Modeliranje izgleda i procesa rasta biljnih organizama

Ćosić, Marin

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU

FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

Sveučilišni studij

MODELIRANJE IZGLEDA I PROCESA RASTA BILJNIH ORGANIZAMA

Završni rad

Marin Ćosić

Osijek, 2019.

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

Obrazac Z1P - Obrazac za ocjenu završnog rada na preddiplomskom sveučilišnom studiju

Osijek, 16.09.2019.

Odboru za završne i diplomske ispite

Prijedlog ocjene završnog rada

Ime i prezime studenta:	Marin Ćosić
Studij, smjer:	Preddiplomski sveučilišni studij Računarstvo
Mat. br. studenta, godina upisa:	R3906, 26.09.2018.
OIB studenta:	79318555478
Mentor:	izv. prof. dr.sc. Josip Job
Sumentor:	
Sumentor iz tvrtke:	
Naslov završnog rada:	Modeliranje izgleda i procesa rasta biljnih organizama
Znanstvena grana rada:	**Obradba informacija (zn. polje računarstvo)**
Predložena ocjena završnog rada:	Izvrstan (5)
Kratko obrazloženje ocjene prema Kriterijima za ocjenjivanje završnih i diplomskih radova:	Primjena znanja stečenih na fakultetu: 2 bod/boda Postignuti rezultati u odnosu na složenost zadatka: 2 bod/boda Jasnoća pismenog izražavanja: 2 bod/boda Razina samostalnosti: 3 razina
Datum prijedloga ocjene mentora:	16.09.2019.
Datum potvrde ocjene Odbora:	25.09.2019.

Potpis mentora za predaju konačne verzije rada u Studentsku službu pri završetku studija:	Potpis:
	Datum:

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA
I INFORMACIJSKIH TEHNOLOGIJA **OSIJEK**

IZJAVA O ORIGINALNOSTI RADA

Osijek, 30.09.2019.

Ime i prezime studenta:	Marin Ćosić
Studij:	Preddiplomski sveučilišni studij Računarstvo
Mat. br. studenta, godina upisa:	R3906, 26.09.2018.
Ephorus podudaranje [%]:	13

Ovom izjavom izjavljujem da je rad pod nazivom**: Modeliranje izgleda i procesa rasta biljnih organizama**

izrađen pod vodstvom mentora izv. prof. dr.sc. Josip Job

i sumentora

moj vlastiti rad i prema mom najboljem znanju ne sadrži prethodno objavljene ili neobjavljene pisane materijale drugih osoba, osim onih koji su izričito priznati navođenjem literature i drugih izvora informacija.
Izjavljujem da je intelektualni sadržaj navedenog rada proizvod mog vlastitog rada, osim u onom dijelu za koji mi je bila potrebna pomoć mentora, sumentora i drugih osoba, a što je izričito navedeno u radu.

Potpis studenta:

SADRŽAJ

1. UVOD

Ljepota biljaka privlači ljudsku pozornost otkada postoji čovječanstvo. Oduvijek je svojim izgledom bila inspiracija različitim sferama života, od dizajna odjevnih predmeta pa sve do izgleda gradova. Objašnjenje također leži u tome zašto nam je jedno ljudsko lice ljepše od drugog ili zašto ne možemo skinuti pogled s određene zgrade, a to je simetrija. Kao što bi Hermann Weyl rekao „Ljepota je povezana sa simetrijom.“[1] Navedena pojava je primijećena još u antičko doba, ali matematičari nisu stali samo na tome nego su proširili to saznanje i na ostatak biljnog svijete koji nije na prvi pogled simetričan. Promatrajući rast biljaka rješenje se pronašlo u samosličnosti i različitim geometrijskim oblicima koji se pojavljuju tijekom razvoja.

U drugom poglavlju će se pričati o prvim idejama modeliranja koje su činile osnovu za ostatak napretka. U drugom potpoglavlju će se spomenuti i objasniti načini modeliranja, najpoznatiji pristup L-sustav, će se detaljnije obraditi u trećem potpoglavlju. Pri opisu sustava objasnit će se metoda prepisivanja, vizualno prikazivanje pomoću kornjače te njegov najjednostavniji oblik DOL-sustav. U četvrtom potpoglavlju znanje će se proširiti na trodimenzionalno modeliranje, te će se kasnije navesti alati i parametri potrebni za prikaz, te će biti jedan primjer prikazan. Objasnit će se i inverzno modeliranje. U trećem poglavlju će se objasniti alati potrebi za prikazivanje biljaka u L-sustavu koji će se pokazati u drugom potpoglavlju. Alati su HTML, JavaScript i p5.js biblioteka.

3.1. Zadatak završnog rada

Cilj završnog rada je objasniti kako ovladati „slučajnošću“ prirode i kako prikazati složenost biljnog svijeta pomoću nekoliko parametara. Poznavanjem metoda koje su prikazane u radu skraćuje se vrijeme potrebno za rekreiranje drveća i ne zahtijeva ručno oblikovanje modela.

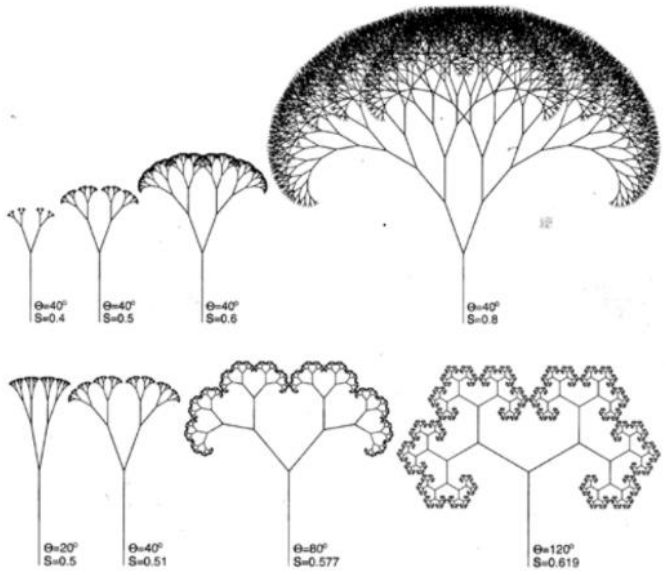
2. MODELIRANJE BILJNIH ORGANIZAMA

Glavni pojam koji se veže uz biljke je pojam grananja. Njezin izgled se često definira kao produkt slučajnosti. U jednu ruku to i je istina jer se ne može u potpunosti objasniti zašto pojedina biljka ima određeni izgled no ujedno se mogu dati određeni faktori (visina, širina grananja itd.) koji će dovesti do modela sličnim onima iz prirode. Ti faktori čine osnove modeliranja koji se mogu naknadno upotpuniti složenijim elementima radi potpune sličnosti, kao što su geotropizam, fototropizam itd.

2.1. Počeci modeliranja

Granje u prirodi je primijećeno ne samo kod biljaka nego i kod tokova rijeka, rasporeda krvnih žila u čovjekovom tijelu, rasporedu ogranaka pahuljica itd. Premda drveća imaju znatno učestaliju primjenu najviše se vremena proučavalo njihovu građu.

Jedni od prvih načina za dobivanje biljnih oblika bili su fraktali. Ako se aproksimaciju uzme da pojedinačni dijelovi drveta imaju jednaku strukturu kao i cijelo drvo onda ova metoda ima smisla. Hallé F., Oldemann R.A.A. i Tomlinson P.B.u fraktale objasnili kao idealan oblik za biljke, ako se kao najvažniji element gleda prijenos energije. Omogućuje brz prijenos vode kroz cijelu strukturu bez da se smanjuje količina površine potrebna za prikupljanje sunčeve energije.

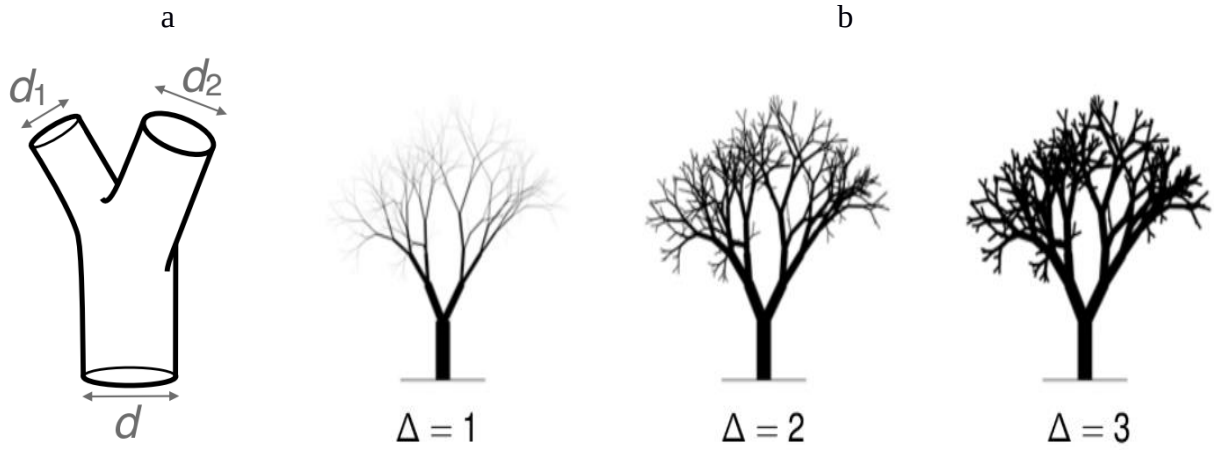


Slika 2.1. Fraktalni prikaz dvije različite biljke [3]

Ako se uzme u obzir debljina grana, dolazi se do zaključka da drveća nisu fraktali. Leonardo da Vinci je još da tezu da je u svakom stadiju debljina dječjih grana u zbroju jednaka debljini roditeljske grane.

