Peter Novák, MFF UK, 4iui
pno@whitestein.com

L-systém – tvorba kríka

(case study)

Úvod

Cieľom tejto práce je študovať rôzne produkčné systémy na generovanie L-systémov pre tvorbu fraktálneho kríka s rôznymi vstupnými parametrami generátora L-systému.

Teoretický úvod

Základy

Lindenmayerove systémy (ďalej len L-systémy) boli prvý raz popísané v druhej polovici tohto storočia Aristidom Lindenmayerom, podľa ktorého aj získali meno, za účelom modelovať vývoj jednoduchých viacbunkových organizmov. V princípe ide o druh paralelných gramatík, ktoré sú definované pracovnou abecedou Σ, počiatočným symbolom σ a množinou pravidiel $\mathbf{P}$, v tvare $\xi \rightarrow \omega$, kde ξ je symbol, alebo slovo z pracovnej abecedy a ω je výstupné slovo. Pri výpočte nerozlišujeme terminálne a neterminálne symboly. Keďže ide o paralelnú gramatiku, všetky symboly sa prepisujú naraz.

Neskôr sa ukázalo, že výstupné slová vygenerované L-systémami majú zaujímavé vlastnosti. Keďže pravidlá L-systémov sú rekurzívne dôsledkom je, že vygenerované slová majú vlastnosť sebapodobnosti, ktorá je charakteristická pre fraktálne štruktúry.

Bezkontextové L-systémy

Bezkontextové L-systémy (nazývame ich aj 0L-systémy) sú systémy, pre ktoré existuje ich gramatika, ktorej množina pravidiel obsahuje len pravidlá v tvare $\xi \rightarrow \omega$, kde ξ je jeden symbol zo vstupnej abecedy. To znamená, že v každom kroku výpočtu prepisujeme každé písmeno zo vstupného slova na výstupný reťazec.

Kontextové L-systémy

Kontextové L-systémy (označujú sa aj ako 1L, 2L, … - systémy) sú systémy, ktoré sa dajú popísať gramatikami, ktorých pravidlá obsahujú na ľavej strane 1, 2, prípadne viac symbolov. L-systémami tohto typu sa v tejto práci nebudem zaoberať.

Súvislosť s modelmi z prírody

L-systémy môžu byť, vďaka ich vlastnosti sebapodobnosti, chápané aj ako modely rastu rôznych štruktúr v prírode, ktoré majú často tiež túto vlastnosť.

Napríklad zjednodušené modely rastu bunkových kolónií sa dajú popísať L-systémami, alebo tiež rôzne zjednodušenia rastu jednoduchých rastlín a podobných organizmov.

Pomocou L-systémov je možné simulovať rast rastlín od jednej bunky cez viacbunkovú štruktúru (akési embryonálne štádium rastliny) až po celú rastlinku s listami, ich štruktúrou žilnatiny, kvetmi, lupeňmi apod.

V tejto práci sa budem neskôr zaoberať modelovaním rastu veľmi jednoduchých rastliniek pomocou 0L-systémov. Táto práca nejde však príliš hlboko a je pomerne jednoduchá. Zameriava sa len na modelovanie rozvoja štruktúry vetvičiek fraktálnych kríkov.

Grafická reprezentácia

Začiatkom ’70-tych rokov začali niektorí žiaci a fanúšikovia L-systémov skúmať možnosti grafického zobrazovania slov vygenerovaných gramatikami L-systémov a ukázalo sa, že grafické reprezentácie týchto slov môžu viesť k veľmi zaujímavým a esteticky príťažlivým obrázkom. Dokonca sa našli umelci, ktorí vo svojej tvorbe použili rôzne produkčné systémy na generovanie obrázkov založených na L-systémoch.

Základným princípom zobrazovania slov vygenerovaných gramatikami L-systémov je použitie princípov známych ako „korytnačia grafika“. L-systémami sa totiž dajú ľahko vygenerovať programy pre korytnačku, ktorá kreslí čiary po ploche na základe sady jednoduchých príkazov ako je ***“choď dopredu o X jednotiek“***, ***„otoč sa o Y stupňov doprava“*** a podobne.

Medzi najznámejšie a najpopulárnejšie gramatiky L-systémov patria fraktálové obrazce ako sú *Kochovej vločka* a *Sierpinského koberec*.

Program *L-system generator*

Úlohou pre túto case-study bolo vytvoriť program, ktorý generuje obrázky trojrozmerného fraktálového stromčeka, ktorého parametre sú náhodné. Teda vetvy sa neskracujú podľa konštantného faktora, ale koeficient skracovania v každom kroku je náhodný. Podobne je to aj pre stupeň vetvenia v uzloch stromu.

