SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 991

ALGORITAMSKA BOTANIKA

Ognjen Šuljagić

Zagreb, siječanj 2010.

mentorici prof. Željki Mihajlović,

što su završetak studija i izrada ovog rada protekli u ugodnoj atmosferi,

prof. Dragici Šestanj Perić,

na pristupu i ustrajnosti da me nauči razliku kvadrata :),

prof. Krešimiru Kovšci,

na svom razumijevanju i šansama,

cijeloj obitelji,

na potpunoj podršci,

mojoj majci,

na povjerenju i strpljenju, čak i kad ih je bilo tako teško imati,

a posebno voljenoj Nikoleti,

bez koje svega ovoga ne bi nikad bilo.

Sadržaj

1. Uvod

U ovom radu rješava se problem predstavljanja botaničkih objekata zapisanih L-sustavom. Rješenje će biti klasičnog oblika, odnosno implementacijom grafike trokutastog pokazivača. Čak i na današnjim računalima, napraviti kompleksan, realan botanički sustav, te omogućiti simulaciju fizike u stvarnom vremenu na takvom sustavu, nije lak zadatak. U radu biti će razrađeni primjeri manje složenosti. Prvo se prezentira opća tematika L-sustava, kao i njihove implementaciju u dvije i tri dimenzije pomoću grafike trokutastog pokazivača. Nakon toga opisuje se izrađena programska implementacija.

2. L-sustav

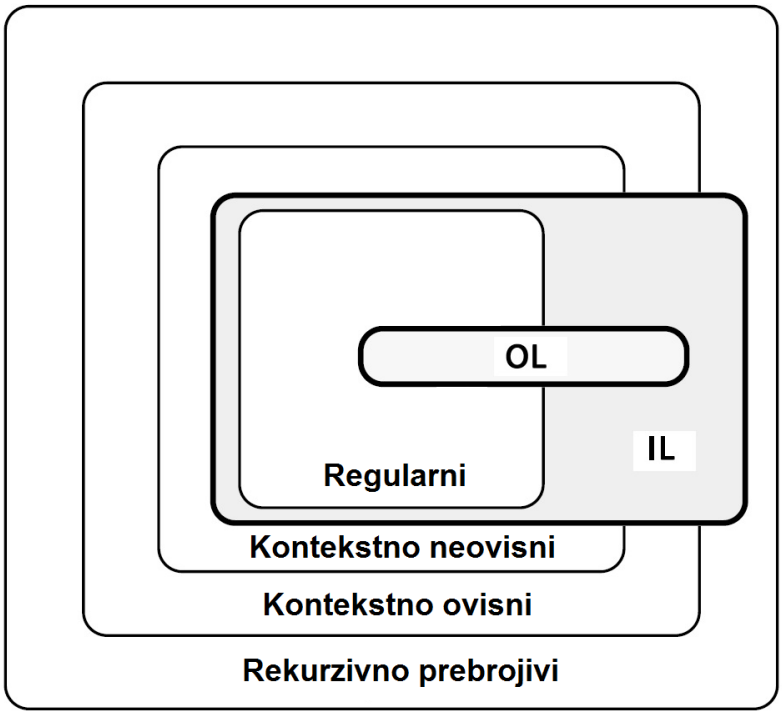
2.1. Povijest i motivacija

Ideju opisa razvoja biljaka matematičkom teorijom realizirao je 1968. Aristid Lindenmayer putem L-sustava. L-sustavi u početku nisu uključivali dovoljno detalja da modeliraju složenije biljke, već je naglasak stavljen na topologiju biljke, odnosno odnos između stanica ili većih dijelova biljaka. Geometrijski aspekti biljaka nisu bili uključeni u teoriju. Tek kasnije se interpretacije tih aspekata pojavljuju u raznim oblicima, jedan od kojih je interpretacija trokutastog pokazivača koja se opisuje u ovom radu.

L-sustavi u osnovi su vrlo slični formalnim gramatikama Noama Chomskya, s razlikom u načinu primjene produkcija. Kod Chomskyevih gramatika produkcije se primjenjuju sekvencijalno, a u L-sustavima paralelno, mijenjajući odjednom sve znakove dane riječi. U toj je razlici reflektirana biološka motivacija L-sustava, u smislu dijeljenja stanica višestaničnog organizma, gdje se proizvoljan broj dijeljenja može dogoditi istovremeno.

2.2. Vrste L-sustava

Razlikujemo kontekstno neovisne (OL-sustave) i kontekstno ovisne (IL-sustave) , odnos tih sustava i klasa Chomskyeve gramatike prikazan je na slici 1.



Slika 1. Odnos klasa Chomskyeve gramatike i L-sustava

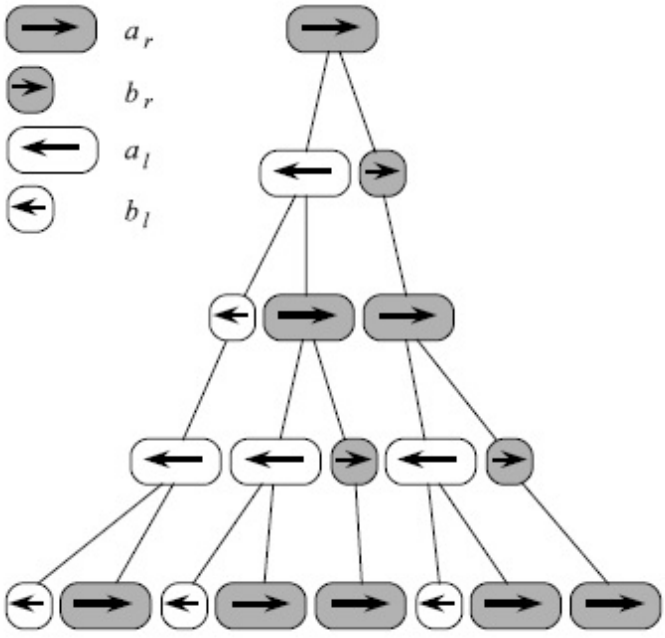
Formalna definicija DOL ili determinističkih kontekstno neovisnih L-sustava, predstavljenih uređenom trojkom G dana je na sljedeći način

$$G = < V, \omega, P>$$

gdje je V abeceda sustava, $\omega \in V^+$ je neprazna riječ, tzv. aksiom i $P \subset V \times V$ konačan skup produkcija. Produkcija $(a, \chi) \in P$ se piše kao $a \rightarrow \chi$, pri čemu je a slovo abecede, a χ riječ iz V^*.

Na sljedećem primjeru [iz 1] simulacije razvoja plavo-crne bakterije *Anabaene catenule* prikazuje se primjena DOL sustava u slučaju višestaničnih organizama (slika 2). Simboli a i b predstavljaju citološka stanja stanica, odnosno njihovu veličinu i spremnost na dijeljenje. Indeksi l i r su polariteti stanica, odnosno smjerovi u kojem će nastati stanica dijete.

$\omega : a_r$
$p_1 : a_r \rightarrow a_l b_r$
$p_2 : a_l \rightarrow b_l a_r$
$p_3 : b_r \rightarrow a_r$
$p_4 : b_l \rightarrow a_l$



Slika 2. Razvoj filamenta Anabaene catenule

Počevši od aksioma a_r, dobivamo sljedeći niz riječi:

1. iteracija: $a_l b_r$
2. iteracija: $b_l a_r a_r$
3. iteracija: $a_l a_l b_r a_l b_r$
4. iteracija: $b_l a_r b_l a_r a_r b_l a_r a_r$

. . .

2.3. Interpretacija grafikom trokutastog pokazivača (engl. turtle)

2.3.1. Povijest

Za razvoj modela kompleksnijih od *Anabaene catenule* trebalo je osmisliti složeniju grafičku interpretaciju. Prvu takvu objavili su 1974. Frijters i Lindenmayer, te Hogeweg i Hesper, čija je namjena prvenstveno bila opisati grananje biljaka. Szilard i Quinton predlažu drugačiji pristup interpretaciji 1979., uz rigorozno definiranu geometriju, pokazali su da naizgled jednostavni DOL sustavi mogu stvoriti iznimno kompleksne oblike, poznate danas kao fraktale.

