SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 39

POSTUPCI PROCEDURALNOG MODELIRANJA GRADOVA

Hrvoje Ribičić

Zagreb, lipanj 2008.

Sadržaj

Uvod

U prošlosti su glavna ograničenja računalne grafike bili zahtjevni izračuni koji su ograničavali kvalitetu i detalje modela koje se moglo prikazati. Napredak računalne grafike je kroz razvoj sklopovlja i algoritama omogućio prikazivanje scena iznimno velike složenosti, ali je to stvorilo i nove probleme. Detalji koji su vidljivi na scenama zahtijevaju mnogo uloženog truda i vremena sa strane umjetnika koji ih stvaraju, te je postalo nužno automatizirati bar dio tog procesa. Proceduralna generacija je postupak kojim je moguće stvoriti mnogo različitih objekata iste vrste, ali je njezino ograničenje vrsta objekata koji se mogu modelirati. Stvaranje ili rast modeliranih objekata moraju se odvijati prema nekim zakonima koje je moguće oponašati matematičkim modelom. Uspješnost ovog postupka je vidljiva na primjerima generacije biljaka [1] ili terena [2].

Makar gradovi naoko ne podliježu nikakvim zakonima koje je jednostavno opisati ili implementirati, arhitekt Christopher Alexander je u svojem djelu „A Pattern Language“ [3] iznio niz sličnosti koje je zapazio tijekom svoje karijere i nazvao ih oblikovnim obrascima. Svaki oblikovni obrazac opisuje neko pravilo dizajna i kako ga, zašto i kada treba primjenjivati. Upravo iz oblikovnih obrazaca dolazi ideja da je moguće odrediti set pravila koja, poput grananja za biljke, određuju kako će se grad razvijati.

U radu Parisha i Müllera „Procedural Modeling of Cities“ izdanom 2001. godine predstavljen je način generiranja virtualnih gradova temeljen na proširenim Lindenmayerovim sustavima (L-sustavima) i Alexanderovom radu koji kao ulazne podatke koristi visinske mape i podatke o gustoći naseljenosti. Cilj generacije je uvjerljiv grad opisan pomoću mreže cesta, modela zgrada i tekstura fasada. Makar su u tom radu opisani principi po kojima su dobiveni rezultati, nedostaju mnogi koraci potrebni za reprodukciju rezultata. Ovaj rad je reprodukcija postupka generacije mreže cesta koji je u originalnom radu dovoljno opširno opisan da bi pokušaj bio moguć.

Pregled područja

Rad koji je jako bitan za originalno ostvarenje je [1], budući da su u njemu definirani otvoreni L-sustavi koji su teoretska osnova procesa grananja. Osim toga, u tom se radu po prvi put u generaciji biljaka dodaje utjecaj okoliša – drugih biljaka i vanjskih faktora poput osvjetljenja, princip koji je vrlo sličan onom korištenom u generaciji gradova.

Autori originalnog rada su nastavili nadograđivati svoj rad, što je rezultiralo programom CityEngine [4], koji postiže jako impresivne rezultate. Još jedna primjena koncepata iz originalnog rada je igra Subversion tvrtke Introversion, čije je impresivne rezultate moguće vidjeti na [5].

1. Ulazni podaci

1. 1. Izbor ulaznih podataka

Proces stvaranja grada je jako složen proces podložan mnogim utjecajima koji se mijenjaju kroz vrijeme. Povijesne okolnosti utječu na priljev ljudi u gradove, koji mijenja način na koji su građeni dijelovi grada u kojima stanuju novopridošlice. Tehnologija i klima utječu na poljoprivredne i industrijske djelatnosti, a sektori u kojima su one smještene su strukturalno različiti od rezidencijalnih četvrti. Očito je da bi modeliranje grada sa svim ovim utjecajima na umu bilo vrlo teško zbog složenosti modela i nepraktično zbog velike količine podataka koje bi trebalo dostaviti algoritmu za modeliranje. Smanjenje skupa utjecaja je moguće, ali se postavlja pitanje kako izabrati kriterije koji će dati uvjerljiv rezultat.