Popis programu

Program *L-system generator* som napísal v programovacom jazyku C++ pod operacným systémom Windows NT 4.0. Po spustení programu sa otvorí okno v ktorom je možné nastaviť parametre generovania stromčeka a spustiť generovanie.

Nastavenie parametrov

V hlavnom okne je možné nastaviť rôzne parametre generovania obrázku. Algoritmus generovania obrázku stromčeka je parametrizovateľný niekoľými atribútmi. Najdôležitejšími z nich sú ***stupeň vetvenia*** *(limbs degree)* stromčeka, ***hĺbka generovania vetiev*** *(level)* a ***koeficient skrátenia*** *(scale factor)* ďalšej úrovne vetvy.

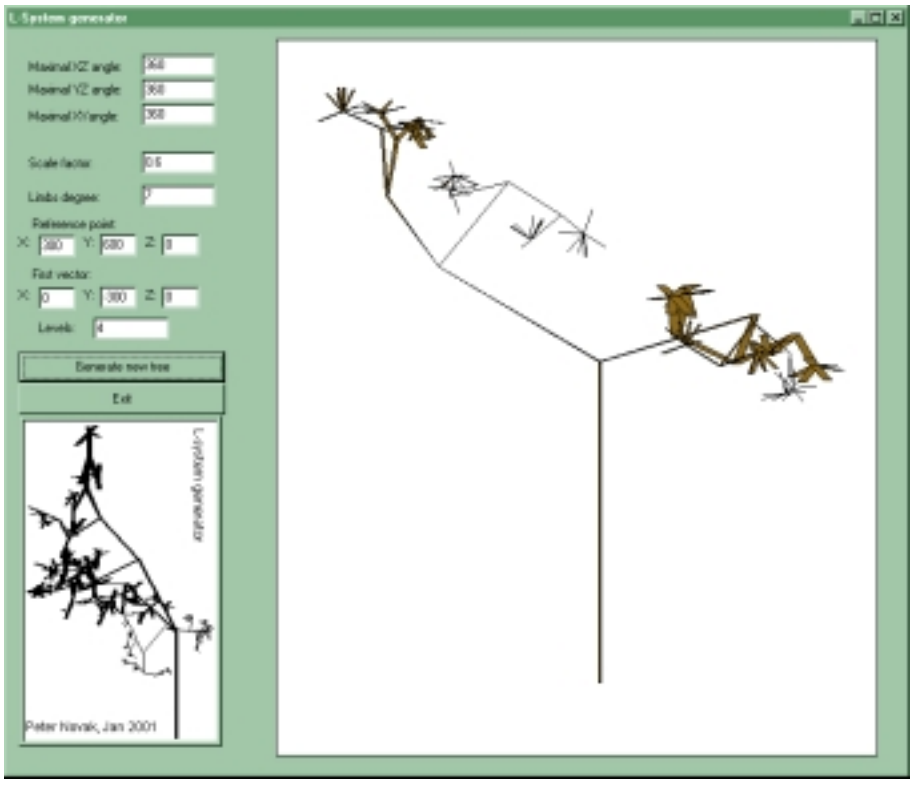
Ak za niektorý z uvedených parametrov užívateľ zadá nulovú hodnotu, algoritmus použije pre daný parameter v každom kroku vetvenia náhodne vygenerovanú hodnotu.

Algoritmus je možné parametrizovať aj maximálnym uhlom otočenia nasledujúcej vetvy v rôznych rovinách priestoru. Užívateľ môže teda nastaviť maximálny uhol otočenia v rovinách XY, XZ a YZ, pričom súradnicová os je umiestenená v ploche určenej na kreslenie obrázku tak, že bod `[x,y] = [0,0]` je umiestnený v ľavom hornom rohu kresliacej plochy. Priestorová súradnica `z` v obrázku určuje hrúbku konárika stromu. Teda čím je konárik umiestnený ďalej od pomyselnej kamery, tým je jeho hrúbka menšia a naopak, čím je konárik bližšie k pozorovateľovi tým je jeho hrúbka väčšia.

Nakoniec je možné nastaviť polohu a veľkosť prvého konárika v súradnicovej sústave. Tento je daný ***začiatočným bodom kreslenia*** *(reference point)* a ***polohou vektora kmeňa*** *(first vector)* stromčeka. Takto môžme nastaviť aby stromček rástol napríklad dole hlavou, smerom k, alebo od kamery, prípadne iným smerom.

Po spustení generovania stromčeka, kliknutím na tlačítko *Generate new tree* sa spustí výpočet a do bielej plochy sa začne kresliť obrázok stromčeka tak ako ho parametrami nastavený algoritmus generuje. Súradnice uzlov vetvenia, ani žiadne iné dáta niesú ukladané do pamäte, čo znamená, že stromček sa vykresľuje priamo tak ako ho algoritmus generuje. Z tohto dôvodu nieje možné zopakovať ten istý výpočet (teda vygenerovanie toho istého stromčeka) viac krát[1].

