SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 763.

Proceduralno generiranje arhitekture virtualne okoline

Andrija Stepić

Zagreb, lipanj 2014.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA
ODBOR ZA DIPLOMSKI RAD PROFILA

Zagreb, 10. ožujka 2014.

DIPLOMSKI ZADATAK br. 763

Pristupnik: **Andrija Stepić (0036450447)**
Studij: Računarstvo
Profil: Računarska znanost

Zadatak: **Proceduralno generiranje arhitekture virtualne okoline**

Opis zadatka:

Proučiti postupke proceduralnog generiranje virtualnih okolina. Generirani elementi trebaju sadržavati sve potrebne strukturne podatke kao što su uv mape za teksture i hijerarhijsku organizaciju u zapisu elemenata. Posebno obratiti pažnju na dinamičke sadržaje. Ostvariti programsku implementaciju koja omogućuje prikaz razrađenog modela. Na različitim primjerima prikazati ostvarene rezultate. Diskutirati utjecaj parametara. Načiniti ocjenu implementiranih algoritama i ostvarenih rezultata.
Izraditi odgovarajući programski proizvod. Rezultate rada načiniti dostupne putem Interneta. Radu priložiti algoritme, izvorne kodove i rezultate uz potrebna objašnjenja i dokumentaciju. Citirati korištenu literaturu i navesti dobivenu pomoć.

Zadatak uručen pristupniku: 14. ožujka 2014.
Rok za predaju rada: 30. lipnja 2014.

Mentor:
Prof.dr.sc. Željka Mihajlović

Djelovođa:
Doc.dr.sc. Tomislav Hrkać

Predsjednik odbora za diplomski rad profila:
Prof.dr.sc. Siniša Srbljić

Sadržaj

1. Uvod 6
2. Osnove proceduralnog generiranja 7
2.1. Definicije 7
2.2. Taksonomija metoda proceduralnog generiranja 7
2.3. Podjela metoda proceduralnog generiranja prema svojstvima 8
2.3.1. Kontinuirane i inicijalne metode 8
2.3.2. Nužne i proizvoljne metode 9
2.3.3. Stupnjevi i dimenzije kontrole 9
2.3.4. Generičke i adaptivne metode 9
2.3.5. Stohastičke i determinističke metode 9
2.3.6. Konstrukcijske i testne metode 10
2.3.7. Automatizirane metode i združeno autorstvo 13
2.4. Poželjna svojstva proceduralnog generiranja 13
2.5. Podjela proceduralno stvorivog sadržaja 15
2.6. Granice proceduralnog generiranja 18
2.7. Prednosti i nedostaci 18
2.8. Praktična primjena proceduralnog generiranja 21
2.8.1. Virtualno preslikavanje stvarnog sadržaja 21
2.8.2. Filmska industrija 21
2.8.3. Računalne igre i simulacije 22
2.8.4. Popis programa koji koriste proceduralno generiranje 24
3. Proceduralno generiranje arhitekture 25
3.1. Definicija 25
3.2. Metode generiranja geometrije 25
3.2.1. L – sustavi i „turtle“ gramatika u arhitekturi 25
3.2.2 Gramatika podjele i kontrolna gramatika 35
3.2.3. „CGA shape“ gramatika 42
3.2.4. Evoluiranje vanjskih blokova građevina 45
3.2.5. Metode generiranja osnovnih elemenata 47
3.2.6. Metode organizacije objekata 48
3.3. Proceduralno teksturiranje arhitekturalnih elemenata 51
3.3.1. Određivanje UV koordinata iz položaja točaka modela 52

3.3.2. Teksturiranje zasebnih cjelina ili poligona modela 54
3.3.3. Gramatika podjele 56
3.3.4. Teksturiranje slojevitom mrežom 57
3.4 Generiranje normala modela 59
3.4.1. Prosječna normala strane 59
3.4.2. Prosječna normala točke 60
3.4.3. Normala na temelju granice zaglađenosti 60
4. Ostvareni rezultati 62
4.1. Pregled ostvarenih tehnika 62
4.1.1. Generiranje pročelja zgrada na temelju plana katova 62
4.1.2 Organizacija interijera dvodimenzionalne uniformne mreže uz povezanost razina 63
4.1.3. Generiranje 2D staze hijerarhijom čvorova linijskih segmenata 65
4.1.4. Gramatika L-sustava 69
4.1.5. Gramatika kornjače 71
4.1.6. Gramatika oblika 72
4.2. Mogućnosti daljnjeg razvoja ostvarene implementacije 73
4.3. Razmatranja za buduća istraživanja u proceduralnom generiranju sadržaja 74
5. Zaključak 76
6. Literatura 77
7. Sažetak 80
8. Privitak 81
8.1. Formalizirane naredbe jezika L-sustava s fiktivnom kornjačom 81
8.2. Proširenje naredbi jezika L-sustava s fiktivnom kornjačom - parametrizirani sustavi 82
8.3. Popis naredbi „CGA shape“ gramatike iz rada „Procedural modeling of buildings“ (Müller 2006.) 83
8.4. Evoluiranje vanjskih blokova građevina – struktura jezika 84

1. Uvod

Ljudi vole izmišljati nove stvari, stvarati, graditi. Neki vole pravilnosti u organizaciji objekata, dok neki preferiraju kaos. Kaos ne postoji, te se sve temelji na pravilima. Svaki sustav ima svoju arhitekturu, koja se može prikazati konačnim brojem pravila njenog razvoja od osnovnih oblika i drugog, predefiniranog sadržaja. Napretkom računalne grafike postalo je moguće prikazati razne sustave pomoću pravila s visokom razinom detalja, uključujući arhitekturu od jednostavnih ukrašenih stupova do kompleksnih gradova. Ovaj rad bavi se pravilima koja definiraju arhitekturalne elemente, strukture i oblike, njihov izgled, povezanost i ulogu u kompliciranijem sustavu složenih formi.

Tema ovog rada je proceduralno generiranje arhitekture u virtualnim okruženjima. Cilj rada je proučavanje, implementacija i usporedba metoda za proceduralno generiranje arhitekture na temelju različitih ulaznih parametara i zahtjeva izlaza. Osnovni podaci rezultata u smislu 3D modela su položaji točaka, normale potrebne za sjenčanje, uv koordinate potrebne za teksturiranje, te hijerarhijska pohrana arhitekturalnih elemenata. Proučene metode pružaju različite raziname kvalitete, ekspresivnosti, jednostavnosti i brzine, te je u navođenju rezultata postignutih njihovim korištenjem navedena njihova detaljna usporedba.

Za praktični dio ovog rada razvijena je programska biblioteka u C# programskom jeziku, koja sadrži kod za generiranje, pregled, modifikaciju i pohranu zapisa pravila, parametara i geometrije modela kreiranih i korištenih u svim ostvarenim metodama, uključujući osnovne i najpoznatije metode navedene u radu te autorove originalne metode, od kojih se neke temelje na izvornim konceptima, a neke su hibrid postojećih metoda. Neke ostvarene metode su temeljno različite od izvornih, prilagođene željenim rezultatima i pojednostavljene kako bi bile praktične za moguću komercijalnu upotrebu u izradi proceduralnog arhitekturalnog sadržaja.

Uz ovaj dokument, priložen je sav sadržaj stvoren pri izradi biblioteke. Između ostalog, tu se nalaze korištene teksture, materijali, programi za sjenčanje, tekstualne datoteke predefiniranih parametara i pravila te scene s primjerima generirane geometrije, organizacije prostora te interaktivnim konzolama za upis naredbi za generiranje sadržaja.

2. Osnove proceduralnog generiranja

Ovo poglavlje daje definiciju proceduralnog generiranja, taksonomiju i podjelu metoda proceduralnog generiranja i njihova poželjna svojstva te pregled sadržaja koji se može proceduralno generirati s naglaskom na vrste sadržaja potrebne u ovom radu, uz prednosti i nedostatke procesa generiranja. Na kraju poglavlja navedeni su resursi za pregled praktične primjene proceduralnog generiranja, te par primjera. Ovo poglavlje služi kao objektivni uvod u proceduralno generiranje, te se svi spomenuti principi izravno primjenjuju na temu ovog rada, koja je fragment ovog velikog poglavlja u računarstvu i računalnoj grafici.

2.1. Definicije

Proceduralno generiranje sadržaja se može definirati kao algoritamsko kreiranje sadržaja s ograničenim ili indirektnim unosom parametara od strane korisnika, ili drugim riječima, računalni program koji može nezavisno ili manipulacijom korisnika kreirati novi sadržaj (Shaker, 2014.). Drugim rječima, to je produciranje medijskog sadržaja algoritamski, bez ručne obrade (u koju ne ulazi inicijalno postavljanje parametara i vođenje procesa generiranja odabirom postupaka i produciranih medija).

Ključna riječ u definiciji je sadržaj. Mogućnosti su neograničene, no u računalnim simulacijama, vizualizacijama te igrama, on se najčešće povezuje s vizualnim i zvučnim elementima, koji su kao konačni proizvod najočitiji korisnicima. Detaljniji pregled kategorija sadržaja dan je u podpoglavlju 2.5.

2.2. Taksonomija metoda proceduralnog generiranja

U nastavku je dana taksonomija metoda proceduralnog generiranja sadržaja, po uzoru na sliku 2 iz (Hendrikx, 2011.).

1. Pseudo-nasumični generatori brojeva

2. Generativne gramatike

   - Lindenmayer sustavi (L sustavi)
   - Gramatike podjele
   - Gramatike zidova

- Gramatike oblika

3. Obrada slika

- Binarna morfologija
- Konvolucijski filtri

4. Prostorni algoritmi

- Postavljanje slojeva i diskretnih ponavljajućih elemenata
- Podjela prostora i uniformne mreže
- Fraktali
- Voronoi dijagrami

5. Modeliranje i simulacija kompleksnih sustava

- Stanični automat
- Tenzorska polja
- Simulacija temeljena na agentima

6. Umjetna inteligencija

- Genetski algoritmi
- Umjetne neuronske mreže
- Planiranje i zadovoljavanje ograničenja

2.3. Podjela metoda proceduralnog generiranja prema svojstvima

S obzirom na velik broj generativnih problema i dostupnih metoda, potrebno je stvoriti podjelu metoda kako bi njihove sličnosti, razlike i primjene bile jasnije. U nastavku je dana podjela po uzoru na podjelu danu u (Togelius, 2011.). Većina danih podjela nije isključivo binarna, već su samo krajnosti između kojih može biti smještena neka metoda.

2.3.1. Kontinuirane i inicijalne metode

Kontinuirane metode generiraju novi sadržaj dok se postojeći istražuje i mijenja, dopuštajući beskonačnu varijaciju i besprijekoran prijelaz kroz virtualnu okolinu prikaza sadržaja. U kontinuiranim metodama teško je ostvariti pohranu sadržaja, te je cijela vizualizacija otežana implementacijom razmjene sadržaja u memoriji. Inicijalne metode kreiraju sadržaj isključivo prije pregledavanja. Ove metode moraju biti brze te omogućiti trajnu pohranu sadržaja, što potiče razmjenu sadržaja i pravila, i vodi do poboljšanja same metode. Primjer kontinuirane metode je slučaj kada korisnik uđe u zgradu, te se tada generira interijer zgrade, a inicijalni ekvivalent bi bio generiranje zgrade do konačnih detalja prije cjelokupne korisničke

upotrebe. Zahtjevi za kontinuirane metode su brzina, predvidljivo vrijeme izvođenja, niski memorijski zahtjevi i predvidljiva kvaliteta rezultata.

2.3.2. Nužne i proizvoljne metode

Nužne metode generiraju glavni sadržaj potreban za ispunjavanje logike zadanih pravila. Proizvoljne metode generiraju sporedan sadržaj koji služi kao ukras, nije nužan za funkcionalnost cjeline i može se ukloniti, u djelovima ili potpunosti zamijeniti s drugim glavnim ili sporednim sadržajem. Primjer nužne metode u generiranju zgrade je ona koja osigurava prohodnost cijele zgrade. Primjer proizvoljne metode je ona čiji rezultat je ukrasni, poput balkona, kipova ili freski na stropu.

2.3.3. Stupnjevi i dimenzije kontrole

Metode se međusobno razlikuju prema broju parametara - stupnju kontrole koji metoda pruža, točnije ekspresivnost metode, koja ne mora nužno smanjiti raznolikost ukoliko su parametri detaljnjih pravila stohastički uzorkovani. Nekad je poželjno da početni oblik sadržaja bude identičan u svakom izvršavanju, pri čemu se u algoritmima koriste fiksni izvorni parametri (engl. *seed*). Moguća je implementacija metode koja će sama razvijati svoja pravila stvaranja i izmjene sadržaja, pri čemu je teoretski moguće generiranje ograničeno jedino sklopovskom opremom.

Različiti mehanizmi kontrole mogu biti ugrađeni s obzirom na svojstva sadržaja kroz iteracije razvoja istog.

2.3.4. Generičke i adaptivne metode

Generičke metode su metode koje djeluju autonomno, neovisno o korisnikovim akcijama i djelovanju na okolinu. Adaptivne metode analiziraju korisnikove akcije i mijenjaju sadržaj kao odgovor na prethodno ponašanje korisnika. Ovakav pristup se često koristi kako bi se okolina prilagodila da pruži korisniku izazov ili veću zabavu, te da odgovara njegovim preferencijama na temelju povijesti akcija.

2.3.5. Stohastičke i determinističke metode

Determinističke metode omogućavaju reprodukciju identičnog sadržaja, dok stohastičke metode u pravilu daju različit rezultat svaki put, što ovisi o rasponu i

tipu parametara. Potpuno determinističke metode se mogu smatrati oblikom kompresije podataka. Izvrstan primjer ovoga je .kkrieger (.theprodukkt, 2004.).

2.3.6. Konstrukcijske i testne metode

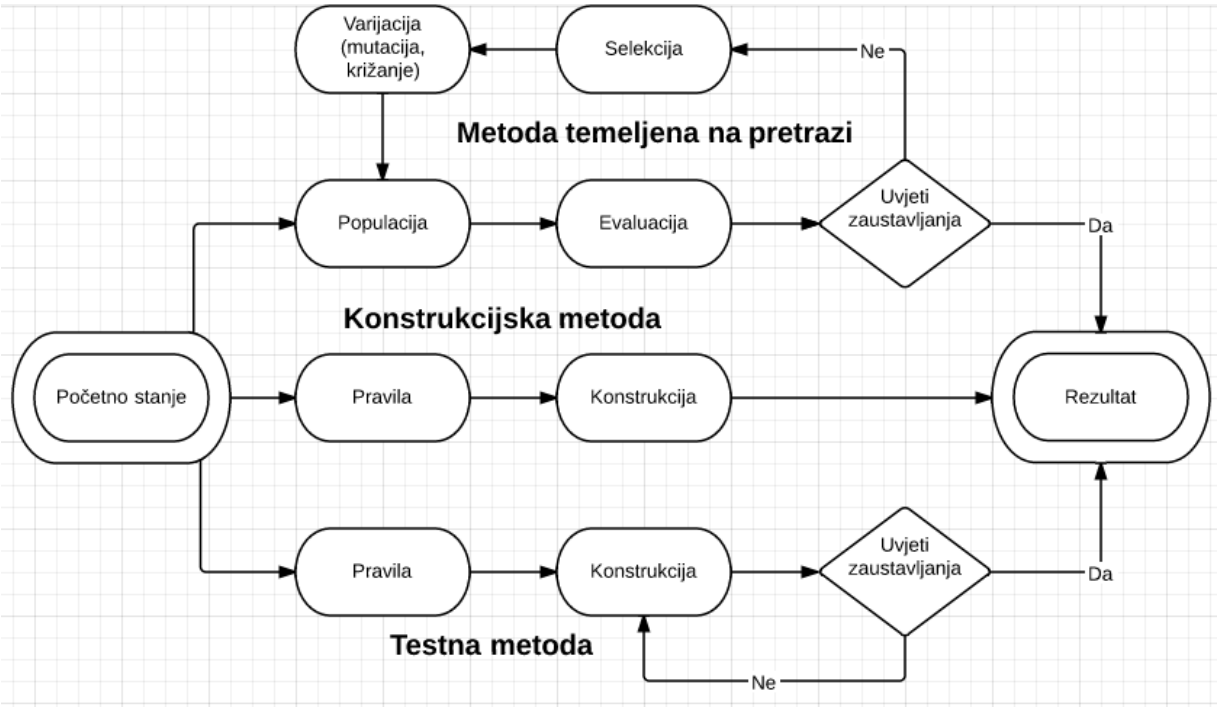
U konstrukcijskim metodama sadržaj je generiran jednom i promjene nisu dozvoljene, najčešće zbog vremenskog ograničenja i izvođenja u svrhu prikaza u stvarnom vremenu, zbog čega je potrebno osigurati da se sadržaj korigira prema pravilima tijekom procesa generiranja. Češće su testne metode koje se pokreću zadani broj puta ili dok se ne dobije zadovoljavajući rezultat, odbacujući prethodno generiran sadržaj nedovoljne kvalitete.

2.3.6.1. Metode temeljene na pretrazi

Posebna grana testnih metoda su metode temeljene na pretrazi koje posjeduju slijedeće dvije karakteristike:

- Testna metoda nema jednostavnu posljedicu poput prihvaćanja ili odbacivanja sadržaja, već ga ocjenjuje skalarom ili vektorom realnih brojeva, poput funkcije dobrote kod neuronskih mreža.
- Generiranje novog kandidata ovisi o dobroti pridruženoj prethodno evaluiranim instancama sadržaja, u svrhu poboljšanja dobrote kod novog kandidata

Unatoč tome što se ove metode temelje na evolucijskim algoritmima, izvorno je odabran termin „temeljene na pretrazi“ kako bi se obuhvatile sve heurističke i stohastičke metode pretrage i optimizacije. Slika 2.3.1. prikazuje razlike u postupcima između metoda iz podpoglavlja 2.2.6.



Slika 2.3.1. – prikaz usporedbe akcija potrebnih za metode temeljene na pretrazi, konstrukcijske te testne metode.

2.3.6.2. Prikaz sadržaja i prostor pretrage

U izvršavanju metode temeljene na pretrazi, svaki generirani sadržaj čini jednu jedinku Način pohrane sadržaja u jedinku određuje dimenzionalnost prostora pretrage algoritma prilikom varijacije postojećeg u novi sadržaj. S obzirom na vrstu prikaza razlikujemo izravan i neizravan zapis. U izravnom zapisu sadržaj je jasno i jednoznačno predstavljen, te je prostor pretrage potpun s obzirom na domenu prikaza sadržaja. U neizravnom zapisu, jedna jedinka sadržaja predstavljena je skraćenim oblikom iz kojeg je posljedično dobiven sadržaj (poput početnih parametara generiranja), no s obzirom da postupci često uključuju neku razinu nedeterminizma, te sadržaji u potpunom zapisu imaju puno veću dimenzionalnost od neizravnih zapisa, korištenje te vrste znatno ograničava prostor pretrage.

Kao primjer ovisnosti prikaza sadržaja i prostora pretrage, dano je 5 zapisa strukture labirinta, poredanih od najizravnijeg do najneizravnijeg:

1. Zapis sadržaja svake ćelije labirinta (poput zida, vrata, praznog prostora, sadržaja – entiteta, resursa..)
2. Zapis koji sadrži položaj, orijentaciju i duljinu zidova labirinta

3. Zapis različitih iskoristivih uzoraka zidova i slobodnog prostora, te lista njihove distribucije uz uniformne transformacije u prostoru sadržaja
4. Zapis željenih svojstava poput broja soba, vrata, entiteta, duljina zidova i faktor grananja zidova
5. Početni broj uzorkovanja pseudonasumičnih brojeva

Prvi zapis pokriva sve mogućnosti prostora pretrage i varijacije nad jedinkama tog zapisa imaju lokalni utjecaj – mogu promijeniti samo sadržaj jedne ćelije, no duljina zapisa odgovara umnošku dimenzija labirinta. Zadnji zapis pretražuje jednodimenzionalni prostor, no nije dovoljan da precizno odredi sadržaj. Zapisi 2 i 3 pružaju način opisa koji može prekriti potpuni prostor uz sub-linearni rast zapisa u odnosu na dimenzije labirinta, što je idealno za praktičnu upotrebu.

2.3.6.3. Funkcije evaluacije

Slično kao u evolucijskim algoritmima, evaluacijska funkcija sadržaja može biti funkcija dobrote ili funkcija kazne. Postoje tri klase fukcija koje se upotrebljavaju u evaluaciji proceduralno generiranog sadržaja:

1. Fukcije izravne evaluacije – iz instance sadržaja ekstrapoliraju se određene informacije koje izravno sudjeluju u funkciji dobrote (poput ukupnog broja puteva od početne točke do izlaza iz labirinta)
2. Funkcije temeljene na simulaciji – agent ispituje sadržaj u simulaciji te su rezultati agentovih akcija (poput vremena prolaska labirinta ili količine prijeđenog puta) ulazi u funkciju dobrote. Ove funkcije se dijele na statične i dinamične s obzirom na to uči li agent prilikom simulacije, gdje svojstvo agentovog učenja i savladavanja sadržaja može biti ulaz u funkciju dobrote. Ove funkcije mogu biti vremenski zahtjevne, tako da se ne koriste u praksi u aplikacijama korištenima u realnom vremenu.
3. Interaktivne funkcije – dobrota sadržaja se evaluira prilikom njegovog korištenja u aplikaciji na temelju korisničkog bodovanja. Podaci se mogu skupljati eksplicitno ispitivanjem korisnika o dojmu, ili implicitno mjerenjem akcija korisnika.

2.3.6.4. Nedostaci metoda temeljene na pretrazi

S obzirom da koriste genetske operatore, ove metode su stohastičke. Ne postoji način određivanja rezultata, ne postoji način reprodukcije identičnog

rezultata (osim pohrane najboljeg rezultata s njegovom dobrotom), niti općeniti dokaz da metaheuristički algoritmi konvergiraju u konačnom vremenu. Uz sve navedeno nije zajamčeno da će ove metode generirati zadovoljavajuće rezultate. Ako se pri evaluaciji sadržaja koriste simulacije, metoda nije prikladna za korištenje u stvarnom vremenu.

2.3.7. Automatizirane metode i združeno autorstvo

Do nedavno, metode su bile isključivo automatizirane, te su autori algoritama bili jedini koji su podešavali parametre do zadovoljivosti problema, no pojavila se nova paradigma zajedničke inicijative koja se fokusira na korištenje autorovih i korisničkih postavki kako bi se ostvario proceduralni sadržaj.

Primjer za to je sustav *Tanagra* [30] gdje dizajner unese osnovne podatke za 2D razinu, te algoritam generira djelove koji nedostaju uz zadržavanje logike danih pravila. Drugi primjer je *SketchaWorld* [27] razvojna okolina u kojoj dizajneri mogu ručno stvarati i uređivati parametre gradova i okoline dok sustav održava postojeći model konzistentnim s pravilima.

2.4. Poželjna svojstva proceduralnog generiranja

Metode proceduralnog generiranja se mogu smatrati rješenjima problema stvaranja sadržaja. Rješenje može biti različitog karaktera, ovisno o željenom sadržaju. U generiranju često postoje kompromisi između brzine i kvalitete, ekspresivnosti, različitosti i pouzdanosti rješavanja problema. U nastavku su dana poželjna svojstva proceduralnih rješenja:

1. Brzina – ovisno o količini sadržaja i praktičnoj upotrebi, potrebna brzina može varirati od maksimalno par milisekundi za prikaz novog sadržaja u realnom vremenu koji se osvježava svakih par okvira prikaza (npr. sadržaj pokrenute igre), do par mjeseci za generiranje okoline koja se može pohraniti i proučavati kada se jednom generira (npr. organizacija prometnica i/ili nastambi na nekoj geografskoj lokaciji prilagođena vremenskim uvjetima, klimi, dostupnim materijalima i slično).

2. Memorijski lagana – metode bi trebale imati svojstvo ekstrapolacije proceduralnog sadržaja iz malenog broja podataka, te imati maleno

zauzeće memorije pri kreiranju, dok sam generiran sadržaj smije imati veće memorijske zahtjeve, no i dalje je poželjno da se određeni djelovi konačnog rezultata mogu rekonstruirati iz dijela početnih podataka i rezultata bez ponavljanja cijelog postupka, kako bi se uštedilo na vremenu pri pregledavanju memorijski zahtjevnog rezultata.

3. Čitkost ulaznih parametara – neki generatori stvaraju sadržaj koji savršeno odgovara rješenju problema, no kako bi se za sličan problem dobilo približno kvalitetno riješenje, potrebno je potrošiti previše vremena na analizu pravila i interpretacije ulaznih podataka (pogotovo ako osoba koja stvara ulazne podatke za rješavanje drugog problema nije ista osoba koja je riješila prethodni problem). U većini slučajeva poboljšanje čitkosti uz dobru brzinu pruža veliku moć eksperimentiranja, što dovodi do bržeg nalaženja boljih parametara ukoliko su oni nejasni iz početnog problema.

4. Čitljivost rezultata – s obzirom da se sadržaj proceduralno generira u memoriji, on je u konačnom obliku pohranjen binarno, te ga je potrebno što jasnije tekstualno ili vizualno prikazati, kako bi se mogla što bolje interpretirati njegova kvaliteta ili uočiti pogreške i način njihovog uklanjanja. Kao primjer, u vizualizaciji proceduralno generiranog interijera zgrade može biti očito kako nedostaju vrata, ima premalo prozora ili postoji prostorija omeđena zidovima, dok iz tekstualnog oblika to ne bi bilo očito, dok je za generiranje rasporeda događaja tekstualni opis uz oznake vremenskih trenutaka prikladniji od isključivo grafičkog prikaza animacije istih. Ukratko, potrebno je jasno označiti, tekstualno zabilježiti ili grafički pregledno prikazati generirane rezultate.