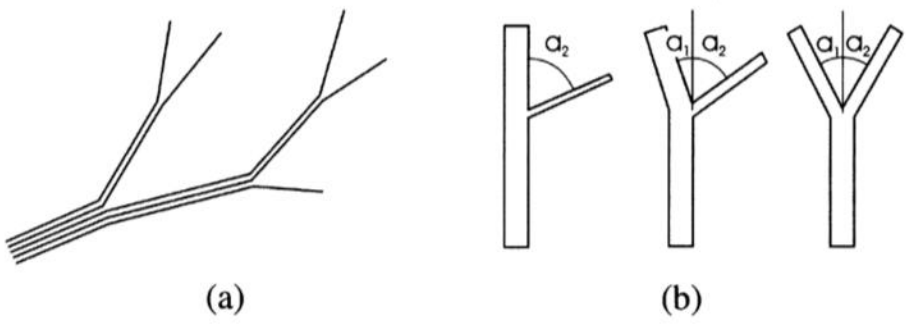
$$d^{\Delta} = d_1^{\Delta} + d_2^{\Delta} \tag{2-1}$$

Njegova procjena je da Δ iznosi 2, što je poprilično točno, ali pronađena je samo jedna vrsta za koju to vrijedi, za Δ = 2.2 pronađeno je 10 vrsta



Slika 2.2. a) Roditeljske i dječje grane
b) Varijacije izgleda biljke promjenom vrijednosti Δ [2]

Da Vinci je došao na tu ideju gledajući na biljku kao skupinu linija gdje se na svakom čvoru odvajaju u nove grane. Ujedno je to povezao i s izlaznim kutom, gdje bi novonastala grana imala manji kut ako je njena debljina veća od druge dječje grane.[2]



Slika 2.3. a) Grana kao kupina linija **b)** Odnos debljine i izlaznog kuta [3]

2.2. Metode modeliranja

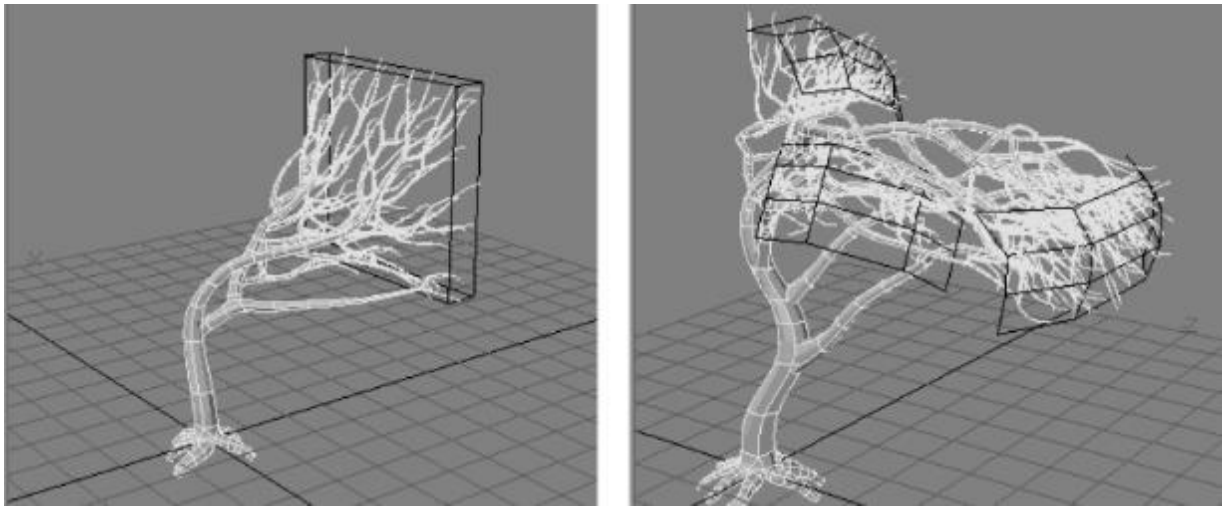
Dva glavna pristupa modeliranju su:

- lokalno u globalno
- globalno u lokalno

Ovisno o namjerama razvoja modela bira se jedan ili drugi pristup. Kod prvog pristupa modeliranje ima prirodan tijek kao što ima i u prirodi od sjemenke (kasnije poznato kao aksiom) do stabla. Kod globalnog u lokalno pristupa počinje se od nekog osnovnog oblika te se detalji dodaju kasnije. Način rada određenih metoda definiran je pomoću pristupa koje koriste, a metode su:

- Proceduralna (treesDesigner)
- Zasnovana na pravilima (L-sustav)
- Hibridna (Xfrog)

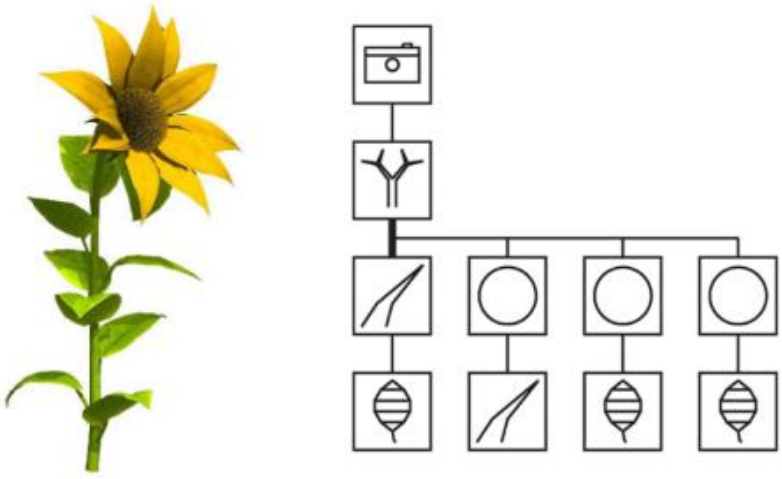
Globalno u lokalno pristup je nešto složenije, zbog pretvaranja globalan oblik u objekt nalik biljci. Metoda u kojoj je ovaj pristup dominantan je proceduralna metoda. Metoda i pristup su nešto složeniji te se rjeđe korite. Na slici 2.4. je prikazan proces izrade objekta pomoću treesDesigner dodatka koji je dostupan za LightWave3D program.



Slika 2.4. treesDesigner [3]

Metoda zasnovana na pravilima je tekstualnog oblika i sadrži elemente kao što su aksiom i pravila produkcije. Zasniva se na rekurziji i samosličnosti. Premda tekst vizualno ne predstavlja biljku, koristi alate koji sadrže interpretaciju putem kornjače koja pretvara tekstualni dio u vizualni prikaz. Omogućuje brzo stvaranje složenih oblika ali je teže dobiti konkretan oblik nego pomoću ostalih metoda. Premda jedina od navedene tri metode sadrži programersku podlogu u svom načinu rade dok su ostale više dizajnerski orijentirane, u radu će se fokusirati na njezino objašnjavanje.

Hibridna metoda ujedinjuje rad prethodno navedenih. Biljka je prikazana pomoću grafa strukture, struktura se sastoji od različitih dijelova koji inkapsuliraju određene procedure modeliranja. Procedure mogu stvarati sadržaj, umnažati sadržaj i izvoditi globalno modeliranje.[3]



Slika 2.5. Graf strukture suncokreta pomoću Xfroga [3]

2.3. L-SUSTAV

2.3.1. O sustavu

Lindenmayerov sustav(često skraćen kao L-sustav) je paralelni sustav rekurzivne naravi koji vodi do samosličnosti i fraktalnog izgleda. Prvotno je osmišljen za opisivanje jednostavnih višestaničnih organizama i ilustriranje susjednih odnosa između biljnih stanica, kasnije je nadograđen za prikaz većih biljaka i složenijih granatih struktura. Sustav je ujedno i tip formalne gramatike, što znači da se sastoji od abecede i simbola koji vode do produkcije, odnosno transformiranje niza znakova ili do unaprijed definiranih postupaka. Ilustrirana verzija se naziva interpretacija putem kornjače, a pojednostavljena verzija se naziva DOL-sustav [4].

Matematički se sustav može prikazati ovako (2-2):

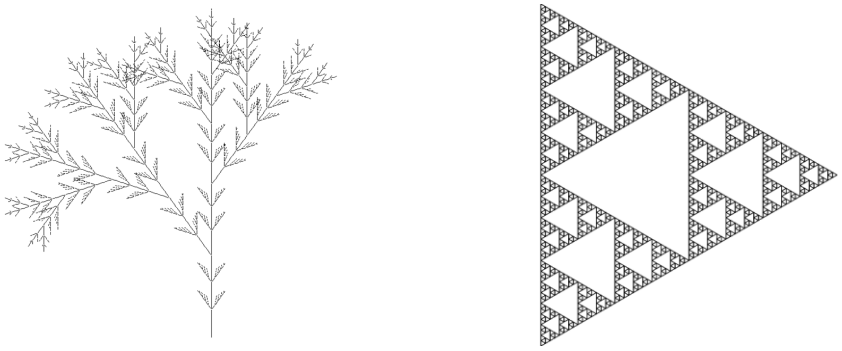
$$G = (V, \omega, P) \tag{2-2}$$

gdje je

- V (abeceda) predstavlja skup simbola koji sadrži zamjenjive (varijable) i nezamjenjive (konstante) elemente
- ω (aksiom) je simbol iz abecede iz kojeg se počinje građenje
- P je produkcija ili skup pravila proizvodnje koji definiraju način na koji će se element zamijeniti s kombinacijama drugih elemenata. Produkcija se sastoji od dva dijela, prethodnika i nasljednika.

Različita kombinacija gore navedenih elemenata dovodi do različitih struktura. Dobivene strukture ne moraju nužno podsjećati na biljke (Slika 2.6. (a)) nego mogu biti i zanimljivog geometrijskog oblika (Slika 2.6. (b)). Svaki smisleni prikaz strukture nosi i svoje ime. Neki primjeri su Grmlje, Sierpinski trokut, Cantorov set, Alga, Kochova krivulja itd.

a b



Slika 2.6. a) Alga [20], **b)** Sierpinski trokut [4]

Kao primjer nastanka jedne strukture uzet će se Binarno stablo(Slika 2.7.)