Prusinkiewicz se usredotočio na interpretaciju baziranu na trokutastom pokazivaču *turtle* u LOGO stilu i prikazao još fraktala i modela sličnih biljkama. Daljnji razvoj L-sustava s interpretacijom trokutastog pokazivača donio je realistične modele biljaka, sintezu muzičkih dijela i automatsko generiranje prostorno popunjavajućih krivulja. Ideje Abelsona i diSesse dovele su do proširenja ove interpretacije u treću dimenziju.

2.3.2. Interpretacija u 2D

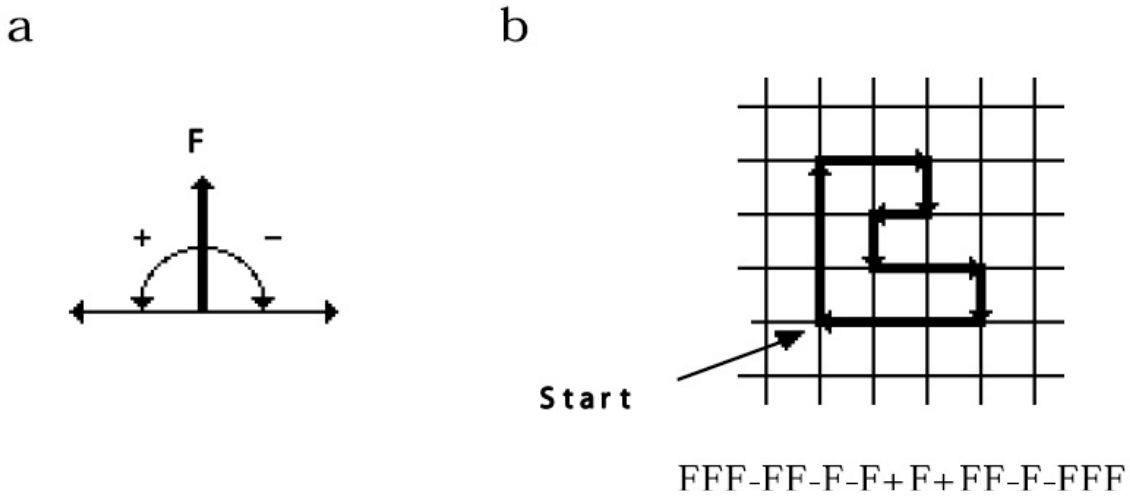
Stanje trokutastog pokazivača definira se kao trojka (x, y, α), pri čemu (x, y) predstavlja koordinate pokazivača u kartezijanskom koordinatnom sustavu, a kut α predstavlja smjer u kojem je trokutasti pokazivač okrenut. Uz duljinu koraka d i povećanje kuta δ, turtle pokazivač slijedi naredbe dane sljedećim simbolima:

F : Kreni naprijed za korak duljine d. Stanje trokutastog pokazivača se mijenja u (x', y', α), gdje je $x' = x + d \cos\alpha$, $y' = y + d \sin\alpha$. Nacrtaj liniju između (x, y) i (x', y').

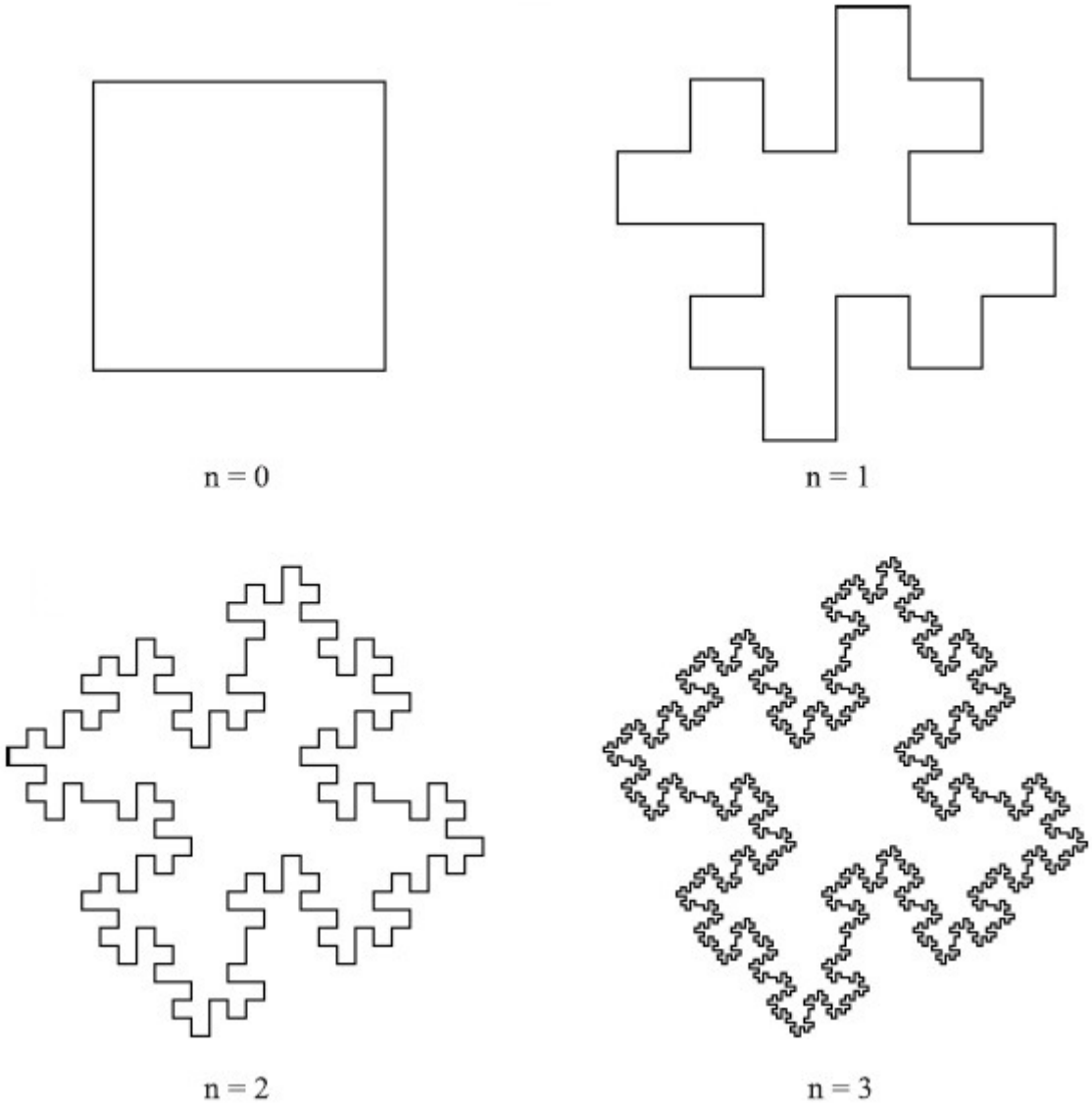
\+ : Okreni se u lijevo za kut δ. Sljedeće stanje trokutastog pokazivača je ($x, y, \alpha + \delta$). Pozitivna orijentacija kutova je u smjeru suprotnom kazaljci na satu

\- : Okreni se u desno za kut δ. Sljedeće stanje trokutastog pokazivača je ($x, y, \alpha - \delta$).

Uz početni niz znakova w, početno stanje pokazivača (x_0, y_0, α_0) i fiksne parametre d i δ, interpretacija dobivena postupkom trokutastog pokazivača od w je niz linija nacrtanih kao odaziv na niz znakova w.



Slika 3. Turtle interpretacija simbola i niza znakova



Slika 4. Kvadratni Kochov otok

Na slici 3(a) prikazana je interpretacija dobivena postupkom trokutastog pokazivača simbola *F, +, -*.
Na slici 3(b) prikazana je interpretacija dobivena postupkom trokutastog pokazivača niza znakova, sa δ=90° i početnoj orijentaciji prema gore.

Slika 4 predstavlja aproksimaciju kvadratnog Kochovog otoka, a dobivena je interpretacijom sljedećeg L-sustava:

$$\omega : F - F - F - F$$

$$p : F \rightarrow F - F + F + FF - F - F + F$$

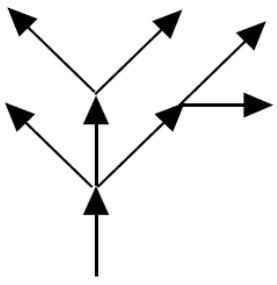
Uz broj koraka od 0 do 3, te povećanjem kuta δ=90°, duljina koraka se smanjuje četiri puta nakon svake slike.