5. Pouzdanost – potrebno je osigurati da će generirani rezultat zadovoljiti sva pravila na način da konačni rezultat ima logičkog smisla, ovisno o sadržaju; ako generirana građevina nije prohodna do ključnog dijela zgrade (poput trezora u banci), to je loš rezultat i popravljati ga ručno nakon generiranja ili pisati pravila koja će naknadno rješavati moguće probleme je nekvalitetno rješavanje problema, potrebno je odmah napisati takav sustav koji će

pouzdano svaki put (ukoliko pravila nisu neispravna ni kontradiktorna) generirati smislen, ispravan, upotrebljiv rezultat.

6. Kontrola – često je poželjno da algoritam generiranja ima kontrolu u obliku pravila razvoja sadržaja, pružanje ulaznih primjera sadržaja, ograničenja performansi (poput maksimalne memorije ili vremena koje smije koristiti za generiranje sadržaja), kreiranje sadržaja određene kategorije (poput drvene kućice ili mramorne fontane), definicije parametara koji omogućavaju nedeterminizam (vjerojatnosti ili drugi parametarski rasponi), ili niza specifičnih slijednih pravila koja generiraju jasno određeni rezultat.

7. Ekspresivnost i raznolikost – često se javlja potreba za jednom od dvije krajnosti, jedna je proceduralna obrada i generiranje sadržaja prema strogim ekspresivnim pravilima, a druga je masovno generiranje što više raznolikog sadržaja. Često s većom ekspresivnošću smanjujemo raznolikost, jer će sve biti predodređeno parametrima pravila, no ukoliko smanjimo ekspresivnost kako bi dobili na raznolikosti, rezultati će imati manje logičkog smisla i neće biti praktično upotrebljivi.

8. Kreativnost i uvjerljivost – u većini slučajeva poželjno je da sadržaj izgleda kao da je stvoren ručno, a ne umjetno. Ljudi su nesavršeni i kreativni, no uz ugradnju minornih pogrešaka ili blagog kaosa u organizaciji, teško je implementirati kreativnost proceduralnom generatoru, ali uz kvalitetan balans ekspresivnosti i raznolikosti, u stvaranju više rezultata moguće je dobiti neočekivan rezultat koji će izgledati kreativno, novo, ljudski.

2.5. Podjela proceduralno stvorivog sadržaja

Na popisu danom u nastavku prikazana je podjela sadržaja koji se može proceduralno generirati, po uzoru na tablicu danu u (Hendrikx, 2011.). Tablica se temelji na podjeli sadržaja u 6 kategorija, stupnjevanu po razini složenosti i realnosti prikaza elemenata pravog svijeta. Uz svaku podkategoriju navedene su grupe metoda iz taksonomije metoda koje su se do sada pokazale uspješnima u

generiranju sadržaja te podkategorije, što ne znači da se u budućnosti neće razviti algoritam koji kvalitetno koristi principe druge grupe metoda.

Podjela proceduralnog sadržaja:

1) Bitovni sadržaj

   Osnovni sadržaj od kojeg se sastoji svaka virtualna okolina:

   a) Teksture – najčešće generirane grupama metoda 1, 3, 4 i 5 iz podjele metoda u poglavlju 2.2.

   b) Zvuk – generiran grupama metoda 1, 2, 5 i 6, gdje su najuspješnije metode generativnih gramatika pri čemu skladatelji definiraju pravila generiranja

   c) Modeli – generirani grupama metoda 2, 4 i 5, gdje su najuspješniji L-sustavi te skupovi pravila oblika koja su definirali korisnici

   d) Ponašanje entiteta (fizikalni zakoni, sile, odgovor, reakcija i promjenja stanja entiteta na interakciju) – 2., 5. I 6. grupa metoda

   e) Efekti – viša razina sadržaja koja se može sastojati od svega ili podskupa ostatka kategorija bitovnog sadržaja (na primjer: vatra, voda, odron zemlje, oblaci) – 1., 3., i 5. grupa metoda

2) Prostor

   Okruženje koje koristi bitovni sadržaj prostorno organiziran u smislenu cjelinu. Postoje uspješni članci generiranja prostora svim grupama metoda generiranja sadržaja

   a) Interijer – opis strukture i relativnog pozicioniranja objekata interijera u sobe povezane hodnicima, stubama i drugim elementima u slojeve

   b) Eksterijer – elevacija i struktura terena i vodenih masa – agenti, nasumični brojevi te genetski algoritmi

3) Sustavi

   a) Ekosustavi – smještaj, evolucija i interakcija flore i faune kroz pravila i algoritme – ova razina složenosti u kojoj dolazi do izmjene sadržaja najkvalitetnije se može implementirati agentima koji stvaraju i agentima koji troše resurse, te tako djeluju jedni na druge umjetnom inteligencijom, iako su se i druge metode pokazale uspješnima

b) Mreža prometnica eksterijera i transportacija unutar te između naselja – sve grupe metoda su se pokazale praktičnima

c) Urbana okruženja - generiranje evolucijskim razvojem, prostornom podjelom i predefiniranim pravilima

d) Ponašanje entiteta – interakcija između korisnika i okoline (reakcija drugih entiteta na korisnikove akcije, grupno ponašanje) – modeliranje metodom zadovoljenja ograničenja

4) Scenariji

Opis ponašanja, akcija te odgovora na moguće korisničke akcije

a) „Slagalice“ su problemi na koje korisnik može naći riješenje na temelju prethodnog znanja ili sistematičnog pretraživanja prostora rješenja – ovisno o željenoj kvaliteti i složenosti, može se primjeniti nasumično generiranje, pravila ili genetski algoritmi

b) Priča – logika događaja, cilj za korisnika – zadovoljenje ograničenja na temelju definiranih pravila

c) Razine – odvojeni prolazni prostori virtualne okoline, u kojoj se korisnik kreće obavljajući zadatke i izbjegavajući prepreke – praktično generirane svim metodama

5) Dizajn

Dizajn sustava koji sadrži pravila, ciljeve te estetske komponente virtualne okoline – ostvareno složenim gramatikama za zapis pravila te evolucijom istih

6) Izvedeni sadržaj

sadržaj koji je kreiran kao sporedni produkt virtualnog svijeta simulacije, te može značajno povećati doživljaj korisniku - s obzirom na kompleksnost, ne postoji velik broj primjera proceduralnog generiranja ove kategorije sadržaja, a postojeći se temelje na predefiniranim pravilima.

a) vijesti i objave događaja u okolini za koje je korisnik izravno odgovoran

2.6. Granice proceduralnog generiranja

Uz napredak znanosti i tehnologije, razvoja algoritama s drugih područja, rješavaju se problemi i pomiču granice proceduralnog generiranja, no i dalje postoji par neriješenih problema koji predstavljaju granicu i viziju budućnosti proceduralnog generiranja. U nastavku je dano par takvih problema.

1. Višerazinsko generiranje raznovrsnog sadržaja – generator sadržaja koji za dana pravila, vremenski (povijesni) period, prostorne, geološke i meteorološke parametre, kulturne značajke i razvojno okruženje može generirati sav sadržaj integrirane okoline koji bi se potom mogao virtualno istražiti. Takav generator bi mogao imati fokus i na ljudski faktor, kao utjecaj na okolinu, povijest, kulturu, ekonomiju i slično, ili biti fokusiran samo na utjecaj prolaska vremena, vremenskih pojava, flore i faune na okolinu, što bi rezultiralo virtualnim ekosustavom. S obzirom na rezultat i slijedeće probleme, u ovom problemu se generiranje odnosi na organizaciju postojećeg sadržaja u koherentnu cjelinu.

2. Generiranje pravila ponašanja sadržaja, poput pasivnih reakcija okoline, ili entiteta upravljanih umjetnom inteligencijom.

3. Generiranje grafičkog i zvučnog sadržaja u virtualnim okolinama – kroz povijest razvoja računala nastao je velik broj algoritama koji do neke granice uspješno rješavaju ove probleme, no oni su trenutačno još u ranom stadiju razvoja s obzirom na njihov potencijal, te se trebaju još puno unaprijediti kako bi mogli pružiti potpunu kontekstno – neovisnu generaciju uz minimalne ručno unesene ulazne podatke.

4. U procesu proceduralnog generiranja sadržaja postoji rekurzivna meta-razina: generiranje pravila koja bi generirala pravila stvaranja sadržaja, te sve slijedeće meta-razine.

2.7. Prednosti i nedostaci

Sva obilježja navedena u podpoglavlju 2.4. mogu se izravno navesti kao prednosti:

Brzina, maleni memorijski zahtjevi, čitljivost ulaza i izlaza, pouzdanost, konrola, ekspresivnost i raznolikost, kreativnost i uvjerljivost.

no potrebno je pogledati svaku od tih karakteristika sa strane stvaranja i korištenja algoritama, procesa, okruženja proceduralnog generiranja.

Brzina - Pri korištenju gotovih programa za proceduralno generiranje, postavljanje parametara, izvršavanje i generiranje sadržaja treba i najčešće je kratkog vremenskog roka, no ukoliko je skupina ljudi stvarala program za korištenje u vlastite svrhe, treba uzeti u obzir vrijeme izgradnje programa za smislen i kvalitetan izlazni sadržaj, te se možda projekt za to vrijeme mogao „ručno“ obaviti prije i kvalitetnije. S druge strane, ako projektni tim koristi tuđi program, treba uračunati vrijeme savladavanja i privikavanja na program, sve dok se ne dobije koristan sadržaj, čime se gubi na vremenu do ostvarenja projekta. Unatoč tome, korištenjem takvih procesa koji obavljaju posao brže od čovjeka omogućava se rapidna izrada prototipa, iteracije generiranja, testiranje i evaluaciju istog te drugog sadržaja poput koda koji koristi privremene proceduralno generirane modele i teksture u virtualnom okruženju (kao u računalnim igrama).

Budžet – korištenjem proceduralnog generiranja sadržaja smanjuje se ili uklanja potreba za ručnim stvaranjem sadržaja, što smanjuje trošak na ljudske resurse, ali moguće je da se gubi na kvaliteti, izražajnosti i osobnosti sadržaja koja se može postići samo ljudskom kreativnošću.

Nerepetitivnost - Proceduralno generiranje može generirati beskonačan nerepetitivan sadržaj. U ovoj stavci riječ „nerepetitivan“ je upitna. Naravno, postoje načini da se parametri i pravila generiranja uzorkuju na temelju ulaznih podataka iz pravog svijeta, koji nikad neće biti isti, poput vremena, korisničkog pomicanja miša, šuma odjeka Velikog praska i slično, ali pitanje je koliko zapravo različit i zanimljiv sadržaj s blago različitim ulazom može biti – to ovisi o kvaliteti proceduralnog sustava i ekspresivnosti pravila, koja je potencijalno neograničena.

Interaktivnost - Kombiniranjem postupka generiranja sadržaja s neuronskim mrežama ili drugim modulima strojnog učenja s povratnom informacijom od korisnika o mjeri njegovog zadovoljstva stvorenim sadržajem, on se može generirati tako da se maksimizira užitak konzumiranja sadržaja na način specifičan za svakog korisnika.

Kreativnost - Proceduralno generiranje može stvoriti neočekivan, neobičan i ljudima nezamisliv sadržaj koji nije izravno očiti rezultat ulaznih podataka, ali je i dalje ispravno rješenje zadanog problema generiranja sadržaja. Izvan područja računalnih igara, taj fenomen je poznat kao evolucijski dizajn (Shaker, 2014.). Stvaranjem neočekivanog može se potaknuti ljude na daljnji razvoj, razmišljanja i proširivanje postupaka proceduralnog generiranja.

Razumijevanje – Proučavanjem procesa proceduralnog generiranja, postiže se shvaćanje procesa prirodnog nastanka ili umjetnog dizajna generiranog sadržaja. Kada se implementira postupak za razvoj sadržaja dobije se bolji uvid u ručni postupak kreiranja istog, što vodi do razvoja kvalitetnijih i intuitivnijih postupaka za ručno i proceduralno generiranje, formaliziranja i proširivanja pravila te njihovih ekspresivnosti.

Glavni nedostatak proceduralnog generiranja je nedostatak shvaćanja pravila postupaka, problema, formalnog i tehničkog zapisa neke domene sadržaja, no taj nedostatak danas nije prisutan, jer postoji niz znanstvenih članaka, amaterskih implementacija i komercijalnih programa za generiranje, te se iz njih može puno saznati i naučiti.

Sa korisničke strane, ovisno o obrazovanju i pozadini moguća je barijera neshvaćanja povezanosti pravila i parametara s izlaznim rezultatom programa, no to se rješava kvalitetnom dokumentacijom, korisničkim forumom, tekstualnim ili video zapisom opisanim procesom korištenja programa te javno dostupnim praktičnim primjerima, tako da se u današnje vrijeme za relevantne programe i ovaj nedostatak uklanja.

Jedini realistični nedostatak koje preostaje je postizanje ravnoteže između brzine, jednostavnosti i ekspresivnosti parametara, koji će uvijek biti prisutan u javno dostupnim programima.

2.8. Praktična primjena proceduralnog generiranja

2.8.1. Virtualno preslikavanje stvarnog sadržaja

Proučavanjem i korištenjem procesa proceduralnog generiranja razvijaju se nove metode, koje će u konačnici moći proceduralno iz stvarnog sadržaja stvoriti virtualni.

Virtualno preslikavanje Zemlje je primjer koji obuhvaća većinu praktičnih primjena virtualnog preslikavanja proceduralnog sadržaja:

- Virtualni turizam – istraživanje mjesta prije posjeta ili zbog nemogućnosti posjeta
- Povijesni turizam – virtualni pregled mjesta koja više ne postoje ili nisu u tako očuvanom stanju
- Edukacija – geografska referenca u edukaciji o izgledu i ljudskom utjecaju na planet i svemir
- Planiranje – planiranje korištenja posjeda zemlje, izgradnje naselja, organizacije prometa prirodnih resursa
- Vizualizacija vremenskih uvjeta, nepogoda te utjecaja na okoliš
- Vizualizacija postojećih nekretnina za prodaju ili proceduralno generiranih za reklamiranje zemljišta
- Sudjelovanje javnosti u oblikovanju i iskorištavanju javnih površina

2.8.2. Filmska industrija

Proceduralno generiranje se često koristi u filmskoj industriji kako bi se u kratkom vremenskom roku stvorio vizualno zanimljiv i realističan prizor.

Jedan od principa primjene naziva se „tvornica nesavršenosti“, u kojem umjetnici generiraju velik broj instanci istog objekta s različitim svojstvima tako da svi objekti izgledaju skladno a budu međusobno različiti, što je potaknuto stvarim svijetom, gdje ne postoje identične instance istog objekta. Na primjer, umjetnik bi mogao stvoriti policu trgovine voćem, te bi mu proceduralno generiranje omogućilo da stvori police koje bi imali različite dimenzije, izgled i raspored sadržaja na njima.

Od svih grupa metoda iz poglavlja 4.2. u praktičnoj upotrebi je najizraženija prva grupa, algoritmi pseudonasumičnih generiranja brojeva. Njihova primjena je vidljiva u čestičnim efektima (korištenim u prirodnim ili vremenskim pojavama) ili uzorkovanju pseudopovršine (korištenim u generiranju vode, terena, oblaka i slično).

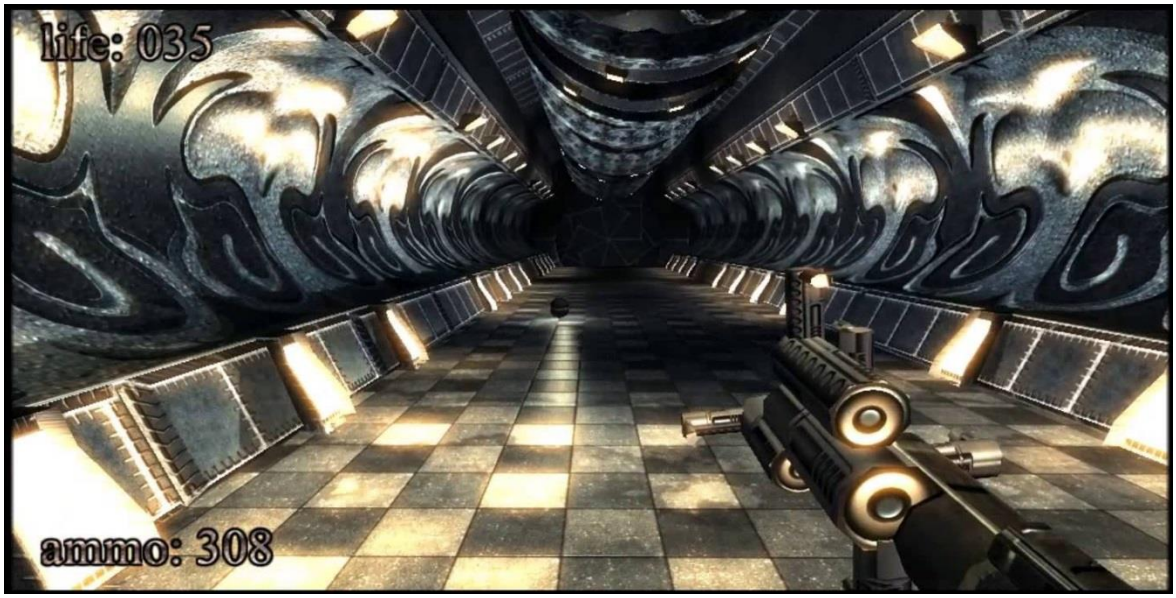
Primjer korištenja proceduralnog generiranja u preprodukciji filmova i serija je E-on Vue, program za stvaranje prikaza virtualne 3d okoline. (E-on vue portfolio).

2.8.3. Računalne igre i simulacije

Stranica virtualne enciklopedije sadrži popis igara koje koriste proceduralno generiranje (andrewdoull, 2008.), a tablica 1 iz (Hendrikx 2011.) pruža pregled računalnih igara uz označene kategorije proceduralnog sadržaja u njima. Zbog velikog broja takvih igara, ovdje će se navesti samo par primjera koji su značajni ili zbog primjene tehnike, ili zbog njihove popularnosti dobivene na temelju proceduralnog sadržaja ili razine igrivosti koja je proizašla iz njega.

- Rescue on Fractalus (1984.) – proceduralno generiranje terena
- The Sentinel (1986.) – više tisuća razina igre pohranjenih u 64kB memorije koda te proceduralno generirano pri izvršavanju
- .kkrieger (2004.) – igra iz prvog lica (engl. first-person shooter) koja se sastoji isključivo od 96kB koda, te generira teksture, zvučne i vizualne efekte, razine, oružja i neprijatelje – cijeli vidljivi sadržaj igre je u potpunosti proceduralan. Slika 2.8.1. prikazuje scene iz igre.
- Dwarf Fortress (2006.) – proceduralno generiran izgled svijeta, entiteti, kultura i povijest
- Spore (2008.) – proceduralno generiranje okoline, modela i tekstura stvorenja
- Minecraft (2009.) – igra u kojoj se inicijalno stanje svijeta (položaj blokova, njihov materijal, drugi entiteti) generira proceduralno. Slika 2.8.2. prikazuje modificiran original igre kojem su dodani napredni programi za sjenčanje.
- Diablo I (1998.), Diablo II (2000.), Diablo III (2012.), Torchlight (2009.), Torchlight 2 (2012.), Path of Exile (2013.) – proceduralno generiranje razina od elementarnih segmenata ili povezivanje gotovih cjelina, smještanje

objekata, entiteta i generiranje skupivih objekata. Slike 2.8.3. – 2.8.5. prikazuju scene iz navedenih igara

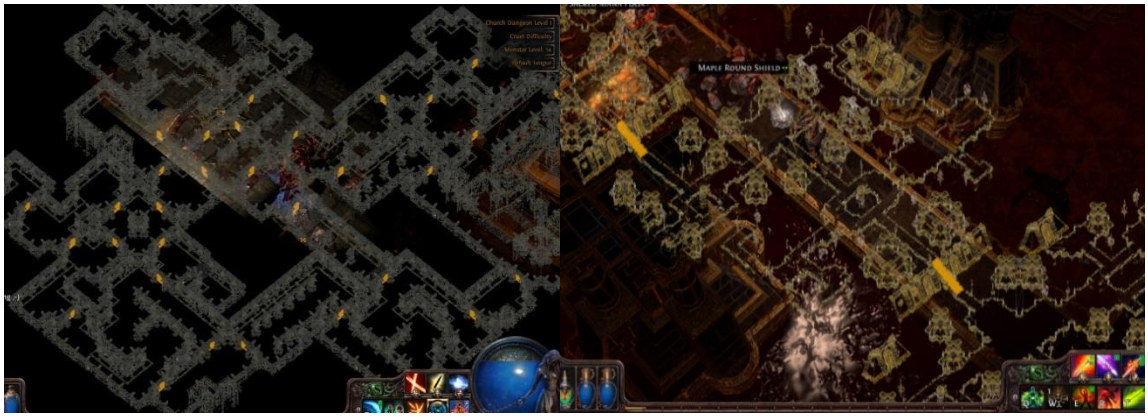


Slika 2.8.1. – prikaz scene iz igre .kkrieger

Slika 2.8.2. – prikaz scene iz igre Minecraft

Slika 2.8.3. – prikaz višeslojne arhitekture razine igre Torchlight

Slika 2.8.4. – prikaz arhitekture interijera razine igre Diablo II



Slika 2.8.5. – prikaz arhitekture interijera razine igre Path of Exile

2.8.4. Popis programa koji koriste proceduralno generiranje

Najopširniji popis programa za prodecuralno generiranje virtualnog sadržaja može se naći na stranici projekta virtualnog terena (Discoe, 1997.), virtualne enciklopedije proceduralnog generiranja sadržaja (andrewdoull, 2008.) te stranici o proceduralnom generiranju na wikipediji (Proceduralno generiranje, 2005.).

3. Proceduralno generiranje arhitekture

3.1. Definicija

Proceduralno generiranje arhitekture je algoritamsko kreiranje arhitekturalnog sadržaja, koja se u osnovi dijeli na generiranje građevnih elemenata i organiziranu podjelu prostora. Podjela prostora se može dalje podijeliti na organizaciju interijera, eksterijera (fasade) ili organizaciju uređenja osnovnih i složenijih elemenata. Pod pojmom generiranje obuhvaćeno je stvaranje modela sa svim podacima koji ga definiraju. U osnovnom obliku modela ti podaci su: položaji točaka, UV koordinata i normale.

3D Modeliranje je proces stvaranja virtualnog modela, što se može ostvariti parametrima ili modifikacijom i spajanjem postojećih modela.

Teksturiranje je stvaranje UV koordinata, njihovo pridruživanje točkama modela, te njihova modifikacija.

Dodavanje normala modelu je najjednostavniji dio generiranja modela, te se tretira kao dio modeliranja koji je automatiziran, te se lako provodi nakon generiranja cijelog modela (položaja točaka).

3.2. Metode generiranja geometrije

3.2.1. L – sustavi i „turtle“ gramatika u arhitekturi

Princip prepisivanja i promjene niza znakova na temelju gramatike je izvorno zamišljen u svrhu modeliranja biljaka i procesa njihovog rasta (Lindenmayer, 1990.) Ovo podpoglavlje istražuje primjenu L-sustava i „turtle“ gramatike u generiranju arhitekture (Hansmeyer, 2003.).

Vizualizacija L-sustava i metode preslikavanja strukture u arhitekturalni sadržaj

Proširenje jezika L – sustava kako bi reagirao na promjene okoline i prilagodio se širokom rasponu zahtjeva arhitekturalnog dizajna.

3.2.1.1. Kratki uvod u L-sustave

L-sustav temelji se na definiciji kontekstno nezavisne gramatike, gdje se konačan broj pravila sastoji od jednog nezavšnog znaka na lijevoj, te niza završnih i nezavršnih znakova na desnoj strani produkcije.

L-sustav se definira pomoću četvorke:

G = (V_N, V_T, P, S) = (nezavršni znakovi, završni znakovi, pravila generiranja, početni nezavršni znak). Provođenje L-sustava dijeli se na dva dijela: generiranje i interpretaciju. Glavni koncept korišten u generiranju je prepisivanje niza znakova, u kojem se svi znakovi iz jednog koraka zamjenjuju s nizom znakova u drugom koraku. Taj postupak se ponavlja željeni broj puta ili do ispunjenja drugih uvjeta. U djelu interpretacije, svaki znak iz niza znakova se može interpretirati kao akcija.