- varijable : 0, 1
- konstante : [,]
- aksiom : 0
- pravila : (1 → 11), (0 → 1 [0] 0)

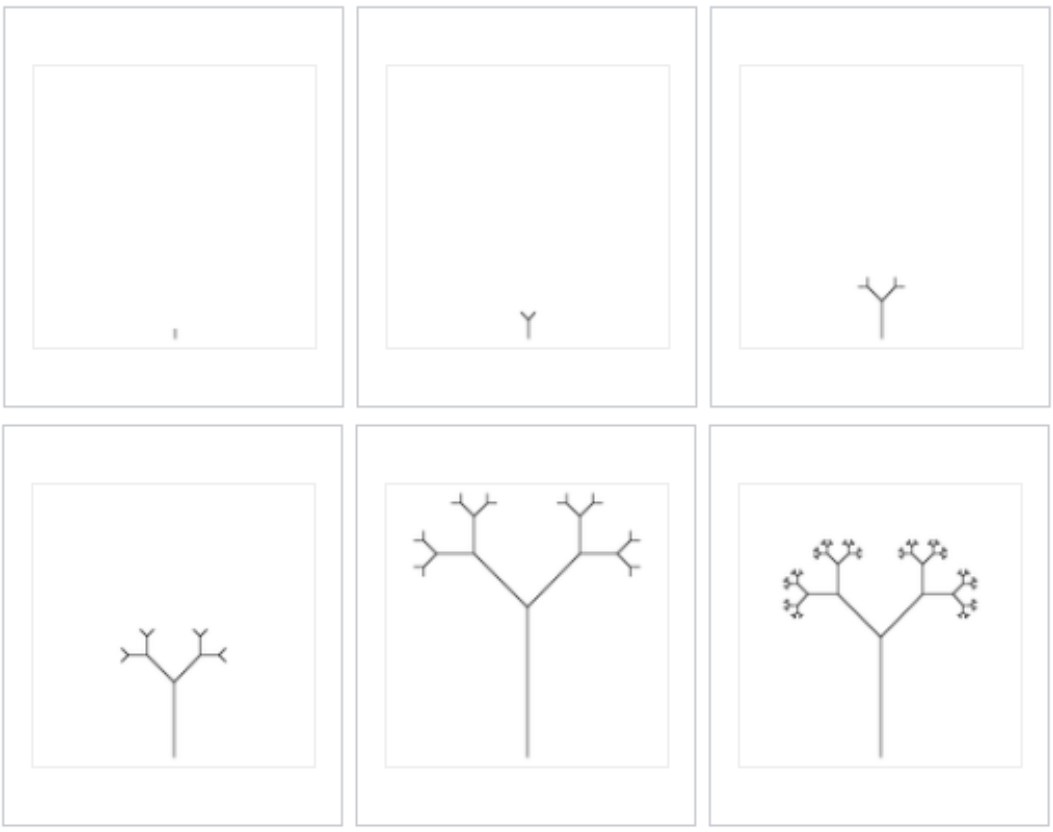
aksiom: 0

1. rekurzija: 1 [0] 0

2. rekurzija: 11 [1 [0] 0] 1 [0] 0

3. rekurzija: 1111 [11 [1 [0] 0] 1 [0] 0] 11 [1 [0] 0] 1 [0] 0

…



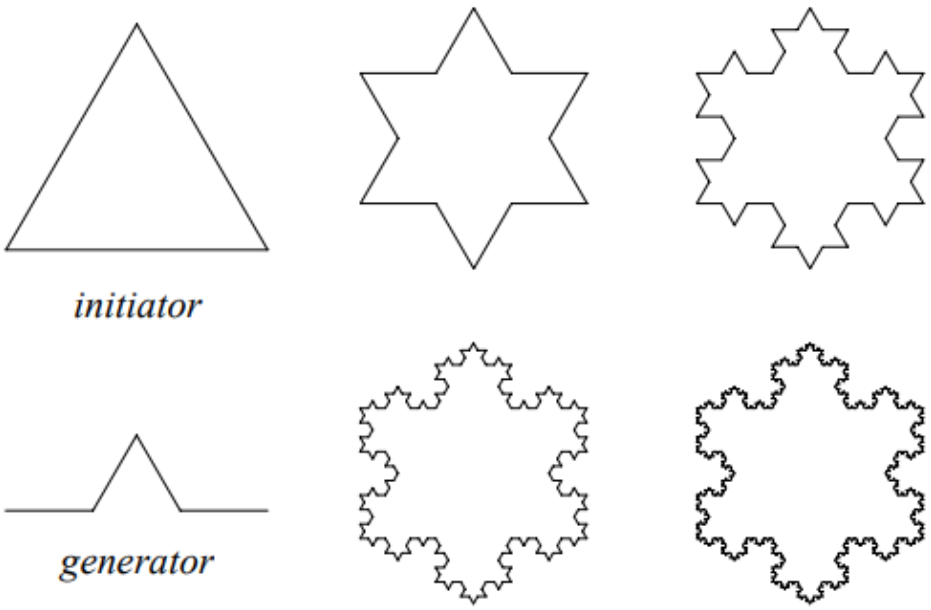
Slika 2.7. Prikaz rasta binarnog stabla [4]

Svaka rekurzija(Slika 2.7.) predočena je pripadajućom slikom, aksiom se nalazi na prvoj slici, prva rekurzija na drugoj itd. U nastavku će se objasniti poveznica između koda i slike i značenje svakog od elemenata Na ovom primjeru mogu se vidjeti tri glavne stvari L-sustava, a to su sustav prepisivanja(rekurzija), grananje i interpretacija putem kornjače.

2.3.2. Sustav prepisivanja

Središnji koncept L-sustava je prepisivanje, odnosno rekurzija. Općenito, prepisivanje je tehnika za definiranje složenih objekata sukcesivnom zamjenom dijelova jednostavnog početnog objekta pomoću skupa pravila prepisivanja ili produkcije. Klasični primjer objekta definiranog pomoću navedenih pravila je krivulja snježne pahuljice (Slika 2.8.) koja počinje s dva oblika, inicijatorom i generatorom.

Svaki element inicijatora se zamjenjuje generatorom. Povećanjem broja iteracija povećava se i broj zamjena jer svakom zamjenom inicijator u idućoj iteraciji postaje kompleksniji. U daljnjem objašnjenju L-sustava pri korištenju sustava rekurzije inicijator će predstavljati aksiom, a generator će biti skup pravila proizvodnje.



Slika 2.8. Konstrukcija snježne pahuljice [5]

2.3.3. DOL-sustav

Navedeni sustav predstavlja najjednostavnije klase L-sustava, one koje su determinističke i bez konteksta. Sustav je bez konteksta ako se svako pravilo proizvodnje odnosi na samo jedan simbol tj. ako pravilo ovisi i o susjedima onda je kontekstno osjetljiv, ako za svaki simbol postoji točno jedna produkcija onda se kaže da je sustav deterministički. Kako bi se lakše objasnilo za primjer će se uzeti sustav čija se abeceda V sastoji samo od dva znaka, a i b. Svako slovo je povezano s jednim pravilom. Pravilo a->ab govori da se slovo a zamjenjuje s nizom ab, drugo pravilo b->a znači da na mjesto b dolazi a. Proces prepisivanja započinje od određenog znaka koji se naziva aksiom[5]. Uzme li se da je aksiom znak b proces deriviranja nakon 5 prepisivanja će izgleda kao što je prikazano na slici 2.9.

Slika 2.9. Primjer derivacije u DOL-sustavu [5]

2.3.4. Interpretacija putem kornjače

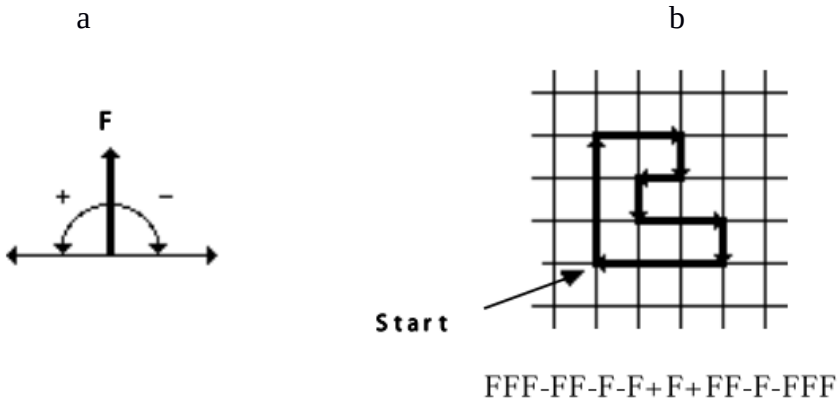
Grafički prikaz niza znakova potrebnog za definiranje strukture stabla naziva se interpretacija putem kornjače. Kornjača je odabrana zato što nema naglih promjena kretanja i giba se poprilično pravocrtno. Metoda se definira kao trojka(x, y, α), gdje (x, y) predstavljaju koordinate položaja kornjače, a kut s oznakom α. Određivanjem veličine koraka d i kutnog porasta δ kornjača može se nacrtati različite rečenice, primjer jedne je na slici 2.10 (b). Kretanje je predstavljena simbolima (Slika 2.10. (a)), a simboli su:

F -> Pomakni se naprijed za dužinu d. Nakon toga se pozicija kornjače mijenja u (x′, y′, α′), gdje je $x' = x + dcos\alpha$ i $y' = y + dsin\alpha$. Linija između (x, y) i (x′, y′) se nacrtala.

f -> Pomakni se naprijed za dužinu d bez crtanja linije.