Dosad obrađenim L-sustavima ne možemo specificirati strukturu podataka kojom bi prikazali drva sa centralnom osi. Taj problem rješavamo uvođenjem uglatih zagrada u niz znakova, a ta dva nova simbola su:

[: stavi na stog (engl. push) trenutno stanje turtle pokazivača na stog. Informacija sačuvana na stogu sadržava poziciju i orijentaciju trokutastog pokazivača, te možda druge atribute npr. boju ili širinu linije koja se crta

] : digni stanje (engl. pop) s vrha stoga, to stanje postavi kao trenutno stanje trokutastog pokazivača.

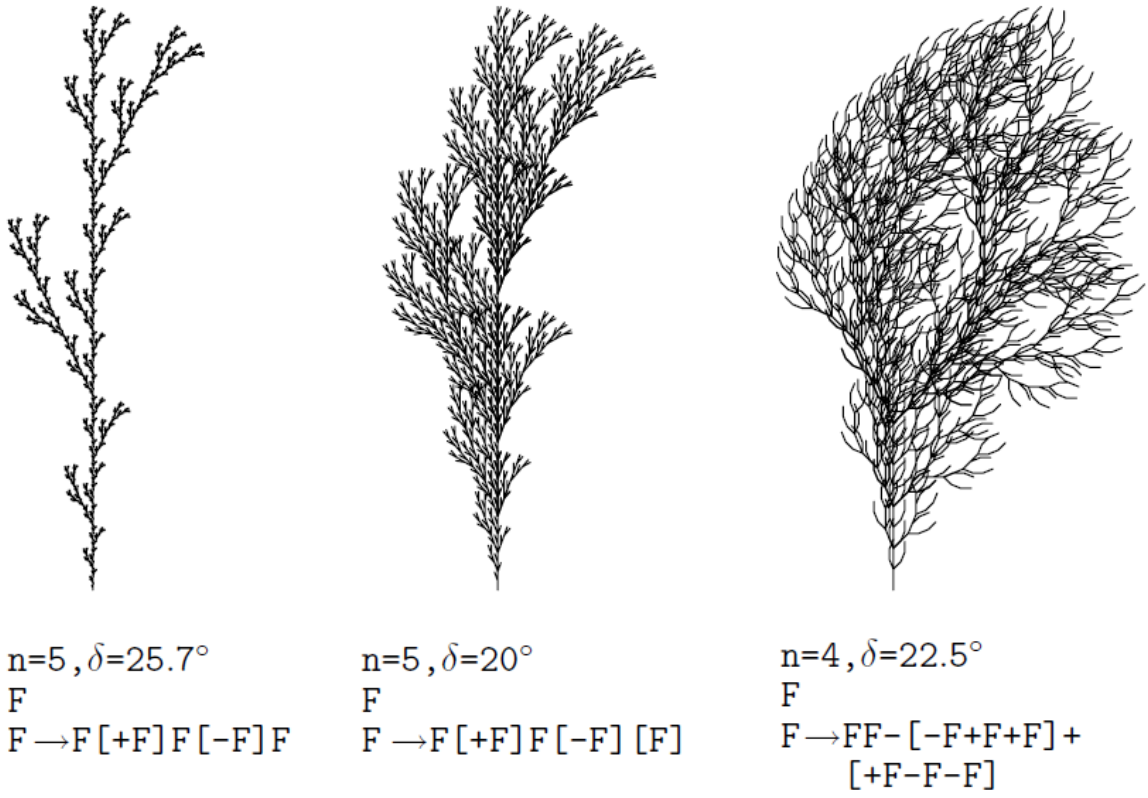
Primjer drva sa centralnom osi i pripadajućeg niza znakova prikazan je na slici 5.



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

Slika 5. Prikaz drva sa centralnom osi pomoću niza znakova sa uglatim zagradama

Primjeri struktura koje mogu nastati pomoću OL-sustava s uglatim zagradama dane su na slici 6.



Slika 6. Razne strukture generirane iz OL-sustava s uglatim zagradama

2.3.3. Interpretacija u 3D

Temeljni koncept interpretacije u 3D je predstavljanje orijentacije trokutastog pokazivača u prostoru pomoću tri vektora $\vec{H}$, $\vec{L}$, $\vec{U}$, koji su smjer trokutastog pokazivača, smjer lijevo i smjer gore, respektivno. Ovi vektori su jedinične dužine, međusobno su okomiti i zadovoljavaju jednadžbu $\vec{H} \times \vec{L} = \overrightarrow{U}$. Rotacije trokutastog pokazivača su izražene jednadžbom

$$[\vec{H}'\ \vec{L}'\ \vec{U}'] = [\vec{H}\ \ \vec{L}\ \ \vec{U}]\boldsymbol{R}$$

pri čemu je **R** rotacijska matrica 3 × 3. Rotacije oko vektora $\vec{H}$, $\vec{L}$, $\vec{U}$, predstavljene su sljedećim matricama:

$$\boldsymbol{R_U}(\alpha) = \begin{bmatrix} \cos\alpha & \sin\alpha & 0 \\ -\sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\boldsymbol{R_L}(\alpha) = \begin{bmatrix} \cos\alpha & 0 & -\sin\alpha \\ 0 & 1 & 0 \\ \sin\alpha & 0 & \cos\alpha \end{bmatrix}$$

$$\boldsymbol{R_U}(\alpha) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha \\ 0 & \sin\alpha & \cos\alpha \end{bmatrix}$$

Sljedeći simboli kontroliraju orijentaciju trokutastog pokazivača u prostoru :

\+ : skreni lijevo za kut *δ,* koristeći $\boldsymbol{R_U}(\delta)$

\- : skreni desno za kut *δ,* koristeći $\boldsymbol{R_U}(-\delta)$

& : okreni se prema gore za kut *δ,* koristeći $\boldsymbol{R_L}(\delta)$

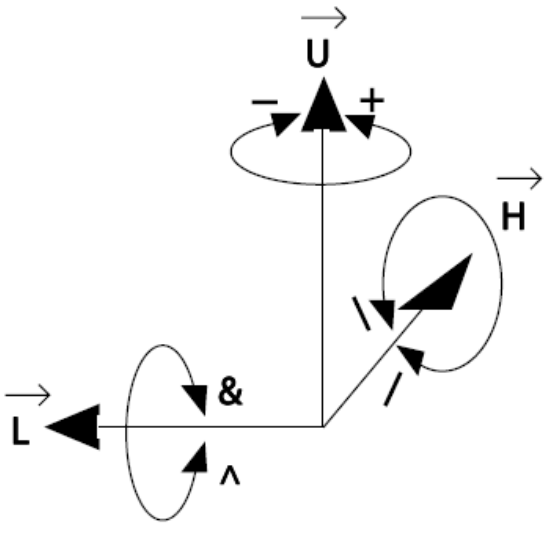
^ : okreni se prema dolje za kut *δ,* koristeći $\boldsymbol{R_L}(-\delta)$

