L-sistemi za fraktalno generiranje 3D objektov
Seminarska naloga
Računalniška grafika
IŠRM

Marinka Žitnik (63080119)

1. junij 2011

Kazalo

Poglavje 1

Pregled

1.1 Uvod

V seminarski nalogi smo razvili program *LSystems* za vizualizacijo L-sistemov. *LSystems* je interaktivni program, ki uporabniku omogoča izbiro gramatike v grafičnem uporabniškem vmesniku, nastavitev gramatično odvisnih parametrov in izbiro vizualizatorja. Vizualizatorji so gramatično neodvisni, kar pomeni, da je možno posamezni vizualizator uporabiti na več gramatikah in obratno, eno gramatiko predstaviti z več vizualizatorji. Pri tem so lastnosti gramatike ob izbiri določenih predstavitev bolj ali manj izpostavljene/opazne/vidne – uporabnik lahko preizkusi več predstavitev in se nato odloči za njemu ustrezno. Nekatere gramatike definirajo tak jezik, da je izris njegovih besed lahko časovno zahteven. Zato je možno izbrati število iteracij nad produkcijami gramatike, kar vpliva na dolžino generirane besede - navodila za izris in časovno kompleksnost izrisa. Tako uporabnik lahko prilagodi delovanje zmogljivosti sistema, na katerem je *LSystems* nameščen. Z osnovnimi transformacijami (rotacija, translacija, skaliranje, zorni kot) je možen boljši pregled na izrisom.

Napredni uporabniki lahko izkoristijo prednosti enotnega programskega vmesnika tako, da po specifikaciji vmesnika za gramatike (ali vmesnika za vizualizatorje) k izvornim datotekam programa dodajo javanske razrede z novo definiranimi gramatikami (vizualizatorji). Ob naslednjem zagonu *LSystems* se dodani razredi prevedejo in so na volju v spustnem meniju za izbiro gramatik (vizualizatorjev).

1.2 Funkcionalna specifikacija

Funkcionalne zahteve *LSystems*:

Enotni programski vmesnik Programski vmesnik enotno obravnava gramatike. Ta zahteva je smiselna in potrebna, če želimo omogočiti uporabniku enostavno dodajanje novih gramatik v *LSystems*. Dodana gramatika mora razširiti razred `Grammar` in najmanj redefinirati metodo `declare()` in `setRequiredParameters()`. V tej metodi se nastavi aksiom – začetni simbol gramatike in doda produkcije – pravila za izpeljevanje. Odvisno od kompleksnosti gramatike je možno definirati aksiome kot zaporedja simbolov in dodati produkcijske družine (npr. v primeru stohastičnih gramatik imamo za eno levo stran določenih več desnih strani produkcij, desna stran se ob izvajanju nato izbere nedeterministično).

Določiti je možno parametre gramatike (imena in privzete vrednosti – ta se izpišejo ob izbiri gramatike v grafičnem uporabniškem vmesniku). Simbolom gramatike je možno dodati metode – odziv na generirani simbol v besedi jezika gramatike, kar omogoča podoporo parametričnim L-sistemom.

```
Method growthMethod = new Method(){
 public boolean calculate(ArrayList<Symbol> sS, int sIdx, Symbol cS)
 {
 Float r = cS.getParameter("radius");
 Float r_mul = cS.getParameter("rM");
 cS.setParameter("radius", r*r_mul);

 Float d = cS.getParameter("distance");
 Float d_mul = cS.getParameter("dM");
 cS.setParameter("distance", d*d_mul+
        (cS.getParameter
        ("distanceVariation")*d*Randomizer.getRandomizer().nextVariation()));

 return true;

 }
};
SymbolClass growthF = new SymbolClass("F","F(grow)");
growthF.addMethod(growthMethod);
```

Podobno kot za gramatike velja tudi za vizualizatorje. Dodani vizualizator mora razširiti abstraktni razred `Visualizator` in najmanj redefinirati metodo `visualize(VisualData vd)`. Predpostavimo lahko, da objekt `vd` vsebuje seznam simbolov in njihovih pridruženih transformacij. Te simbole z vizualizatorjem (npr. valj, sfera, deblo, piramida, grm, praprot, cvetoče drevo itd.) izrišemo. Tudi za vizualizatorje je mogoče določiti parametre (npr. velikost in starost lista pri vizualizatorju `Tree`). Simboli iz seznama so objekti, ki med drugim hranijo pripadajoče transformacije – te določi interpreter besed, ki jih zgeneriramo s pravili gramatike.