+ -> Okreni se lijevo za kut δ. Stanje pozicije nakon rotacije je(x, y, α+δ)

- -> Okreni se desno za kut δ. Stanje pozicije nakon rotacije je(x, y, α-δ)



Slika 2.10. a) Interpretacija simbola F,+,- **b)** Interpretacija stringa s kutom porasta δ 90º [5]

2.3.5. Rubno i čvorno prepisivanje

Korištenje navedenih načina, iako na prvi pogled djeluje znatno drugačiji, dovesti će do istih rezultata. Princip kreiranja figura je jednak već navedenim načinima gdje postoji zadani simbol koji se zamjenjuje određenim stringom. Glavna razlika ove dvije metode je što rubno prepisivanje koristi sve elemente u stringu za crtanje figure tj. pored *pop(), push()* i zakretanja slova su varijacije znaka F(npr Fr, Fl), jer se tako rješavaju kompleksnije figure.

$$\omega : Fl$$
$$p1 : Fl \rightarrow Fl + Fr+$$
$$p2 : Fr \rightarrow -Fl - Fr$$

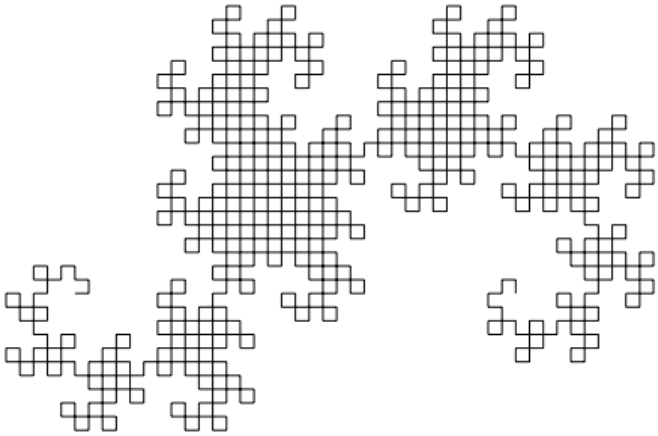
Produkcijska pravila 2.1. Rubno prepisivanje

Čvorno prepisivanje također ima sve navedene elemente ali uvodi i druga slova (npr. l, r) koji nemaju za svrhu funkciju crtanja nego povećanje broja prepisivanja u svrhu dobivanja složenijih struktura. Razlika u izgleda kod-a će biti u tome što će rubno prepisivanje imati složenije rečenice, a čvorno će imati znatno više rekurzija za kompleksnije primjere.

$$\Omega : Fl$$
$$p1 : l \rightarrow l + rF+$$
$$p2 : r \rightarrow -Fl - r$$

Produkcijska pravila 2.2. Čvorno prepisivanje

Korištenjem prethodno navedenih pravila produkcije nakon 10 iteracija i za kut δ = 90 dobije se struktura prikazana na slici 2.11.[6]:



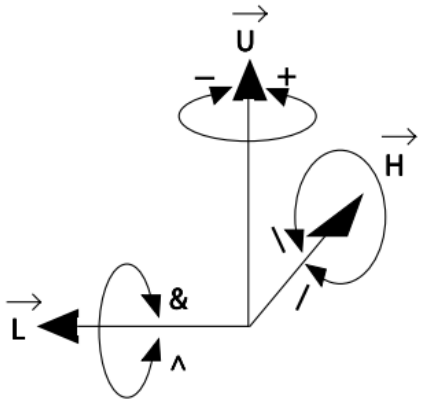
Slika 2.11. Zmajeva krivulja [4]

2.4. 3D MODELIRANJE

Najvažniju ulogu u modeliranju zadnjih godina vjerojatno ima trodimenzionalno modeliranje, gdje svoju primjenu nalazi od igara preko filmova s CGI efektima pa sve do virtualne stvarnosti. Znatno je kompliciranije od dvodimenzionalnog modeliranja ako se nastoji ići u krajnje detalje kao što su teksture, pri tome prestaje biti ekonomično koristiti alate kao što je p5.js a u pomoć dolaze programi kao što su Lsystem4 ili Direct3D.

2.4.1. Parametri modeliranja

Interpretacija putem kornjače može se proširiti i na 3D modele L-sustava. Načelo rada je da se orijentacija kornjače prikaže pomoću tri vektora $\vec{H}$, $\vec{L}$, $\vec{U}$. Vektori imaju jediničnu dužinu i okomiti su jedan na drugi i zadovoljavaju jednadžbu $\vec{H} \times \vec{L} = \vec{U}$.



Slika 2.12. Vektori u prostoru [5]

Rotacije kornjače se prikazuju pomoću jednadžbe $[\overrightarrow{H'}, \overrightarrow{L'}, \overrightarrow{U'}] = [\vec{H}, \vec{L}, \vec{U}]\mathbf{R}$. Gdje je **R** 3×3 rotacijska matrica, koja rotira kornjaču za kut δ.

$$\boldsymbol{R_U(\delta)} = \begin{bmatrix} \boldsymbol{cos\delta} & \boldsymbol{sin\delta} & \mathbf{0} \\ \boldsymbol{-sin\delta} & \boldsymbol{cos\delta} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{1} \end{bmatrix} \tag{2-3}$$

$$\boldsymbol{R_L(\delta)} = \begin{bmatrix} \boldsymbol{cos\delta} & \mathbf{0} & \boldsymbol{-sin\delta} \\ \mathbf{0} & \mathbf{1} & \mathbf{0} \\ \boldsymbol{sin\delta} & \mathbf{0} & \boldsymbol{cos\delta} \end{bmatrix} \tag{2-4}$$

$$\boldsymbol{R_H(\delta)} = \begin{bmatrix} \mathbf{1} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \boldsymbol{cos\delta} & \boldsymbol{-sin\delta} \\ \mathbf{0} & \boldsymbol{sin\delta} & \boldsymbol{cos\delta} \end{bmatrix} \tag{2-5}$$

Sljedeći simboli kontroliraju orijentaciju kornjače u prostoru:

+ Rotacija lijevo za kut δ, koristeći rotacijsku matricu $\boldsymbol{R_U(\delta)}$

- Rotacija desno za kut δ, koristeći rotacijsku matricu $\boldsymbol{R_U(-\delta)}$

& Nagib dolje za kut δ, koristeći rotacijsku matricu $\boldsymbol{R_L(\delta)}$

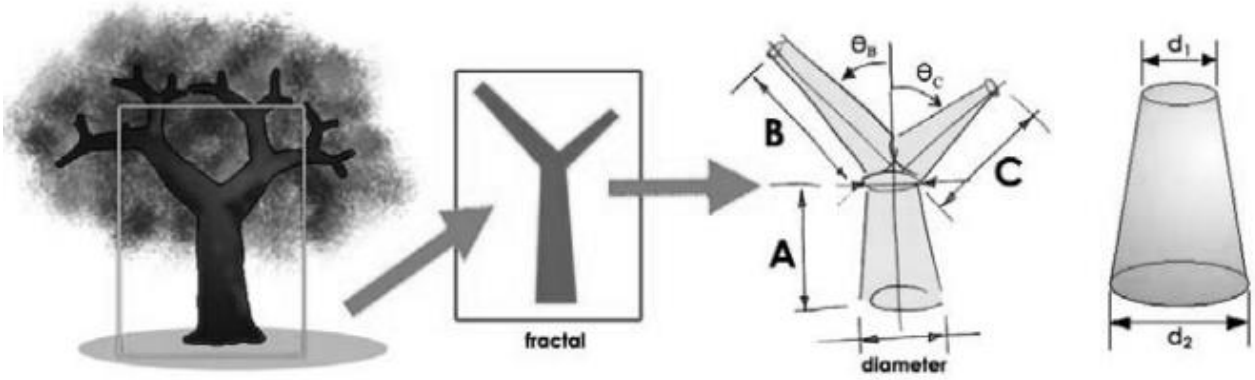
^ Nagib gore za kut δ, koristeći rotacijsku matricu $\boldsymbol{R_L(-\delta)}$

\ Kotrljanje lijevo za kut δ, koristeći rotacijsku matricu $\boldsymbol{R_H(\delta)}$

/ Kotrljaj desno za kut δ, koristeći rotacijsku matricu $\boldsymbol{R_H(-\delta)}$

| Okretanje, koristeći rotacijsku matricu $\boldsymbol{R_U(180°)}$

Biljke se sastoje od puno dijelova kao što su grane, debla, listova i cvijeća. Kako bi se pojednostavilo modeliranje, bez gubljenja realističnih aspekata, prikazivat će se samo deblo i grane. Kao i kod 2D modela gdje se, radi lakšeg prikaza, promatrala samosličnost pri grananju i rastu, tako se i pri 3D modeliranju traže uzorci koji će pojednostaviti proces i smanjiti broj varijabli.



Slika 2.13. Proces pojednostavljenja strukture drveta [7]

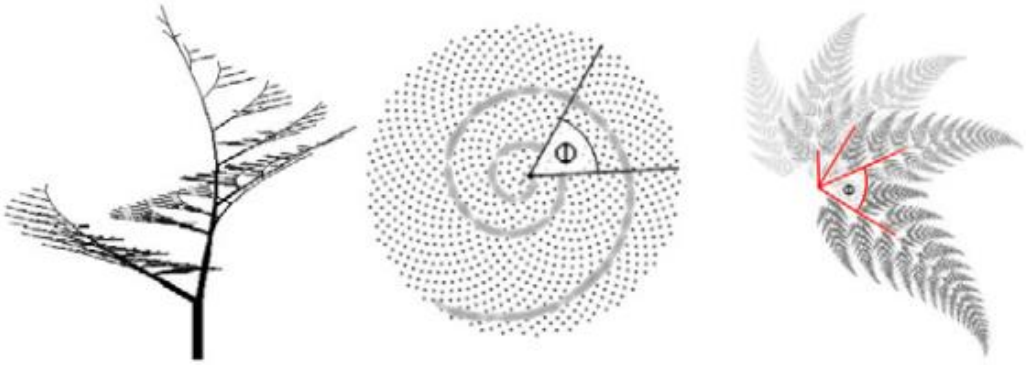
Varijable za 3D modeliranje koje nisu uključene grananjem:

Omjer Duljina – proporcije između dužine roditeljskih grana i grana djece. Na primjer sa slike je odnos duljina između B/A 1 a između C/A 0.5

Kut grananja(θ) – kut između roditeljskih i dječjih grana

Phyllotaktički kut(Φ) – kut između ravnine grana djece i ravnine roditeljske grane(uzorak samosličnosti)

Omjer promjera – parametar koji govori o odnosu promjera vrha i dna grane



Slika 2.14. Slika Phyllotaktičkog kuta(Φ) [7]

2.4.2. Primjer modeliranja

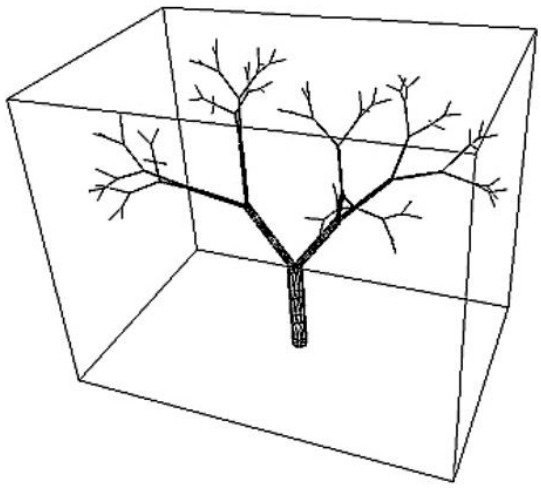
Aksiom: F
P1: F->Y[++++++MF][-----NF][^^^^^OF][&&&&&PF]
P2: M->Z-M
P3: N->Z+N
P4: O->Z&O
P5: P->Z^P
P6: Y->Z-ZY+
P7: Z->ZZ