\ : povećaj nagib u lijevo za kut *δ,* koristeći $\boldsymbol{R_H}(\delta)$

/ : povećaj nagib u desno za kut *δ,* koristeći $\boldsymbol{R_H}(-\delta)$

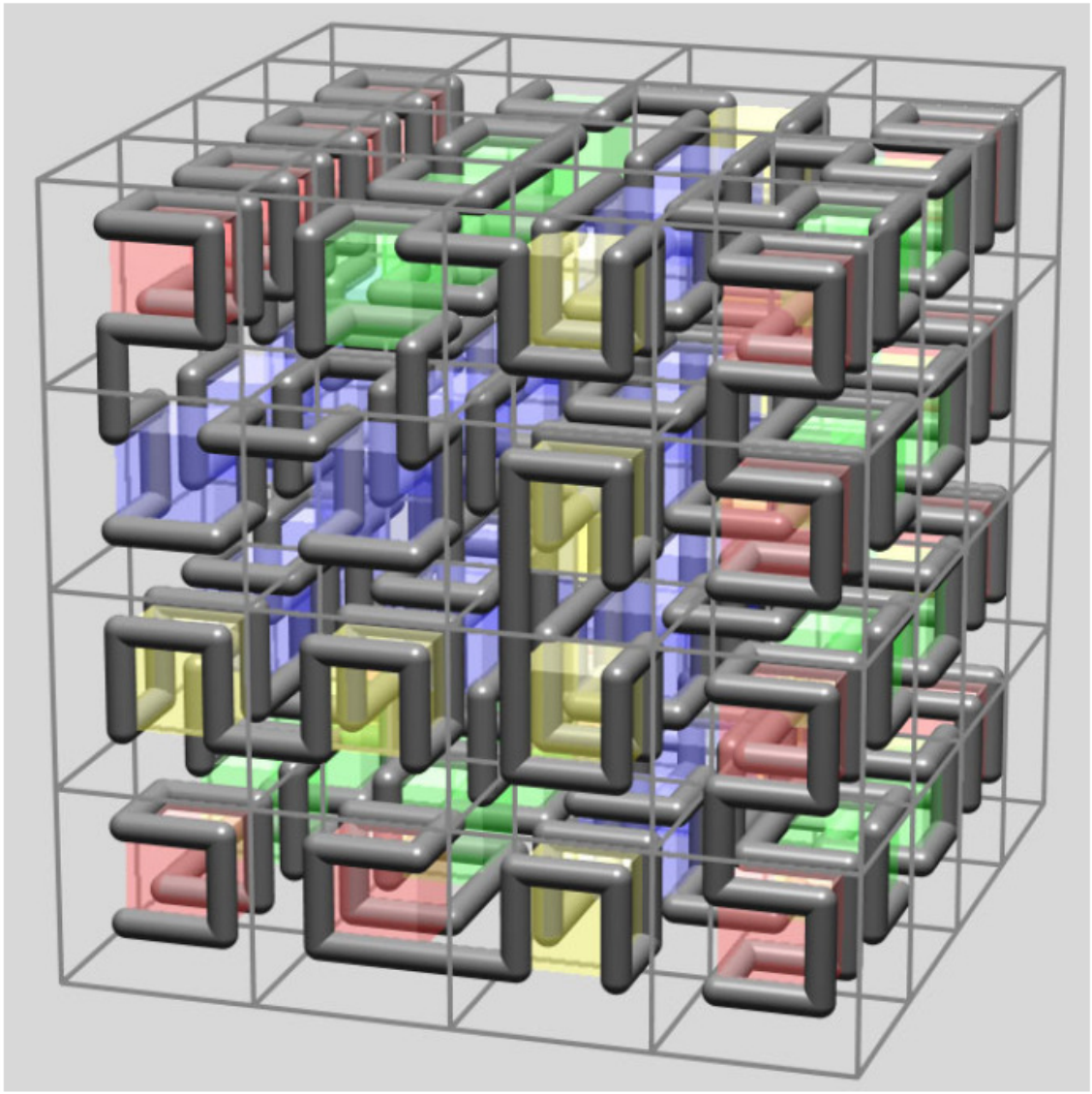
| : okreni se u suprotni smjer, koristeći $\boldsymbol{R_U}(180°)$

Slika 7 prikazuje orijentacione vektore $\vec{H},\ \vec{L},\ \vec{U}$, i efekte koje kontrolirajući simboli imaju na orijentaciju.



Slika 7. Kontroliranje turtle pokazivača u 3D

Slika 8 prikazuje proširenje Hilbertove krivulje u treću dimenziju kao primjer trodimenzionalnog objekta stvorenog pomoću interpretacije L-sustava grafikom trokutastog pokazivača.



n=2, δ=90°
A
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//

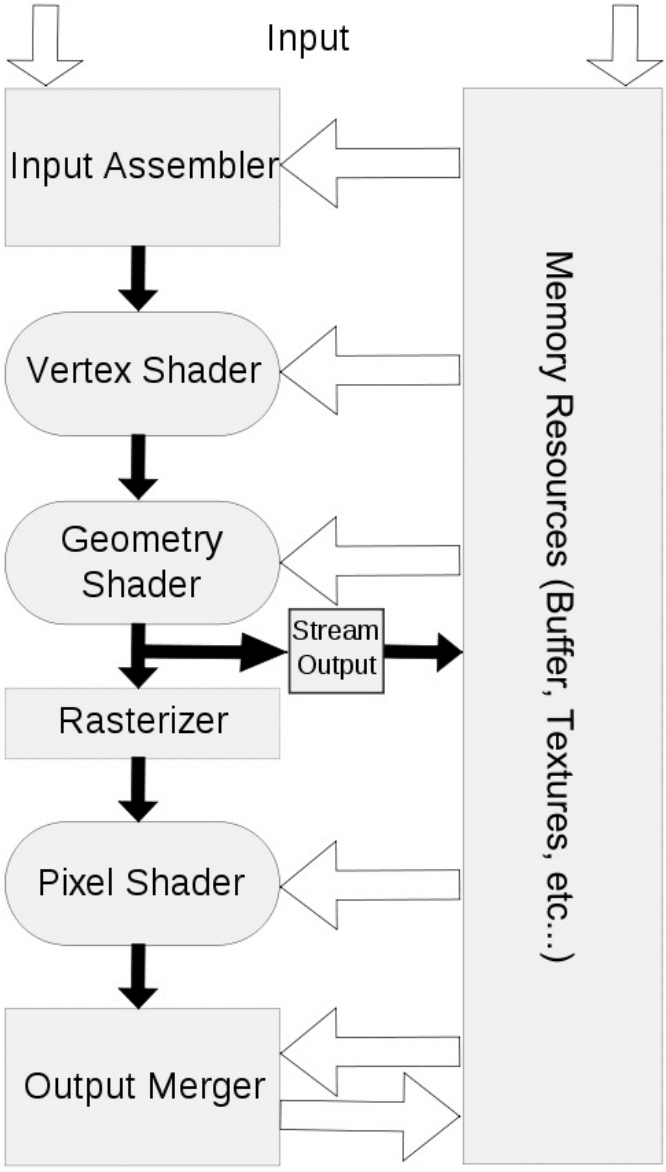
Slika 8. Trodimenzionalno proširenje Hilbertove krivulje

3. Programska implementacija

3.1. Korištene tehnologije

Programska implementacija izrađena je u razvojnom okruženju Microsoft Visual C++ 2008 Express Edition.

Za iscrtavanje objekata korišten je Microsoft Direct3D 10 koji je dio Microsoft DirectX API (Application Programming Interface). Direct 3D 10 omogućuje sklopovsku akceleraciju i potpunu programibilnost 3D grafičkog protočnog cjevovoda pomoću programa za sjenčanje (engl. shadera) verzije 4.0.



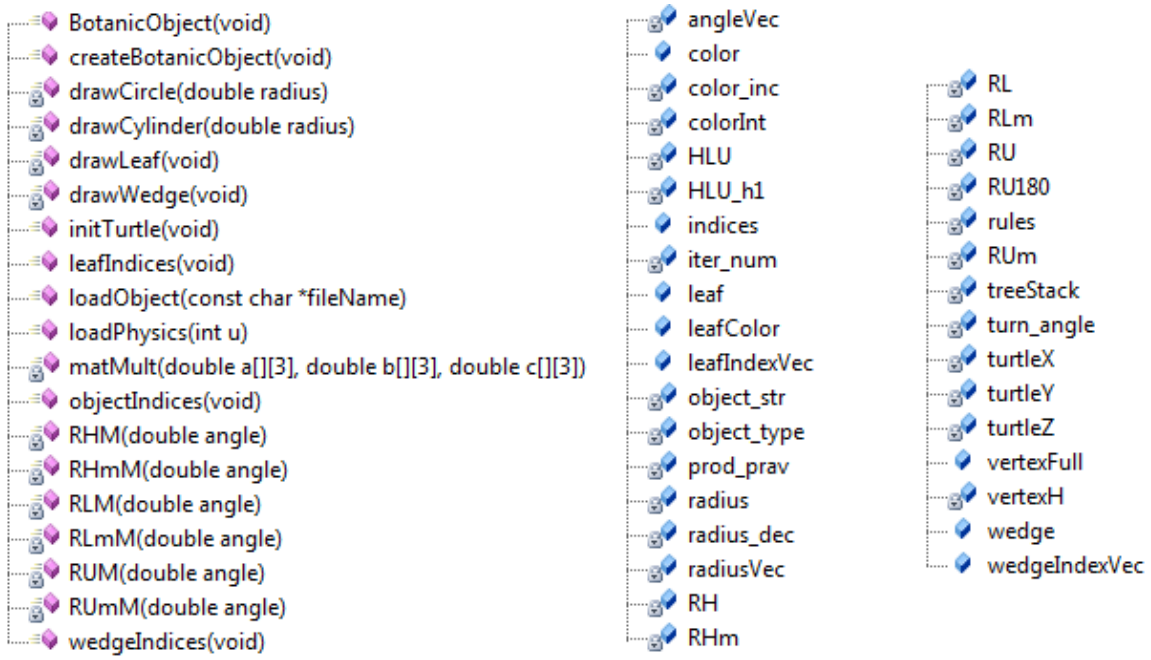
Slika 9. Proces grafičkog cjevovoda u Direct3D 10