Primjer gramatike L-sustava:

({A,B}, {c}, A->ABA, B->Ac, A)

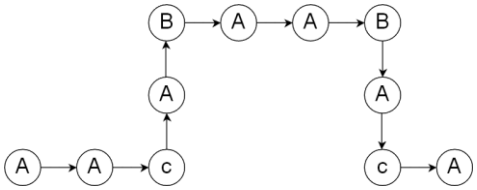
Primjer koji sadrži prvih par generacija dane gramatike:

0) A
1) ABA
2) ABAAcABA
3) ABAAcABAABAcABAAcABA

Ukoliko su znakovi preslikani kao akcije na slijedeći način:

A = idi naprijed, B = skreni desno, c = skreni lijevo

Vizualna interpretacija 3. generacije izgledala bi ovako:



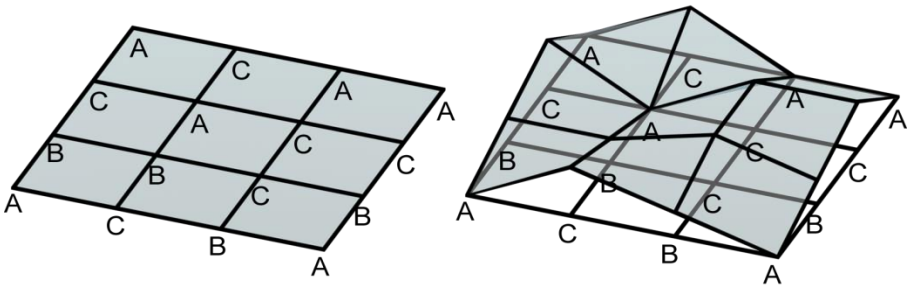
Slika 3.2.1. - Prikaz interpretacije 3. generacije L-sustava navedenog u primjeru

3.2.1.2. Vizualizacija L-sustava preslikavanjem

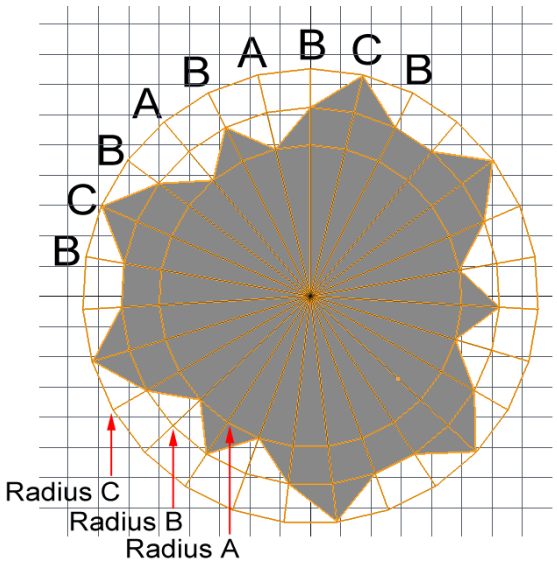
U teoriji, uz dovoljno kompleksnu gramatiku, preslikavanjem L-sustava na akcije moguće je postići svaki mogući oblik. Ovdje su navedene 3 jednostavnije verzije preslikavanja:

1) Preslikavanje na površine
   - Generacije (nizovi znakova) su preslikane na proizvoljnu površinu tako da svaki znak predstavlja amplitudu pomaka
   - Varijacije:

- Definiranje 3D pomaka, korištenjem 3 znaka po točci površine
- Korištenje različitih početnih modela, poput kocke, prizme, cilindra
- Preslikavanje nizova na različitim razinama, poput preslikavanja znaka na 1 točku, ili na mrežu od 4x4 točke, ili različitim osima
- Korištenje lokalnog koordinatnog sustava svake točke



Slika 3.2.2. – Prikaz primjene znakova niza interpretiranih kao vertikalni pomak (A=0, B=1, C=2) na ravnu plohu



Slika 3.2.3. – Prikaz interpretacije znakova niza kao iznosa radijalnih pomaka (prikazani su znakovi iznosi trećine niza, koji se ukupno tri puta ponavljaju u krugu)

Praktična primjena ove vrste preslikavanja svodi se na generiranje osnovnih elemenata iz više nizova koji se interpretiraju kao pomaci i putanje po različitim osima, rotacije, skaliranja i druga svojstva kao instanciranja drugih elemenata.

Primjer primjene jednostavnog niza u kreiranju stupa:

- Generiranje niza koji se interpretira kao radijalni pomak
- Niz visina točaka na y osi, koje bi kreirale vertikalne segmente
- Niz rotacija vertikalnih segmenata oko y osi
- Niz faktora skaliranja radiusa po y osi

Primjer primjene jednostavnog niza u kreiranju stepenica:

- Niz točaka staze koja definira položaj i poravnavanje stepenica (može biti pravocrtni, kružni, ili proizvoljan niz točaka, te je moguća dodatna interpolacija kako bi generirane stepenice bile ekvidistantne)
- Niz faktora rotacije
- Niz faktora skaliranja
- Dodatni nizovi za ogradu sa strana stepenica, među kojima može biti i niz za povremeno instanciranje dekorativnih modela

Primjer primjene niza u kreiranju lukova:

- Niz točaka koje definiraju temeljni 2D oblik
- Niz točaka koje čine luk
- Dodatni parametri granica luka odozgo i sa strane

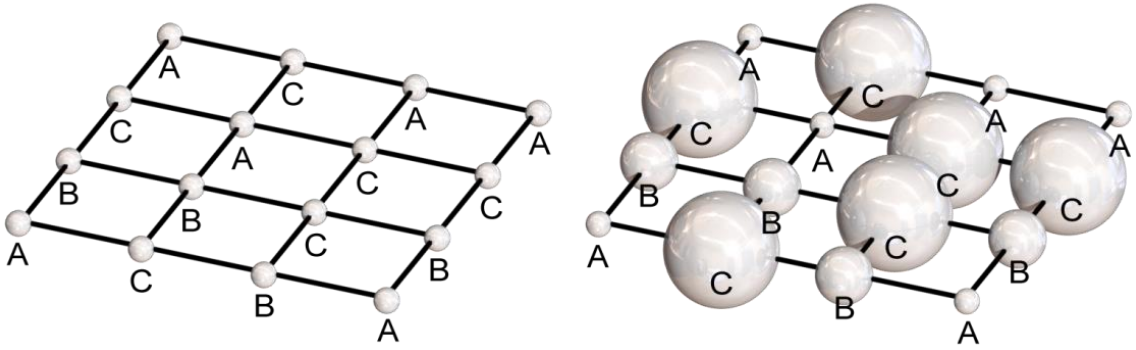
Vrijednosti svih nizova nakon interpretacije se mogu zagladiti ili podijeliti u međusegmente uz linearnu, polinomijalnu, Bezierovu, Catmull-Romovu interpolaciju i slično. Generirani model se također može „zagladiti“ podjelom poligona.

1) Preslikavanje na individualne objekte
    - Osnovna razlika između ovog i prethodnog preslikavanja je u tome što se u ovom slučaju ne stvara nova geometrija, već se modificira postojeća
    - Interpretacije nizova se primjenjuju na uređene skupove identičnih objekata
        - Na primjer pravocrtni niz, niz po putu, dvodimenzionalna ili trodimenzionalna matrica objekata

- Najjednostavniji primjer je kolonada ili krug stupova ili lukova

- Nizovima se mogu pridružiti proizvoljni atributi, koji ovdje nisu generativne, već transformacijske prirode
- Niz za instanciranje dodatnih objekata
- Niz za promjenu vidljivosti i materijala prikaza objekata
- Vrijede sve varijacije navedene za prethodno preslikavanje

Najjednostavniji primjer primjene ovog preslikavanja prikazan je na slici 5.2.2.3., uz korištenje kugle kao inicijalni model. Korišten je samo jedan niz interpretiran kao niz faktora skaliranja, gdje je A=1, B=2, C=3.



Slika 3.2.4. – Prikaz primjene interpretacije niza uz preslikavanje na individualne objekte

Praktična primjena ove vrste preslikavanja nalazi se u generiranju pravilnih poredaka postojećih modela, što se može primjeniti u generiranju skladnih nizova modela, od prolaznih građevina i strukturih, potpornih djelova, sve do detalja u dekoracijama drugih modela.

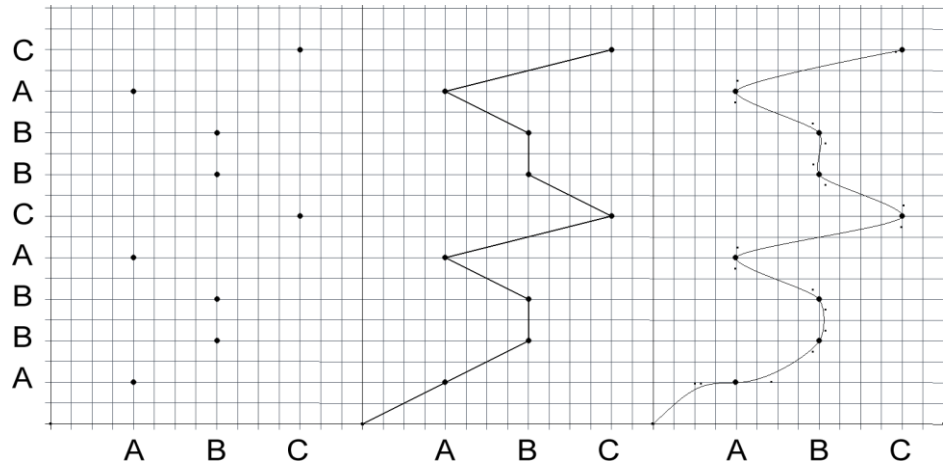
2) Derivirani oblici

Dobiveni nizovi interpretiraju se kao definirani položaji točaka: znak se interpretira kao pomak po jednoj osi, a njegov položaj u nizu kao pomak po drugoj osi. U ovom preslikavanju rezultat interpretacije se dodatno geometrijski obrađuje kako bi se stvorio model.

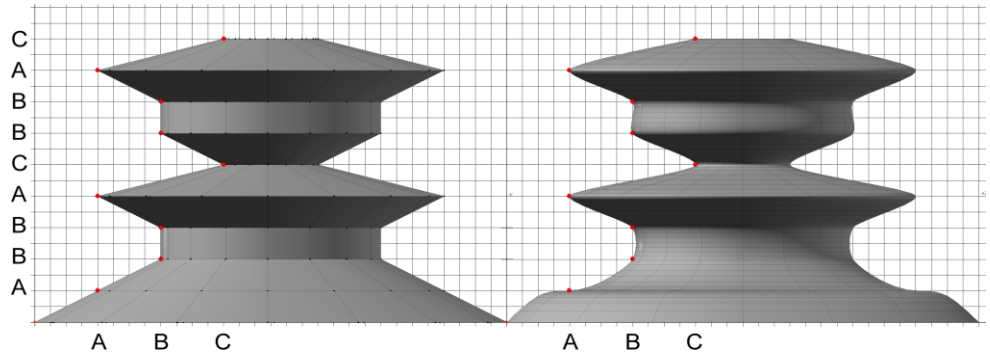
- Sve varijacije iz prethodnih preslikavanja su primjenjive i u ovom preslikavanju
    - Dodatna obrada niza točaka:

- Interpolacija ili paremetriziranje krivulje interpretiranim točkama
    - Zaglađivanje ili uklanjanje unutrašnjih oblika
    - Pretvaranje obrađenog niza točaka u geometriju:
    - Radijalno proširivanje

Slika 3.2.5. prikazuje primjenu interpretacije niza u svrhu stvaranja linearnog niza bridova, te niza vrhova povezanih Bezierovom kubnom krivuljom. Na slici 3.2.6. prikazan je rezultat radijalnog proširenja oba niza točaka, te su stvoreni kružni modeli nalik podnožju stupa. Korištena interpretacija preslikava pomake po X osi identično pomacima u prethodnim primjerima (A=1, B=2, C=3).



Slika 3.2.5. – prikaz niza točaka dobivenog interpretacijom niza znakova (lijevo), linearnog segmenta od točaka (sredina) te krivulje zadane točkama (desno),



Slika 3.2.6. – prikaz modela dobivenih radijalnim proširenjem linearnog segmenta (lijevo) te radijalnim proširenjem krivulje (desno)

3.2.1.3. „Turtle graphics“ – L-sustavi s fiktivnom kornjačom

„Turtle graphics“ ili grafika s fiktivnom kornjačom temelji se na interpretaciji znakova u kojoj se oni preslikavaju u naredbe koje slijedi fiktivna kornjača, te se

na različite načine vizualizira njen prođeni put. Minimalni dvodimenzionalni grafički sustav fiktivne kornjače dan je slijedećim naredbama:

A = korak unaprijed, B = okret lijevo, C = okret desno

Navedene naredbe mogu se proširiti na razne načine, a najčešće proširenje uključuje grananje, potpunu trodimenzionalnu rotaciju te modularne komponente. U privitku je priložen formalizirani vokabular L-Sustava koji koristi koncept stanja kornjače kao stanja sustava koji se mijenja naredbama interpretiranih iz znakova niza.

Stanje sustava se u svakom trenutku može prikazati stanjem kornjače, koje se sastoji od slijedećeg:

- Položaj – x,y,z koordinata
- Rotacija – 3 jedinična vektora koji čine ortonormalni koordinatni sustav
- Pomak – iznos pomaka (x,y,z) koji se primjenjuje pri translaciji kornjače
- Radijus – radijus cijevi koja se može dobiti proširenjem pređenog puta
- Kutevi rotacije – iznosi koji se primjenjuju pri rotacijij kornjače

Kako bi se jednom definicijom pravila i sustava mogli postići raznovrsni rezultati, uvode se stohastičke produkcije, gdje je svakoj produkciji zajedničkog nezavršnog znaka s lijeve strane dodijeljena vjerojatnost njenog izbora, čija suma mora biti 1 za sve produkcije istog znaka. Na primjer, dane su produkcije

A -> A; p = 0.5, A -> B; p = 0.3, A -> AC; p = 0.2

Te one mogu od početnog nezavršnog znaka A u dvije generacije stvoriti slijedeće nizove s danim vjerojatnostima:

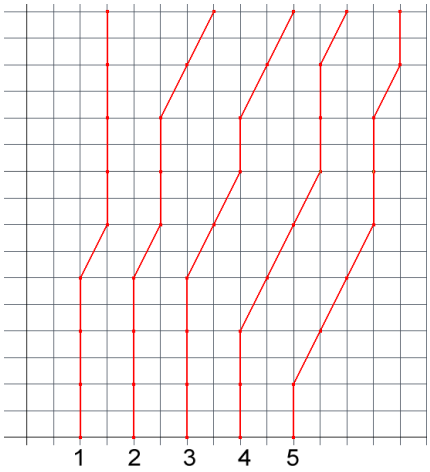
A – 0.25, AB – 0.15, AAC – 0.1, B – 0.3, AC – 0.1, BC – 0.06, ACC – 0.04

Stvorene generacije nizova slova mogu se istovremeno iskoristiti za kreiranje nezavisnih elemenata, slojevitog sadržaja, poput niza stupova, lukova ili stepenica koje se međusobno blago razlikuju, ili organizacije zidova unutar katova zgrade. U nastavku dane su slike koje prikazuju točke povezane u segmente nastale interpretiranjem nizova više uzastopnih generacija (slika 5.2.5.) te slika koja prikazuje radijalno proširenje tih linijskih segmenata (slika 5.2.6.).

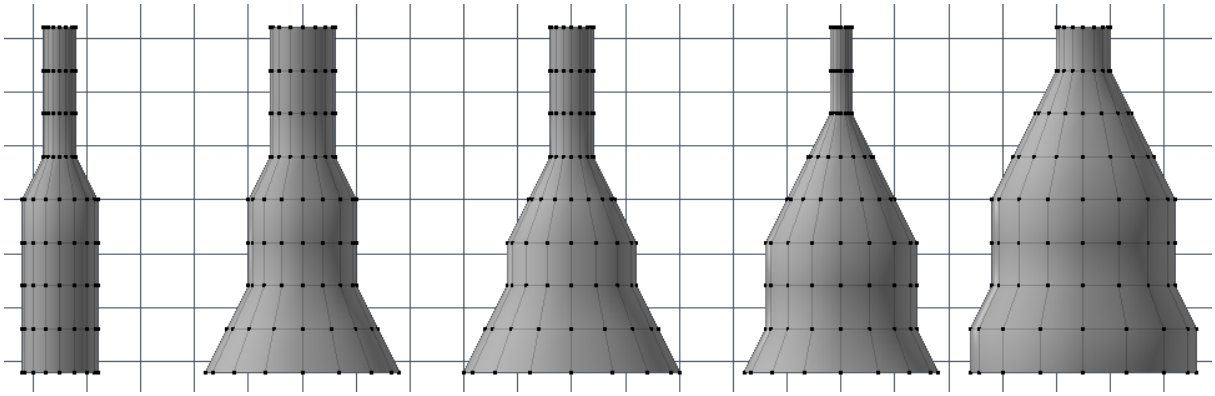
Nije nužno da se svi znakovi interpretiraju direktno kao naredbe, moguće je i definirati module – znakovi koji prelaze produkcijom u niz naredbi, ili drugih znakova modula, čime se efektivno dobivaju funkcije. Pri korištenju modula potrebno je više generacija kako bi niz u potpunosti prešao u znakove konkretnih naredbi, osim u slučaju rekurzivnih pravila. Moguće je proširenje interpretiranje gramatike tako da se definiraju rasponi ispravnih interpretacija u ovisnosti o broju generacije. Na primjer, nezavršni znak A se u prve tri generacije zanemaruje pri interpretacijij, a od četvrte na dalje tretira se kao pomak kornjače unaprijed. Primjer modula danog u nastavku prema formaliziranim naredbama iz privitka generira kvadrat, vraćajući kornjaču u početni položaj, rotiranu za -90 stupnjava po y osi u odnosu na početno stanje:

A -> Y(=90) F + F + F + F

Za postizanje bržeg izvođenja i manjeg troška memorije, pretvorba znakova modula u desnu stranu produkcije se ne mora izvršiti u stvaranju generacija, već se nailaskom na znak modula desni znakovi mogu izravno interpretirati i primjeniti na stanje kornjače i sustava.



Slika 3.2.7. – prikaz linijskih segmenata nastalih iz interpretacije uzastopnih generacija (broj generacije je naznačen ispod segmenta)



Slika 3.2.8. – prikaz radijalnog proširenja segmenata sa slike 3.2.7. uz različite unutrašnje radijuse

Proširenja sustava s fiktivnom kornjačom s naglaskom na okolinu:

- Odnos i interakcija okoline s trenutnim stanjem kornjače mogu utjecati na novo stanje kornjače. Primjer:ako je kornjača barem 2 jedinice od nekog brida, usmjerava se od tog brida.
- L-sustav može imati liste produkcija, gdje je u jednom trenutku samo jedna lista aktivna, te se koriste samo njene produkcije u generacijama
- Kornjača može mijenjati listu aktivnih produkcija s obzirom na interakciju s okolinom
- Produkcija i interpretacija se biraju na temelju odnosa trenutnog stanja kornjače i okoline

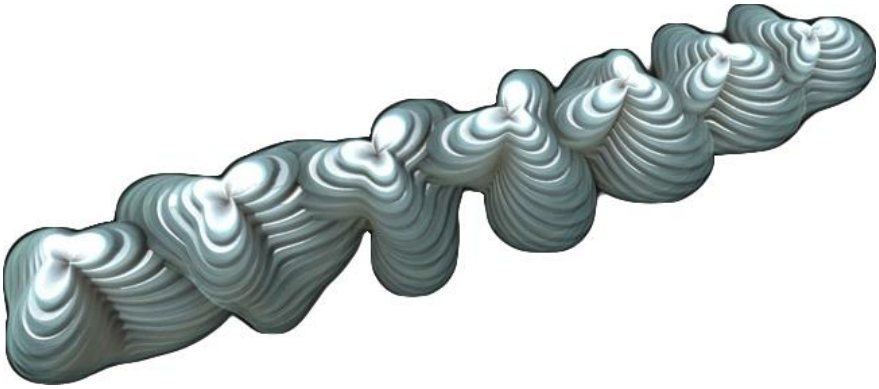
3.2.1.4. Parametarski sustavi

U definiciji L-sustava koji koristi stanje virtualne kornjače uvedena je sintaksa koja omogućava promjenu njenih parametara, kao definiciju, inkremenalne promjene, te nasumičan odabir produkcija.

Navedenom sustavu i dalje nedostaju složenije matematičke funkcije, definiranje odnosa između parametara, nezavisne varijable i nedostatak proizvoljnih uvjeta, koji bi tu gramatiku učinili daleko ekspresivnijom. U privitku, podpoglavlju 8.2, dan je popis mogućih proširenja gramatike L-sustava s fiktivnom kornjačom koji značajno povećava njene mogućnosti.

U ovoj metodi korištenja sustava postoje dvije podvrste sustava s obzirom na očuvanje vrijednosti parametara između pokretanja sustava, i nezavisnih

parametara koji se mogu nezavisno mijenjati u uzastopnim ili paralelnim izvršavanjima operacija sustava.



Slika 3.2.9. – prikaz interpretacije niza znakova temeljenih na parametarskim sustavima

Slika 3.2.9. prikazuje radijalno proširenje putanje kornjače koja se u sedam modula nastalih grananjem kretala putanjom od 16 krugova, a radijus kruga sinusoidalno varira na temelju parametara nezavisnih varijabli.

U međusobno povezanim parametarskim sustavima oni su povezani memorijom – nezavisnim varijablama, te primarni sustavi stvaraju sekundarne, te pohranjuju specifične topografske podatke svakog objekta sekundarnog sustava, kao što su položaj i granice definirane proširenjem kornjačine putanje unutar svih obrađenih sustava. Sekundarni sustav raste i širi se iz lokacija zapisanih u primarnom sustavu, te putanjom ne prelazi granice zadane primarnim sustavom. Praktična primjena ovakvih dualnih sustava je u generiranju gradova i rasta zgrada unutar područja grada, ili definiranje zgrada i razvoja infrastrukture - katova, soba, hodnika te sadržaja interijera unutar zgrade.

3.2.1.5. – Zaključak L-sustava

L-sustavi omogućavaju vrlo moćan pristup dizajniranju, no u izradi definicije gramatike L-sustava postoji kontrast između ekspresivnosti i jednostavnosti. L-sustavi inherentno podržavaju forme poput grananja, rekurzije i modula (funkcija). Rekurzivna priroda omogućava generiranje podprostorno organizirane logike, te porast kompleksnosti i detalja sadržaja na većim dubinama podjele.

L-sustavi se mogu dodatno proširiti iznad parametarskih sustava, no postavlja se pitanje kada će proširivanja dovesti do ekspresivnosti pravog

programskog jezika. L-sustavi omogućavaju arhitektima iznimnu i uvijek proširivu kontrolu, daleko iznad nedeterminističnog izbora akcije iz nekog skupa karakterističnog za neke jednostavne postupke. Operacije koje uzimaju u obzir okolinu dopuštaju korisnicima implementaciju prirodne i konstrukcijske logike u svrhu ostvarivanja njihovih ciljeva.

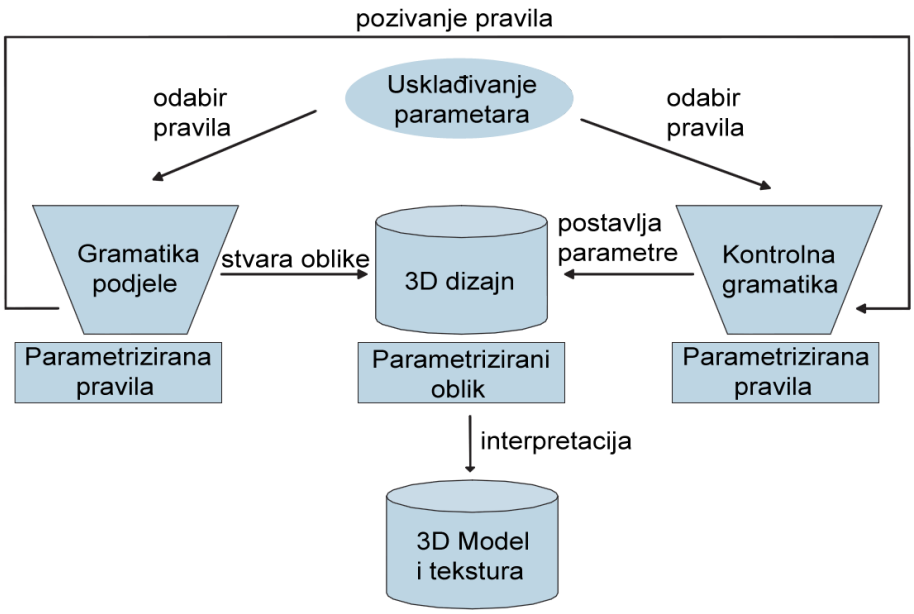
3.2.2 Gramatika podjele i kontrolna gramatika

U radu nazvanom „Instantna arhitektura“ (Wonka, 2003.) predložena je nova metoda automatiziranog modeliranja arhitekture – gramatika podjele prostora.

Ova gramatika je primjenjiva kao iterativno generiranje modela i kao teksturiranje poligona postojećih modela.

U primjeni produkcija potrebno je pažljivo organizirati proces odabira produkcija, jer potpuno stohastički izbor rezultira kaotičnim modelima koji ne bi imali praktičnu primjenu. U radu (Wonka, 2003.) pokazano je da je stohastički izbor kaotičan već za malenu bazu produkcija, te se povećava eksponencijalno s porastom baze. Smislenost i pravilnost se može postići pažljivom prostornom raspodjelom atributa po oblicima. Doprinosi ovog rada u modeliranju građevina su slijedeći:

- Koncept gramatike podjele s ograničenjem na tip dozvoljenih pravila, koja pružaju ekspresivnost te jednostavnost automatizacije kontrole slijeda pravila.
- Sustav usklađivanja parametara koji omogućava specifikaciju ciljeva dizajna na visokoj razini i kontrolira nasumičnost uz konzistentan izlaz
- Kontrolna gramatika, vrsta jednostavne kontekstno neovisne gramatike koja na uređeni način prostorno organizira pojedinačne elemente dizajna, po uzoru na principe arhitekture.