Produkcijska pravila 2.3. Pravila za model Alstonia Scholaris drveta

Omjer duljina	8/12
Kut grananja	48, 40, 40, 40
Phyllotaktički kut(Φ)	56
Omjer promjera	0.95

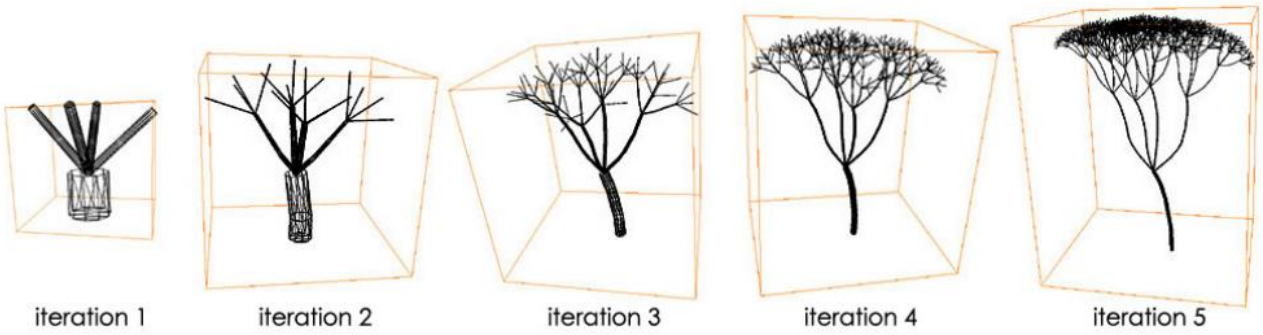
Tablica 2.1. Dodatni elementi potrebni za 3D modeliranje

Nakon upoznavanja rada L-sustava u 2D oblik, razumijevanja 3D modela dolazi intuitivno. Kao što se u prethodnim primjerima trebala pravila produkcije, aksiom i kut grananja, tako je za 3D model to prošireno s elementima kao što su omjer promjera,duljina i phyllotaktički kut razlog tome je što moramo uzeti u obzir grananje i u novu dimenziju te smanjenjem promjera grana drveta njihovim rastom u visinu. Stvari postaju znatno kompliciranije ako u proces uzmemo i rast lišća i cvjetova. Zadane varijable unesemo u program predviđen za to (p5.js također podržava rad s 3D modelima) i dobiva se objekt prikazan na slici.



Slika 2.15. 3D prikaz Alstonia Scholaris drveta [7]

Kao što je bio primjer za razvoj biljke u 2D modelu binarnog stabla prikazanog na slici 2.7. i razvoj korova što će biti prikazan na slici 3.2. tako i na slici 2.16. je primjer iteracija 3D modela gdje svaka iteracija pridonosi realističnosti drveta. Razvoj strukture drveta ne staje samo na zadnjoj slici nego iteracije mogu ići do beskonačnosti, gdje svaka nosi sve sitnije detalje, no ujedno dovodi i do velike potrošnje memorije, što može biti problem, ovisno o performansama računala [7].



Slika 2.16. Konstrukcija drveta [7]

2.4.3. Inverzna metoda modeliranja

Proceduralna metoda, metoda zasnovana na pravilima i hibridna metoda pružaju mogućnost 3D modeliranja, gdje se za unesene parametre efektivno dobije model tražene biljke. Glavni problem je što je te parametre teško odrediti i vremenski je zahtjevno pojedinačno za svako drvo određivati parametre iako se žele postići samo male varijacije npr. kreiranje šume s puno istih biljaka. Radno okruženje koja se pruža kao rješenje je inverzna metoda. Glavna primjena je za modeliranje stohastičkih drveća, gdje pomoću jednog ulaza se može proizvesti više modela tog drveta ali s varijacijama u izgledu tako da izgleda kao da su iste vrste.[8]

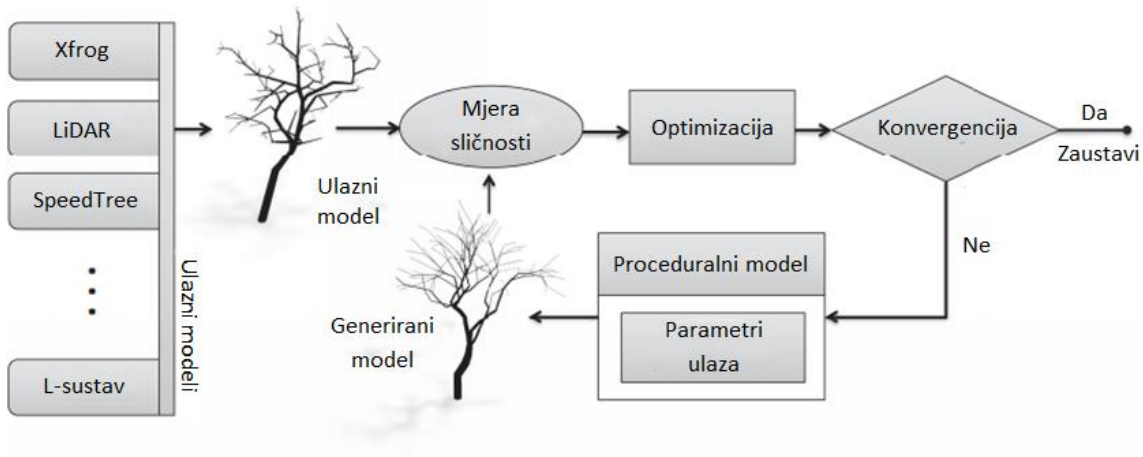
Slika 2.17. Različite verzije iste vrste pomoću jednog ulaza [8]

Metoda može koristiti jedan ili više modela kao ulaz, kao što su LiDAR skeniranja, Xfrog i SpeedTree modeli ili drveća kreirana pomoću L-sustava. Ulaz se nastoji reproducirati što preciznije koristeći Monte Carlo Markov lanac [9] optimizaciju na parametrima ulaza. Model se kontrolira s 24 $\bar{\varphi}$ parametra koji opisuju unutarnje i vanjske faktore drveća. Parametri se mogu svrstati u tri grupe geometrijski, ekološki i oni koji određuju sudbinu pupoljka.

Geometrijski parametri – Određuju phyllotaksiju biljke, brinu se o relativnoj orijentaciji pupoljaka, kutovima između elemenata i pridodaju slučajnosti izgleda

Ekološki parametri – Određuju kako okolina utječe na biljku. Svjetlo i gravitacija utječu na prirodan rast pupoljka. Brine i o starosti biljke, tijekom starenja grane postanu teže te se savijaju prema dolje

Parametri koji određuju sudbinu pupoljka – Pupoljak može postati cvijet, slomiti se, postati internodij s bočnim i apikalnim pupoljcima ili umrijeti

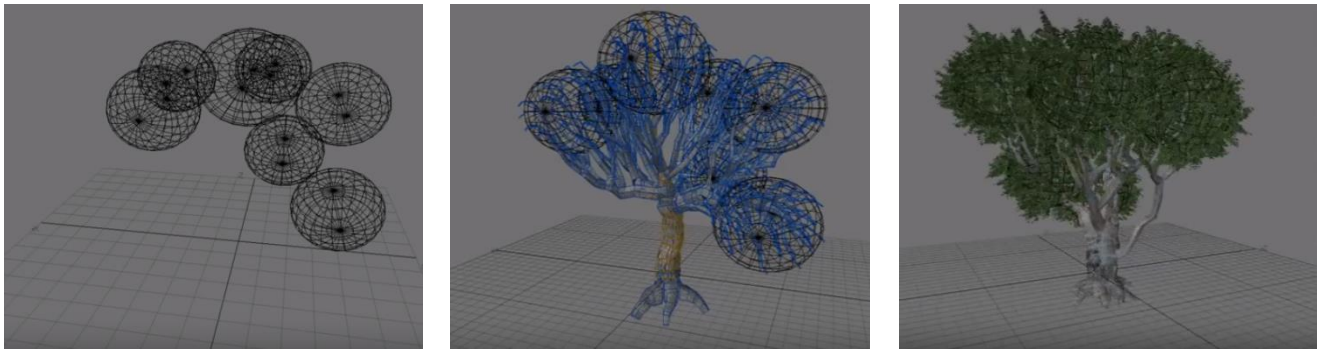


Slika 2.18. Pregled načina rada radnog okruženja [8]

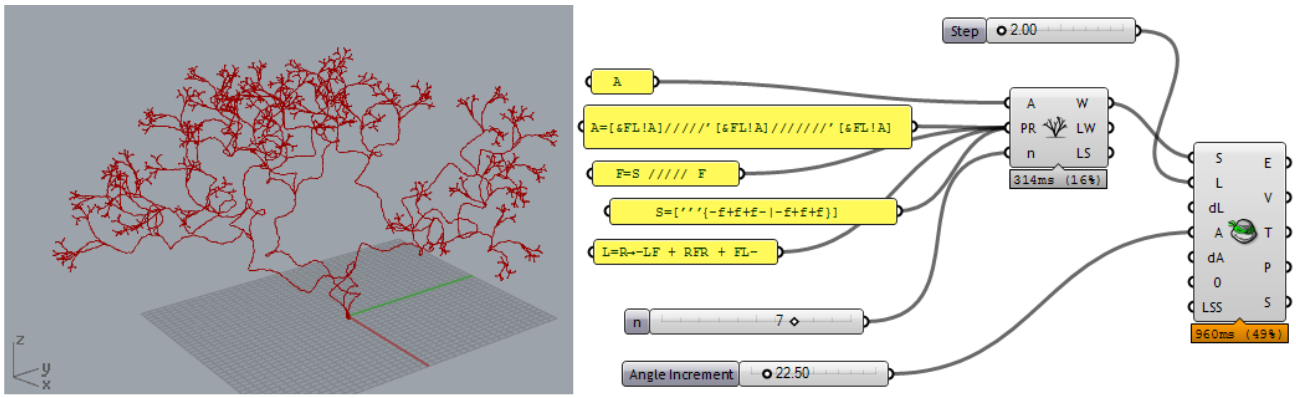
2.4.4. Alati

Razvojem tehnologije olakšano je 3D modeliranje, te je svoju ulogu našlo u različitim životnim sferama. Nekad korišteno uglavnom u igrama, danas pronalazi svrhu i u filmografiji, arhitekturi i virtualnoj stvarnosti. Kreiranje modela biljaka naročito je važan posao jer uglavnom sastavan dio okoline svih navedenih sfera. Pri modeliranju biljnog svijeta može se birati je li potrebno rekreirati objekt iz prirode bez botaničke podloge ili se želi da simulira rast kao u prirodi. .