Na slici 9 prikazan je proces grafičkog cjevovoda u Direct3D 10. Faze tog procesa su sljedeće:

1. **Input Assembler**: Učitava podatke o vrhovima iz međuspremnika vrhova u aplikaciji i šalje ih niz cjevovod.
2. **Vertex Shader**: Izvodi na jednom po jednom vrhu operacije transformacije, osvjetljenja, ...
3. **Geometry Shader**: Procesira cijele primitive kao što su trokuti, točke ili linije. Danu primitivu u ovoj se fazi ili zanemaruje ili se iz nje stvara jedna ili više primitiva.
4. **Stream Output**: Može zapisati rezultate prošle faze u memoriju, što je korisno za recirkulaciju podataka niz cjevovod.
5. **Rasterizer**: Konvertira primitive u slikovne elemente, ubacujući ih u program za sjenčanje slikovnih elemenata (engl. pixel shader). Rasterizer može izvoditi i druge zadatke, primjerice odsijecanje (engl. clipping) ili interpolaciju vrhova u podatke po pojedinom slikovnom elementu.
6. **Pixel shader**: Određuje konačnu boju slikovnog elementa koji će se zapisati u prikazni sustav i može izračunati dubinsku vrijednost, koja se zapisuje u međuspremnik dubine.
7. **Output Merger**: Spaja razne tipove izlaznih podataka (vrijednosti rezultata programa za sjenčanje slikovnih elemenata, međuspremnik dubine ...) da formira konačan rezultat.

3.2. Klasa BotanicObject

Srce cijele programske implementacije klasa je BotanicObject. U toj klasi učitava se objekt iz datoteke, inicijalizira se grafika trokutastog pokazivača, te se stvaraju svi vrhovi botaničkog objekta i pripadni indeksi. Dizajn klase ne ovisi o API-u, pa se može koristiti i uz drugi API za 3D grafiku, npr. OpenGL. U nastavku se opisuju glavne metode ove klase. Na slici 10 prikazan je dijagram klase BotanicObject.



Slika 10. Dijagram klase BotanicObject

3.2.1. Metoda loadObject

Metoda loadObject učitava botanički objekt iz datoteke zadane kao parametar. Za potrebe definiranja biljke uvodi se četiri nova simbola za kontrolu trokutastog pokazivača.

! : dekrementiraj radijus segmenta

' : inkrementiraj trenutni intenzitet boje

f : pomakni se u smjeru trokutnog pokazivača i nacrtaj liniju lista

w : pomakni se u smjeru trokutnog pokazivača i nacrtaj liniju latice

Objekt mora biti zapisan u datoteci na sljedeći način:

1. x → prvi red, cijeli broj, predstavlja broj iteracija primjene produkcijskih pravila na početni niz
2. x → drugi red, decimalni broj, predstavlja početnu promjenu kuta skretanja
3. x → treći red, decimalni broj, predstavlja početni radijus elemenata biljke
4. x → četvrti red, decimalni broj, predstavlja moguće smanjenje radijusa (ako je zapisano u produkcijama)
5. x → peti red, decimalni broj, predstavlja početni intenzitet boje
6. x → šesti red, decimalni broj, predstavlja moguće povećanje intenziteta boje
7. ω → sedmi red, niz znakova, predstavlja početni niz botaničkog objekta

8. p_1 → osmi red, niz znakova (oblika znak→niz_znakova), predstavlja prvu produkciju

...

n. p_n → n-ti red, niz znakova (oblika znak→niz_znakova), predstavlja n-tu produkciju

Dane znakove s lijeve strane produkcija, koji čine abecedu gramatike, niz znakova s desne strane produkcija, sam niz znakova koji će predstavljati konačni objekt (prvo u njega stavljamo početni niz znakova) i sve druge varijable iz datoteke stavljamo svaki u poseban spremnik. Iz početnog niza, sljedećim algoritmom dobivamo konačan niz:

```
for (int i = 0; i<broj_iteracija; i++)
{
	if((pos1=object_str.find_first_of(abc))!=string::npos)
	{
		rule_pos = abc.find(object_str[pos1]);
		object_str.replace(pos1, 1, rules[rule_pos]);
	}
	do{
		pos2=pos1;
		if((pos1=object_str.find_first_of(abc, pos2 +
			rules[rule_pos].size()))!=string::npos)
		{
			rule_pos = abc.find(object_str[pos1]);
			object_str.replace(pos1, 1, rules[rule_pos]);
		}
	}while(pos1!=string::npos);
}
```

Dakle u svakoj iteraciji, potražimo u *object_str*, koji predstavlja spremnik konačnog niza, prvo pojavljivanje nekog znaka abecede. Kad nađemo prvo pojavljivanje nekog znaka abecede postavljamo pokazivač na tu poziciju, primjenjujemo odgovarajuću produkciju, odnosno mjenjamo taj znak nizom znakova. S obzirom da u se u svakoj iteraciji L-sustava paralelno primjenjuju sve produkcije, mičemo pokazivač za duljinu upravo primjenjene produkcije i tražimo sljedeći znak abecede na koji ćemo primjeniti produkciju. To ponavljamo dok pokazivač ne dođe do kraja spremnika konačnog niza.

3.2.2. Metoda initTurtle

U metodi *initTurtle* postavlja se početna pozicija trokutastog pokazivača, početne vrijednosti matrica HLU, koja predstavlja matricu vektora orijentacije trokutastog pokazivača i početne vrijednosti matrica RU, RUm, RU180, RL, RLm, RH i RHm, gdje matrice s dodatkom *m* imaju kut negativnog predznaka.

3.2.3. Metoda createBotanicObject

Metoda *createBotanicObject* sastoji se od jednog switch-case mehanizma unutar while petlje. Ta while petlja "kreće" se znak po znak od početka do kraja konačnog niza, a switch-case mehanizam šalje pojedini znak odgovarajućem izvršnom dijelu koda. Za znakove se radi sljedeće:

F : prvo poziva metodu *drawCircle* koja stavlja 11 vrhova oko početne lokacije trokutastog pokazivača u spremnik vrhova, čineći tako prvu bazu prvog cilindra. Ostali pozivi rješavaju se na isti način, tako da se pomoću metode *drawCylinder* pomiče trokutasti pokazivač u smjeru vektora $\vec{H}$, i oko te lokacije ponovo puni spremnik s 11 vrhova, čineći tako drugu bazu prethodnog odnosno prvu bazu sljedećeg cilindra. Ako je pokazivač mijenjao smjer, prije upisivanja 11 novih vrhova upisuje se kontrolni broj kako bi se izbjegli nepoželjni efekti pri iscrtavanju. Na taj način pomoću znaka *F* iscrtavaju se stabljike i grane raznih biljaka.

f : pomoću metode *drawLeaf* prvo se pomiče trokutni pokazivač u smjeru vektora $\vec{H}$, potom se upisuje vrh na toj lokaciji u spremnik vrhova lišća. Na taj način pomoću znaka *f* iscrtava se lišće raznih biljaka.

w : pomoću metode *drawWedge* prvo se pomiče trokutni pokazivač u smjeru vektora $\vec{H}$, potom se upisuje vrh na toj lokaciji u spremnik vrhova latica. Na taj način pomoću znaka *w* iscrtavaju se latice raznih biljaka.