Slika 3.2.10. – pregled sastava i interakcije sustava korištenog u radu (Wonka, 2003.), slika preuzeta iz istog rada

Kontrolna gramatika izračunava i pridjeljuje atribute oblicima generiranim gramatikom podjele. Sustav atributa (parametara) jedinstven je po tome što se atributi pridjeljuju i pravilima i oblicima. Pri odabiru pravila koja se mogu primjeniti na trenutni oblik razmatraju se njihovi parametri, te se odabire pravilo čija je ocjena kompatibilnosti najveća. Jedan korak u procesu primjene pravila je slijedeći:

1. Gramatika podjele odabire prikladna pravila iz baze za primjenu na trenutnom obliku
2. Sustav usklađivanja parametara se poziva kako bi odabrao pravilo koje po parametrima najbolje odgovara trenutnom obliku
3. Atributi se kopiraju iz trenutnog oblika u sve oblike koji su stvoreni odabranim pravilom podjele
4. Kontrolna gramatika prostorno distribuira ideje dizajna (na primjer, postavlja atribute za prizemlje na različite vrijednosti od onih za katove zgrade). Nasljeđivanje parametara u kontrolnoj gramatici identično je nasljeđivanju iz gramatike podjele
5. Gramatika podjele se rekurzivno poziva za oblike generirane trenutno odabranim pravilom

Koraci se ponavljaju do određenog broja iteracija ili dok se generacija ne sastoji u potpunosti od završnih znakova.

3.2.2.1. Gramatika podjele

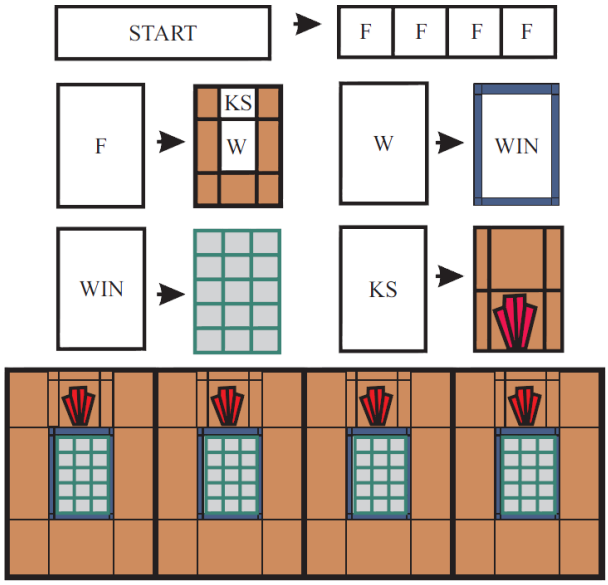
U ovom podpoglavlju daje se opis trodimenzionalne gramatike podjele koja je korištena u (Wonka, 2003.) za prostornu podjelu dizanja eksterijera zgrada.

Osnovni primitiv u ovoj gramatici je koncept oblika (engl. shape), koji je osmišljen u gramatici oblika (Stiny, 1980). Definicija oblika glasi:

Oblik je konačno, ograničeno uređenje ravnih linija u trodimenzionalnom euklidskom prostoru.

Gramatike koje se koriste za prostorno dizajniranje zahtjevaju općenitiju definiciju od kontekstno neovisnih gramatika ili L-sustava. U nastavku je dano par definicija:

Linija l = p1,p2 je definirana svojim krajnjim točkama. Oblici mogu sadržati skup označenih točaka. Označena točka sastoji se od koordinata i oznake A: <p,A>. Označen oblik se zapisuje kao <s,P> gdje je P oznaka oblika. Oblik se parametrizira položajem i veličinom volumena prostora kojeg umeđuje, te se u njemu nalaze sve linije tog oblika.



Slika 3.2.11. – prikaz primjera nezavršnih simbola (bijeli pravokutnici s oznakama simbola) te pravila gramatike podjele (gore) i deterministčkog rezultata primjene simbola i pravila (dolje). Slika preuzeta iz (Wonka, 2003.)

Gramatika $G = (N, T, R, I)$ sastoji se od nezavršnog riječnika (skupa oznaka) $N \subseteq U$, završnog skupa oznaka $T \subseteq U$, početnih oznaka $I \subseteq N$ i produkcija $R \subseteq UxU$, gdje je U skup svih znakova gramatike.

Gramatika skupova sastoji se od gramatike rječnika B, gdje je B podskup seta znakova U, operacija + (unija skupa) i – (razlika skupova), relacije podskupa ≤, te skupa operacija S, gdje je f(S) skup $\{f(s)|s \in S\}$ za podskup $S \subseteq B$.

Gramatika podjele je posebna vrsta gramatike skupova koja transformira oblike. Prikladna je za oblikovanje zgrada zbog ekspresivnosti dizajna te mogućnosti automatizacije smislenog odabira produkcija.

Rječnik gramatike podjele je skup $B = \{f(b)|b\ je\ osnovni\ oblik, f \in F|F\ je\ skup\ afinih\ transformacija\}$

Po definiciji, podjela je dekompozicija osnovnog oblika u oblike iz rječnika B.

Skup oblika $B \subseteq B'$ nastao gramatikom podjele naziva se dizajn zgrade samo ako se sastoji u potpunosti od završnih oblika. Slika 3.2.11. prikazuje rječnik pravila (gore) i konačni dizajn zgrade (dolje).

U gramatici podjele dozvoljena su slijedeća pravila:

Pravilo podjele: $a \rightarrow b$, gdje je *a* povezani podskup *B*, i *b* sadrži iste elemente kao *a* osim jednog, na koji se primjenjuje podjela

Pravilo pretvorbe: $a \rightarrow b$ koje pretvara jedan osnovni oblik u drugi. *a* i *b* su identični povezani podskupovi od B koji se sastoji od jednog osnovnog oblika, *b* sadrži iste elemente kao *a*, osim osnovnog oblika koji je bio zamijenjen drugim. Jedino ograničenje je da oblik *b* treba biti sadržan u volumenu oblika *a*.

3.2.2.2. Kontrolna gramatika za propagaciju atributa

Kontrolna gramatika koristi se za propagaciju atributa koji opisuju detalje dizajna kroz hijerarhiju oblika. To je zapravo kontekstno neovisna gramatika koja se u pravilima sustava može kombinirati s gramatikom podjele, te sadrži atribute u svojim završnim znakovima.

Atributi se koriste u dvije svrhe:

- Kodiranje podataka o materijalu na različitim razinama detalja – od boje na niskoj razini do stila zgrade koji dolazi do izražaja u kasnijim koracima derivacije oblika.
- Atributi upravljaju procesom odabira pravila, što znači da imaju utjecaj na prostorni raspored oblika, kao i na kontrolnu gramatiku.

Atributi se postavljaju na tri načina:

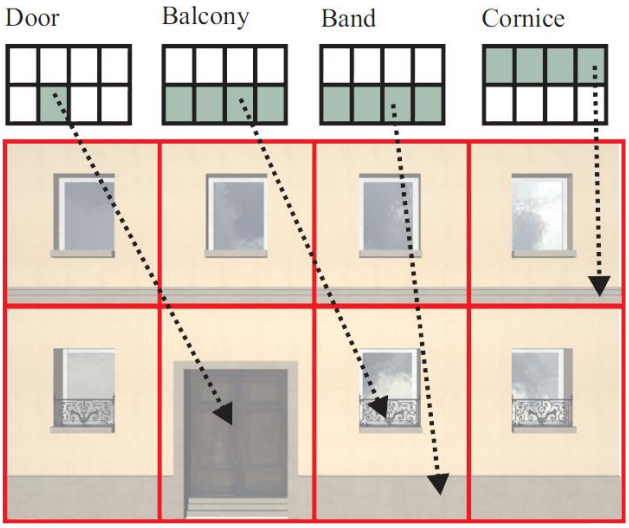
- Ručno postavljanje vrijednosti atributa početnih simbola
- Prepisivanje atributa roditelja na sve generirane oblike djecu
- Kontrolna gramatika

Kontrolna gramatika je jednostavna atributna kontekstno neovisna gramatika koja prostorno organizira pridjeljivanje vrijednosti atributa.

Završni znakovi ove gramatike su u obliku skupa (c, a, v), gdje je c prostorni lokator unutar podjele oblika, *a* je ime atributa, a *v* je vrijednost atributa. Lokator *c* u određenoj mjeri ovisi o vrsti podjele u trenutnom pravilu, ali u općem slučaju može se definirati kao položaj u redu, stupcu ili sloju (u 3D slučaju).

Atributi određuju kada će se pozvati kontrolna gramatika; u primjeni pravila podjele, desna strana može sadržavati završni znak kontrolne gramatike koji se interpretira kao naredba postavljanja atributa novih oblika nastalih tom podjelom.

Lokator se koristi kao oznaka raspona oblika u toj podjeli kojima se atribut *a* postavlja na vrijednost *c*. Slike 3.2.12. i 3.2.13. prikazuju primjer predložaka podjele (3.2.12. gore) stvorenu geometriju ili teksturu (3.2.12. dolje), te pravila generiranja (3.2.13.) među kojima zadnje navedeno, s nezavršnim znakom DOOR na lijevoj strani sadrži znakove kontrolne gramatike na desnoj strani.



Slika 3.2.12. – prikaz predložaka podjele fasade i generirane teksture, uz oznaku izvornog predloška djelova. Slika preuzeta iz (Wonka, 2003.)

```
FACADE_CONTROL ::- DOOR_PATTERN, RANDOM_PATTERN,
                   RANDOM_PATTERN, RANDOM_PATTERN
RANDOM_PATTERN ::- CORNICE | BAND | BALCONY
                   | QUOIN | PILASTER
DOOR_PATTERN ::- DOOR | GARAGE
DOOR ::- <[0,1], door, 1> | <[0,2], door, 1>
         | <[0,MAX], door, 1>
```

Slika 3.2.13. – prikaz pravila za generiranje podjeljenih oblika koji odgovaraju slici 3.2.12.

3.2.2.3. Usklađivanje atributa za odabir pravila

Odabir pravila se vrši tako da stvoreni oblici budu koherentni, vjerojatni u stvarnom svijetu, da postoji dovoljna varijacija u konačnim modelima zgrada te da su zadovoljeni kriteriji korisnika.

Pri odabiru pravila ocjenjuje se kompatibilnost atributa svakog pravila s atributima oblika na kojeg se pravilo primjenjuje

Atribut pravila definira se kao skup vrijednosti: donja granica, gornja granica intervala vrijednosti, zastavica prisutnosti, statistička distribucija.

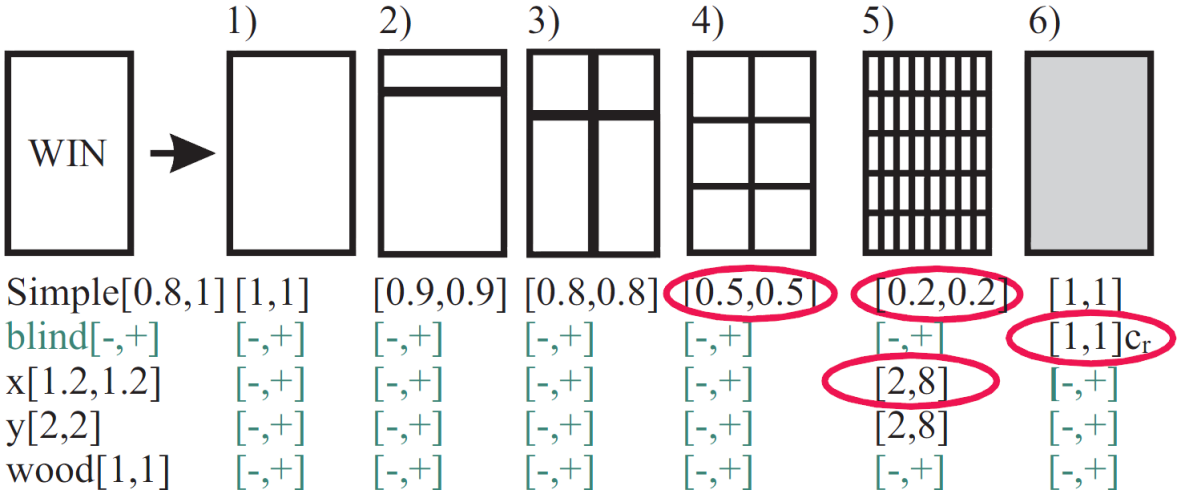
Usklađivanje atributa obavlja se u 2 faze: prva faza je provjera preklapanja intervala atributa i dodjela prioriteta pravilima, a druga je odabir konkretnog pravila iz pravila najvišeg prioriteta na temelju statističkih distribucija pridjeljenih tom pravilu. Korištenjem ovog pristupa omogućavaju se slijedeći načini selekcije pravila:

Isključivi interval ili vrijednost – na primjer, vrijednost pravila 0.5 je izvan intervala atributa [0.8,1], što je slučaj pravila 4 u slici 3.2.14., ili slučaj gdje vrijednost intervala u pravilu ne uključuje vrijednost atributa.

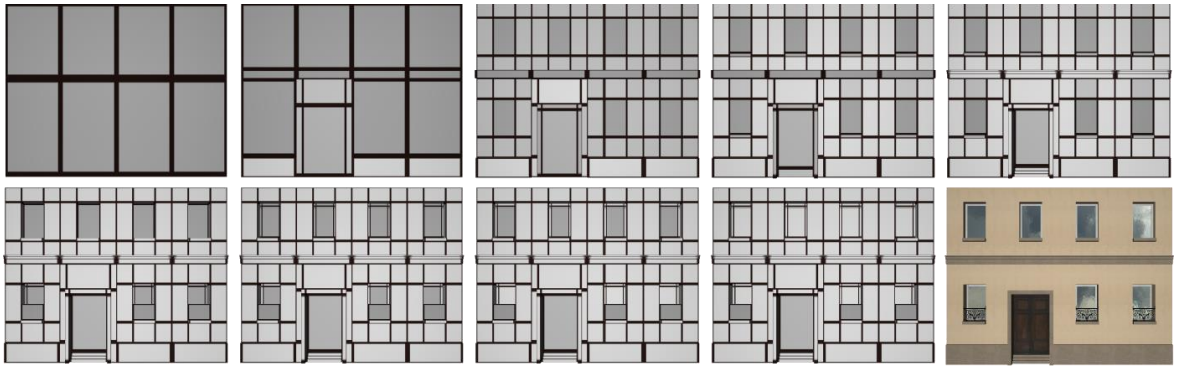
Inicijalna vrijednost atributa – na primjer: atribut čiji raspon je definiran tako široko da ga sva pravila zadovoljavaju, te svako pravilo prolazi usklađivanje s tim atributom

Isključiva inicijalna vrijednost – neka pravila se trebaju primjeniti samo kada se to isključivo zahtjeva, poput udubljenja u zidu bez prozora. Za to je potrebno da interval ili vrijednost atributa budu potpuno sadržani u rasponu vrijednosti atributa pravila.

Slika 3.2.14. prikazuje oblik s nezavršnim nizom WIN, te niz mogućih pravila, od kojih su 4., 5. I 6. isključena zbog nezadovoljavanja uvjeta atributa u lijevom stupcu, označena crvenom elipsom. Raspon $[-,+]$ predstavlja raspon $[-\infty,\infty]$, ili riječima „bilo koja vrijednost“.



Slika 3.2.14. – prikaz usklađivanja intervala pravila (gornji red) i intervala atributa (lijevi stupac). Slika preuzeta iz (Wonka, 2003.)



Slika 3.2.15. – prikaz postupne podjele pročelja zgrade po koracima primjene produkcija. Slika preuzeta iz (Wonka, 2003.)

3.2.3. „CGA shape“ gramatika

Ovo poglavlje daje uvid u detalje gramatike nazvane „CGA shape“, gramatike generiranja virtualne arhitekture koju su združenim naporima osmislili Müller, Wonka i ostali u radu „Proceduralno generiranje građevina“ (Müller 2006.).

Gramatika se temelji na podjeli „masenog modela“, što je autorski naziv za virtualni volumen, koji se nastavlja dijeliti i odvajati u podvolumene i ravnine kako bi se dobila podjela fasade, te se na njenim djelovima u zadnjoj fazi instanciraju gotovi modeli poput vrata, prozora, stubišta, ograde i slično. Osim naredbi za podjelu, ova gramatika sadrži i kontrolnu gramatiku za modificiranje geometrije volumena, u smislu označavanja, promjene i uklanjanja strana, bridova i vrhova, što se tretira kao geometrija sadržana unutar virtualnog volumena.

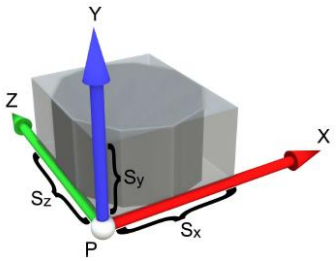
Ova gramatika sastoji se od niza kontekstno ovisnih pravila, čiji se redoslijed primjene temelji na njihovim prioritetima uz mogućnosti dodatnih uvjeta koji uključuju trenutni broj generacije izraza. Gramatika je zapisana u notaciji sličnoj gramatici L-sustava, te je preuzela mogućnost dodavanja, skaliranja, rotacije i translacije oblika iz nje, ali za razliku od paralelnih gramatika (poput L-sustava) koje su prikladne za simuliranje rasta oblika kroz vrijeme, ova gramatika je sekvencijalna, što kao rezultat omogućava karakterizaciju strukture; prostornu distribuciju osobina i komponenata.

CGA oblik je temelj ove gramatike. Oblik se sastoji od:

- Simbola – niza znakova

- Geometrije – definiranog volumena, strana, bridova, vrhova i drugih geometrijskih atributa
- Ostalih proizvoljnih numeričkih atributa

Simbol oblika identificira oblik, te može biti ili završni ili nezavršni simbol, te se oblici s obzirom na simbol nazivaju završnim ili nezavršnim oblicima. Najvažniji geometrijski atributi su oni koji definiraju volumen oblika: referentni položaj volumena P, tri ortogonalna vektora X, Y i Z koji definiraju koordinatni sustav oblika, te veličina oblika po osima dana trodimenzionalnim vektorom S. Položaj P je definiran kao lijevi, donji i stražnji vrh volumena. Primjer definiranog volumena koji sadrži osmostranu prizmu kao unutrašnju geometriju (koja je definirana u nastavku) dan je na slici 3.2.16. Iz definicije volumena jasno je da je on uvijek u obliku kvadra. U slučaju kada se primjene naredbe kontrolne gramatike za modificiranje geometrije, ona je pohranjena odvojeno od njenog obuhvaćajućeg volumena.



Slika 3.2.16. – prikaz volumena definiranog „CGA shape“ gramatikom uz označene osnovne geometrijske atribute (Müller 2006.)

Proces provođenja produkcija:

1. Definira se početni oblik – volumen i njegov nezavršni simbol
2. Odabere se aktivan oblik A s nezavršnim simbolom B
3. Izabere se produkcijsko pravilo koje na lijevoj strani sadrži nezavršni simbol B,
4. Oblik A se označi neaktivnim, te se u hijerarhiju oblika kao njegova djeca pohrane oblici stvoreni desnom stranom odabranog produkcijskog pravila
5. Koraci 2-4 se ponavljaju sve dok postoje aktivni oblici s nezavršnim znakovima ili dok se ne ispune drugi uvjeti zaustavljanja poput broja generacija

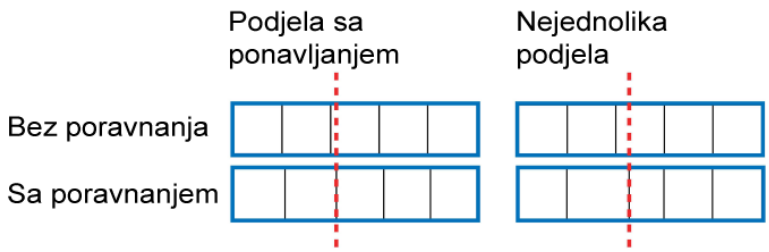
S obzirom da definirani proces ne ograničava izvođenje na primjenu produkcija po širini ili dubini, a niti jedan od tih koncepata ne pruža dovoljnu kontrolu oko izvođenja, uvodi se koncept prioriteta, gdje je svakom pravilu pridružen prioritet koji označava redoslijed izvođenja produkcija. Kod procesa je bitno za primjetiti da se oblici ne brišu, već se zadržavaju u hijerarhiji kako bi pri interpretaciji omogućili pretragu prostora i povezanosti.

Osnovne operacije ove gramatike (osim transformacija) su nejednolika podjela oblika, podjela s ponavljanjem, te podjela oblika po komponentama.

Jedan od posebnih principa koje ova gramatika koristi je kombiniranje apsolutnih i relativnih vrijednosti varijabli kao parametra u nekoj operaciji. Ako je u operaciji potreban samo jedan parametar, relativne vrijednosti se izračunaju i dodaju apsolutnima, no ako operacija zahtijeva više parametara među kojima su apsolutni i relativni (primjer 8.2.2. u privitku), prednost se daje asolutnim, te vrijednost relativnih ovisi o preostaloj vrijednosti nakom primjene apsolutnih parametara.

Drugi princip je provjera zaklanjanja oblika drugim oblicima (engl. occlusion test). Objekt može biti potpuno i djelomično zaklonjen, ili potpuno nezaklonjen drugim oblikom ili hijerarhijom oblika. Naredbe testiranja omogućavaju i specifikaciju vrste oblika na koje se operacija odnosi; to mogu biti oblici posebne oznake, aktivni ili neaktivni, te „roditelji“ ili „djeca“ drugih oblika. Praktična primjena rezultata provjere zaklanjanja je upotreba u uvjetima za primjenu pravila poput instanciranja modela prozora na strani fasade volumena koja je okrenuta prema ulici i nije zaklonjena drugim volumenima.

Treći princip koristi koncept poravnanja parametara na najbliži sadržaj (engl. snapping). Naredbama gramatike mogu se definirati globalne osi poravnanja, koje su definirane imenom, ravninom te položajem na pravcu okomitog na ravninu. Volumeni koji se dijele slijedećim naredbama uzimaju u obzir sve presjecajuće osi poravnanja, što rezultira oblicima podijeljenim na položajima osi, olakšavajući sadržajno povezivanje tih oblika. Rezultat primjene osi poravnanja vidljiv je na slici 3.2.17.



Slika 3.2.17. – prikaz primjene naredbe poravnanja pri izvršavanju podjele s ponavljanjem (lijevo) u kojoj se nakon podjele izvornog volumena oba dijela dijele dalje na jednolike segmente, te pri nejednolikoj podjeli (u kojoj se mijenja samo granica između segmenata najbliža osi poravnanja)

U privitku, podpoglavlju 8.3. dan je popis i objašnjenje svih operacija „CGA shape“ gramatike.

Ova gramatika je karakterizirana visokom ekspresivnošću sa svoju domenu modeliranja vanjskog izgleda prostora, uz instanciranje gotovih modela, te čitljivosti definicije pravila. Krivulja učenja sintakse i semantike „CGA shape“ gramatike identična je onoj za učenje nekog skriptnog jezika. Kombiniranjem zapisa pravila iz L-sustava, ideje stanja preuzete iz L-sustava temeljenog na stanju kornjače koja se ovdje ostvaruje kao stanje trenutnog oblika, gramatike podjele i kontrolne gramatike za modeliranje u jednoj, dvije ili tri dimenzije nastala je moćna gramatika za kontroliranu izradu niza prototipa modela zgrada koje se mogu upotrijebiti u simulacijama te industriji filmova i računalnih igara.

3.2.4. Evoluiranje vanjskih blokova građevina

U izradi igre Subversion (Martin, 2010.) istražen je postupak razvoja virtualnih 3D struktura građevina koji se provodi u suradnji s korisnicima aplikacije. Zgrade se opisuju strukturiranim jezikom u kojem se naredbe stvaranja i promjene djelova zgrada zapisuju u cjeline, koje se mogu međusobno referencirati i sadržavati jedna drugu. Korišteni opisi mogu se nasumično generirati te međusobno križati i mutirati, a selekcija se provodi korisničkim odabirom i ocjenjivanjem.

Glavna prednost ovog postupka je ta da korisnik ne treba imati jasnu viziju ili pisati pravila procesa generiranja sadržaja, već mu se nudi izbor ocjenjivanja proceduralno generiranih struktura. Korisnik u bilo kojem trenutku može pregledati i promijeniti parametre kreiranja ili samu strukturu pojedine građevine. Ostale

prednosti ovog postupka proceduralnog generiranja su jednostavnost korištenja, brzina izvođenja te korisničko usmjeravanje razvoja sadržaja. Nedostatak jezika je premalena ekspresivnost za kompleksne, realistične zgrade, što čini ovaj postupak pogodnim samo za stvaranje vanjskog oblika i fasade, uz mogućnost daljnje obrade strukture drugim metodama.

Korišten jezik je razvijen s namjerom da bude ljudski čitljiv i razumljiv, te osoba koja razumije sintaksu može iz strukture zgrade znati kako će izgledati njen model.

Jezik opisuje svaku građevinu kao skupinu trodimenzionalnih objekata opisanih dvodimenzionalnim oblicima koji se proširuju u treću dimenziju (najčešće je to y os, koja pokazuje prema gore u virtualnom prostoru). Dvodimenzionalni oblici korišteni za temelj građevine su pravokutnici, elipse, proizvoljne krivulje ili skupovi planarnih točaka. Nad njima se mogu vršiti operacije konstruktivne čvrste geometrije (engl. constructive solid geometry), koje djeluju poput unije, razlike i presjeka 2D površine ili 3D volumena. Nad temeljnim oblicima mogu se vršiti transformacije rotacije, skaliranja, translacije, no ideja je da je konačna građevina geometrijski zapisana kao unija planarnih zatvorenih skupina točaka, kako bi se mogle konstruirati zatvorene plohe.