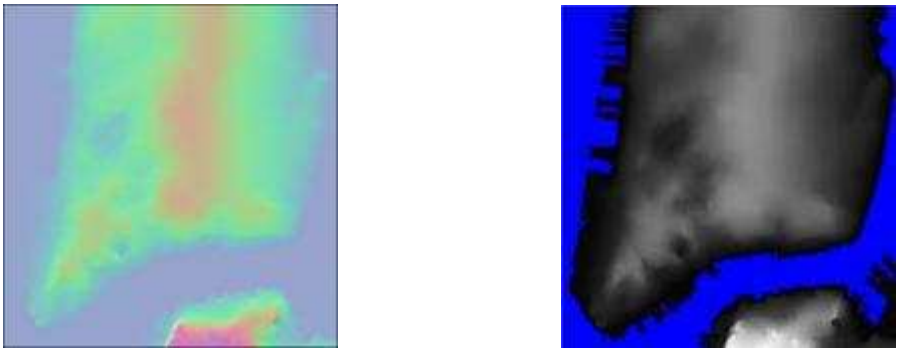
Odgovor na to pitanje je da je umjesto navođenja utjecaja i dobivanja rezultata koji odgovara utjecajima poželjnije zadati parametre koji opisuju svojstva željenih rezultata. Idealan parametar za tu svrhu je mapa gustoće stanovništva, slika koja opisuje varijacije u gustoći stanovnika promjenama u boji. Uz nju, potrebna je i visinska mapa koja sadrži podatke o vodi i terenu na kojem se generira virtualni grad. Podaci mogu biti temeljeni na stvarnim gradova, ali je prikupljanje podataka o gustoći stanovništva problematična jer su oni dostupni samo za najveće gradove.

1. 2. Visinska mapa

U visinskoj mapi su sadržana dva zasebna ali važna utjecaja. Prisutnost vode ima velik utjecaj na formiranje grada, budući da su mnoge ekonomske aktivnosti vezane uz nju, što utječe na način na koji su građene ceste u blizini vode. Drugi utjecaj je nagib terena, koji ograničava područja na kojima je moguće graditi ceste i čini isplativijim izgradnju prometnih cesta na manjim nagibima.

Prikladnu visinsku mapu je moguće dobiti jednom od metoda proceduralne generacije terena, ali ako nije potrebno koristiti slučajno napravljen teren, bolje je koristiti

podatke uzete iz prirode. Na slikama 1.1. a) i b) su primjeri visinskih mapa preuzetih sa stranica USGS-a [6] prije nego što su bili prilagođeni potrebama rada i poslije.



a) b)

Slika 1.1 a) visinska mapa u originalnom obliku i
b) visinska mapa s dodanom vodom i paletom boja koji omogućava razlikovanje vode i tla

1. 3. Mapa gustoće stanovnika

Gustoća stanovnika je jedan od boljih indikatora kako će mreža cesta izgledati, budući da su prometna čvorišta i velika gustoća naseljenosti inherentno povezani. Ona nije savršen pokazatelj jer postoje područja koja zbog svoje funkcije (trgovina, uprava, industrija etc.) moraju biti dobro prometno povezana, ali u njima nema mnogo stanovnika. Ukoliko se zadaje proizvoljna mapa gustoće, to nije problem, i zato je preporučljivo raditi s podacima koji nisu napravljeni na temelju pravih gradova ili ih modificirati kako bi se uklonili takvi nedostaci.