+, -, &, ^, \, /, | : množi se matrica HLU sa matricom odgovarajućeg znaka i rezultat se sprema u HLU. Na taj način trokutasti pokazivač mijenja orijentaciju .

[: stavlja na stog trenutnu poziciju trokutastog pokazivača, matricu HLU, radijus elementa i intenzitet boje.

] : podiže s vrha stoga poziciju trokutastog pokazivača, matricu HLU, radijus elementa i intenzitet boje.

! : smanjuje radijus elementa za veličinu *radius_dec*.

' : pojačava intenzitet boje za *color_inc*.

3.2.3. Metode objectIndices, leafIndices, wedgeIndices

Ove metode popunjavaju odgovarajuće spremnike indeksa, tako da se mogu iscrtati topologijom *trianglestrip*. S obzirom da se indeksiraju posebno, same grane ili stabljika, lišće i latice, tretiraju se kao odvojeni entiteti prilikom iscrtavanja.

3.2.4. Metoda loadPhysics

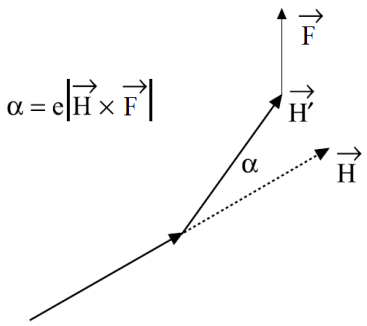
Metoda *loadPhysics* preslika je metode *createBotanicObject* uz izmjenu da ne popunjava dodatno spremnike vrhova, lišća i latica, već ih prepisuje novim vrijednostima, koje se dobivaju tako da se u slučaju promjene orijentacije trokutnog pokazivača mijenja i kut pomaka pokazivača. To se događa na sljedeći način:

```
angle = 0.001  * (1-radius) * sqrt(tempX*tempX +
            tempY*tempY + tempZ*tempZ);
turn_angle += angle;
```

Dakle trenutnoj vrijednosti kuta pomaka pokazivača pribraja se vrijednost *angle*. Ta vrijednost dobivena je iz heurističke formule

$$\alpha = 0.001 \cdot (1 - radius) \cdot \left|\vec{H} \times \vec{F}\right|$$

pri čemu je *angle* označen sa α, a vektor $\vec{F}$ predstavlja vektor sile koja djeluje na cijelu biljku, npr. vjetar ili gravitacija. Ova formula motivirana je fizikom. Ako $\vec{F}$ interpretiramo kao silu na vrh vektora $\vec{H}$ i ako $\vec{H}$ može rotirati oko svoje početne točke, onda je $\vec{H} \times \vec{F}$ moment sile. Slika 11 prikazuje promjenu orijentacije trokutastog pokazivača zbog utjecaja sile.



Slika 11. Promjena orijentacije trokutastog pokazivača zbog utjecaja sile

S obzirom da promjena kuta ovisi o radijusu (vrijednosti od 0 do1), deblje grane ujedno i tamnije predstavljaju starije grane i manje se savijaju od tanjih, odnosno mlađih grana. Također kod lišća, s obzirom da nemaju radijus, boja određuje podložnost savijanju, tako da se svjetliji, odnosno mlađi listovi savijaju lakše od tamnijih, odnosno starijih listova. Može se reći da boja i radijus u ovakvoj implementaciji predstavljaju masu pojedinih elemenata objekta.

3.3. Klasa DXManager

Klasa *DXManager* inicijalizira Direct3D i iscrtava botanički objekt. Unutar ove klase postoje i metode za promjenu lokacije pogleda na scenu, te se povezuje simulacija fizike iz klase *BotanicObject* sa glavnim dijelom programa koji obrađuje događaje s tipkovnice.

Vrhovi objekta, lišća i latica, u metodi *renderScene* ove klase popunjavaju međuspremnik vrhova, a indeksi objekata, lišća i latica popunjavaju međspremnik indeksa. Nakon toga se sa tri poziva metode *DrawIndexed* iscrtava dani objekt na sustavu prikaza.

3.4. Korisničko sučelje

Korisničko sučelje implementirano je preko tipkovnice na sljedeći način:

- tipke F1 do F4 mijenjaju lokaciju pogleda na scenu
- tipke F5 do F8 mijenjaju botanički objekt koji se prikazuje
- tipke F9 do F12 pokreću simulaciju fizike, tj. puhanje vjetra raznih jakosti