Početni set zapisa struktura zgrada sastoji se od prethodno generiranih ili novih nasumičnih struktura. Strukture se parsiraju i zapišu u stablastu strukturu koja preslikava tekstualni oblik zapisa, te je prikladna za evolucijske operatore. Opis njihovog provođenja dan je u nastavku.

Nad populacijom inicijalno nasumičnih opisa zgrada izvršavaju se evolucijski operatori. Proporcionalna selekcija se obavlja na temelju dobrota koje korisnik pridjeljuje jedinkama trenutne generacije.

Križanje između dvije strukture se svodi na kombiniranju i zamjeni kompatibilnih blokova s obzirom na njegove naredbe; na primjer, naredbe *Circle* i *Square* su kompatibilne, jer obje označavaju temeljni 2D oblik. Jedan od korisničkih parametara korišten u križanju je broj zamjena blokova između roditelja.

Mutacija se dijeli na strukturnu i parametarsku. Strukturna mutacija se temelji na zamjeni blokova s novim, nasumično generiranim blokovima, uz ograničenje da

je broj blokova konstantan. Snaga mutacije je izražena u broju zamjena blokova. Parametarska mutacija je provodi promjenom iznosa parametara postojećih blokova.

U privitku rada, poglavlje 8.4., priložena je struktura jezika te jednostavan primjer evolucije zgrada.

3.2.5. Metode generiranja osnovnih elemenata

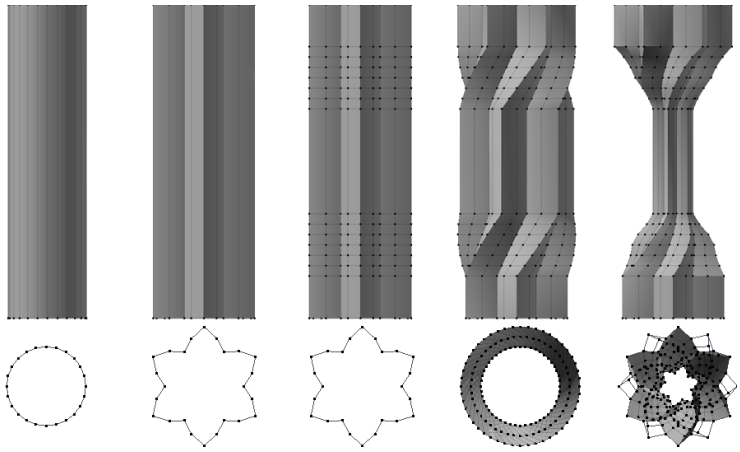
Programi za 3D modeliranje imaju veliki broj zajedničkih alata i operacija koje znatno olakšavaju ručno modeliranje. Proceduralno modeliranje ima slobodu primjene svih operacija uz razne kontrolirane vrijednosti parametara.

Većina osnovnih elemenata arhitekture može se rastaviti na manje djelove koji primjenom akcija ponavljanja ili neke vrste osne promjene oblika daju osnovne elemente. Popis osnovnih operacija dan je u nastavku:

- Osno proširenje strane modela – ova operacija za danu skupinu poligona generira proširenje danih poligona tako da ih poveže novim poligonima s njihovim vanjskim rubom. Moguća proširenja ove operacije su skaliranje pomaknutih poligona, segmentacija povezujućih poligona te njihova rotacija. Osno proširenje može se uzastopno primjeniti za neki poligon i put (niz linijskih segmenata), te je prikladno za generiranje lukova i mostova.
- Radijalno proširenje – za danu skupinu poligona ili povezanih rubova (niz linijskih segmenata) ova operacija stvara zadani broj međusobno povezanih identičnih skupina koje su jednako udaljene od osi proširenja. Ova metoda prikladna je za generiranje stupova.
- Osna rotacija – za danu skupinu poligona ova operacija ih rotira oko osi rotacije, pri čemu se može definirati funkcija snage rotacije, tako da se točke poligona rotiraju za nejednoliki kut. Ova operacija prikladna je za naknadnu obradu stupova.
- Instanciranje postojećih objekata po stazi (nizu linijskih segmenata) ili implicitnoj krivulji – ova operacija stvara geometriju definiranog objekta na predefiniranim položajima, postižući pravilan skup. Ova metoda kombinira se s radijalnim proširenjem te koristi pri generiranju ograda, niza stupova ili stepenica. Može se proširiti na instanciranje po površini ili volumenu.

Slika 3.2.18. prikazuje korištenje ove metode za generiranje stupa, pri čemu je više nizova interpretirano u niz brojeva, te je svaki niz iskorišten za drukčije parametre. Slijeva na desno prikazani su:

- Početni model cilindra zadane visine te 24 radijalna segmenta
- Radijalno proširenje: 6 ponavljanja radijusa (1, 1.25, 1.5, 1.25)
- Osna podjela na segmente duljina (4, 6x1, 10, 6x1, 4)
- Osna rotacija po y osi za slijedeće iznose u stupnjevima, lokalne u odnosu na rotaciju nižeg sloja: (0, 0, 6x7.5, 0, 6x7.5, 0)
- Skaliranje radijusa uzduž y osi faktorima: (1, 1, 0.85, 0.75, 0.7, 0.6, 0.45, 0.4, 0.4, 0.45, 0.6, 0.7, 0.75, 0.85, 1, 1)



Slika 3.2.18. – prikaz modeliranja stupa metodom interpretacije nizova

S obzirom da se svaki proces ručnog modeliranja može zapisati kao diskretan broj točno definiranih operacija, ukoliko se izgradi gramatika koja će pozivati metode ugrađene u program za modeliranje, u teoriji moguće je postići niz završnih znakova te gramatike koji bi interpretacijom izgradili bilo koji model.

3.2.6. Metode organizacije objekata

U poglavlju 3.2.2. i 3.2.3. navedene su gramatike koje dijele prostor u manje djelove koje su hijerarhijski povezane u implementaciji. One se mogu koristiti i za prostornu organizaciju objekata interijera ili objekata eksterijera sa svim sadržanim objektima interijera, te se taj pristup neće spominjati jer je identičan primjeni gramatika u prethodnim poglavljima.

Drugi, poznati primjeri pohrane objekata u hijerarhiju su kvadratno (2D) i oktalno stablo (3D), koji se koriste za pretragu objekata virtualne scene, izračun kolizija, zaklanjanja pogleda i druge tehnike. U praktičnoj primjeni, objekti se smještaju u sve ćelije strukture koje ih sadrže u nekom vremenskom trenutku, te ćelije mogu biti fiksne veličine, ili se dinamički prilagođavaju gustoći objekata koje sadrže.

U ovom poglavlju biti će riječ o podjeli prostora i smještanju objekata u uniformnu mrežu. Navedeni primjeri su više fokusirani na dvodimenzionalnu razinu, no svi principi primjenjivi su u tri dimenzije.

3.2.6.1. Uniformna mreža

U nastavku su navedene i ukratko opisane tehnike generiranja povezanog interijera uz korištenje uniformne mreže:

1. Centralno raspoređivanje soba – inicijalno prazna mreža se postupno popunjava sobama: kad se soba doda interijeru, udalji se na zadanu udaljenost od svih postojećih soba te poveže s najbližim sobama ili hodnicima, ili se generira druga soba, a trenutna briše.
2. Binarna podjela prostora – definirana prazna mreža se rekurzivno binarno dijeli do definirane granice minimalne površine, te se sobe stvorene u istoj podjeli međusobno povežu i povezuju dalje postupno s drugim čvorovima istih razina i roditelja
3. Stvaranje zidova – inicijalno prazna mreža popunjava se zidovima raznih duljina u debljina na način da ne postoji zid koji potpuno odjeljuje prostor na dva međusobno nedostupna dijela – nasumično se generiraju otvori u zidovima
4. Kombiniranje postojećih cjelina – ova metoda postavlja postojeće grupe soba i hodnika u potpuno povezanu cijelinu. Medota se svodi na rotaciju često kvadratnih segmenata i uspoređivanje povezanosti na „spojevima“ segmenata
5. Uniformne ćelije – uniformna mreža podjeli se na veće ćelije tako da se svaka sastoji od istog broja ćelija mreže, a zatim se unutar svake veće ćelije smjesti soba ili niz hodnika, tako da je zajamčeno da se svaka soba nalazi unutar granica uniformne ćelije. Ovo svojstvo omogućava jednostavno građenje soba

u skladu s definiranim uzorkom, praćenje povezanosti i dinamičku promjenu djelova interijera.

6. Višeslojno raspoređivanje – ovo je zasebni princip koji se može kombinirati s bilo kojim drugim. Temelji se na generiranju sadržaja na jednoj razini (definiranoj veličini ćelije), nakon čega se svaka ćelija podijeli u manje podćelije, koje se onda koriste za daljnje generiranje soba, hodnika i smještanje sadržaja
7. Tehnike temeljene na labirintima – ovaj princip može se koristiti za popunjavanje neispunjenog (praznog) prostora u mreži, ili za ograničavanje prevelikih prostorija.

Niti jedna od navedenih tehnika nije strogo ograničena na uniformnu mrežu, ili uopće prostorno ograničena, no najčešće se primjenjuju upravo u takvom okruženju, zbog jednostavnosti implementacije u smislu praćenja povezanosti pojedinih djelova te diskretnosti veličina i koraka pomaka pri generiranju.

3.2.6.2. Agenti

Metoda koja je na razini jednostavnosti implementacije stohastičkog raspoređivanja, no pruža veću kontrolu te puno bolje definiranje pravila naziva se metoda organizacija korištenjem agenata. Agent je entitet koji prolazi zadanim prostorom te s obzirom na stanje interijera i drugih agenata može promijeniti svoje stanje, stanje interijera, ili poslati poruke drugim agentima kako bi oni poduzeli neku akciju.

U praksi, agenti se koriste za „bušenje“ prostora, stvaranje prolaznog prostora poput hodnika i soba, te se u pravilu zaustavljaju ili vraćaju na zadnje vlastito raskrižje ukoliko naiđu na područje kojim je prošao neku drugi agent, ili dođu u njegovu neposrednu blizinu.

Moguće je implementirati više verzija agenata, koji se aktiviraju u interijeru u različitim fazama generiranja:

- stvaranje hodnika
- stvaranje soba koje graniče s hodnicima
- pretvaranje sjecišta hodnika u sobe
- postavljanje vrata i prozora interijeru

- omogućavanje povezivanja trenutne razine s drugim postavljajuće povezne elemente poput dizala i stepenica
- ukrašavanje ulaza u prostorije i širokih hodnika stupovima i slično

Agenti se mogu primjeniti i na metode opisane u prethodnom podpoglavlju, pri čemu bi ostvarili paralelno generiranje sadržaja uz komunikaciju i mehanizme odlučivanja zajedničke akcije (poput glasanja ili pareto optimalne strategije), što bi povećalo kompleksnost implementacije, ali istovremeno pružilo i ekspresivnost u pravilima koje agenti moraju zadovoljiti kroz vlastite akcije.

3.2.6.3. Zadovoljavanje ograničenja

Sadržaj interijera može se popunjavati i metodom zadovoljavanja ograničenja. Ova metoda nije najekspresivnija, no pogodna je za kombiniranje s drugim metodama. Ovisno o pristupu i redoslijedu prolaza postojeće hijerarhije interijera ili vanjskih granica, ova metoda daje iznimno raznolike rezultate.

Jedno ograničenje predstavljeno je pravilom organizacije objekata. Pravilo može biti čvrsto i meko. Čvrsta pravila moraju biti zadovoljena za uspješno generiranje, jer bi u protivnom dio modela bio logički neispravan (na primjer: jedan dio zgrade bio bi nedostupan). Meka ograničenja ne moraju se nužno zadovoljiti, no ukoliko se zadovolje, takva organizacija postići će bolju ocjenu te će elementi biti skladnije složeni.

Korištenje mekih i tvrdih ograničenja praktično je u kombinaciji s genetskim algoritmima u organizaciji interijera, jer se tako postiže kontrolirana raznolikost rasporeda uz zadovoljavanje svih nužnih i što više vrijednijih ograničenja.

3.3. Proceduralno teksturiranje arhitekturalnih elemenata

Proceduralno teksturiranje je opširno istraženo na području tekstura koje imitiraju prirodne površine poput vode, drveta, mramora, oblaka i slično, no na području teksturiranja građevina najbolje je koristiti teksture nastale od pravih fotografija. Za materijale zidova, dasaka prozora, stupova, ograda i stepenica mogu se koristiti i proceduralne teksture drveta.

Ovo poglavlje objašnjava kako se generiranim modelima pridružuju UV koordinate, te kako se koriste teksture nastale iz fotografija.

3.3.1. Određivanje UV koordinata iz položaja točaka modela

Položaj točaka modela može se iskoristiti za uzorkovanje slikovnih elemenata postojeće ili proceduralne teksture računate u stvarnom vremenu. Točke se mogu koristiti kao UV koordinate u svim sustavima kroz koje prolaze u programima za sjenčanje: prostor modela, svijeta (scene), prostor pogleda (kamere) ili prostor ekrana (normaliziran ili skaliran na veličinu izlaznog spremnika).

Od svih navedenih metoda, samo jedna ima praktičnu primjenu: korištenje točaka lokalnog prostora modela za određivanje UV koordinata. Unatoč tome što je ovaj princip moguće koristiti uz dobre performanse za vrijeme iscrtavanja virtualne okoline, češće se provodi određivanje statičnih UV koordinata za vrijeme generiranja modela.

Prednosti korištenja lokalnog prostora modela u odnosu na druge su slijedeće: Lokalne koordinate su stalne, te ne ovise o transformacijama modela, položaju kamere ili rezoluciji ekrana, tako da će UV koordinate biti konzistentno s oblikom modela. Postoji više načina korištenja lokalnih koordinata točaka:

1. Direktno preslikavanje svih točaka modela u UV koordinate – odabirom dvaju osi, ili funkcije preslikavanja trodimenzionalne točke u dvodimenzionalnu dobivaju se UV koordinate – ovaj način ima loše rezultate ako se odaberu dvije osi, jer u tom slučaju sve točke poligona neke ravnine imaju samo jednu UV koordinatu različitu, što će uz uzorkovanje tekstura rezultirati linijama istog slikovnog elementa. Slika 3.3.1 prikazuje model krova zgrade (lijevo), te preslikavanje lokalnih položaja točaka uniformnim skaliranje u prvi kvadrant UV prostora (sredina) i preslikavanje uz skaliranje za maksimalno iskorištavanje prostora (desno).

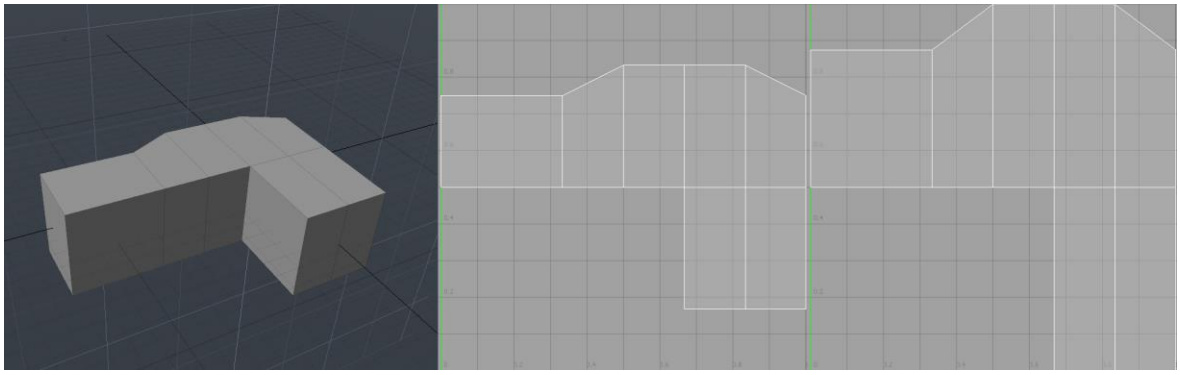
2. Preslikavanje točaka jedne strane u prvi kvadrant kartezijevog koordinatnog sustava s obzirom na položaje točaka u lokalnom prostoru te strane. Lokalni položaj točaka u prostoru strane (poligona) izračunava se iz centra, normale i tangente strane, dobivene aritmetičkim sredinama i normaliziranjem istovjetnih podataka svih točaka te strane.

3. Korištenje kubne, sferne ili cilindrične projekcije: uz definiranje položaja centra projekcije, glavne osi projekcije i preostalih parametara (dimenzije kvadra,

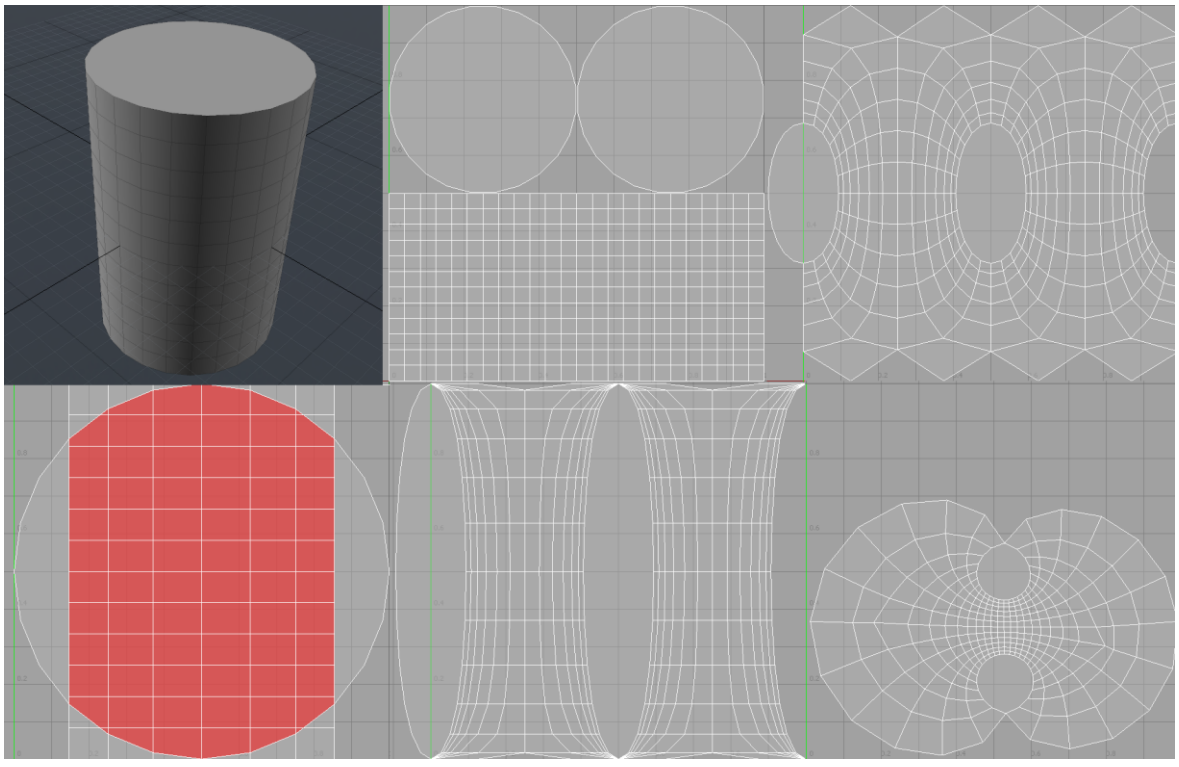
prostorne elipse ili cilindra), svaka točka modela se projicira na definiranu geometriju projekcije te se iz njenog preslikavanja odrede UV koordinate točke modela.

4. Korištenje sustava opruga za smanjivanje napetosti bridova, uz određene grupe rubova koje će biti granice reza između poligona modela.

Slika 3.3.2. prikazuje model valjka (gore lijevo) s različitim projekcijama njegovih poligona na prostor UV koordinata: ručno UV preslikavanje (gore sredina), sferna projekcija (gore desno), kubna projekcija (dolje lijevo), cilindrična projekcija (dolje sredina), raspuštanje napetosti bridova sustavom opruga (dolje desno).



Slika 3.3.1. – prikaz modela zgrade te UV koordinata poligona krova

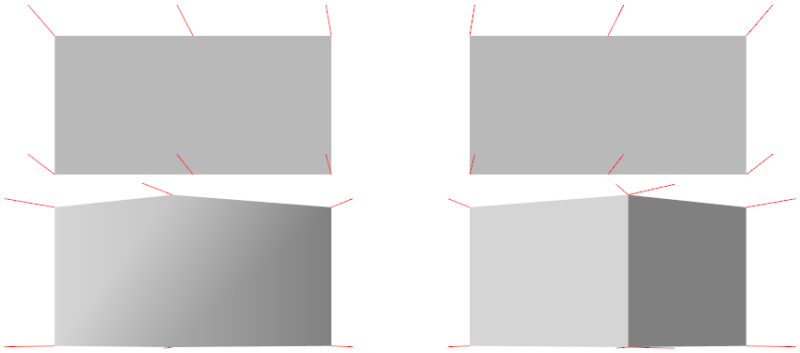


Slika 3.3.2. – prikaz modela valjka i projekcija njegovih poligona u UV prostor

3.3.2. Teksturiranje zasebnih cjelina ili poligona modela

U svrhu objašnjavanja dijeljenja UV koordinata između točaka, u nastavku je dan primjer dijeljenja normala.

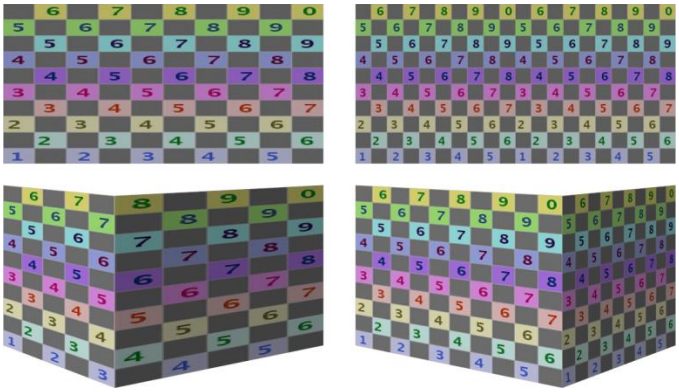
Pri generiranju građevina nužno je da normale svih strana budu identične normali strane kojoj pripadaju. Iz tog razloga potrebno je imati različite točke modela za isti rub susjednih strana, jer jednoj točki može biti pridjeljena samo jedna normala. Slika 3.3.3. prikazuje segmente koji se sastoje od 4 trokuta. Segmenti na lijevoj strani modelirani su zajedno, tako da su srednje točke iste, te njihove normale dijeljene, a segmenti se sastoje od 6 točaka. Segmenti na desnoj strani su zapravo dva odvojena kvadrata, te se sastoje od 8 točaka, od kojih svaka ima vlastitu normalu, te nema normala koje su dobivene od normala više strana. Normale svake točke predstavljene su crvenim linijama. Na donjem djelu slike, u kojem su kvadrati segmenata pod kutem, vidljivo je da su normale dijeljene (lijevo), to jest odvojene (desno), te je očita razlika u sjenčanju.



Slika 3.3.3. – prikaz dijeljenja (lijevo) i nedijeljenja (desno) normala između točaka segmenata

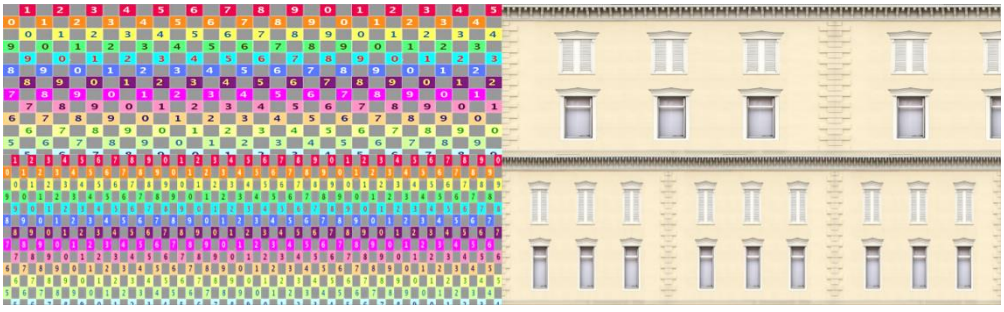
Točke „dijele“ uv koordinate na isti način kao i normale, te je za mogućnosti nezavisnog teksturiranja svake strane zgrade potrebno imati zasebne točke na istim položajima. Slika 3.3.4. prikazuje teksturiranje segmenata identičnih onima sa slike 3.3.3.: segmenti lijeve strane obuhvaćaju cijelu teksturu, a segmenti desne strane mogu imati nezavisne UV koordinate, što omogućava proizvoljno skaliranje koordinata uzorkovanja tekstura.

U jednostavnom modeliranju vanjskog izgleda zgrada često se koriste modeli čije su strane odvojene cjeline upravo zbog slobodnijih mogućnosti teksturiranja.

