné kroky odvodenia a následne interpretácia korytnačkou. Tlačidlom Stop môžeme výpočtový proces kedykoľvek zastaviť – v takom prípade sa vykreslí toľko, koľko sa stihlo vypočítať.
4. Po dokončení výpočtu sa v čiernej ploche vykreslí daný L-systém. Teraz ho môžeme posúvať, škálovať a otáčať tak, ako ho potrebujeme.
   V pravom hornom rohu okna sa nachádzajú tlačidlá X, Y, Z, ktoré určujú, podľa ktorých osí sa transformácie budú vykonávať (ak napríklad chceme L-systém posúvať iba vodorovne, musí byť stlačené iba tlačidlo X, a pod., transformácie majú však vplyv aj na tieto osi, čiže keď napríklad L-systém najskôr otočíme vodorovne o 90° a potom ho budeme posúvať pozdĺž osi x, nebude sa posúvať vodorovne, ale bude sa vzďalovať resp. približovať...).
   Posúvanie sa robí tak, že držíme stlačený kláves SHIFT a pohybujeme myšou so stlačeným ľavým tlačidlom.
   Škálovanie sa robí tak, že držíme stlačený kláves ALT a pohybujeme myšou so stlačeným ľavým tlačidlom.
   Otáčanie sa robí tak, že držíme stlačený kláves CTRL a pohybujeme myšou so stlačeným ľavým tlačidlom.
   Tlačidlo Reset zruší všetky aplikované transformácie a vráti všetko do pôvodného stavu.
5. Napísaný L-systém je možné uložiť na disk – stlačením tlačidla Save. Tlačidlom Load je možné predtým uložené L-systémy z disku znova načítať. S L-systémom sa ukladá aj aktuálne nastavený počet iterácií.
6. Vygenerovaný obrázok je možné kedykoľvek uložiť tlačidlom Save Image, ukladá sa vo formáte .bmp.

Mutovanie L-systémov

Program L-System Laboratory nedokáže L-systémy iba vizualizovať, ale dokáže dokonca vytvárať aj ich mutácie. Pod mutáciou sa rozumie, že do pravidiel alebo axiómu niečo pribudne, niečo z nich zmizne alebo sa niečo zmení. V programe je možné definovať “pravdepodobnosti” pre každý druh mutácie, s ktorou nastanú. Tiež je možné vylúčiť axióm z mutačného procesu. Ak sa pred nejakým produkčným pravidlom vyskytne znak +, sú všetky pravidlá, ktoré za týmto pravidlom nasledujú (vrátane pravidla so znakom +) vylúčené z mutačného procesu a určite ním nebudú zasiahnuté.

Keďže tento mutačný proces je veľmi primitívny, môže L-systémy úplne “pokaziť”, t.j. jednou mutáciou môže nastať zásadná zmena a môžeme dostať úplne odlišný obrázok ako predtým. Toto by mutácie teoreticky spôsobovať nemali. Z tohto dôvodu sa mutačnými schopnosťami programu nebudeme ďalej zaoberať, táto funkcia je v programe implementovaná skôr ako zaujímavosť.

Výsledky experimentov s L-systémami

Nasleduje posledná časť tejto práce – prezentovanie výsledkov, dosiahnutých v programe L-System Laboratory pri pokusoch so stochastickými L-systémami. Bude vždy uvedený L-systém tak, ako bol zadaný do programu, nejaké výsledné obrázky a prípadne nejaký komentár. Okrem výsledkov vyplývajúcich zo zadaní tejto práce sú tu uvedené aj niektoré ďalšie L-systémy, ktoré modelujú rastliny. Prílohu k programu tvoria aj ďalšie L-systémy, ktoré v tomto texte nie sú uvedené z rozsahových dôvodov. Väčšina týchto systémov pochádza z knihy The Algorithmic Beauty of Plants (pozri zoznam literatúry).

```
C
C=F[AC][BC]
A=+(40)*(~(0.7)(0.1))
B=-(33)*(~(0.7)(0.1))
```