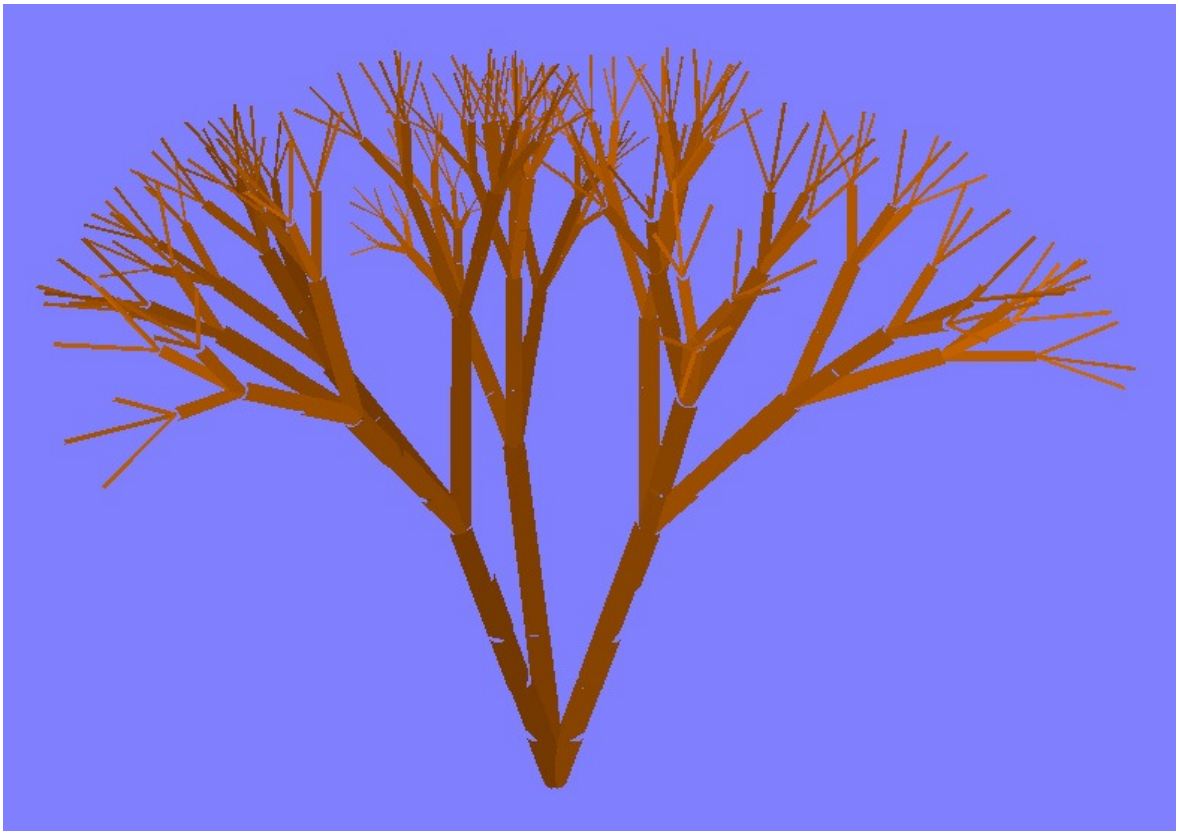
3.5. Rezultati

Primjer 1

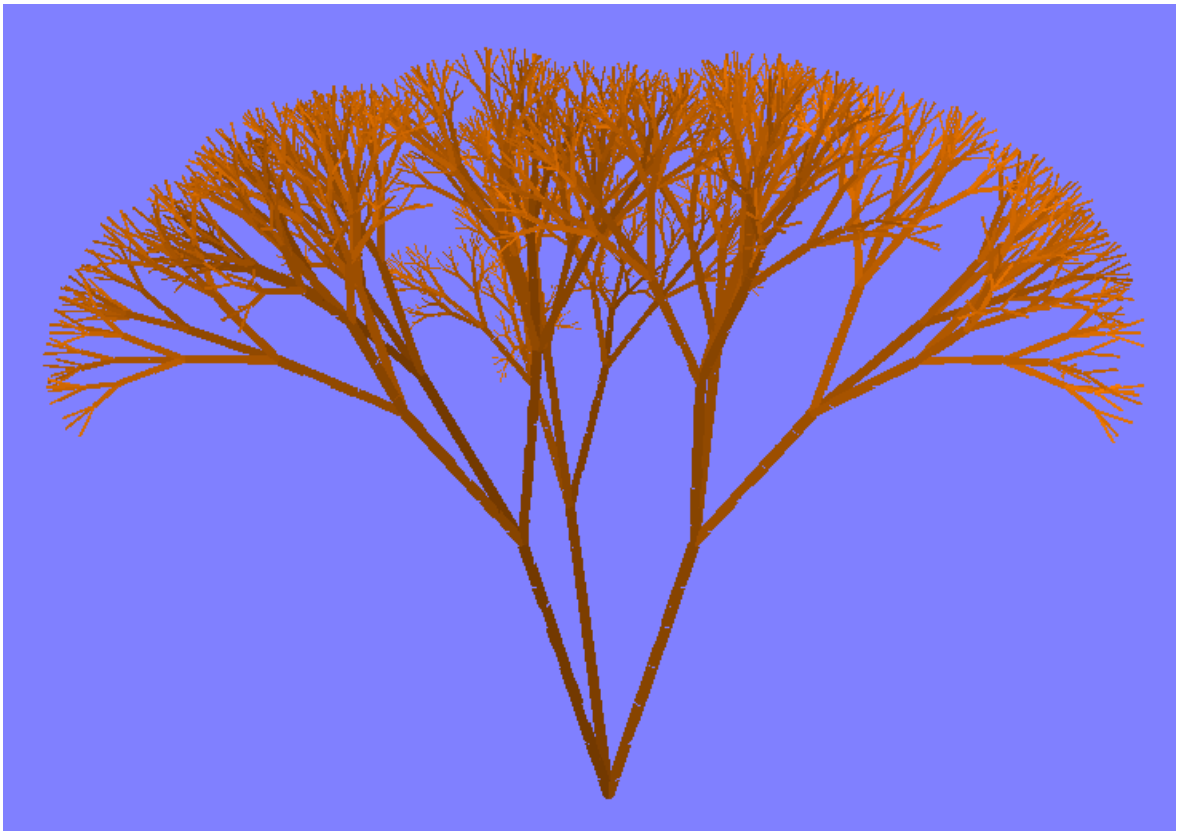
Uz zadane sljedeće parametre u ulaznoj datoteci

5 (12a) / 7 (12b), 22.5, 0.20, 0.045, 0.45, 0.04
A
A → [&FL!A]/////'[&FL!A]///////'[&FL!A]
F → S/////F
S → FL
L → ["'^^{-f+f+f-|-f+f+f}]

i bez iscrtavanja lišća dobivamo grmolik botanički objekt na slici 12 i 13. Na slici 13 lokacija pogleda na scenu je tri puta bliža nego na slici 13



Slika 12. Botanički objekt dobiven uz parametre iz primjera 1



Slika 13. Botanički objekt dobiven uz parametre iz primjera 1

Primjer 2

Uz zadane sljedeće parametre u ulaznoj datoteci

7, 22.5, 0.25, 0.035, 0.45, 0.04
A
A → [&FL!A]/////'[&FL!A]///////'[&FL!A]
F → S/////F
S → FL
L → ['"^^{-f+f+f-|-f+f+f}]

dobivamo grmolik botanički objekt na slici 14.

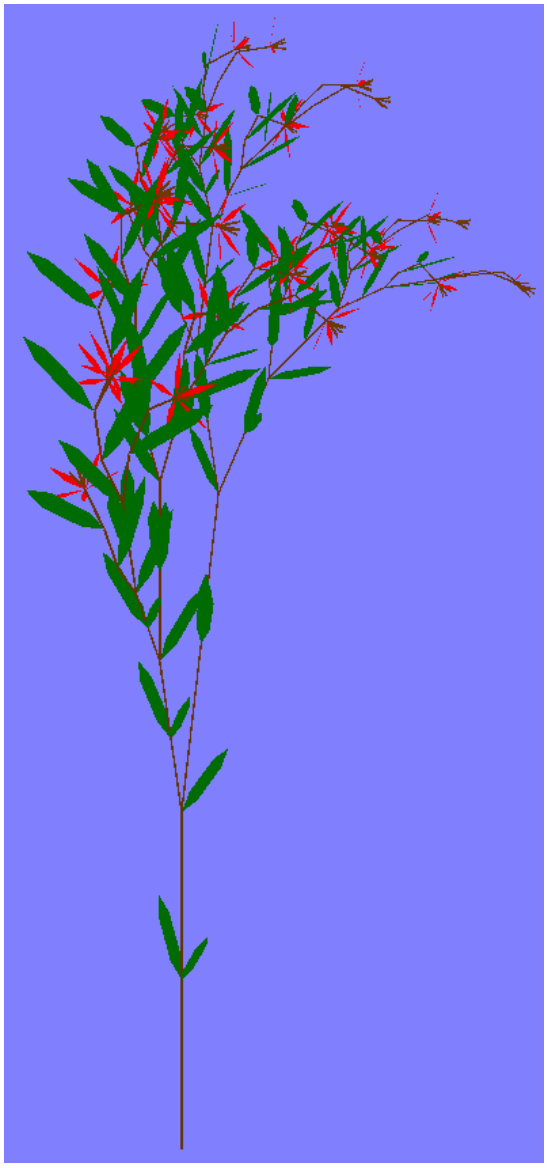
Slika 14. Botanički objekt dobiven uz parametre iz primjera 2

Primjer 3

Uz zadane sljedeće parametre u ulaznoj datoteci

5, 18, 0.05, 0.01, 0.4, 0.1
P
P->I+[P+C]--//[--L]I[++L]-[PC]++PC
I->FS[//&&L][//^^L]FS
S->SFS
L->['{+f-ff-f+|+f-ff-f}]
C->[&&&G'/W////W////W////W////W]
G->FF
W->['^F][{&&&&-w+w|-w+w}]

dobivamo botanički objekt na slici 15.



Slika 15. Botanički objekt dobiven uz parametre iz primjera 3

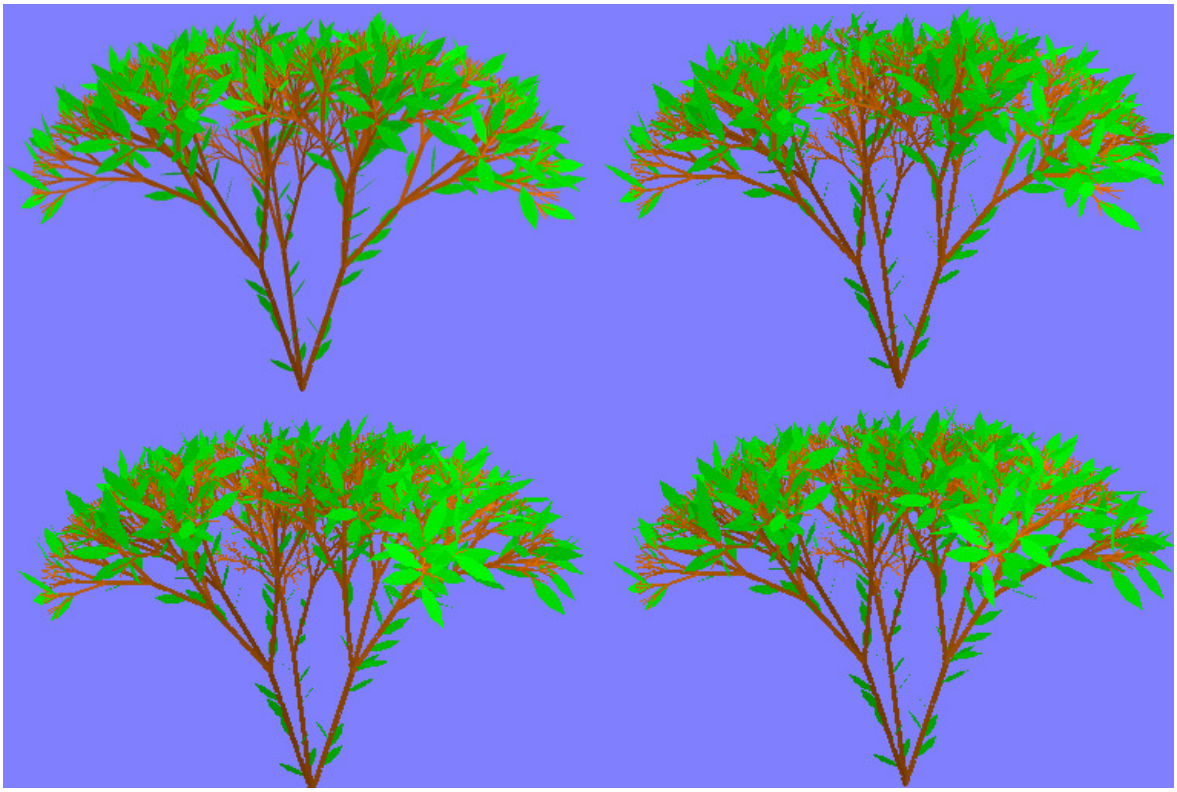
Puhanje vjetra implementirano je na sljedeći način:

```
for (int i = 0; i<150; i++)
{
      tree.loadPhysics(i);
      renderScene();
}

for (int i = 150; i>0; --i)
{
      tree.loadPhysics(i);
      renderScene();

}
```