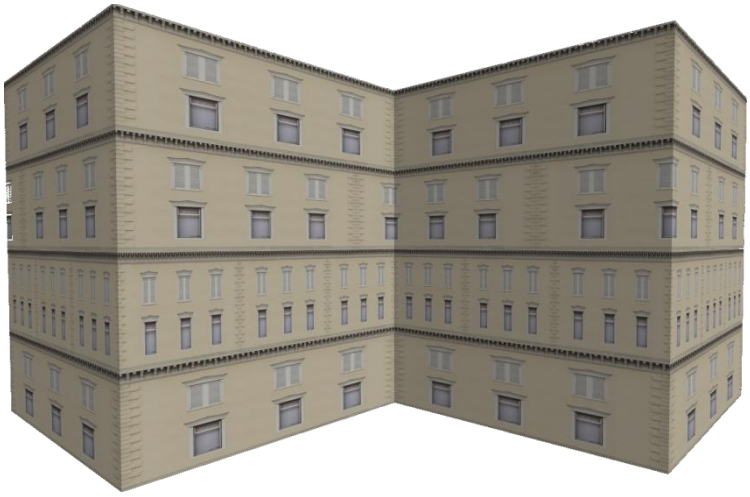


Slika 3.3.4. – prikaz dijeljenja (lijevo) i nedijeljenja (desno) uv koordinata između točaka segmenata

U praktičnoj primjeni ove metode, uz odvojene točke kako bi se dobile zasebne UV koordinate, koristi se i skaliranje UV koordinata na temelju udaljenosti točaka istog trokuta u lokalnom prostoru modela. Za primjenu skaliranja potrebno je samo definirati parametar skaliranja udaljenosti dvije točke kako bi se dobile njihove UV koordinate. U praksi, korištena tekstura mora biti horizontalno ponovljiva (tekstura je ponovljiva po osi ako se njenim ponovljenim iscrtavanjem ne vidi granica između lijevog i desnog kraja teksture), te je poželjno da konačna vrijednost UV koordinata bude cijeli broj, kako se ne bi dogodilo da je samo dio teksture vidljiv na jednom od krajeva strana. Slika 3.3.5. ilustrira primjenu ovog principa i njegove probleme. U gornjem redu prikazana je strana zgrade kojoj su UV koordinate postavljane na vrijednost tako da se tekstura omota zadani broj puta oko zgrade, te je vidljivo kako dolazi do neželjenog djelomičnog prikaza teksture. U donjem dijelu slike vidi se rezultat metode koja pridjeljuje broj ponavljanja ovisno o udaljenosti točaka u prostoru svijeta, uz zaokruživanje na niži cijeli broj.



Slika 3.3.5. – prikaz dva kata proceduralno generirane zgrade s naglaskom na skaliranje uv koordinata



Slika 3.3.6. – prikaz četiri kata proceduralno generirane zgrade

Slika 3.3.6. prikazuje četiri kata zgrade od kojih svaki ima zasebne parametre za generiranje UV koordinata. Faktor ponavljanja koordinata za prvi kat je 1 po svakih 0.15 prostornih jedinica udaljenosti točaka u xz ravnini (y os je prema gore), drugi kat ima faktor 0.3, no s obzirom na zaokruživanje konačnog faktora na niži cijeli broj, drugi kat na vidljivim stranama ima tri ponavljanja teksture, dok prvi kat ima samo jedno. Treći i četvrti kat su generirani uz pridjeljivanje UV koordinata na način da se cijela tekstura omota oko svih strana istog kata zadani broj puta: broj omotavanja za treći kat je 10, a za četvrti 8 puta. Na trećem katu su očiti problemi ove tehnike – prekidanje teksture i savijanje oko kutova. Četvrti kat izgleda identično prvom jer ova zgrada ima točno 8 segmenata, pa je faktor 8 postigao preslikavanje jednog ponavljanja teksture po jednoj strani zgrade.

Ova tehnika je zadovoljavajuće rješenje teksturiranja ukoliko se želi generirati samo vanjski izgled zgrade bez puno detalja i uz jednolikost ponavljanja određene skupine tekstura.

3.3.3. Gramatika podjele

Za postizanje veće kontrole i kombiniranja različitih elemenata teksture u smislenu cjelinu, u teksturiranju se može koristiti metoda objašnjena u poglavlju 3.2.2.1. koja koristi gramatiku podjele. U ovom slučaju, gramatika bi se primjenila na podjelu jedne strane zgrade u manje cjeline, koje bi morale imati odvojene točke kako bi se različiti djelovi tekstura mogli nezavisno uzorkovati. Kako bi se

postigle bolje performanse, sve često zajedno korištene teksture spremaju se u jednu veću teksturu, koja se naziva atlas. S obzirom da su tada teksture dio veće teksture, više ne postoji mogućnost ponavljanja UV koordinata (vrijednosti veće od 1) na istom poligonu, te se za svako ponavljanje elementa moraju generirati novi poligoni, što rezultira daleko većim ukupnim brojem geometrije modela u odnosu na prethodne tehnike, no ovom tehnikom su moguće neograničene varijacije u izgledu teksturiranja zgrade, te je konačni model puno detaljniji i realističniji.

3.3.4. Teksturiranje slojevitom mrežom

U radu (Parish 2001.) predložen je novi princip teksturiranja, jer je vrijeme pripreme tekstura bilo preveliko u odnosu na vrijeme proceduralnog generiranja modela građevina.

Za teksture korištene su slike pravih građevina koje su modificirane te projicirane na geometriju modela pomoću UV koordinata njegovih točaka.

Osmišljen princip temelji se na algoritamskom kombiniranju gotovih tekstura, nazvan slojevita mreža (engl. layered grids). Princip se temelji na slijedećim zapažanjima nastalim promatranjem fasada pravih zgrada.

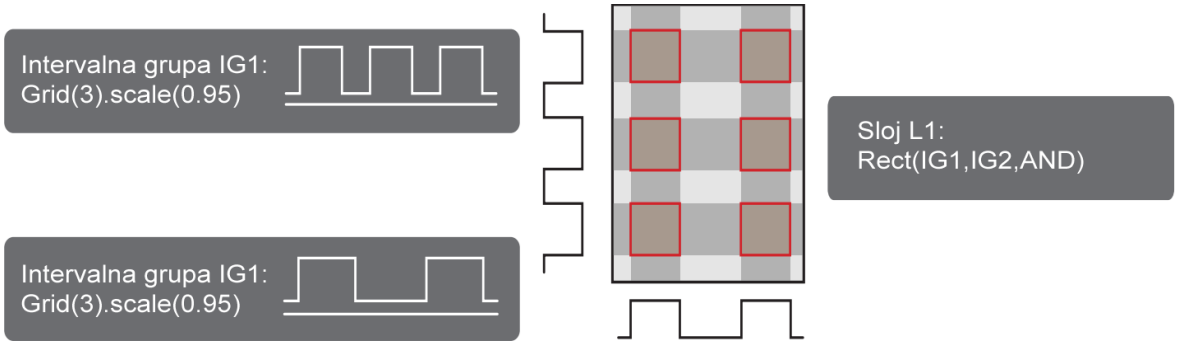
- Fasade (pročelja) se sastoje od višeslojne mrežaste strukture, gdje većina ćelija mreže obavlja istu strukturu (na primjer: ponavljanje prozora, vrata i drugih okvira za otvore)
- Neke ćelije mreže utječu na položaj i veličinu drugih ćelija koje ju okružuju (na primjer, prozori u prizemlju zgrade imaju različitu veličinu od onih iznad vrata)
- neregularnosti u strukturi jedne ćelije mreže utječu na cijeli njen red i stupac

Princip kombinira izdvojene djelove fasada gotovih, pripremljenih tekstura u nove fasade s obzirom na danu veličinu i ulogu ciljne ćelije mreže, izvornu teksturu i druge atribute (poput faktora ulančavanja i skaliranja UV koordinata)

Hijerarhija mreže fasade temelji se na intervalnim grupama. Intervalna grupa je set nepreklapajućih uređenih intervala, matematički gledano, suma jednoliko udaljenih step funkcija iste amplitude.

Prednost kombiniranja jednodimenzionalnih intervalnih grupa je u tome što se broj i veličina redova i stupaca može jednostavno promijeniti promjenom atributa grupe odgovarajuće osi. Svaka točka čiji položaj je evaluiran s vrijednošću

1 smatra se aktivnom točkom te dimenzije. Aktivne točke sloja su one koje su evaluirane u 1 po svim osima koje definiraju taj sloj. Sloj se može odvojiti u aktivne ćelije mreže, koje su zapravo pravokutnici definirani aktivnim točkama sloja. Slika 3.3.7 prikazuje primjenu dvije intervalne grupe na generiranje mreže ćelija jednog segmenta fasade, gdje su aktivne ćelije mreže prikazane crvenim okvirom.



Slika 3.3.7. – prikaz kombiniranja intervalnih grupa u aktivne ćelije logičkom funkcijom I. Slika preuzeta iz (Parish, 2001.)

Aktivne ćelije generirane slojevima se mogu kombinirati uvjetima i logičkim funkcijama, te tvoriti složene proceduralne kombinacije tekstura.

U samoj primjeni tekstura za neki objekt fasade preporuča se koristiti nasumični odabir između identičnih instanci tog objekta u različitom stanju, kako bi se postigla neuniformnost, nasumičnost i realizam. Najjednostavniji primjer toga je tekstura jednog prozora, otvorenog, na pola te potpuno zatvorenog.

Slika 3.3.8 prikazuje ciglenu fasadu zgrade s prozorima i ulazom generiranu različitim parametrima te različitim kombinacijama fukcija između slojeva.

Slika 3.3.8 – prikaz generiranih fasada te kombiniranja i nasumičnog poretka identičnih elemenata (prozora). Slika preuzeta iz (Parish, 2001.)

Ograničenje ovog principa teksturiranja je u tome što svaka izvorna tekstura mora biti ručno definirana: podijeljena na djelove po UV koordinatama teksture (u rasponu [0,1] po obje osi), te svaki dio mora biti označen po svojoj svrsi, poput prozora, vrata, ciglenog zida, balkona, stupova, ukrasa i slično. U trenutku objave članka označavanje se izvršavalo poluautomatski, gdje je algoritam odvajao linearno i pravokutno odvojive cjeline po sličnosti, te ih je samo bilo potrebno ručno kategorizirati. Slika 3.3.9 prikazuje primjer kategorizacije djelova fasade te primjere fasada generiranih na temelju kategorizirane izvorne teksture.

Slika 3.3.9 – lijevo: prikaz izvorne teksture fasade koja je djelomično označena i kategorizirana u segmente, sredina i desno: fasade proceduralno generirane na temelju označene teksture, u kojima se može vidjeti iskorištavanje označenih segmenata izvorne teksture.

3.4 Generiranje normala modela

U potpunom modelu svaka točka mora imati pridružen normalizirani trodimenzionalni vektor, koji predstavlja njenu normalu u lokalnom koordinatnom prostoru te točke, te se koristi pri sjenčanju i drugim algoritmima.

U općem slučaju postoje dva načina izračuna normala. Oba načina opisana su u nastavku. Navedeni načini primjenjivi su na sve tipove modela, iako se u posebnim slučajevima modeliranja vegetacije normale ručno promijene zbog postizanja realističnijeg sjenčanja.

3.4.1. Prosječna normala strane

Prosječna normala strane računa se kao normalizirana suma normala dobivena od normala svih vrhova te strane. Uz pretpostavku da se radi o trokutu, normala strane je jednostavno:

$$\vec{n} = normalize(cross(normalize(v2 - v1), normalize(v3 - v1))$$

gdje je *normalize* funkcija normaliziranja trodimenzionalnog vektora, *cross* je funkcija vektorskog umnoška dvaju trodimenzionalnih vektora, a *v1, v2* i *v3* su vrhovi trokuta.

Nakon što se za svaku točku izračunaju prosječne normale svih njenih strana, radi se novi geometrijski model koji se sastoji od onoliko vrhova koliko je bilo normala, dakle od jedne točke koja se nalazi u 10 strana, nastati će 10 vrhova, svaka sa svojom zasebnom normalom. Pri tom postupku potrebno je i promijeniti niz indeksa koji definiraju strane modela. Duljina niza indeksa će ostati ista, ali više se neće svaka točka referencirati iz svih njenih strana, već će indeksi biti jedinstveni za točku na svim stranama međusobno različitih normala.

3.4.2. Prosječna normala točke

Prosječna normala točke računa se kao normaliziran vektor sume svih normala točke. Za jednu točku može se odrediti po jedna normala za svaki poligon u kojem se točka nalazi, gdje su poligoni najčešće rastavljeni u trokute, jer ih takve crta grafička kartica.

Korištenje prosječnih normala točke poželjno je u zaglađenim modelima najčešće organskih objekata, jer se dodatna glatkoća u sjenčanju postiže automatiziranom interpolacijom normala točaka preko slikovnih elemenata trokuta iz prijelaza od programa za sjenčanje vrhova do programa za sjenčanje slikovnih elemenata.

Model čije se normale računaju ovim postupkom zadržava isti broj vrhova.

S obzirom da su u arhitekturalnim elementima češći kutevi od 90 stupnjeva ili oštriji, ovdje je prikladnija metoda izračuna normala po strani.

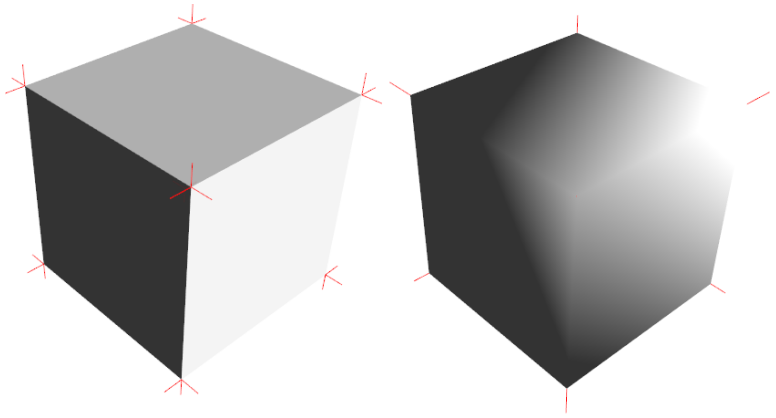
3.4.3. Normala na temelju granice zaglađenosti

Ovo je „treća“ metoda izračuna normala koja kombinira pristupe iz dvije prethodne metode, pa ju je možda bolje zvati izvedenom hibridnom metodom, jer ne unosi nove koncepte.

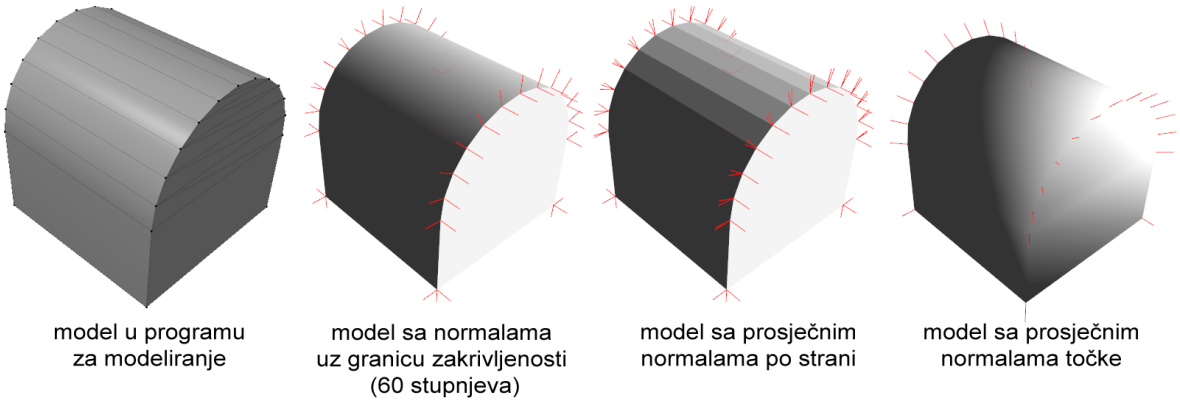
Ova metoda koristi se na modelima koji imaju i oštre i glatke kutove između strana, tako da je na nekim mjestima poželjno imati odvojene, a na drugim prosječne normale.

Granica zaglađenosti je zapravo kut koji predstavlja granicu između primjene prosječne normale točke i strane. Ako je kut između dvije strane manji od granice, onda se točke i njihove normale razdvajaju, inače se točkama pridjeljuje prosječna normala za te dvije strane. Postupak se ponavlja za sve parove susjednih strana modela.

Slika 3.4.1. prikazuje model kocke sa zasebnim normalama po strani (lijevo) te prosječnim normalama za svaki vrh (desno). Na slici je vidljiva razlika u sjenčanju proizašla iz različitih normala. U programima za modeliranje ugrađeni model kocke obično odgovara lijevom prikazu, koji zapravo ima 24 točke, jer su normale zasebne za svih 8 točaka, svih 6 strana različitih skupina normala. Na slici 3.4.2 vidljiv je primje primjene svih navedenih metoda.



Slika 3.4.1. – prikaz normala modela kocke – prosječna normala strane (lijevo), prosječna normala točke (desno). Normale svake točke označene su crvenim linijama.



Slika 3.4.2 – prikaz modela kocke s lukom na jednoj strani. Na slici je označena vrsta izračuna normala.

4. Ostvareni rezultati

Ovo poglavlje navodi ostvarene tehnike, modele generirane tim tehnikama, mogućnosti daljnjeg razvoja implementacije proceduralnog generiranja arhitekture, te autorova razmatranja za buduća istraživanja na području proceduralnog generiranja.

4.1. Pregled ostvarenih tehnika

4.1.1. Generiranje pročelja zgrada na temelju plana katova

Poglavlje 3.3.2. temelji se na ostvarenoj implementaciji ove metode.

Metoda se temelji na zapisu zgrade koje se sastoji od odvojenih višerazinskih cjelina. Svaka cjelina sastoji se od niza dvodimenzionalnih točaka koje definiraju njen temeljni oblik, niza katova koji dijele temeljni oblik, početnog položaja na y osi, visine i krova. Svaki kat definiran je vlastitom visinom, načinom i parametrima teksturiranja bočnih strana (pročelja) tog kata, te referencom na materijal i područje teksture koji se primjenjuju na taj kat.

Svaka višerazinska cjelina ima definiran vlastiti krov. Svaka vrsta krova ima vlastite parametre, poput visine, izbočenja, kuta bočnih strana te parametara za teksturiranje.

Ova metoda ne zahtjeva definiranje gramatike, samo namještanje parametara teksturiranja da konačni rezultat izgleda skladno uz primjenu određene teksture uz skaliranje u lokalnom prostoru modela. Metoda omogućava iznimno brzo generiranje velikog broja zgrada različitog izgleda zbog varijacija u broju cjelina, obliku temelja (koji nije nužno ostvaren isključivo pravim kutevima između segmenata različitih duljina), broju katova te vrsti i parametrima krovova. Ukoliko se u različitim djelovima zgrade koristi isto područje iste teksture, preporuča se da parametri skaliranja UV koordinata budu identični za postizanje vizualne atraktivnosti.

Slika 4.1.1. prikazuje primjer zgrade nastao korištenjem metode generiranja pročelja zgrada.

Zbog jednostavnosti, niske ekspresivnosti te malene strukture modela, ova metoda je prikladna za korištenje u generiranju nedostupnog okruženja u virtualnoj

okolini koja služi samo za postizanje dojma kompleksnosti objekata na većoj udaljenosti.

Slika 4.1.1. - prikaz pročelja zgrade u Unity-ju izrađene metodom generiranja pročelja zgrada na temelju plana katova

4.1.2 Organizacija interijera dvodimenzionalne uniformne mreže uz povezanost razina

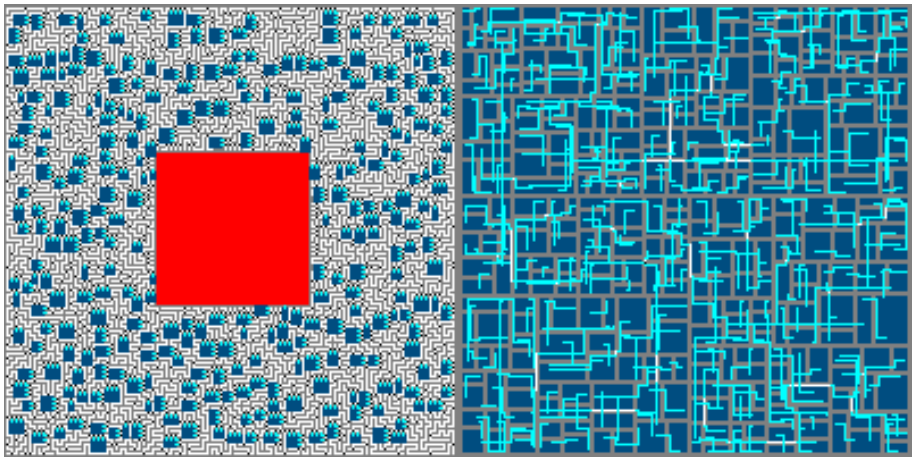
U sklopu ovog rada ostvarene su dvije metode smještanja soba i hodnika u uniformnu mrežu: metoda smještanja soba na temelju uzorka, te metoda binarne podjele prostora.

Glavna struktura je uniformna mreža, koja je definirana pomoću dimenzija te dvodimenzionalnog polja ćelija. Svaka ćelija ima vlastiti položaj u mreži, oznaku posjećenosti (korištenu u algoritmu koji sadrži agente), vrstu ćelije (prazna, zid, blokirana, soba, hodnik, stepenice), te 4 oznake za vrstu strana ćelije (prazno, zid, vrata, ograda, stepenice gore ili dolje).

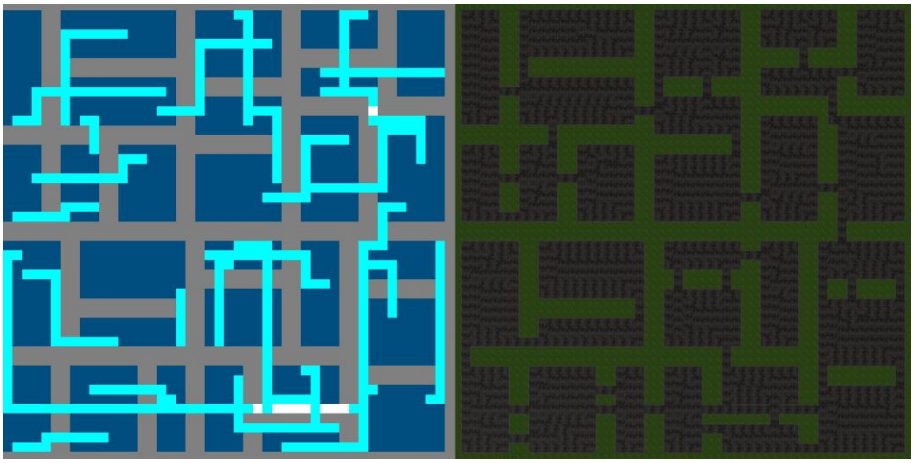
Algoritam smještanja soba na temelju uzorka pretražuje slobodna mjesta za postavljanje soba, smješta sobe različite veličine koje sve povezuje hodnicima, a preostale ćelije koriste se za postavljanje stepenica ili stvaranje dodatnih slijepih hodnika. Uzorak se zadaje matricom nula i jedinica koja se skalira na punu veličinu mreže, te se na ćelijama na kojima se u skaliranoj matrici nalazi oznaka blokade ne može postaviti nikakav sadržaj (soba, hodnik, stepenice). Na lijevom djelu slike 4.1.2. je vidljiv crveni kvadrat koji predstavlja blokirane ćelije na kojima

nije generiran sadržaj, tamno plave kvadrate koji predstavljaju sobe, svijetlo plave ćelije ulaza u sobe te bijele ćelije hodnika.

Algoritam binarne podjele prostora pretpostavlja da su sve ćelije prazne, te dijeli prostor na dva dijela zadana parametrima poput minimalnih dimenzija podjele, minimalne površine, sve dok nije dosegnuta najveća dopuštena dubina ili je preostali prostor premalen za podjelu. Nakon toga u svaki od podprostora se smješta soba te se povezuje sa sadržajem iste dubine, a potom sa sadržajem više razine sve dok cijela mreža nije povezana. Na desnom djelu slike 4.1.2. vidljiv je raspored mreže za primjenu binarne podjele prostora, uz isto značenje boja kao i na lijevom djelu.



Slika 4.1.2. – prikaz kontrolne teksture s kodiranim ćelijama mreže:generiranje sadržaja interijera po uzorku (lijevo) te binarna podjela prostora (desno)



Slika 4.1.3. – prikaz binarne podjele prostora – mapa odgovara organizaciji instanciranih postojećih modela

Slika 4.1.4. – prikaz generirane organizacije interijera

Slika 4.1.5. – prikaz generirane organizacije interijera

4.1.3. Generiranje 2D staze hijerarhijom čvorova linijskih segmenata

Ova metoda temelji se na generiranju povezanog niza segmenata koji se u kasnijim koracima koristi kao jednodimenzionalna ili dvodimenzionalna osnova za proširivanje oblika kroz prostor. Na primjer, ova metoda može se upotrijebiti za generiranje temelja razine zgrade, temelja stupa, radijusa stupa po y osi, osnove mosta ili lukova i slično.

Gramatika korištena u metodi je kontekstno neovisna gramatika u kojoj su pravila produkcije grupirana po zajedničkom nezavršnom znaku lijeve strane produkcije – svako pravilo sastoji se od lijeve strane (jedinstvenog identifikatora) te skupine desnih strana. Svaka desna strana pravila sastoji se od jedinstvenog imena te desne strane, vjerojatnosti njenog odabira između svih ostalih desnih strana, te tri skupine nezavršnih znakova:

- skupina početnih nezavršnih znakova
- skupina ponavljajućih nezavršnih znakova
- skupina konačnih nezavršnih znakova

Ovisno o vrsti čvora, iz jednog linijskog segmenta može nastati veći broj linijskih segmenata čije su krajnje točke identične izvornom segmentu. Pri tome, nastalim segmentima pridjeljuju se nezavršni znakovi, koji će se zamijeniti novim čvorovima. Prvo se dodjeljuju početni, zatim konačni te na kraju ponavljajući nezavršni znakovi. Ukoliko čvor nema dovoljno segmenata za neku skupinu, dodjeljivanje segmenata staje, te se izvršavanje pravila produkcija nastavlja.