Otáčanie je konštantné, mení sa iba dĺžka kroku – násobí sa náhodným číslom s Gaussovským rozdelením so stredom v 0.7 a disperziou 0.1.

```
$(12)C
C=F[AC][BC]
A=+(40)*(~(0.7)(0.1))"(-2)
B=-(33)*(~(0.7)(0.1))"(-2)
```

Otáčanie je konštantné, mení sa iba dĺžka kroku – násobí sa náhodným číslom s Gaussovským rozdelením so stredom v 0.7 a disperziou 0.1.
Hrúbka sa každým vnorením zmenšuje o 2.

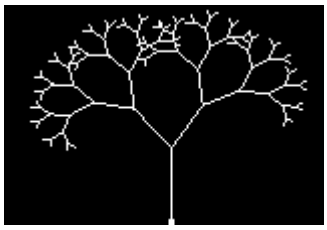
```
$(12)C
C=F[AC][BC]
A=+(_(50)(30))*(~(0.7)(0.1))"(-2)
B=-(_(50)(30))*(~(0.7)(0.1))"(-2)
```

Otáčanie nastáva o uhol s rovnomerným rozdelením medzi 30 a 50.
Ostatné je ako u predošlého L-systému.

```
$(12)C
C=F[AC][BC]
C=F[AC][DC][BC]
A=+(_(50)(30))*(~(0.7)(0.1))"(-2)
B=-(_(50)(30))*(~(0.7)(0.1))"(-2)
D=+(~(0)(10))*(~(0.7)(0.1))"(-2)
```

Rovnaký ako predošlý L-systém, ale počet vetiev je určený náhodným číslom 2 alebo 3 s rovnakou pravdepodobnosťou.
Prípadná tretia vetva má uhol určený Gaussovským rozdelením so stredom 0 a disperziou 10.

Súbor: ZADANIE.ls



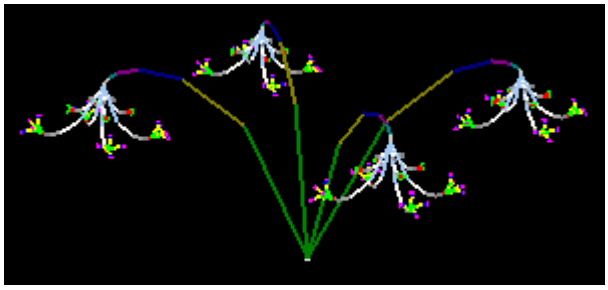
Súbor: BOP00.ls

Súbor: BOP02.ls

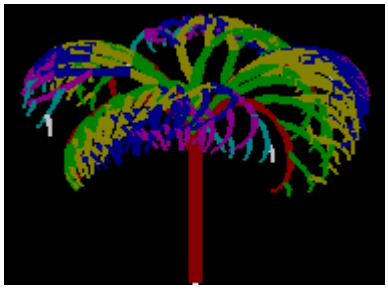
Súbor: SPIRAL2.ls

Súbor: BOP01.ls

Súbor: PLANT01.ls



Súbor: SPIRAL1.ls



Literatúra

1. Prusinkiewicz P, Lindenmayer A (1990). *The Algorithmic Beauty of Plants*. Springer-Verlag:New York

2. Lindenmayer A (1968). Mathematical models for cellular interaction in development I. Filaments with one-sided inputs. *Journal of Theoretical Biology* 18:280-289

3. Prusinkiewicz P, Hanan J (1989). *Lindenmayer systems, fractals, and plants.* Lecture Notes in Biomathematics Springer-Verlag:Berlin

4. Smith AR (1984) Plants, fractals and formal languages. *Computer Graphics* 18:July 1-10

5. http://www.biologie.uni-hamburg.de/b-online/e28_3/lsys.html

6. http://www.math.okstate.edu/mathdept/dynamics/lecnotes/node12.html