Tlačítkom *Exit* sa program *L-system generator* ukončí.



[1] Samozrejme ak odhliadneme od toho, že algoritmus generovania pseudonáhodných čísel z času na čas vygeneruje tú istú sekvenciu náhodných čísel po sebe, čo má za následok vyprodukovanie toho istého stromčeka.

Metóda generovania obrázku stromčeka

Algoritmus generovania stromčeka

V programe *L-system generator* sa stromček vytvára rekurzívnym volaním funkcie `GenerateTree` triedy `LTree`.

Funkcia `GenerateTree` dostáva ako vstup tri argumenty:

- `startPt` - je pozícia bodu v ktorom aktuálne vypočítavame vetvenie.
- `vec` - je vektor vetvičky, ktorá viedla do bodu startPt (teda vetvičky, ktorá je o jednu úroveň vetvenia bližšie ku kmeňu stromu pred všetkými vetvičkami, ktoré budú v tele funkcie `GenerateTree` vygenerované).
- `level` – úroveň vetvenia pre ktorú práve generujeme konáriky.

Najskôr si vypočítame stupeň vetvenia v danom uzle stromu a potom začneme generovať jednotlivé vetvy stromu. Každú vetvu vypočítavame tak, že náhodne určíme priestorový uhol pod ktorým sa má v priestore otočiť nasledujúca vetvička vzhľadom na vetvičku, ktorá bola určená vektorom `vec` a ktorá viedla do bodu `startPt` v ktorom práve vypočítavame vetvenie podstromu (tento uhol bude maximálne tak veľký ako zadal užívateľ v parametrizácii algoritmu - generátora). Potom určíme vektor skrátenia nasledujúcej vetvičky vzhľadom na predchádzajúcu, následne pomocu jednoduchých algebraických výpočtov vypočítame priestorové súradnice konca novej vetvičky a nakoniec novú vetvičku nakreslíme na obrazovku a zavoláme funkciu `GenerateTree` pre ďalšiu úroveň vetvenia na konci práve vygenerovanej vetvičky.

```
void LTree::GenerateTree(LVector startPt, LVector vec, int level)
{
 double alpha;    //rotation in XY plane
      double beta;       //rotation in XZ plane
      double gama;       //rotation in YZ plane
      double scale;      //scaling factor

      int sign;
      LVector resVec;
      int lmbCnt;

      if (level > cntLevels) return;

      //initialize random generator
      srand((unsigned)time(NULL));

      //set count of limbs in this vertex
      lmbCnt = cntLimbs;
      if (cntLimbs == 0) lmbCnt = rand() % 20;

      //generate limbs
      for (int i=0; i<lmbCnt; i++)
```

```
        {
                //compute vector rotation angles and scaling factor
                alpha = d2r((rand() % maxXYAngle));
                beta = d2r((rand() % maxXZAngle));
                gama = d2r((rand() % maxYZAngle));

                //set scaling factor
                scale = scaleFactor;
                if (scaleFactor == 0)
                scale = 0.8*((double) rand() / (double) (RAND_MAX));

                //compute resulting vector using vector algebra
                resVec = vec;

                //compute vector direction...
                resVec = GetXYRotationMatrix(alpha).Multiply(resVec);
                resVec = GetXZRotationMatrix(beta).Multiply(resVec);
                resVec = GetYZRotationMatrix(gama).Multiply(resVec);

                //scale the vector
                resVec = resVec.Multiply(scale);

                //draw the vector
                DrawTransVector(startPt, resVec);

                //call me again and again and again and ...
                GenerateTree(startPt.Add(resVec), resVec, level + 1);
        }
}
```

Algebra použitá na výpočet vetvičiek

Na začiatku generovania každej vetvičky algoritmus určí (zo vstupu, alebo náhodne) uhly otočenia novej vetvičky v priestore v jednotlivých rovinách priestoru. Tieto roviny sú určené súradnicovými osami `X`, `Y` a `Z`. Uhol `alpha` je uhlom otočenia v rovine `XY`, uhol `beta` v rovine `XZ` a nakoniec uhol `gama` určuje otočenie v rovine `YZ`.