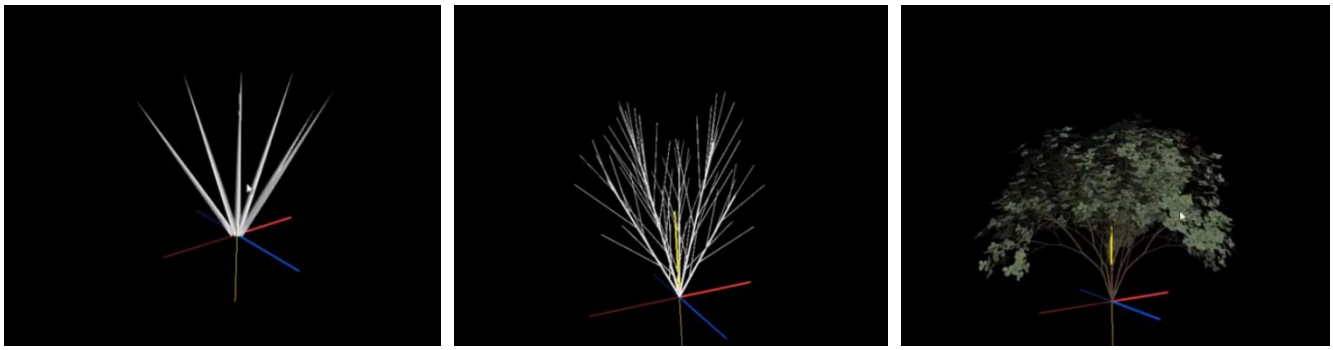
Neki od alata pogodnih za modeliranje su spomenuti treesDesigner (Slika 2.19.) koji koristi proceduralnu metodu, Rabbit (Lika 2.20.) i FractaL-Tree koji koriste L-sistem i Xfrog (Slika 2.21.) koji je zasnovan na hibridnoj metodi. Također moguće je koristiti i Houdini kao i dodatke za Blender kao što su Arbaro, ngPlant,TreeGen itd.



Slika 2.19. koraci modeliranja u treesDesigneru [10]



Slika 2.20. Rabbit [11]



Slika 2.21. stadiji u Xfrogu [12]

3. MODELIRANJE BILJNIH ORGANIZAMA POMOĆU P5.JS

Jedan od alata praktičan za vizualizaciju L-sustava je p5.js. Ako su se dobro razumjeli prethodno faktori i proučen je načina rada u JavaScriptu, proces modeliranja rasta biljaka će biti znatno olakšan.

3.1. PREGLED KORIŠTENIH TEHNOLOGIJA

3.1.1. HTML opisni jezik

HyperText Markup Language (često skraćen kao HTML) je prezentacijski jezik za izradu web stranica popularan zbog svoje jednostavnosti i praktičnosti. Hipertekst dokument se stvara pomoću HTML-a. Njegova temeljna zadaća je uputiti web preglednik kako prikazati taj isti dokument i da pri tome izgleda jednako bez obzira o pregledniku, operacijskom sustavu ili računalu se radi. HTML nije programski jezik, niti se njime može izvršiti ikakva zadaća, čak ni ona najjednostavnija poput zbrajanja ili oduzimanja. Kako su HTML datoteke obične tekstualne datoteke, često se za njihov rad koristi i CSS(Cascading Style Sheets). CSS definira kako prikazati elemente, te uređuje izgled i raspored stranice

Osnovna svojstva HTML-a su [13]:

- Jednostavan za naučiti i koristiti
- Korišten na svim preglednicima
- Alat imamo na svakom sustavu

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta http-equiv="X-UA-Compatible" content="ie=edge">
  <title>Document</title>
</head>
<body>

</body>
</html>
```

Programski kod 3.1. Prikaz početnog HTML koda

3.1.2. JavaScript programski jezik

JavaScript(često skraćen kao JS) je skriptni, interpretirani, više paradigmatski programski jezik s funkcijama prve klase[14]. Nalazi se u <script></script> dijelu statičke HTML stranice. (Programski kod 3.2) Uz HTML i CSS poznata je kao jedna od glavnih tehnologija web aplikacija tako što pruža interaktivne stranice [15]. Web pretraživači imaju posebne JS mehanizme kako bi izvršili navedene radnje. Iako je najpoznatiji za internet usluge, koristi se i u mnogim ne pretraživačkim okruženjima. Pokreće se na strani klijenta i premda se izvršava samo jednom, kada se HTML dokument učita, ne pruža interaktivnost tako. Zato se definiraju „OnClick“ događaji, koji izvršavaju dio koda kada pritisnemo na određeni dio HTML dokumenta. Primjer ovoga će biti kasnije prezentiran pomoću tipke *generate* koja će služiti za promjenu izgleda modela biljke. Premda je navedeni jezik snažno okrenut korisniku putem interakcije i slanjem dokumenata putem interneta tako da je prisutnost zlćudnih programa uvijek moguća. Zato su tvorci osigurali da programi JavaScripta nema pristup lokalnim datotekama čime ne mogu zaraziti druge datoteke ili brisati postojeće.

Osnovna svojstva JavaScript-a su [16]:

- Jednostavnost
- Dinamičnost
- Sigurnost.

```
<!DOCTYPE html>
<html lang="">
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>p5.js example</title>
    <style> body {padding: 0; margin: 0;} </style>
    <script src="../p5.min.js"></script>
    <script src="../addons/p5.dom.min.js"></script>
    <script src="../addons/p5.sound.min.js"></script>
    <script src="sketch.js"></script>
  </head>
  <body>
  </body>
</html>
```

Programski kod 3.2. Proširenje HTML dodavanjem p5.js biblioteke

3.1.3. P5.js biblioteka

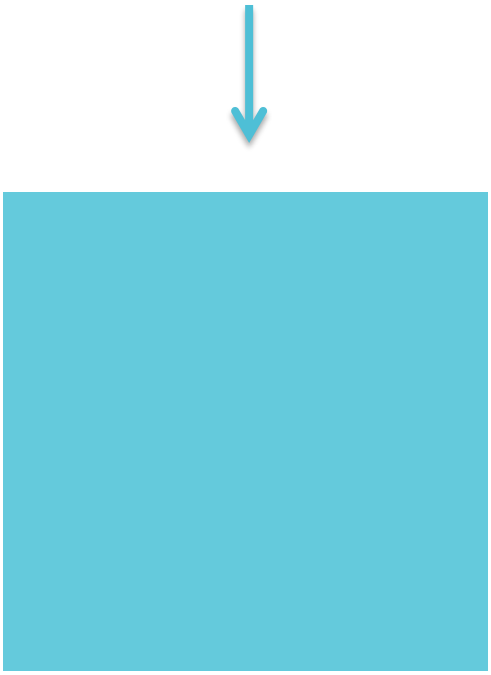
P5.js je biblioteka JavaScript-a koja nastavlja prvotni cilj procesiranja (obrade). Procesiranje je grafička biblioteka otvorenog koda i integrirano razvojnog okruženja (IDE) napravljena za zajednice elektroničkih umjetnosti, novih medija i vizualnog dizajna u svrhu podučavanja početnika osnovama računalnog programiranja u vizualnom kontekstu. P5.js je nastao kao prijelaz starije verzije procesiranja koja je koristila Java jezik na noviju verziju koja koristi JavaScript jezik [17]. Početni kod se sastoji od praznih *setup()* i *draw()* funkcija.

Funkcija *setup()* koristi se za definiranje svojstava okoline kao što je određivanje veličine platna i služi za učitavanje medija kao što su slike i fontovi.[18] Izvrši se samo na početku programa. *Draw()* služi kada želimo da se određena radnja kontinuirano izvodi npr. ponavljajuće animacije. Izvršava se sve dok se program ne zaustavi, a nakon zadnje linije iznova ide prva [19].

```
function setup() {
  createCanvas(400,400);
  background(100,202,220);
}

function draw() {
}
```

Programski kod 3.3. Primjer početnog p5.js koda



Slika 3.1. Prikaz u web pregledniku kao rezultat programskog koda 3.3.

3.2. IZRAĐENI MODELI BILJAKA

3.2.1. Model korova

aksiom = F F -> FF-[XY]+[XY] X -> +FY Y -> -FX δ = -22.5

Produkcijska pravila 3.1. Pravila za model korova

Proces pisanja programskog koda započinje s postavljanjem aksioma, rečenice(stringa) i dužine koja predstavlja koliki će korak F raditi(Programski kod 3.4.). Rečenica je na početku jednaka aksiomu jer se od njega kreće s rekurzijom. Pravila proizvodnje su prikazana pomoću polja, radi jednostavnosti prolaženja kroz njih, a broj članova polja ovisi o kompliciranosti figure koja se nastoji dobiti.

```
var axiom = "F";
var sentence = axiom;
var len = 100;

var rules=[];
rules[0] ={
  a: "F",
  b: "FF-[XY]+[XY]"
}
rules[1]={
  a:"X",
  b: "+FY"
}
rules[2]={
  a:"Y",
  b: "-FX"
}
```

Programski kod 3.4. Aksiom i pravila produkcije zapisani putem koda

Radi pojednostavljenja cijelog proces određivanja broja rekurzija potrebnih za dobivanje željenog oblika (stadija) biljke tj. kako se ne bi moralo stalno vraćati u kod i mijenjati broj rekurzija, postavlja se gumb *generate*, čijim će se pritiskom pozvati sustav prepisivanja koji se nalazi u njegovoj funkciji *generate()* (Programski kod 3.5.). Osim što vrši prolazak kroz cijeli ulazni niz znakova radi provjere i primjene produkcijskih pravila, vrši izmjenu stringa i skaliranje biljke jer u suprotnom bi prerasla dimenzije ekrana i već kod treće rekurzije bi vidjeli samo dno korova. Skaliranje se vrši tako

što se varijabla *len* množi proizvoljnim koeficijentom, kojeg pogađamo jer ovisi o veličina platna i konačnoj veličini figure.