Generirana staza može biti otvorena i zatvorena. To ovisi o vrsti početnog čvora gramatike. U nastavku je ukratko opisan svaki od implementiranih vrsta čvorova.

1. Kružni modifikator

   Kružni modifikator (engl. Circle modifier) prikladan je inicijalni čvor hijerarhije, jer se sastoji od zadanog broja segmenata jednake duljine, čiji vrhovi se nalaze na zadanoj udaljenosti od parametra središnje točke. Ovaj modifikator (vrsta čvora) nazvan je kružnim jer se koristi za stvaranje zatvorenog niza linijskih segmenata koji su u obliku kruga. Naravno, za tri segmenta ovo je jednakostraničan trokut, za četiri - kvadrat, za šest – šesterokut, te su ovo češći načini korištenja ovog modifikatora.

2. Modifikator jednostavnog proširenja

   Modifikator jednostavnog proširenja (engl. Extrude modifier) koristi se za jednostavno proširivanje linijskog segmenta jednom ili dvjema točkama, te se može smatrati osnovnom operacijom ove metode. Slika 4.1.6. prikazuje moguće primjene ovog modifikatora.

3. Modifikator podjele

   Ovaj modifikator koristi se za podjelu linijskog segmenta u manje djelove, i to na temelju zadanih udaljenosti koje mogu biti apsolutne ili relativne. Pri podjeli prvo se odvoji prostor za segmente apsolutne duljine, a zatim se preostala duljina izvornog segmenta podijeli na nove relativne segmente.

4. Modifikator ponavljanja

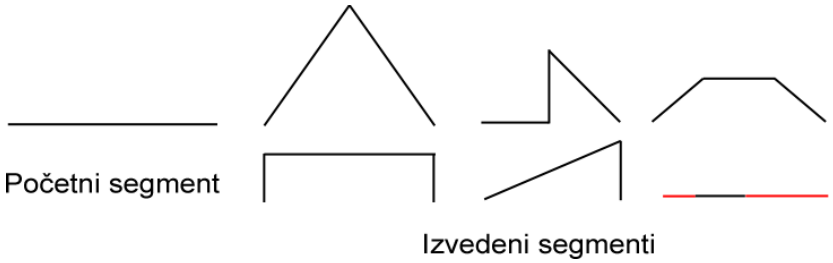
   Ovaj modifikator ima sličan efekt kao modifikator podjele, osim što ima samo parametar duljine novog segmenta, koja može biti apsolutna i relativna, te parametar načina primjene podjele: od početka, od kraja, od sredine izvornog segmenta, ili uz proizvoljni pomak od početka. Slika 4.1.7. prikazuje moguće rezultate primjene ovog modifikatora.

5. Modifikator kružnog luka

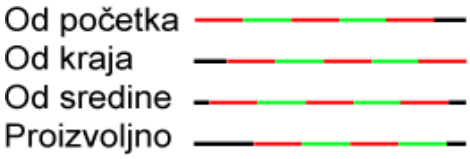
   Ovaj modifikator stvara kružni luk zadanog radijusa i broja segmenata s određene strane izvornog segmenta, gdje su početna i krajnja točka izvornog segmenta početna i krajnja točka stvorenog luka.

6. Bezierov kubni modifikator

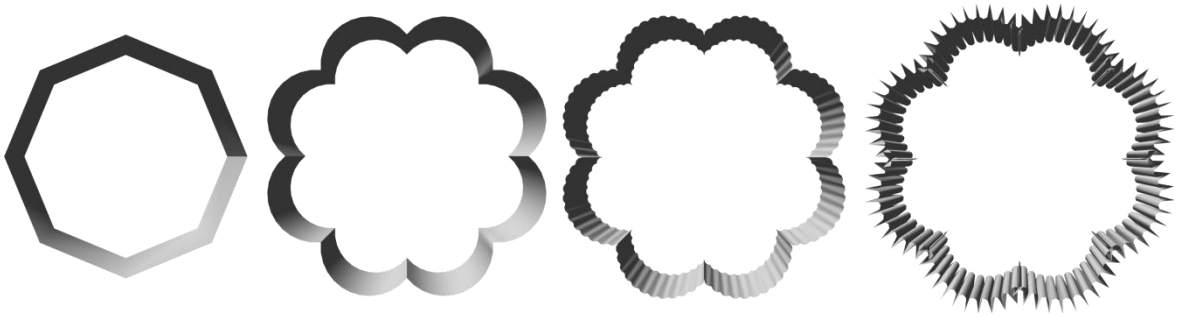
   Ovaj modifikator zadan je preko pomaka od početne i krajnje točke izvornog segmenta. Te četiri točke zajedno čine segment Bezierove kubne krivulje, koja se uzorkuje na zadanom intervalu određeni broj puta.



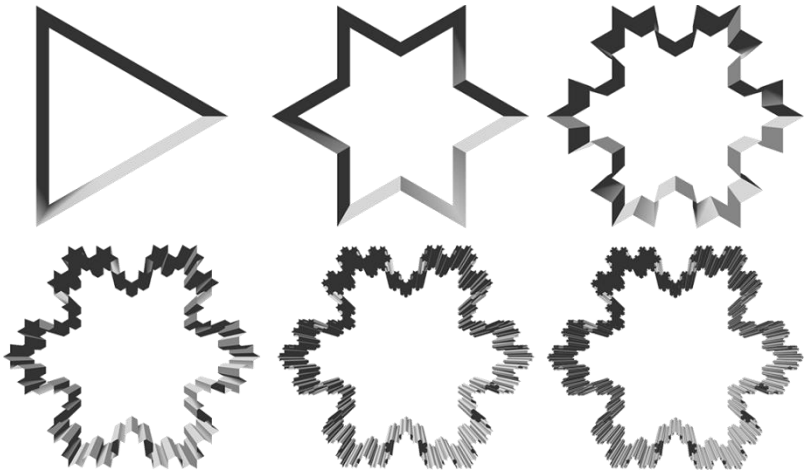
Slika 4.1.6. – prikaz primjene modifikatora jednostavnog proširenja linijskog segmenta



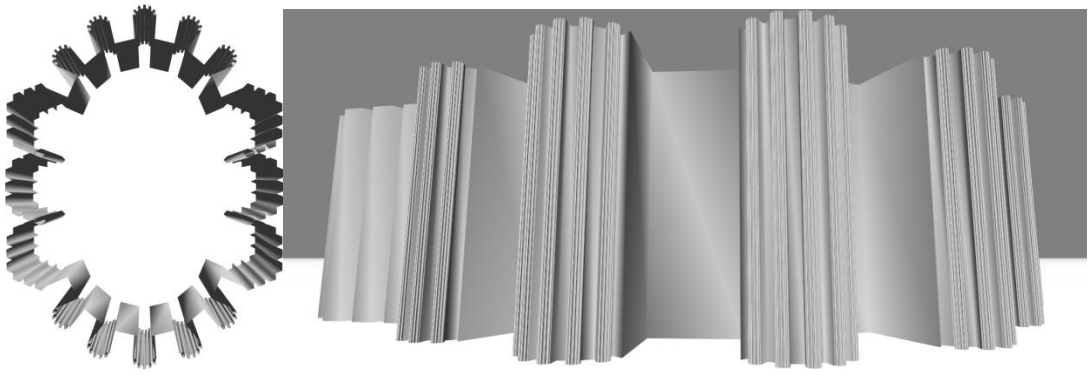
Slika 4.1.7. – prikaz rezultata primjene modifikatora ponavljanja za različite vrijednosti parametara načina primjene podjele (vrijednosti parametra označene na slici)



Slika 4.1.8. – prikaz generiranja staze modifikatorima linearnih segmenata: slijeva na desno: kružni modifikator (8 segmenata, radijus 1), kružni luk (12 segmenata, radijus 0.15r), 1. proširenje (pomaci po liniji 0.33, tangencijalni 0.2), 2. proširenje (pomaci po liniji 0.2, tangencijalni ± 4)



Slika 4.1.9. – prikaz 6 iteracija generiranja Kochove pahuljice nizom linijskih segmenata korištenjem kružnog modifikatora te jednostavnog linijskog proširenja



Slika 4.1.10. – prikaz rezultata kombinacija više modifikatora

Zaključak ove metode je da pruža veliku ekspresivnost u spajanju pravila te oblikovanju hijerarhije oblika, te pruža zadovoljavajuće performanse jer se cijelo

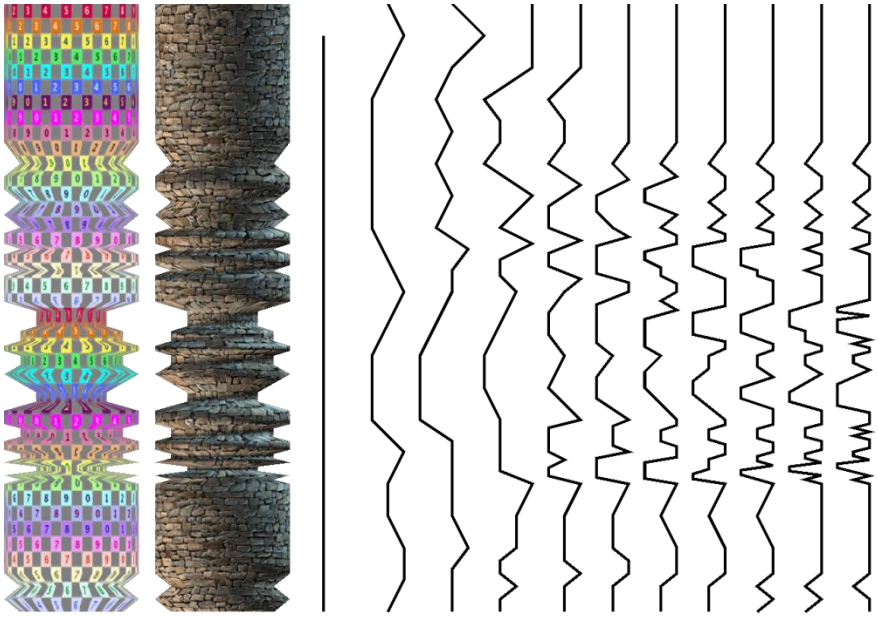
izvršavanje gramatike i oblikovanje modela iz interpretacije svodi na jednostavne petlje, dohvaćanje podataka iz rječnika (tablice ključeva), te stvaranje djece u hijerarhiji čvorova, te konačni oblilazak stabla. Nedostatak ove metode je velik broj parametara i kompleksna definicija postupka za jednostavan rezultat: potrebno je definirati pravila i vjerojatnosti gramatike (uz potencijalne uvjete u proširenju gramatike), a zatim je potrebno definirati svaki jedinstveni modifikator (što je ublaženo inicijalnim vrijednostima svih parametara koje odgovaraju najčešće korištenim i najpoželjnijim vrijednostima). Ukratko, potrebno je definirati velik broj nizova znakova te u prosjeku oko 5 parametara po modifikatoru. Na sreću, primjenama identičnih modifikatora u različitim redoslijedima te na različitim granama hijerarhije stabla postižu se raznovrsni i zanimljivi rezultati, koji su primjenjivi u drugim tehnikama navedenim u ovom radu.

4.1.4. Gramatika L-sustava

Ostvarena implementacija gramatike L-sustava koristi se za generiranje osnovnih elemenata po uzoru na metodu opisanu u poglavlju 3.2.5. Definiraju se pravila produkcije (lijeva, te nizovi desnih strana uz pripadajuće vjerojatnosti), broj iteracija, te se stvori niz koji se interpretira na jedan od mogućih načina:

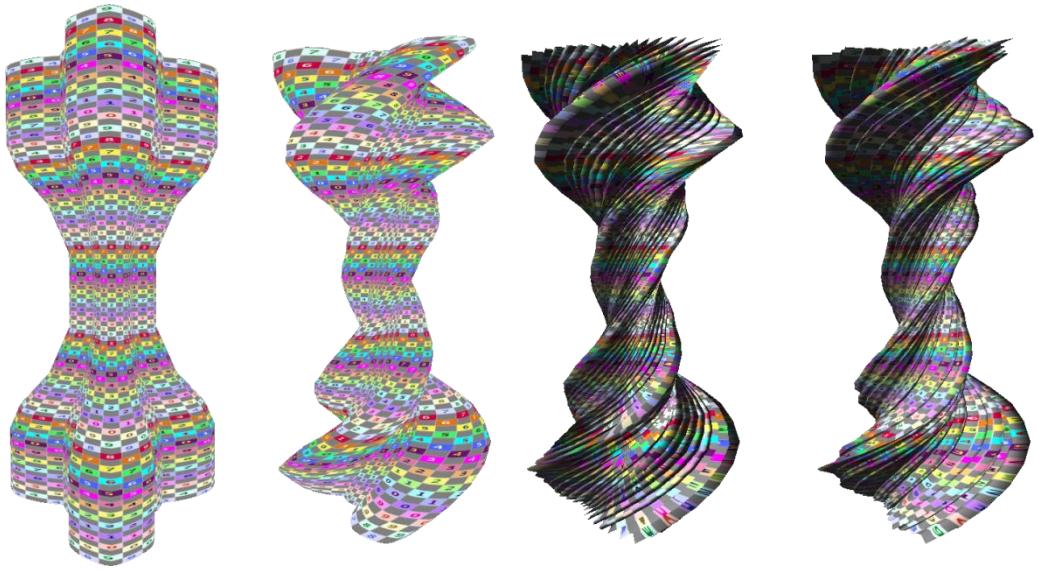
- Položaji točaka na osnovnom putu (inicijalno pomaci za stvaranje točaka na y osi)
- Radijusi kružnih segmenata konačnog modela gdje se jedan radijus pridjeljuje točci osnovnog puta
- Faktori skaliranja radijusa djelova kružnog segmenta
- Kutovi između točaka kružnog segmenta (zajednički cijeloj strukturi)
- Kutovi međusobne rotacije kružnih segmenata
- Kutovi savijanja oko sporedne osi (za generiranje lukova i mostova)

Slika 4.1.11. prikazuje stup generiran korištenjem ove metode, nastao radijalnim proširenjem linijskog segmenta koji je interpretiran iz niza znakova generacije L gramatike. Desno od modela stupova vidljive su generacije primjene gramatike (slijeva na desno), gdje su znakovi interpretirani kao pomaci po x osi. Na lijevom modelu stupa vidljiva je pravilna raspoređenost UV koordinata točaka, a na desnom primjena konkretne teksture kamenog uzorka.



Slika 4.1.11. – prikaz stupa generiranog L gramatikom

Slika 4.1.12. prikazuje stup generiran korištenjem ove metode, gdje je za razliku od prethodne metode vidljivo korištenje niza za osnu rotaciju između segmenata. Slika 4.1.14. prikazuje praktičnu primjenu modela generiranog korištenjem L-gramatike u svrhu generiranja virtualne scene u programu za modeliranje.



Slika 4.1.13. – prikaz modela uz korištenje niza za osnu rotaciju segmenata: krajnje lijevo prikazan je model bez osne rotacije, a ostali prikazi su modeli s različitim parametrima rotacije i sjenčanja.

Slika 4.1.14. – korištenje generiranog modela u stvaranju virtualne scene

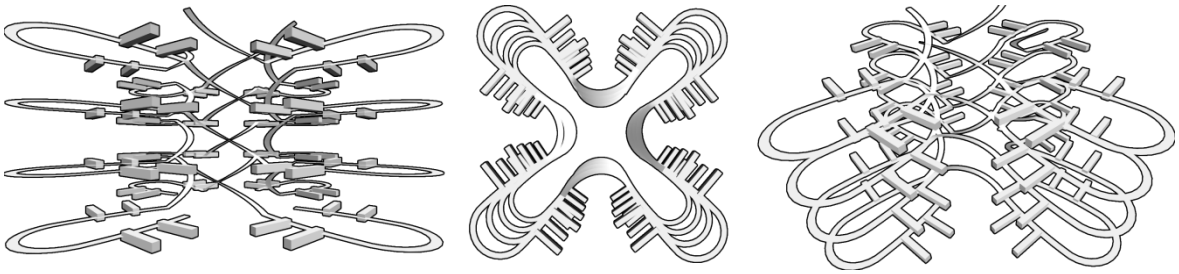
4.1.5. Gramatika kornjače

Po uzoru na poglavlje 3.2.1.3., implementirana je gramatika fiktivne kornjače. Stanje kornjače sastoji se od položaja, rotacije (sustava određenog kvaternionom), vektora pomaka po osima, vektora promjene rotacija po osima, te debljine linije koju kornjača može ostavljati. U nastavku je dan popis implementiranih naredbi kornjače:

- Push i pop – spremanje trenutnog i obnavljanje prethodnogstanja kornjače sa stoga
- Promjena položaja uz mogućnost crtanja linije od trenutnog do novog položaja
- Promjena trenutne rotacije, promjena pomaka ili rotacijskih pomaka
- Podržane su operacije postavljanja, dodavanja, oduzimanja, množenja i dijeljenja konstanti ili varijabli pomaka
- U bilo kojem koraku moguće je stvoriti instance postojećih objekata, gdje se položaj i rotacija stvorenog objekta uzorkuju u odnosu na trenutno stanje kornjače u tom koraku

S obzirom da i ova metoda funkcionira na temelju kontekstno neovisne gramatike, oblik pravila produkcija je isti: svako pravilo ima lijevu stranu i niz desnih strana, gdje je svakoj pridružena vjerojatnost odabira.

U odnosu na prethodnu navedenu implementaciju L-sustava, ova metoda ima veće zahtjeve definicije te je teže pratiti njeno stanje iz samih produkcija, no pruža veću ekspresivnost i slobodu kretanja. Ukoliko se ukupni put kornjače može rastaviti na označene djelove, te se kornjačini pomaci u kojima nije „iscrtavala put“ ne uzimaju u obzir kao segmenti, ova gramatika potpuno je ekvivalentna L-gramatici.



Slika 4.1.15. – prikaz staze i objekata generiranih gramatikom kornjače

4.1.6. Gramatika oblika

Po uzoru na metodu iz poglavlja 3.2.3. te razmatranja komentara čitatelja članka „Proceduralno generiranje građevina“ (Müller 2006.), implementirana je gramatika oblika.

Oblik je definiran preko trodimenzionalnog položaja, veličine, rotacije, referentne točke operacija nad njime te završnim ili nezavršnim znakom. Dodatno, svaki oblik ima referencu na roditeljski oblik (koji u potpunosti sadrži sve oblike nastale iz njega), vlastitu djecu, može sadržavati konkretnu geometriju (poput primitivnih elemenata ili prethodno izrađenih modela), oznake korištene u pretrazi hijerarhije oblika, te oznaku zaklanjanja drugih oblika.

Gramatika generiranja oblika temelji se na izgrađivanju hijerarhije oblika, gdje se oblici modificiraju te stvaraju manje oblike podjelama vlastitog volumena prostora. Uz oblik, u ovoj gramatici postoji još jedna važna struktura, a to je modul.

Modul je definiran pomoću vlastitog imena koje je ekvivalentno imenu funkcije, ili nezavršnog znaka na lijevoj strani produkcije. Uz ime, modul sadrži i popis uvjeta te naredbi.

Gramatika se definira pomoću niza naredbi, koje služe definiranju inicijalnih parametara početnog oblika, globalnih varijabli, globalnih osi poravnjanja ili modula. Za modul istog imena moguće je definirati više blokova naredbi, te se pri izvršavanju aktiviraju naredbe prvog navedenog modula u tekstualnoj datoteci definicije gramatike čiji su uvjeti u potpunosti zadovoljeni.

Uvjeti podržani implementacijom su usporedbe vrijednosti trenutnog oblika ili globalnih varijabli međusobno ili s konstantama, te provjera zaklanjanja trenutnog oblika s roditeljima, djecom te oblicima željenog identifikatora.

Podržane osnovne naredbe nad oblikom su pomak, rotacija i skaliranje pri čemu se mogu koristiti izrazi s globalnim varijablama, vrijednostima trenutnog oblika, konstantama, zagrade te operatori zbrajanja, oduzimanja, množenja, dijeljenja, te logičke operacije I, ILI i NE, a između brojeva se mogu koristiti operatori $<, >, =, \neq \leq, \geq$.

Od preostalih naredbi implementirane su obična te ponavljajuća podjela (engl. repeat), instanciranje postojećih modela unutar trenutnog oblika, korištenje poravnjanja, instanciranje geometrije unutar oblika te označavanje strana oblika.

Ova gramatika pruža ograničenu funkcionalnost modeliranja, te je prikladnija za prostorno organiziranje gotovih modela. Ukoliko bi se gramatika proširila, mogla bi se postići zavidna moć modeliranja izravnim korištenjem ove metode, no to nije njena primarna uloga.

4.2. Mogućnosti daljnjeg razvoja ostvarene implementacije

Ovo podpoglavlje daje prijedloge za mogućnosti poboljšanja implementacije stvorene tijekom izrade ovog rada.

- Implementacija interaktivne definicije i povezivanja građevnih elemenata metoda za proceduralno generiranje arhitekture u vizualne čvorove koji se mogu vezama povezati u hijerarhije (engl. Node-based editor)
- Spajanje implementacija različitih metoda na način da se pojedini djelovi mogu objektno uključiti u druge dijelove. Formaliziranje i proširenje zasebnih gramatika te zajedničke gramatike. U smislu prostorne organizacije to znači porast ekspresivnosti i izražajnosti gotove metode. Sa

strane implementacije, organizacije i čitljivosti, to znači porast kompleksnosti i neintuitivnosti.

- Implementacija prikaza međurezultata i metaparametara generiranog sadržaja korisniku, kako bi mogao vidjeti izravan utjecaj
- Promjena prikaza generiranog sadržaja (modela) koja reagira na promjenu parametara u stvarnom vremenu
- Stvaranje tekstualnih i video primjera za korištenje implementiranih algoritama.
- Implementacija biblioteka u drugim jezicima kako bi se omogućila portabilnost i uključivanje u druge programe i razvojna okruženja
- Implementacija ili uključivanje biblioteke trećeg izvora koja pruža funkcionalnosti konstruktivne čvrste geometrije (engl. Constructive solid geometry), kako bi se omogućile 3D Booleove operacije nad objektima
- Implementacija ili uključivanje biblioteke trećeg izvora koja pruža funkcionalnosti predstavljanja geometrije pomoću volumnih elemenata (engl. voxel)

4.3. Razmatranja za buduća istraživanja u proceduralnom generiranju sadržaja

U ovom podpoglavlju dana su autorova razmatranja o budućem istraživanju i razvoju proceduralnog generiranja.

Potrebno je uložiti vremena i truda u razvoj metoda koje će generirati vjerodostojnu cjelinu virtualnog sadržaja i time obuhvaćati sve ostale grane jednostavnijeg sadržaja proceduralnog generiranja. Na primjer, uz dana pravila arhitekture i izgleda materijala nekog mjesta u povijesti, uz povijesne i kulturne činjenice i nagađanja, socijalne običaje i ekonomsko stanje, bilo bi moguće stvoriti virtualnu okolinu u kojoj bi se mogli proživjeti povijesni trenuci, komunicirati s povijesnim ličnostima, te akcijama korisnika „promijeniti“ i utjecati na ishod virtualno reproducirane povijesti, što bi mogla biti nova razina interaktivne edukacije i način proučavanja ljudske povijesti.

Metode za generiranje nekih grana bitovnog sadržaja, poput tekstura ili modela terena su detaljno istražene, te je sada potrebno usmjeriti napore na stvaranje i poboljšanje metoda u područjima koja nisu dovoljno istražena, poput generiranje animacija i modela životinja, vozila, ljudi. Taj sadržaj bi se mogao koristiti u simulacijama prirodnih nepogoda, fizikalnom testiranju vozila, simulaciji biosfera i djelovanja ljudi na okoliš i općenito u vizualizacijama tog sadržaja, uključujući računalne igre.

S obzirom na Moore-ov zakon, razina detalja koja se može generirati neprekidno raste, no kako društvo u smislu prosječnog potrošača zaostaje za napretkom tehnologije, potrebno je obratiti pozornost na ubrzanje, optimizaciju i paralelizaciju algoritama generiranja sadržaja, no to nije toliki problem u usporedbi s prikazom istog sadržaja, jer je trenutačno brzina iscrtavanja sadržaja usko grlo u procesu izrade i prikaza proceduralnog sadržaja, što upućuje na potrebu poboljšanja povezanosti i performansi glavnog i grafičkog procesora te popratnih memorija, dok su algoritmi prikaza već dovoljno napredovali.

Konačno, potrebno je osmišljanje metoda za auto-evaluaciju generiranog sadržaja, kako bi se dobila ušteda na vremenu u automatiziranom generiranju, povratna informacija o raznim karakteristikama sadržaja za promijenjene parametre generiranja, te podigla njegova ukupna kvaliteta.

5. Zaključak

Područje proceduralnog generiranja je u naglom razvoju, te je sve veći dio naših života. S obzirom na porast računalne snage, uskoro ćemo moći šetati, voziti se, ploviti i letjeti virtualnim prikazom našeg stvarnog svijeta, a ne samo pregledavati satelitske i diskretne ulične slike Google Maps-a. No to nije razlog da se prepustimo virtualnom, već da počnemo više cijeniti ono što je priroda stvorila, te što je čovjek sagradio kroz cijelu povijest. U svoj uređenosti objekata prisutna su neizrečena pravila, te je na izgled utjecala pseudonasumičnost rasporeda atoma njenih materijala, njihove kvalitete potvrđene su genetskim algoritmom vremena, i estetski prosuđene ljudskim očima.