Funkcie `GetXYRotationMatrix(angle)`, `GetXZRotationMatrix(angle)`, `GetYZRotationMatrix(angle)` vypočítavajú na základe daného uhlu v radiánoch (funkcia `d2r(deg)`vracia hodnotu uhla `deg` skonvertovanú zo stupňov na radiány) matice otočenia v jednotlivých rovinách. Tieto matice sú dané nasledovne:

```
XYRotationMatrix:
         cos(angle)    sin(angle)        0
        -sin(angle)    cos(angle)        0
             0              0             1


XZRotationMatrix:
         cos(angle)         0        sin(angle)
             0              1             0
        -sin(angle)         0        cos(angle)
```

```
YZRotationMatrix:
              1             0             0
              0        cos(angle)    sin(angle)
              0       -sin(angle)    cos(angle)
```

Každá nová vetvička fraktálového stromu je vypočítaná z vektora, ktorým bola určená vetvička o úroveň vyššie (vektor `vec`) postupným násobením s maticami otočenia a nakoniec skrátením výsledného vektora podľa koeficientu skrátenia pre danú vetvičku:

```
resVec = vec*XYRotationMatrix*XZRotationMatrix*YZRotationMatrix
```

Matice majú implementované operácie násobenie maticou, vektorom a skalárom. Vektory podobne majú implementované násobenie iným vektorom a skalárom.

Na začiatku výpočtu sa nakreslí počiatočný kmeň stromu zadaný užívateľom a jeho koncový bod je použitý ako argument prvého volania funkcie `GenerateTree`.

Poznámka k implementácii

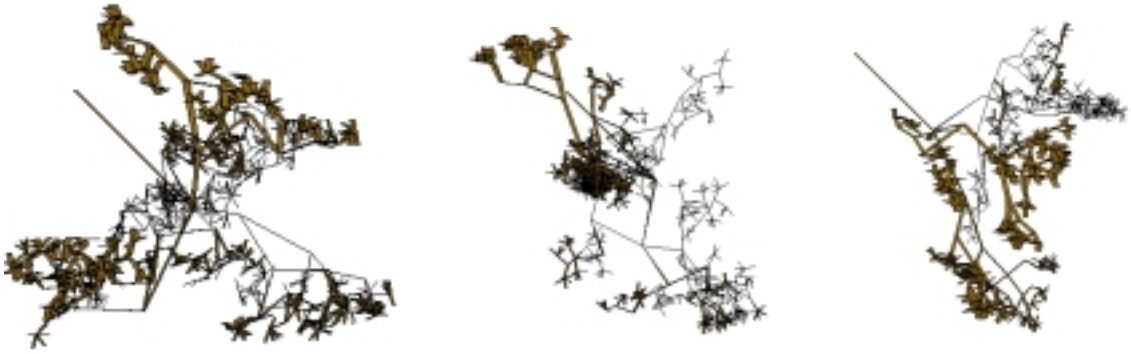
Pri implementácii som nepoužil žiadnu optimalizáciu a preto je výpočet pre veľké stupne vetvenia a pre veľké hĺbky generovania stromov pomerne pomalý. Samozrejme nebolo by náročné implementovať niekoľko optimalizácií na rýchlosť výpočtu, no keďže implementácia rýchleho algoritmu nebola explicitne v zadaní riešenia úlohy, nepoužil som žiadne zrýchlenie.

Vygenerované obrázky

Na záver prikladám niekoľko obrázkov vygenerovaných programom *L-system generator* aj s parametrami pre ktoré boli vygenerované.

```
[MaxXZAngle, MaxYZAngle, MaxXYAngle]     =      [360, 360, 360]
  [RefPointX, RefPointY, RefPointZ]      =       [300, 600, 0]
  [FirstVecX, FirstVecY, FirstVecZ]      =        [0, -200, 0]
 [ScaleFactor, LimbDegree,  Levels]      =          [0, 0, 5]
```

```
[MaxXZAngle, MaxYZAngle, MaxXYAngle]    =      [360, 360, 360]
 [RefPointX, RefPointY, RefPointZ]      =       [160, 170, 0]
 [FirstVecX, FirstVecY, FirstVecZ]      =       [100, 100, 0]
 [ScaleFactor, LimbDegree,  Levels]     =        [0.7, 5, 6]
```



```
[MaxXZAngle, MaxYZAngle, MaxXYAngle]    =       [50, 100, 80]
 [RefPointX, RefPointY, RefPointZ]      =      [50, 600, -100]
 [FirstVecX, FirstVecY, FirstVecZ]      =     [100, -200, -100]
 [ScaleFactor, LimbDegree,  Levels]     =        [0.6, 4, 4]
```

Literatúra

[1] – Kvasnička, Pospíchal, Tiňo; Evolučné algoritmy; Vydavateľstvo STU Bratislava 2000

[2] – L-systems tutorial; http://alife.tuke.sk/projekty/l-sys/index.html

[3] – T. Katriňák; Algebra a teoretická aritmetika I.; Vydavateľstvo Alpha