Podpora parametričnim kontekstno odvisnim L-sistemom Zaradi enotne obravnave gramatik in vizualizatorjev smo podporo relativno enostavno dodali.

Podpora determinističnim in stohastičnim L-sistemom Zaradi enotne obravnave gramatik in vizualizatorjev smo podporo relativno enostavno dodali.

Podpora iterativnim funkcijskim sistemom (IFS) za fraktale Zaradi enotne obravnave gramatik in vizualizatorjev smo podporo relativno enostavno dodali.

Uporabniški vmesnik za izbiro gramatik, vizualizacij in parametrov sistema

- izbira gramatike iz spustnega menija,
- izbira vizualizatorja iz spustnega menija,
- vpis števila iteracij za generiranje besede v jeziku gramatike,
- izbira izrisa naslednje iteracije (inkrementalno),
- vnos vrednosti parametrov izbrane gramatike in vizualizatorja.

OpenGL 3D vizualizacija izbranih L-sistemov Dodana umestitev objektov v prostor in omogočene osnovne transformacije nad objekti. Za realizacijo simbola `[` in `]` za shranjevanje/priklic trenutnega položaja in orientacije uporabljamo sklad za transformacijske matrike, ostale simbole interpretira `LSystemsInterpreter`.

Tudi pomen ostalih simbolov je standarden [1]:

- `[, ]` – shranjevanje in priklic transformacijske matrike,
- `F` – pomik naprej za `distance`,
- `+` – obrat za `+ turn_angle`,
- `-` – obrat za `- turn_angle`,
- `<` – kotaljenje za `- roll_angle`,
- `>` – kotaljenje za `roll_angle`,
- `R` – naključnost,
- `/` – naklon za `pitch_angle`,
- \\ – naklon za `- pitch_angle`,
- `|` – obrat za `180`°,
- `S` – skaliranje.

Poglavje 2

Rešitev

2.1 Ogrodje in jezik

Za implementacijo seminarske naloge *LSystems* smo uporabili programski jezik Java in programski vmesnik OpenGL ter knjižnico LWJGL. Izdelali smo nekaj preprostih 3D modelov, ki smo jih uporabili v vizualizacijah.

2.2 Arhitektura

Pri razvoju sistema *LSystems* smo upoštevali strukturiranost in modularnost, ki omogočata enostavno razširljivost in skalabilnost aplikacije. Končna rešitev s kratkimi opisi funkcij modulov:

- `lsystems`
  Modul skrbi za upravljanje z zunanjimi viri *LSystems*. Če je potrebno, naloži in prevede javanske razrede, ki definirajo gramatike in vizualizatorje in jih doda na uporabo ostalim modulom. Modul vsebuje vmesnik za nalaganje 3D modelov - ta vmesnik poskrbi, da se vsak 3D model naloži v sistem kvečjemu enkrat, tako da hrani modele in vodi evidenco o njihovi uporabi. Za razhroščevanje in spremljanje procesa vizualizacije je koristno obveščanje uporabnika o napakah, obvestilih in opozorilih sistema. Razredi modula implementirajo odzive na operacije simbolov gramatik, ki definirajo transformacije transformacijskih matrik OpenGL. Modul vsebuje zagonski razred aplikacije, ki poskrbi za vzpostavitev grafičnega vmesnika in okna za izris.

- `lsystems.formallanguage` Nadzornik za interpretacijo simbolov v besedi jezika gramatike. Objekti, ki predstavljajo različne vrste simbolov, hranijo pridružene podatke o njihovi transformaciji in globini izrisa.

  - `lsystems.formallanguage.grammar` Definicija gramatik, pravil izpeljevanja in produkcijskih družin.
  - `lsystems.formallanguage.parser` Interpreter nizov, ki predstavljajo besedo v jeziku neke gramatike.
  - `lsystems.formallanguage.symbol` Definicija metod za obravnavo parametrov simbolov je skupna razredu simbolov `ClassOfSymbols`, vsak objekt simbol pa hrani parametre. Izogibamo se nepotrebenemu dupliciranju razredov simbolov.