```
function generate() {
  len *=0.54;
  var nextSentence ="";
  for(var i=0; i<sentence.length;i++){
    var current = sentence.charAt(i);
    var found = false;
    for(var j=0;j<rules.length;j++){
    if(current == rules[j].a){
      found= true;
      nextSentence += rules[j].b;
      break;
    }}
    if(!found){
      nextSentence +=current;
  }}
    sentence = nextSentence;
    createP(sentence);
  turtle();
}
```

Programski kod 3.5. Funkcija generate()

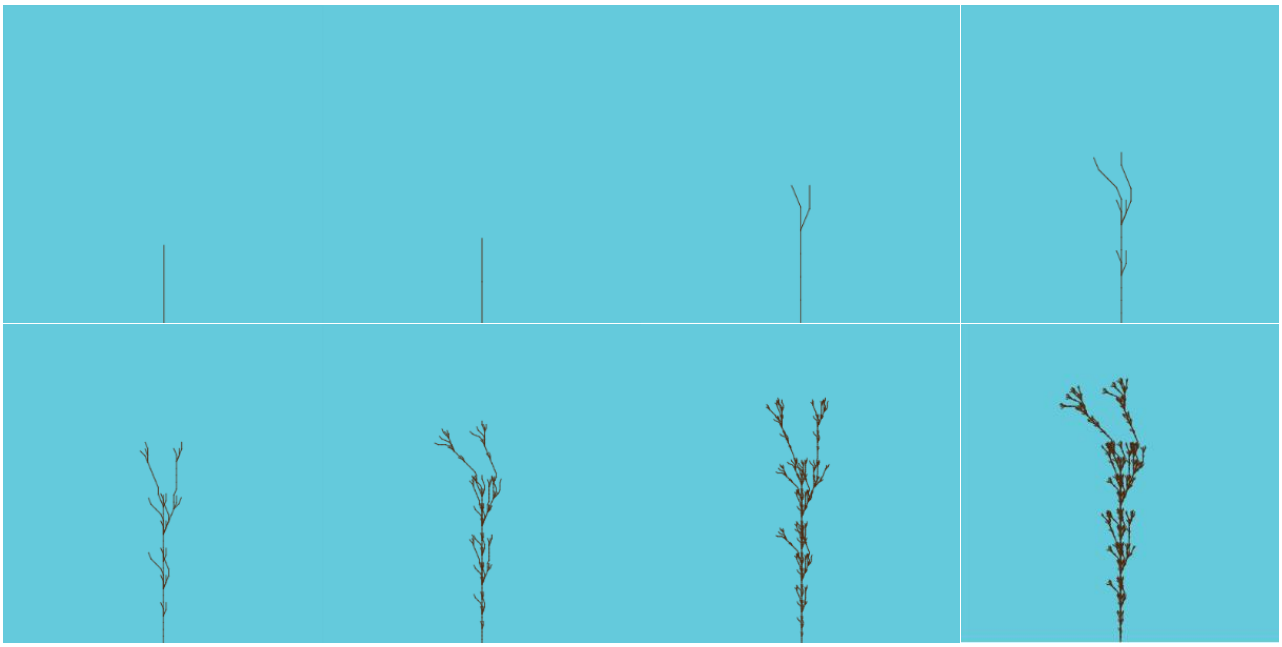
```
function turtle(){
  background(100,202,220);
  resetMatrix();
  translate(width/2, height);
  stroke(77,39,13);
  for(var i=0; i<sentence.length;i++){
  var current = sentence.charAt(i);
    if(current=="F"){
      line(0,0,0, -len);
      translate(0, -len); }
    else if(current=="+"){
      rotate(radians(22.5));}
    else if(current=="-"){
      rotate(-radians(22.5));}
    else if(current == "["){
      push();}
    else if(current=="]"){
      pop();
    }}}
```

Programski kod 3.6. Funkcija *turtle()*

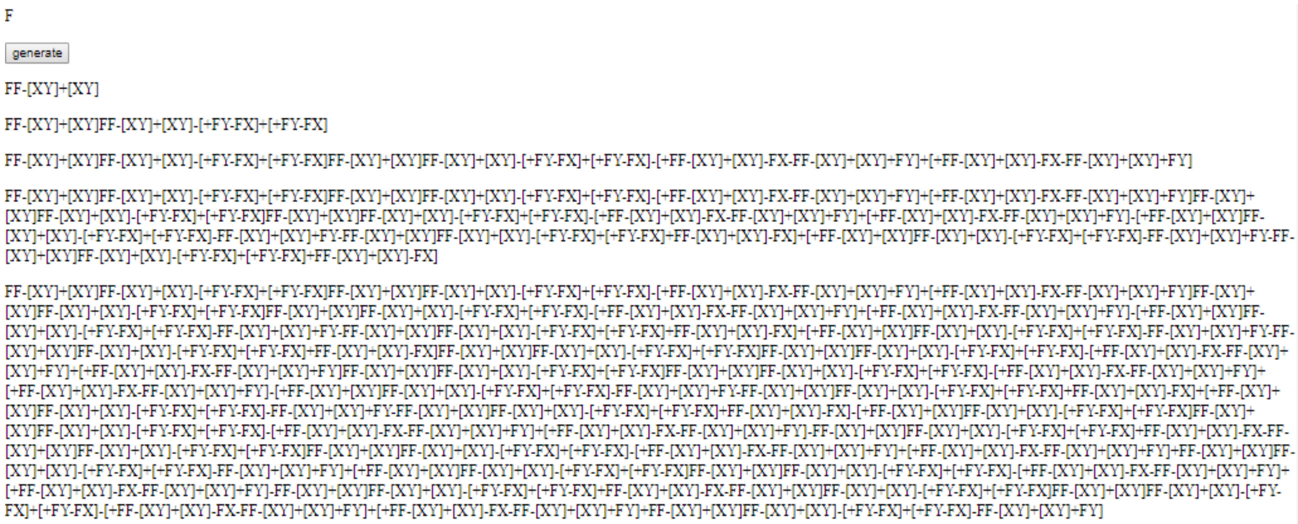
Sam proces crtanja obavlja funkcija *turtle()* (Programski kod 3.6.). Prolazi kroz varijablu string i provjerava sve simbole te obavlja funkcije ovisno o njima. Jedina novost su simboli „[,]“ koji predstavljaju *push()* i *pop()* funkcije koje su unaprijed definirane. *Push()* služi da zaključa poziciju na platnu na kojoj se taj simbol pojavio i nastavlja s crtanjem iz te točke sve dok se ne pojavi *pop()*, onda se kornjača vraća na mjesto zapamćeno *push()* funkcijom. Funkcija *setup()* (Programski kod 3.7.) predstavlja kreiranje platna, gumba *generate* i poziva *turtle()* funkciju.

```
function setup() {
  createCanvas(400,400);
  background(15,145,169);
  createP(axiom);
  turtle();
  var button = createButton("generate");
  button.mousePressed(generate);
}
```

Programski kod 3.7. Funkcija setup()



Slika 3.2. Biljka tijekom 8 rekurzija



Slika 3.3. String nakon 5 rekurzija

Slika 3.2. je rezultat napisanog koda, gdje rast biljke ne staje nakon 8 rekurzija nego može ići u beskonačnost. Svaka rekurzija donosi nova grananja biljke čime raste njena kompliciranost. Na slici 3.3. je odlomcima prikazano kako izgleda niz znakova za svaki stadij biljke. Već nakon četvrte rekurzije niz znakova postaje nepregledan i jako složen, tako da nije prikazan ostatak razvoja biljke tim putem. Za usporedbu složenosti mogu se uzeti rečenice binarnog stabla sa slike 2.8.

3.2.2. Model grma

aksiom = VZFFF
V -> [+++W][---W]YV
W -> +X[-W]Z
X -> -W[+X]Z
Y -> YZ
Z -> [-FFF][+FFF]F
$\delta = 20$

Produkcijska pravila 3.2. Pravila za model grma

Kod prikaza „Grm“ objekta [20], koristit će se gotovo jednaki elementi kao kod prikaza „Korova“. Glavna razlika će biti u početnom dijelu gdje se postavlja aksiom i polje pravila (Programski kod 3.8.). Osim toga može se i promijeniti skaliranje veličine objekta u funkciji *generate()*. Razlog zašto moramo mijenjati faktor skaliranja za svaki objekt je jer svaka figura raste različitom brzinom.

```
var axiom = "VZFFF";
var sentence = axiom;
var len = 100;
var rules=[];
rules[0] ={
  a: "V",
  b: "[+++W][---W]YV"}
rules[1]={
  a:"W",
  b: "+X[-W]Z"}
rules[2]={
  a:"X",
  b: "-W[+X]Z"}
rules[3]={
  a:"Y",
  b: "YZ"}
rules[4]={
  a:"Z",
  b: "[-FFF][+FFF]F"}
```

Programski kod 3.8. Aksiom i pravila produkcije

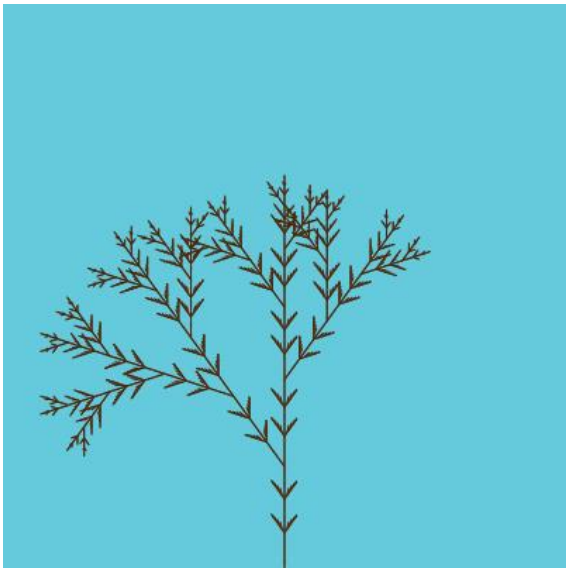
Slika 3.4. Biljka nakon 10 rekurzija

3.2.3. Model travke