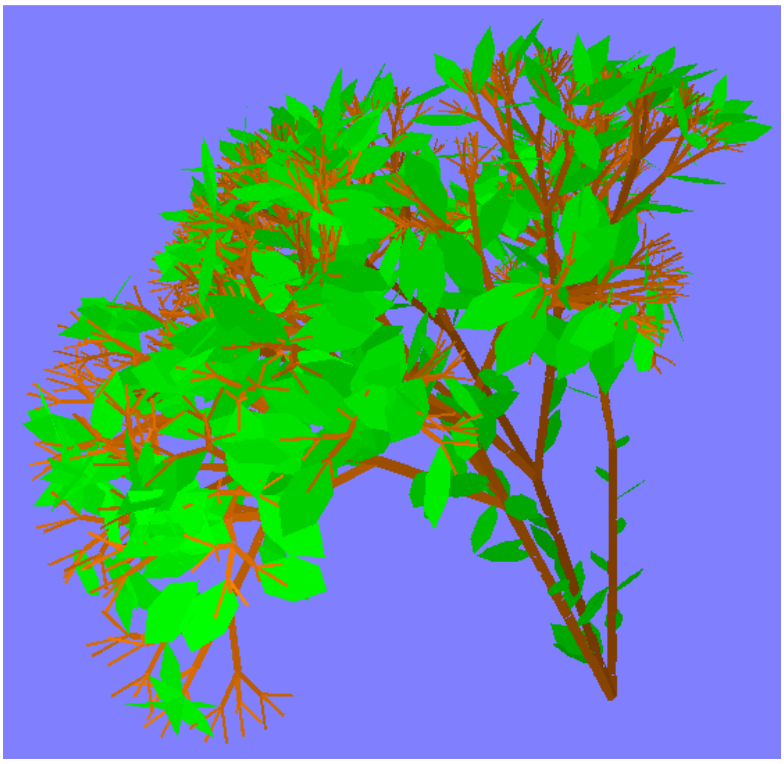
Dakle prvo pojačavamo silu za određeni iznos i potom je smanjujemo, pri tome se dobiva dojam laganog povjetarca.

Ovakva implementacija simulacije fizike pokazala se iznimno stabilnom. Bez obzira danu na jačinu sile, elementi objekta ostali su povezani.

Primjer 4, fizikalna simulacija puhanja vjetra na modelu iz primjera 2 prikazana je kao niz slika, na slici 16.



Slika 16. Fizikalna simulacija modela iz primjera 2



Slika 17. Deformacija botaničkog objekta iz primjera 2 prilikom puhanja snažnog vjetra

Na slici 17, prikazana je deformacija botaničkog objekta iz primjera 2 prilikom puhanja snažnog vjetra, odnosno primijenjene velike sile.

Rezultati mjerenja trajanja izvođenja programa, zapisani u tablici 1 pokazali su očekivani vrijednosti s obzirom na veću složenost objekta u primjeru 3. No pomnijim mjerenjem neočekivano se pokazalo da je upravo metoda *loadObject* odgovorna za drastično povećanje vremena izvođenja nakon desete iteracija u oba slučaja. Od 46 sekundi koliko traje izvođenje primjera 2, metoda *loadObject* traje 45 sekundi i analogno u primjeru 3.

Tablica 1. Rezultati mjerenja trajanja izvođenja programa

	Vrijeme mjerenja (~sek)	
Broj iteracija	Primjer 2 (grm)	Primjer 3 (grm s cvijećem)
7	0	0
8	0	1
9	1	2
10	6	15
11	46	>120

Očigledno je da daljnja poboljšanja programske implementacije moraju biti u smjeru razrade kvalitetnijeg algoritma učitavanja botaničkog objekta. Ostale komponente pokazale su dostatnu brzinu pri testiranom broju iteracija, a daljnje testiranje onemogućio je broj vrhova koji je u slučaju primjera 3, već za 11 iteracija bio oko 10 000 000. Mjerenje je vršeno na računalu sa CPU Intel i7 920, GPU Radeon 4850 1GB, 6 GB RAM, 7200 ok/min HDD.

Program se može prilagoditi da učitava i kompleksnije botaničke objekte. Poboljšanje realnosti fizikalne simulacije jedino je moguće uz razvijanje sustava puno složenijeg od ovdje implementirane jednostavne heurističke funkcije.

4. Zaključak

L-sustavi i njihova implementacija grafikom trokutastog pokazivača, moćan je alat za prikaz botaničkih objekata. Napravljena programska implementacija pokazala se dostatnom za srednje složene L-sustave. Programska implementacija nije ograničena samo na iscrtavanje botaničkih objekata, L-sustavi koji oblikuju razne prostorno popunjujuće krivulje i fraktale također se mogu generirati.

Dva obrađivana primjera u stvarnom vremenu izvodila su simulacije fizike. Fizikalnoj simulaciji manjka realnosti, što je očekivano s obzirom na jednostavnom implementacije. Čak i s tim nedostatkom postižu se dovoljno realne simulacije puhanja vjetra. Naprednije simulacije fizike u botaničkim objektima vrlo su kompleksne, pa je stoga i njihova implementacija iznimno zahtjevna (detaljnije u [3]).

5. Popis tablica

6. Popis slika

7. Literatura

[1] P. Prusinkiewicz, A. Lindenmayer. The Algorithmic Beauty of Plants. New York: Springer-Verlag, 1990.

[2] B. Anguelov, Bobby Anguelov's Blog, http://takinginitiative.wordpress.com, studeni 2009.

[3] J. Taylor-Hell, Biomechanics in Botanical Trees http://algorithmicbotany.org/papers/juliath.th2005.pdf, 2005.

[4] Direct3D graphics, http://msdn.microsoft.com/en-us/library/ee415646%28VS.85%29.aspx, studeni 2009.

8. Sažetak

Algoritamska botanika

U radu se obrađuje problematika prikaza botaničkih elemenata zapisanih pomoću L-sustava. Za implementaciju korištena je grafika trokutastog pokazivača u kombinaciji s Direct3D API. U radu je također opisana i razrađena fizikalna simulacija promjena uslijed djelovanja vjetra za zadani botanički objekta u stvarnom vremenu.

Ključne riječi: L-sustav, Algoritamska botanika, grafika trokutastog pokazivača, Direct3D 10

Abstract

Algorithmic botany

In this work the problematics of rendering botanical elements described by L-systems is analyzed. The L-systems are implemented in turtle graphics, with Direct 3D API. The physics simulation of botanical objects is also described and implemented using the force of wind as an example.

Keywords: L-system, Algorithmic botany, Turtle graphics, Direct3D 10