- `lsystems.grammars` Množica razredov, ki definirajo gramatike. Vizualizacije nekaj gramatik so v Dodatku A.
- `lsystems.gui` Okno za izris fraktalov in modelov.
  - `lsystems.gui.options` Grafični uporabniški vmesnik in proženje izrisov fraktalov.
- `lsystems.loaders` Nalaganje 3D modelov.
- `lsystems.util` Dodatne operacije - npr. merjenje časa.
- `lsystems.visualizator` Vmesniki vizualizatorjev in definicija primitivov.
- `lsystems.visualizators` Množica razredov, ki definirajo vizualizatorje. Uporaba nekaterih vizualizatorjev je prikazana v Dodatku A.
  - `Tree`
  - `Flowering Tree`
  - `SpongeBox`
  - `Bush`
  - `Fern`
  - `Mosaic`
  - `Line`
  - `Sphere`
  - `Pyramid`

2.3 Gramatike

Objekti, ki jih generiramo z deterministčnim L-sistemom, so identični. Vsak poskus vključitve več takih objektov na vidno polje bi povzročil umetni regularni videz. V ta namen smo v seminarski nalogi dodali gramatikam parametre variacije, ki ohranjajo splošni vtis objekta, a spreminjajo detajle. Variacije v objektih lahko dosežemo z naključnim spreminjanjem interpretacije simbolov v nizu jezika gramatike ali z naključnimi spremembami v produkcijskih pravilih gramatike. V seminarski nalogi smo se odločili za drugo možnost, saj ima naključno spreminjanje interpretacije simbolov omejen učinek. Spremembe v interpretaciji simbolov namreč vplivajo na geometrijske lastnosti predmetov (npr. dolžina veje, koti razvejanj ipd.), a topologija predmeta ostane nespremenjena. Nasprotno, stohastična aplikacija produkcijskih pravil vpliva na geometrijske in topološke lastnosti. Eden izmed primerov stohastičnega L-sistema v seminarski nalogi je `cvetoče drevo`. `Cvetoče drevo` je tudi primer parametričnega L-sistema.

Dobro znani objekti, ki jih generirajo iterativni funkcijski sistemi (IFS), so listi praproti. Primer takega funkcijskega sistema v seminarski nalogi je `IFS Praprot`. Funkcijske sisteme opišemo z iterativnim postopkom računanja serij, ki opisujejo koordinate položaja. Enačbe opisujejo translacije, skaliranje, rotacijo in striženje točk (afine transformacije). Pri tem je pomembna omejitev, da so transformacije skrčitve, $d(f(x), f(y)) \leq d(x, y)$ in primerna izbira začetnih točk, ki v iterativnem postopku dajo želeni izris.

V sintezi modela objekta je priročno ločevanje vejitvenih produkcij (opisujejo vzorce vejitev rastline) od prirastnih produkcij (opisujejo rast segmentov rastline). V enostavni gramatiki za generiranje drevesa

```
B->F[+B][>+B][>>+B][B]
F->FF
```

druga produkcija opisuje rast objekta. V gramatiki za generiranje Hilbertove krivulje

```
A->B-F+CFC+F-D\\F/D-F+\\\\CFC+F+B>>
B->A\\F/CFB/F/D//-F-D/|F/B|FC/F/A>>
C->|D/|F/B-F+C/F/A\\\\FA\\F/C+F+B/F/D>>
D->|CFB-F+B|FA\\F/A\\\\FB-F+B|FC>>
```

običajna prirastna funkcija ne nastopa. Število simbolov `F` v gramatiki drevesa se v nizu, ki izvira iz ene pojavitve `F`, v vsaki iteraciji podvoji, kar pomeni, da je rast eksponentna. Prirastna funkcija $f_G(n)$ je funkcija, ki določa število simbolov v besedi jezika gramatike v odvisnosti od dolžine izpeljave. Prirastna funkcija v deterministični kontekstno neodvisnih sistemih je neodvisna od vrstnega reda simbolov v produkcijah in izpeljanih besedah ter je v splošnem kombinacija polinomskih in eksponentnih funkcij $f_G(n) = \sum_{i=0}^{s} P_i(n)\rho^n$. Žal mnogo naravnih prirastnih sistemov ne moremo opisati s to enačbo, zato se uporabljajo drugi pristopi – sigmodalne funkcije in kontekstno odvisni sistemi.