Proceduralno generiranje kao moćan koncept ljudskog stvaranja u svrhu zabave, vizualizacije i edukacije, postaje grana računarstva koja stvara, objašnjava, sistematizira i testira pravila za smisleni nastanak, održavanje, evoluciju i povezanost elemenata u definiranu, skladnu cjelinu.

Napredak u razvoju proceduralnog generiranja pridonijet će dubljem razumijevanju postanka, pohrane i promjena u sadržaju, te će osim alata za teorijski beskonačno generiranje interaktivnih i zanimljivih sadržaja pružiti algoritme koji će se praktično koristiti u potpuno ručnom generiranju, a razviti će se i hibridan način razvoja sadržaja u kojem se ljudska kreativnost i algoritamska ekspresivnost izmjenjuju u stvaranju i evaluaciji virtualnih umjetničkih djela, umjetnosti, svjetova.

6. Literatura

[1] – Lindenmayer, A., & Prusinkiewicz, P. (1990). *The Algorithmic Beauty of Plants*. Springer-Verlag., http://algorithmicbotany.org/papers/#abop

[2] – Unity Engine 4.2.1f - http://unity3d.com/unity/download/archive

[3] – Shaker, N., Togelius, J., Nelson, M. J.: Procedural Content Generation in Games: A Textbook and an Overview of Current Research, Springer, 2014. http://pcgbook.com/

[4] Hendrikx, M., Meijer, S., van der Velden, J., Iosup, A.: Procedural content generation for games: a survey. ACM Transactions on Multimedia Computing, Communications and Applications, 2011. http://www.pds.ewi.tudelft.nl/~iosup/pcg-g-survey11tomccap_cr.pdf

[5] PCG Wiki: Procedural content generation wiki. URL http://pcg.wikidot.com/

[6] Togelius, J. et al. "Search-based procedural content generation: A taxonomy and survey." *Computational Intelligence and AI in Games, IEEE Transactions on* 3.3 (2011): 172-186., http://julian.togelius.com/Togelius2011Searchbased.pdf

[7] .theprodukkt, .kkrieger, 2004., http://web.archive.org/web/20110717024227/http://www.theprodukkt.com/kkrieger

[8] Remo, C.: MIGS: http://www.gamasutra.com/php-bin/news_index.php?story=21165

[9] Martin, A., Lim, A., Colton, S., Browne, C., "Evolving 3d buildings for the prototype video game subversion," in *Proceedings of EvoApplications*, 2010. http://ccg.doc.gold.ac.uk/papers/martin_evogames10.pdf

[10] Müller, P. et al, "Procedural modeling of buildings," *ACM Transactions on Graphics*, vol. 25, pp. 614–623, 2006. https://www.vision.ee.ethz.ch/publications/papers/proceedings/eth_biwi_00399.pdf

[11] Golding, J., "Building blocks: Artist driven procedural buildings," Presentation at Game Developers' Conference, 2010. http://gdcvault.com/play/1012655/Building-Blocks-Artist-Driven-Procedural

[12] Tutenel, Tim, et al. "Generating consistent buildings: a semantic approach for integrating procedural techniques." *Computational Intelligence and AI in Games, IEEE Transactions on* 3.3 (2011): 274-288. http://graphics.tudelft.nl/~rval/papers/tutenel.tciaig11.pdf

[13] Sorenson, Nathan, and Philippe Pasquier. "Towards a generic framework for automated video game level creation." *Applications of Evolutionary Computation*. Springer Berlin Heidelberg, 2010. 131-140. http://www.researchgate.net/publication/220867545_Towards_a_Generic_Framework_for_Automated_Video_Game_Level_Creation/file/d912f510ac2bed57d1.pdf

[14] Wang, W., Zmeureanu, R., & Rivard, H. (2005). Applying multi-objective genetic algorithms in green building design optimization. *Building and Environment*,*40*(11), 1512-1525. http://cumincad.architexturez.net/system/files/pdf/caadria2003_b5-2.content.pdf

[15] 2D CSG slika preuzeta s https://upvoid.com/devblog/2013/07/terrain-engine-part-2-volume-generation-and-the-csg-tree/

[16] Hansmeyer, M., L-Systems in Architecture (2003), http://www.michael-hansmeyer.com/projects/l-systems_info.html?screenSize=1&color=1

[17] Murta, A., GPC – General polygon clipper library, http://www.cs.man.ac.uk/~toby/gpc/, pristup 16.5.2014.

[18] Proceduralno generiranje, članak, http://en.wikipedia.org/wiki/Procedural_content_generation, nastanak stranice 11.7.2005., pristup 16.5.2014.

[19] E-on Vue portfolio, prikaz primjene proceduralnog generiranja u filmskoj industriji, http://www.e-onsoftware.com/showcase/, pristup 18.5.2014.

[20] Discoe, B., Virtual Terrain Project, http://vterrain.org/, nastanak stranice 1997., pristup stranici 20.5.2014.

[21] andrewdoull, Software featuring procedural content generation, http://pcg.wikidot.com/category-pcg-software, nastanak stranice 17.5.2008., pristup stranici 16.5.2014.

[22] andrewdoull, Games featuring procedural generation, http://pcg.wikidot.com/category-pcg-games, nastanak stranice 20.7.2009., pristup stranici 22.5.2014.

[23] Larive, M., & Gaildrat, V. (2006, November). Wall grammar for building generation. In *Proceedings of the 4th international conference on Computer graphics and interactive techniques in Australasia and Southeast Asia* (pp. 429-437). ACM. ftp://ftp.irit.fr/IRIT/VORTEX/Larive_graphite2006small.pdf

[24] Parish, Y. I., & Müller, P. (2001, August). Procedural modeling of cities. In Proceedings of the 28th annual conference on Computer graphics and interactive techniques (pp. 301-308). ACM. http://graphics.ethz.ch/Downloads/Publications/Papers/2001/p_Par01.pdf

[25] Kuang, Z., Chan, B., Yu, Y., & Wang, W. (2013). A compact random-access representation for urban modeling and rendering. *ACM Transactions on Graphics (TOG)*, *32*(6), 172. http://i.cs.hku.hk/~yzyu/publication/NUT-sigasia2013.pdf

[26] Wonka, P., Wimmer, M., Sillion, F., & Ribarsky, W. (2003). *Instant architecture* (Vol. 22, No. 3, pp. 669-677). ACM. http://www.cg.tuwien.ac.at/research/vr/instantarchitecture/instant_architecture.pdf

[27] Smelik, R., Tutenel, T., de Kraker, K., Bidarra, R.: Integrating procedural generation and manual editing of virtual worlds. In: *Proceedings of the Workshop on Procedural Content Generation in Games*, p. 2. ACM, 2010. https://blog.itu.dk/mpgg-e2010/files/2010/10/a5-smelik.pdf

[28] Stiny, G., Production systems and grammars: a uniform characterization. *Environment and Planning B 7*, 399.408., 1980.

[29] Greuter, S., Parker, J., Stewart, N., Leach, G., Real-time procedural generation of 'pseudo infinite' cities. In International Conference on Computer Graphics and Interactive Techniques in Australasia and South East Asia. 87–95., 2003.

[30] Smith, G., Whitehead, J., Mateas, M.: Tanagra: A mixed-initiative level design tool. In: Proceedings of the Fifth International Conference on the Foundations of Digital Games, pp. 209– 216. ACM, 2010. http://games.soe.ucsc.edu/sites/default/files/smith-tanagra-fdg10.pdf

[31] Kelly, G., & McCabe, H. (2007). Citygen: An interactive system for procedural city generation. In *Fifth International Conference on Game Design and Technology* (pp. 8-16). http://goo.gl/KI299i

7. Sažetak

Proceduralno generiranje arhitekture virtualne okoline

Ovaj rad bavi se osnovnim konceptima proceduralnog generiranja arhitekture. U uvodu je dana motivacija za istraživanje ove teme. Drugo poglavlje daje definiciju, taksonomiju metoda i podjelu metoda proceduralnog generiranja po karakteristikama, opisuje njihove poželjne značajke te navodi prednosti, nedostatke i praktične primjere. Treće poglavlje pruža pregled metoda generiranja arhitekture, te smještanja i oblikovanja postojećeg sadržaja. U nastavku je dana kratka podjela djelova arhitekture, te slijedi poglavlje koje opisuje ostvarenu implementaciju, rezultate, performanse i buduća razmatranja razvoja ove grane računarstva. Za kraj, dan je zaključak za daljnje mogućnosti i značenje razvoja tehnologije i znanosti na području ove teme.

Ključne riječi: proceduralno, generiranje, arhitektura, labirint, gramatika, L-sustav, automat, genetski algoritam, računalna grafika, vizualizacija, organizacija, prostor

Abstract

Procedural architecture generation in virtual environments

This paper describes the basic concepts of procedural architecture generation. Introduction provides the motivation for researching this topic. Second chapter contains the definition, taxonomy and listing of procedural generation methods by their characteristics, describes their desirable features and offers an overview of advantages, disadvantages and examples for their practical usage. Third chapter offers an overview of basic, common and famous methods for architecture generation, transformation and placement of existing content. The following chapter has a listing of architectural elements by category, together with their breakdown into rules for their generation. Sixth chapter describes the practical part of this thesis; implementation, its performance and generated results. In conclusion, there is a discussion of further possibilities of technology and science on this topic.

Keywords: procedural, generation, architecture, labyrinth, grammar, L-system, turtle graphics, scope grammar, genetic algorithm, automaton, computer graphics, vizualization, organization, space

8. Privitak

8.1. Formalizirane naredbe jezika L-sustava s fiktivnom kornjačom

Popis formaliziranih naredbi dan je u tablici 8.1.1.

Ime i opis naredbi	Znakovi zapisa naredbe
Kretanje: kornjača se može kretati kroz prostor s ili bez iscrtavanja linija putanje	F – kretanje unaprijed uz crtanje linije U – kretanje unaprijed bez linije
Rotacija. Kornjača se može slobodno rotirati oko osi ortonormalnog sustava.	+, - – rotacija oko lokalne y osi & ^ - rotacija oko lokalne x osi / | - rotacija oko lokalne z osi
Varijabilni kutovi rotacije – kutovi koji se koriste pri pozivu naredbi za rotaciju	X(n) – postavljanje kuta rotacije za y os Y(n) – postavljanje kuta rotacije za x os Z(n) – postavljanje kuta rotacije za z os
Pomak – udaljenost koju kornjača prođe pri pozivu naredbi za kretanje	L(n) – postavljanje duljine pomaka
Radius – radijus cijevi koje kornjača iscrtava tijekom pomaka	R(n) – postavljanje radijusa
Grananje – stanje kornjače se može spremiti na stog, te se može obnoviti uklanjanjem s stoga, što omogućava grananje naredbi	[- spremanje stanja kornjače na stog] – obnavljanje stanja kornjače sa stoga
Površine – segmenti linija koje kornjača iscrtava pri kretanju mogu biti radijalno proširene u površinu	P – početak proširivanja segmenata u površinu Q – prekid proširivanja segmenata u površinu

Tablica 8.1.1. – popis formaliziranih naredbi jezika L-sustava s fiktivnom kornjačom

Popis formaliziranog načina promjene parametara s primjerima dan je u tablici 8.1.2., gdje je za ime varijable korištena oznaka X, a za brojčane parametre oznake a,b i c.

Operacije i sintakse	Značenje
Postavljanje vrijednosti varijable	
X(=a)	postavlja varijablu x na iznos a
Promjena prethodne vrijednosti varijable	
X(+a)	zbraja varijablu x s iznosom a
X(-a)	oduzima iznos a od varijable x
X(*a)	množi iznos a s varijablom x
X(/a)	dijeli varijablu a s iznosom x
Korištenje nasumične vrijednosti za varijablu	

X(=a,b)	postavlja varijablu x na nasumičnu vrijednost iz raspona [a,b]
X(=a,b,c)	postavlja varijablu x na nasumičnu vrijednost iz raspona [a,b] koja je uzorkovana u intervalima iznosa c (na primjer X(=10,30,5) bi postavilo X na jednu od vrijednosti iz skupa (10,15,20,25,30)
X(*2,3)	množi varijablu X s nasumičnim brojem iz raspona [2,3]

Operacija	Sintaksa
Postavljanje vrijednosti varijable	X(=a) – postavlja varijablu x na iznos a
Promjena prethodne vrijednosti varijable	X(+a) – zbraja varijablu x s iznosom a X(-a) - oduzima iznos a od varijable x X(*a) – množi iznos a s varijablom x X(/a) – dijeli varijablu a s iznosom x
Korištenje nasumične vrijednosti za varijablu	X(=a,b) – postavlja varijablu x na nasumičnu vrijednost iz raspona [a,b] X(=a,b,c) – postavlja varijablu x na nasumičnu vrijednost iz raspona [a,b] koja je uzorkovana u intervalima iznosa c (na primjer X(=10,30,5) bi postavilo X na jednu od vrijednosti iz skupa (10,15,20,25,30) X(*2,3) – množi varijablu X s nasumičnim brojem iz raspona [2,3]

Tablica 8.1.2. – prikaz promjene parametara u gramatici fiktivne kornjače

8.2. Proširenje naredbi jezika L-sustava s fiktivnom kornjačom - parametrizirani sustavi

Operacije i sintakse	Značenje
Matematičke funkcije	
X(^2)	potenciranje variable x s potencijom 2
X(=2^3)	postavljanje varijable x na treću potenciju od dva
X(=sin(135)*2)	postavljanje varijable x na dvostruki iznos sinusa kuta od 135 stupnjeva
Nezavisne varijable	
VA(=5)	uvođenje varijable VA inicijalne vrijednosti 5
VB(=VA*3)	uvođenje varijable VB inicijalne vrijednosti trostrukog iznosa trenutne vrijednosti varijable VA
X(=VB)	postavljanje varijable X stanja kornjače

	na vrijednost nezavisne varijable VB
Kondicionalnost	
X(=If(Y>5,3,2))	ako je Y vrijednost veća od 5, X se postavlja na 3, inače na 2
VA(=if(>VB,VB,+1))	ako je VA veći od VB, postavlja se na vrijednost VB, inače se povećava za 1

8.3. Popis naredbi „CGA shape“ gramatike iz rada „Procedural modeling of buildings“ (Müller 2006.)

Pravila gramatike su oblika:

Identifikator: prethodnik : uvjet -> sljedbenik : vjerojatnost

Gdje je *Identifikator* jedinstveni za svako pravilo, *predhodnik* je nezavršni znak, *uvjet* je logički izraz koji mora biti istinit kako bi se pravilo moglo izabrati i izvršiti, *sljedbenik* je skup naredbi koje će kreirati nove ili promijeniti ovaj oblik, te ostaje *vjerojatnost* izbora ove desne strane danog pravila.

Naredba	Objašnjenje	Primjer
Osnovne transformacije	Mijenjaju podatke trenutnog oblika	
T(xa,ya,za)	Translacija trenutnog oblika za iznos xa po x osi, ya po y osi i za po z osi	T(10, 20, 30)
S(xa,ya,za)	Skaliranje trenutnog oblika za iznos xa po x osi, ya po y osi i za po z osi	S(10, 20, 30)
Rx(a)	Rotacija trenutnog oblika za kut a oko x osi	Rx(10)
Ry(a)	Rotacija trenutnog oblika za kut a oko x osi	Ry(20)
Rz(a)	Rotacija trenutnog oblika za kut a oko x osi	Rz(30)
Instanciranje	Stvaraju geometriju modela u sceni	
I(„ime_modela“)	Instanciranje modela ime_modela uz transaciju, rotaciju i skaliranje trenutnog oblika	I(„kocka“)
Podjele volumena	Dijele prostor trenutnog oblika na manje oblike	
Subdiv(os, iznosi...){simboli...}	Podjela prostora po osi na djelove navedenih veličina (apsolutnih i	Subdiv(„x“,1,2,3){A,B,A}

	relativnih)	
Repeat(os,iznos){simbol}	Podjela prostora po osi na jednolike djelove navedene veličine	Repeat(„x“,2){B}
Comp(tip, parametri){simboli...}	Pridruživanje simbola djelu geometrije danog tipa (poput strana, vrhova, bridova) i indeksa	Comp(„edge“,3){C} Comp(„faces“,2,3){D,E}
Dodavanje osi poravnanja	Dodaje os poravnanja u globalni rječnik	
Snap(os,ime)	Korištenjem centra trenutnog oblika postavlja os poravnanja zadanog imena	Snap(„XY“, „tilesnap“)
Provjera zaklanjanja	Provjerava je li trenutni oblik i kako je zaklonjen drugim oblicima	
Scope.occ(imeOblika)	Vraća rezultat provjere kao „nema preklapanja“, „djelomično“ i „potpuno preklapanje“. Ako se za ime oblika koristi „noparent“ provjeravaju se svi oblici koji nisu roditelji trenutnog, „all“ za sve, inače oblici danog imena	Scope.occ(„noparent“) == „none“ Scope.occ(„wall_east“) != „part“ Scope.occ(„floor“) == „full“

8.4. Evoluiranje vanjskih blokova građevina – struktura jezika

Jezik se temelji na navođenju jedne naredbe po tekstualnom redu zapisa strukture građevine. U danim primjerima indentacija je čisto estetične prirode. Jedna struktura sastoji se od samo jednog *Creation* i jednog *Sectors* bloka, koji su odgovorni za stvaranje i transformaciju temelja, tim redom.

Jezik se sastoji od blokova koji su omeđeni ključnim riječima *BEGIN* i *END*, gdje se nakon riječi *BEGIN* navodi naredba na koju se odnosi cijeli blok obuhvaćen tim parom riječi. Ključne riječi nakon begin su:

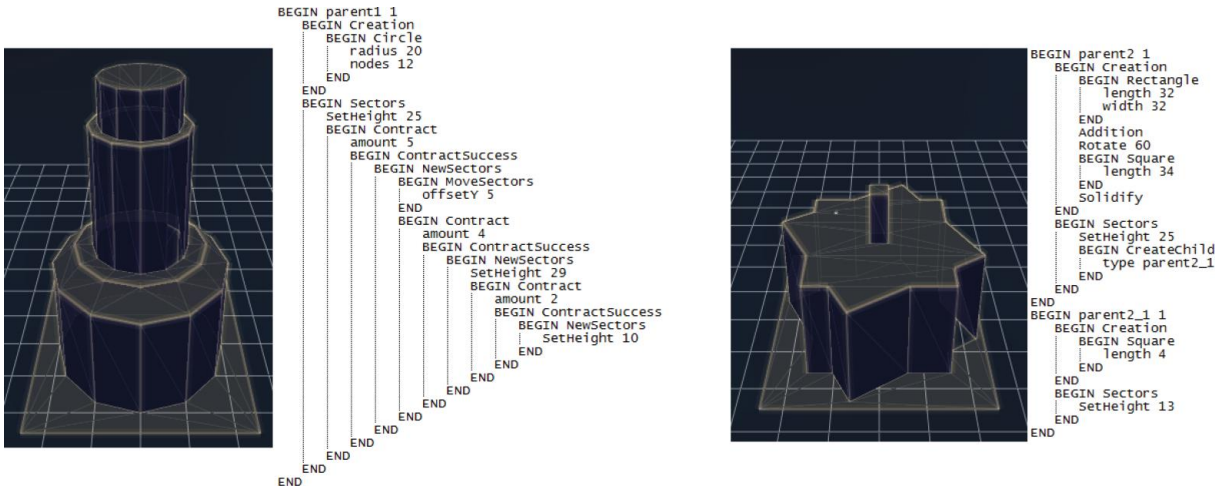
- *Creation* – početak bloka u kojem se stvara skup točaka koje čine temelj geometrije za daljnje transformacije i proširenja dimenzionalnosti
- *Circle*, *Rectangle*, *Square*, *VertexSet* i drugi – početak bloka koji definira planaran set točaka

- *Sectors* – početak bloka koji sadrži transformacije temelja
- *ContractSuccess* – početak bloka koji se izvodi ako je skaliranje bilo uspješno, bez presjeka putanja točaka pri skupljanju
- *NewSectors* – početak bloka koji kreira novi set točaka transformirajući trenutno aktivan set
- *MoveSectors, Contract, Translate* – početak bloka za transformaciju aktivnog seta točaka ili cijelog oblika

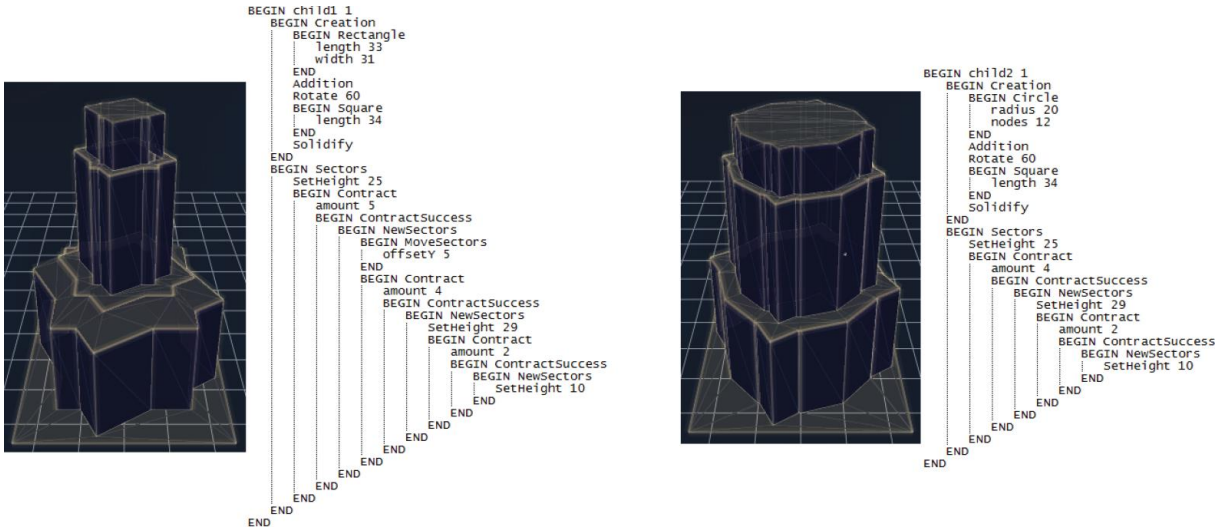
Riječi koje su navedene unutar drugih blokova:

- *Rotate* – naredba koja rotira cijeli dosad stvoreni oblik oko Y osi
- *Addition, Intersection, Subtraction* – CSG naredbe koja se izvršavaju između prethodnog i idućeg temeljnog oblika
- *Solidify* – primjena svih prethodno navedenih CSG naredbi kako bi se stvorio konačni set točaka
- *SetXY* i ostale nenavedene varijable– postavljanje varijable XY ili drugih parametara trenutne naredbe (bloka) na vrijednost danu u nastavku.

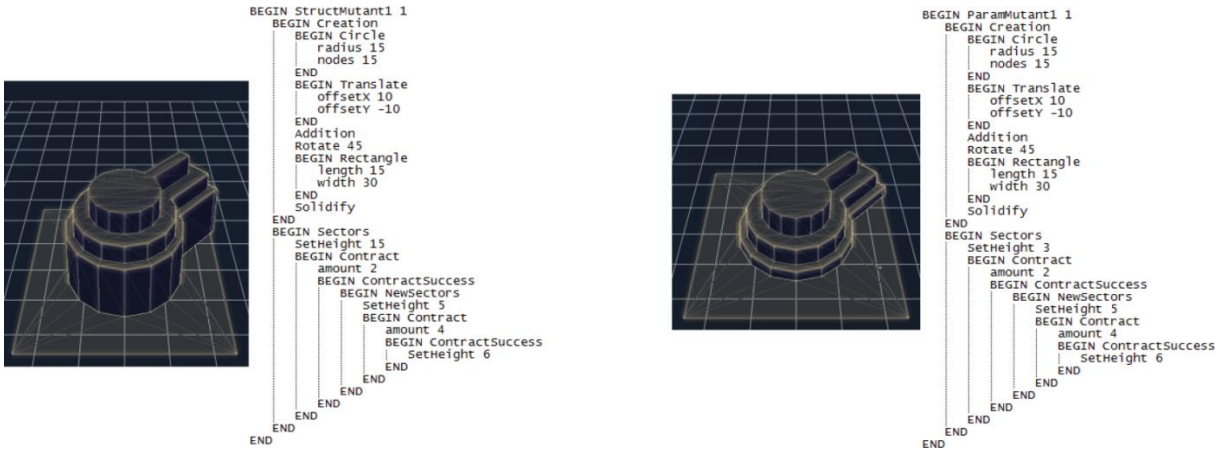
U nastavku su dane slike koje predstavljaju primjer dva građevine korištene kao roditelje (8.2.1.), dva djeteta nastala križanjem roditelja (8.2.2.), te dva djeteta nakon strukturne (8.2.3. lijevo) i parametarske mutacije (8.2.3. desno) (Martin, 2010.).



Slika 8.2.1. – prikaz modela i izvorne strukture dvaju roditeljskih građevina



Slika 8.2.2. – prikaz modela i izvorne strukture dvoje djece nastalih križanjem roditelja sa slike 8.2.1.



Slika 8.2.3. – prikaz modela i izvorne strukture dvoje djece, nakon strukturne (lijevo) i parametarske mutacije (desno)