```
aksiom = aF
a -> FFFFFv[+++h][---q]fb
b -> FFFFFv[+++h][---q]fc
c -> FFFFFv[+++fa]fd
d -> FFFFFv[+++h][---q]fe
e -> FFFFFv[+++h][---q]fg
g -> FFFFFv[---fa]fa
h -> ifFF
i -> fFFF[--m]j
j -> fFFF[--n]k
k -> fFFF[--o]l
l -> fFFF[--p]
m -> fFn
n -> fFo
o -> fFp
p -> fF
q -> rfF
r -> fFFF[++m]s
s -> fFFF[++n]t
t -> fFFF[++o]u
u -> fFFF[++p]
v -> Fv
δ = 12
```

Produkcijska pravila 3.3. Pravila za model travke

Primjer „Travke“ dobro prikazuje kompliciranost zapisivanja koja nastaje s porastom složenosti biljke. Uloga čvornog prepisivanja je gotovo neizbježna, rubno prepisivanje bi smanjilo količinu pravila produkcije ali bi povećalo broj simbola svakog stringa jer bi se svaki korak morao crtati, tj. ne bi bilo elemenata za stvaranje putanje bez se da pri tome crtaju liniju. Radi velikog broja pravila produkcije kod se nalazi u prilogu.



Slika 3.5. Biljka nakon 18 rekurzija

4. ZAKLJUČAK

Glavna zadaća rada bila je objasniti elemente koji čine izgled biljke „slučajnim“, te iz tih elemenata izvući samo nužne parametre potrebne za vlastiti prikaz oblika. Prva ideja za rješenje ovog problema bilo je uvođenje fraktala, iako se zna da biljke nisu fraktali znatno je pomoglo daljnjem razvoju. Metoda koja je djelomično nastala na toj teoriji i samosličnosti je L-sustav, koji omogućuje jako brzo kreiranje raznolikih struktura. Unatoč praktičnosti, pruža mjesta napretku u području realističnih elemenata, kao što u teksture i pupoljci, i pojednostavljenja procesa određivanju produkcijskih pravila. Iako imaju svoje mane, metode koje dolaze kao pomoću u navedenim problemima su proceduralna i hibridna metoda, koje su manje zasnovane na botaničkim pravilima. Objašnjen je osnovni način rada L-sustava pomoću interpretacije putem kornjače i prepisivanja, te je znanje prošireno na trodimenzionalno modeliranje uz nekoliko novih parametara. Metoda koja se su istaknula kao korisna, iako jako kompleksna, je inverzna metoda. Pruža rekreiranje biljaka uz varijacije ali zahtjeva kvalitetan ulazni model ili nekoliko njih. Rad L-sustava na kraju je potpuno je približen pomoću primjera prikazanih pomoću p5.js biblioteke JavaScripta. Primjeri su prikazali ne samo konačne modele nego i stadije rasta biljke. Iako su u radu sve metode prikazane kao efikasno rješenje za modeliranje biljaka, to i dalje ne čini proces pretvaranja biljnog svijeta u par naredbi laganim. Stvari se znatno zakompliciraju čim se malo odmakne od mikroorganizama ili pojednostavljenih verzija stabala, te će različiti modeli zahtijevati različite metode.

LITERATURA

[1] H., Weyl , Symmetry, Princeton Science Library, Princeton, 1952

[2] C. Eloy, Hiérarchisation de l'architecture des arbres, IRPHE, Aix-Marseille Université, CNRS, MarseillE, 2013 http://www.plantnum.agropolis.fr/uploads/talk73.pdf [18.09.2019]

[3] T., Dapper, Practical Procedural Modeling of Plants, Universität Bremen, Bremen, 2003 http://www.td-grafik.de/artic/talk20030122/index.html [18.09.2019]

[4] L-system, Wikipedia, 2019 https://en.wikipedia.org/wiki/L-system, [18.09.2019]

[5] M. Emmerich, L-systems, 2017 https://liacs.leidenuniv.nl/~csnaco/NC/slides/9_L-Systems.pdf?v=14-11-2017&fbclid=IwAR3AfT_dbTkSeEhG_womq54cgm782SxBv2wk0ZEhRcMELwyLcfviKNgHECI, [18.09.2019]

[6] P., Prusinkiewicz, A., Lindenmayer, The Alghoritmic Beauty of Plants, Springer-Verlag, New York, 1990

[7] R., Viruchpintu, N., Khiripet, Real-time 3D Plant Structure Modeling by L-System with Actual Measurement Parameters, National Electronics and Computer Technology Center, Pathumthani 12120, Thailand, 2006

[8] O. Stava, S. Pirk, J. Kratt, B. Chen, R. Mech, O. Deussen and B. Benes, Inverse Procedural Modelling of Trees, COMPUTER GRAPHICS forum, sv. 33, str. 6 pp. 118-131, 2014

[9] Markov chain Monte Carlo, Wikipedia, 2019 https://en.wikipedia.org/wiki/Markov_chain_Monte_Carlo, [18.09.2019]

[10] pOlas, treesDesigner 1.6, YouTube https://www.youtube.com/watch?v=BxR79tuiGkw [18.09.2019]

[11] Alex, Grasshopper https://www.grasshopper3d.com/forum/topics/rabbit-2?overrideMobileRedirect=1 [18.09.2019]

[12] Jan Walter Schliep, Xfrog 3.5 - modeling a shrub, YouTube, https://www.youtube.com/watch?v=F4_0G2r8Qy0&t=1339s [18.09.2019]

[13] Wikipedia, HTML, 2018 https://hr.wikipedia.org/wiki/HTML, [18.09.2019

[14] Wikipedia, JavaScript, 2019 https://en.wikipedia.org/wiki/JavaScript, [18.09.2019]

[15] Wikipedia, Funkcije prve klase, 2019 https://en.wikipedia.org/wiki/First-class_function [18.09.2019]

[16] D., Stančer, Osnove JavaScripta, Srce, Zagreb, 2015

[17] L. McCarthy, p5.js, 2019 https://p5js.org/, [18.09.2019]

[18] p5.js funkcije, Lauren McCarthy, 2019 https://p5js.org/reference/ , [18.09.2019]

[19] sarthak_ishu11, p5.js Introduction 2018
https://www.geeksforgeeks.org/p5-js-introduction/, [18.09.2019]

[20] P., Borke, L-system User Notes, 1991 http://paulbourke.net/fractals/lsys/, [18.09.2019]

SAŽETAK

U radu osim teorijskog i praktičnog dijela objašnjeni su i alati potrebni za samostalnu izradu modela biljaka. Alati su HTML koji je upotpunjen JavaScriptom za vizualne svrhe i bibliotekom p5.js koja ima glavnu ulogu u samom modeliranju. Izrada je prikazana postepenim koracima krenuvši od tekstualnog zapisa produkcijskih pravila, objašnjavanja crtanja putem kornjače i implementacija modela u programu. Pojašnjen je i način na koji se dolazi do složenih struktura putem prepisivanja i uočavanja samosličnosti. Modeliranje i prikazivanje rasta biljaka prikazano je kroz slikovne primjere i zapise pomoću kod-a. Prikazan je i primjer za trodimenzionalno modeliranje gdje je iskorišteno svo dotad stečeno znanje koje je upotpunjeno s par novih elemenata.

Ključne riječi: modeliranje biljaka, L-sustav, grafičko crtanje

ABSTRACT

Modeling the appearance and growth process of the plant organisms

In addition to the theoretical and practical part, the paper also explains the tools needed to independently design the plant models. The tools are HTML together with JavaScript for visual purposes and the p5.js library, which plays a major role in the modeling itself. Production is shown in gradual steps, starting with a textual record of production rules, explaining the drawing via the turtle, and implementing models in the program. The way in which complex structures are obtained through transcription and identification of self-similarity is also explained. Modeling and plant growth is shown through pictorial examples and records using a code. An example of three-dimensional modeling is also presented where all the knowledge gained so far is utilized, complemented by a couple of new elements.

Keywords: plants modeling, L-system, graphic drawing

ŽIVOTOPIS

Marin Ćosić rođen je 22. siječnja 1998. godine u Zagrebu. Od 2004. do 2012. pohađa Osnovnu školu Blaž Tadijanović u Slavonskom Brodu. Godine 2012. upisuje opću gimnaziju Matija Mesić Slavonski Brod koju završava 2016. polaganjem ispita državne mature. Godine 2016. upisuje Elektrotehnički fakultet Osijek na Sveučilištu Josipa Juraja Strossmayera u Osijeku na sveučilišni studij računarstva.

Marin Ćosić

_____

Potpis

P.3.2.3.

Prilog za pravila produkcije modela travke.

```
var rules=[];
rules[0] ={
  a: "a",
  b: "FFFFFv[+++h][---q]fb"
}
rules[1]={
  a:"b",
  b: "FFFFFv[+++h][---q]fc"
}
rules[2]={
  a:"c",
  b: "FFFFFv[+++fa]fd"
}
rules[3]={
  a:"d",
  b: "FFFFFv[+++h][---q]fe"
}
rules[4]={
  a:"e",
  b: "FFFFFv[+++h][---q]fg"
}
rules[5]={
  a:"g",
  b: "FFFFFv[---fa]fa"
}
rules[6]={
  a:"h",
  b: "ifFF"
}
rules[7]={
  a:"i",
  b: "fFFF[--m]j"
}
rules[8]={
  a:"j",
  b: "fFFF[--n]k"
}
rules[9]={
  a:"k",
  b: "fFFF[--o]l"
}
rules[10]={
  a:"l",
```

```
  b: "fFFF[--p]"
}
rules[11]={
  a:"m",
  b: "fFn"
}
rules[12]={
  a:"n",
  b: "fFo"
}
rules[13]={
  a:"o",
  b: "fFp"
}
rules[14]={
  a:"p",
  b: "fF"
}
rules[15]={
  a:"q",
  b: "rfF"
}
rules[16]={
  a:"r",
  b: "fFFF[++m]s"
}
rules[17]={
  a:"s",
  b: "fFFF[++n]t"
}
rules[18]={
  a:"t",
  b: "fFFF[++o]u"
}
rules[19]={
  a:"u",
  b: "fFFF[++p]"
}
rules[19]={
  a:"v",
  b: "Fv"
}
```