Parametrični L-sistemi delujejo nad parametričnimi besedami – nizi simbolov s pridruženimi parametri (simboli so elementi abecede gramatike, parametri so realna števila) $A(a_1, a_2, \ldots, a_n),\ A \in V,\ a_i \in R$. L-sistem lahko vsebuje formalne parametre, konstante, aritmetične in logične operatorje, kar prikazuje spodnji L-sistem, ki smo ga uporabili v seminarski nalogi. Primer osnovnega parametričnega sistema za gramatiko `Vrsta dreves`

```
F->F(x*p,2)+F(x*h,1)--F(x*h,1)+F(x*q,0)
F->F(x,t-1)
```

V parametričnih sistemih produkcije sestavljajo:

- predhodnik (leva stran izpeljav neparametričnih sistemov),
- pogoj,
- izpeljava.

Pravimo, da produkcija ustreza modulu v parametrični besedi, če so izpolnjeni naslednji pogoji:

- simboli v modulu in predhodnik produkcije se ujemata,
- število dejanskih parametrov v modulu je enako številu formalnih parametrov predhodnika,
- z zamenjavo formalnih parametrov z dejanskimi je pogoj v produkciji izpolnjen.

V parametrični gramatiki za drevo `Drevo`

```
K->F[+RB][>>+RB][>>>>+RB][RB(main)]
B->>F[+RBRL][>>+RBRL][>>>>+RBRL][RBRL]
F->F(grow)
```

v seminarski nalogi obravnavamo pogoje za zaključitev vejitev na sledeč način

```
Production p = SimpleParser.parseProduction("B->>F[+RBRL][>>+RBRL][>>>>+RBRL][RBRL]");
p.setCondition(new Condition(){
 public boolean check(ArrayList<Symbol> symbolSequence, int symbolIndex) {
        Symbol calculatedSymbol = symbolSequence.get(symbolIndex);
        return Randomizer.getRandomizer().getRandom().nextFloat() >
                                calculatedSymbol.getParameter("PrBudDeath");
 }
});
addProduction(p);
```

Produkcije v parametričnih L-sistemih so kontekstno neodvisne, njihova aplikacija je možna neodvisno od konteksta predhodnika. Potrebna je kontekstno odvisna razširitev za modeliranje izmenjave informacij med sosednimi moduli. V parametričnih L-sistemih to pomeni, da so komponente predhodnika parametrične besede s simboli iz abecede gramatike in formalnimi parametri. V pogoju in izpeljavi produkcije se lahko pojavi katerikoli formalni parameter. Parametrični kontekstno odvisni L-sistemi so učinkovito orodje za opis razvojnih modelov, ki vključujejo razširitev substanc v biološkem organizmu.

Poglavje 3

Zaključek

3.1 Predlogi za nadaljnje delo

Področje, ki smo ga obravnavali v seminarski nalogi, je zelo zanimivo in ima številne uporabe, zato bi vsebino seminarske naloge bilo možno zlahka razširiti.

Predlagamo simulacijo tropizma pri generiranem 3D rastlinju (npr. spremenjena rast rastlinja ob prisotnosti naravnih spodbud - veter, nagnjeno pobočje, pomanjkanje svetlobe ipd.).

Izris fraktalnih objektov za večje globine izrisa in število iteracij zahteva zmogljivo strojno opremo. V seminarski nalogi smo skušali optimizirati izrisovanje - npr. izrisovanje do določene globine/iteracije, izogibanje nepotrebnim ponovnim izrisovanjem ipd., a je možno izris še dodatno pohitriti z razbremenitvijo centralne procesne enote s senčilniki. Uporabniška izkušnja na zmogljivi strojni opremi je boljša, ker lahko uporabnik izbira izrise večjih globin objektov.

3.2 Sklep

Seminarska naloga L-sistemi za fraktalno generiranje 3D objektov je zanimiva, saj združuje teoretične principe gramatik iz osnov teoretičnega računalništva in njihovo praktično uporabo v vizualizaciji fraktalnih objektov. Ob študiju literature nas je presenetil in navdušil razvoj algoritmične botanike.

Literatura

[1] Lindenmayer, A. Prusinkiewicz, P. The Algorithmic Beauty of Plants. Springer-Verlag, New York. 2004.

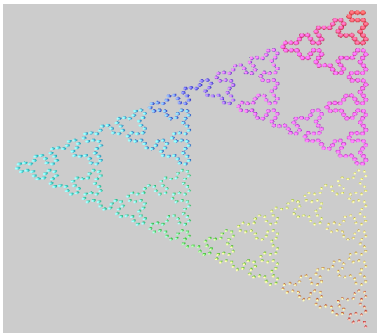
[2] Mech, R., Prusinkiewicz, P. Visual Models of Plants Interacting with Their Environment Visual Models of Plants. In Computer Graphics Proceedings, Annual Conference Series, 1996, ACM SIGGRAPH, pp. 397−410

[3] Fuhrer, M. Jensen, W. H. Prusinkiewicz, P. Modeling Hairy Plants. In Proceedings of Pacific Graphics 2004, pp. 217−226.

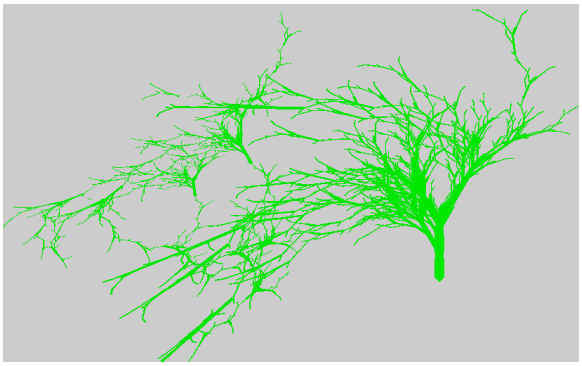
Dodatek A

Primer uporabe

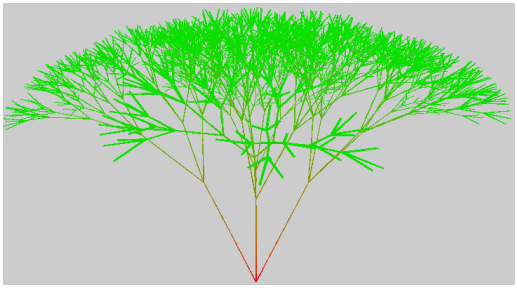
Na slikah A.1 do A.16 so prikazani primeri vizualizacije različnih gramatik z različnimi vizualizatorji. Slike so primerno pomanjšane.



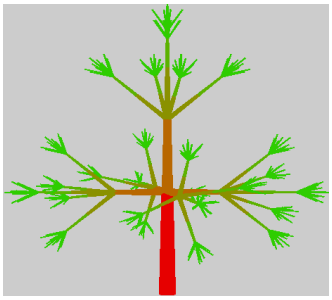
Slika A.1: Gramatika: Krivulja Sierpinskega. Vizualizator: Sfera. 6 iteracij.



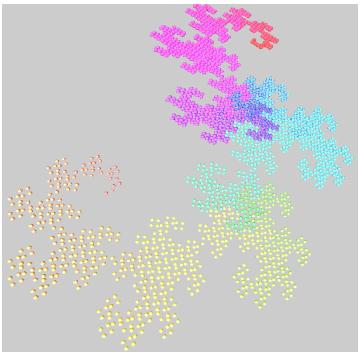
Slika A.2: Gramatika: Veja. Vizualizator: Praprot. 5 iteracij.



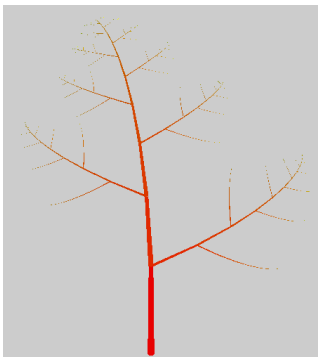
Slika A.3: Gramatika: Grm. Vizualizator: Drevo. 6 iteracij.



Slika A.4: Gramatika: Božično drevo. Vizualizator: Drevo. 5 iteracij, *pitch = 66, roll = 77.*

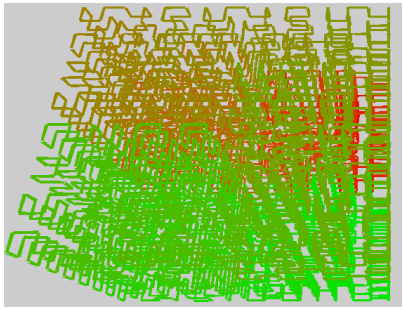


Slika A.5: Gramatika: Zmajeva krivulja. Vizualizator: Sfera. 11 iteracij.

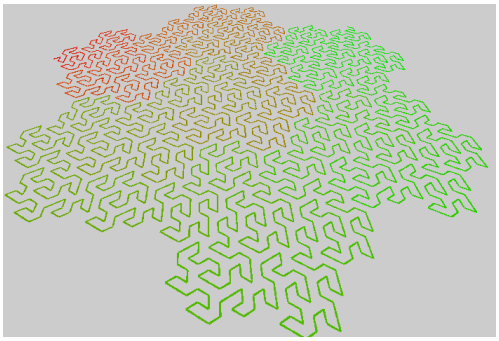


Slika A.6: Gramatika: Praprot. Vizualizator: Praprot. 3 iteracije.

Slika A.7: Gramatika: Cvetoče drevo. Vizualizator: Cvetoče drevo. 7 iteracij in 14613 simbolov, *PrDeathBud = 1*.

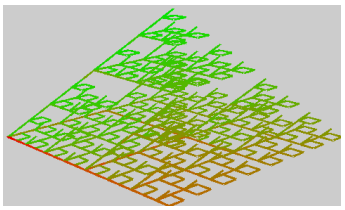


Slika A.8: Gramatika: Hilbertova krivulja. Vizualizator: Drevo. 4 iteracije.

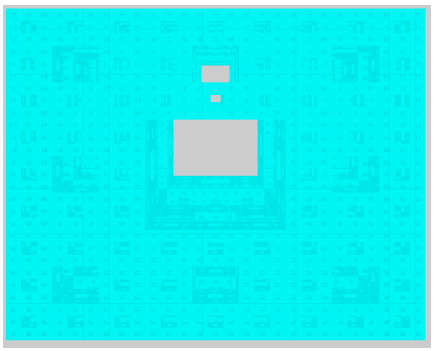


Slika A.9: Gramatika: Peanova krivulja. Vizualizator: Drevo. 4 iteracije.

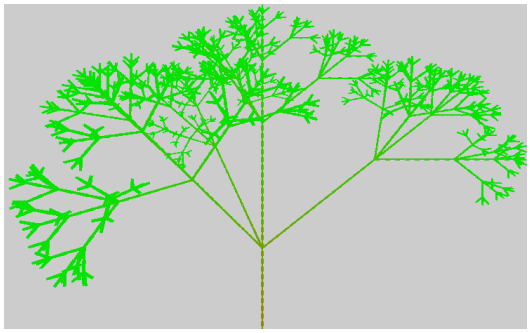
Slika A.10: Gramatika: Vrsta dreves. Vizualizator: Črta. 8 iteracij.



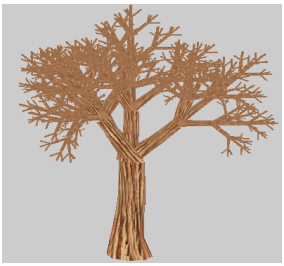
Slika A.11: Gramatika: Piramida Sierpinskega. Vizualizator: Drevo. 4 iteracije.



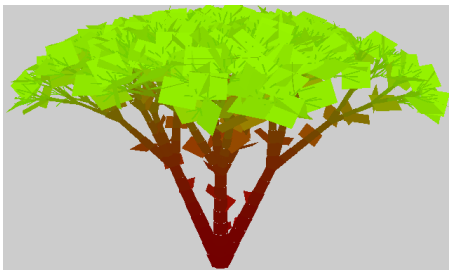
Slika A.12: Gramatika: Goba. Vizualizator: Goba. 2 iteraciji in 2332 simbolov.



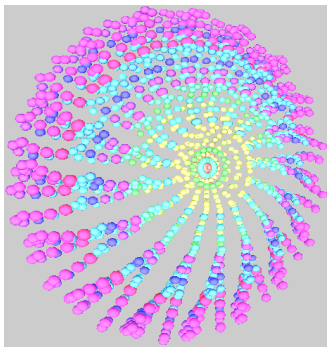
Slika A.13: Gramatika: Drevo. Vizualizator: Drevo. 6 iteracij.



Slika A.14: Gramatika: Drevo 2. Vizualizator: Drevo 2. 6 iteracij in 34126 simbolov.



Slika A.15: Gramatika: Grm. Vizualizator: Grm. 6 iteracij.



Slika A.16: Gramatika: Cvet. Vizualizator: Sfera. 4 iteracije.