Izrada podataka na temelju stvarnih vrijednosti nije jednostavna iz nekoliko razloga. Mnogi gradovi ne objavljuju podatke dobivene popisima stanovništva, ili ih ne povezuju s prostornim rasporedom. Uz to, metodologija skupljanja je takva da se gustoća stanovništva mjeri po organizacijskim jedinicama, što kao rezultat ima mapu u kojoj su granice oštre i koja ne pogoduje metodi generacije koja se koristi. Na slici 1. 2. a) je prikazana stranica iz izvještaja o gustoći stanovništva u New York-u [7]. Nakon uklanjanja nepotrebnog teksta i razdvajanja područja različitih gustoća stanovništva po bojama, nastala je mapa gustoće stanovništva prikazana na 1. 2. b). Problem sa ovim zapisom je potreba za legendom koja objašnjava značenje pojedinih boja i prevladavajući oštri oblici. Korištenjem filtera koji stvaraju smetnje na rubovima jasnih oblika i

dodavanjem Gaussovog šuma stvorena je poboljšana verzija te mape, u kojoj tonovi sive boje označavaju gustoću stanovništva, vidljiva na slici 1. 2. c).

a) b) c)

Slika 1.2 Mapa gustoće stanovništva u nastajanju: a) originalna mapa b) očišćena mapa s legendom c) mapa s dodanim šumom i prilagođenom skalom boja.

2. Model sustava i teorijska pozadina

2. 1. Lindenmayerovi sustavi

Kao osnova sustava za generaciju gradova koriste se Lindenmayerovi sustavi ili skraćeno L-sustavi. Osmislio ih je biolog Aristid Lindenmayer 1968. kao alat kojim je moguće opisati rast i razvoj jednostavnih višestaničnih organizama i odnose između stanica biljaka. Najmanji dio L-sustava su znakovi koji sačinjavaju nizove. Niz znakova predstavlja jedno stanje L-sustava, iz kojeg se prelazi u sljedeće stanje primjenom najvećeg mogućeg broja pravila transformacije kako bi se dobio novi niz. Pravila pretvaraju jedan znak u drugi znak ili niz znakova, a mogu ovisiti i o susjednim znakovima. Početno stanje L-sustava naziva se aksiomom i označava se grčkim slovom ω.

Način na koji teče proces generacije je najlakše vidjeti na primjeru jednostavnog L-sustava koji u koraku g_n generira broj znakova jednak n-tom Fibonnacijevom broju.

ω: a p_1: a → b p_2: b → a b

g_0: a g_2: a b g_4: a b b a b
g_1: b g_3: b a b g_5: b a b a b b a b

Postoji velik broj fraktala i zanimljivih matematičkih nizova koje je moguće generirati koristeći L-sustave, ali je od posebne zanimljivosti za ovaj rad model interakcije biljaka s okolišem izveden L-sustavima napravljen u [1]. Modeliranje biljaka L-sustavima je prvi put detaljno opisano u [8], gdje je rezultat rada L-sustava bio skup naredba čija je sintaksa bila slična programskom jeziku Logo i čijim se izvršavanjem dobio crtež biljke. Takav zapis je koristan za direktno iscrtavanje, ali nije primjeren za složenije postupke u kojima je važna interakcija s okolišem i položaj već stvorenih dijelova biljke. Ovo ograničenje je u [1] bilo nadjačano dodavanjem posebnih znakova – upita.

Upiti od svojih susjeda prilikom stvaranja primaju parametre i u ovisnosti o znaku koji obilježava upit vraćaju vrijednost koja određuje sljedeće primijenjeno pravilo. Značaj upita je njihova komunikacija s „okolišem“ - entitetom koji sadrži parametre poput osvjetljenja ili prisutnosti vode i putem upita saznaje i pamti informacije o novostvorenim biljkama. Postojanje okoliša omogućava složene interakcije između biljaka poput natjecanja za svjetlost i vodu.

Prošireni model uveden u tom radu zove se otvoreni L-sustav i donosi nekoliko promjena. Moguće je da znakovi sadrže atribute koji se navode u zagradama nakon imena znaka. Atributi mogu poprimiti proizvoljne vrijednosti, ali svi znakovi iste vrste moraju imati isti tip i broj atributa. Upiti se označavaju dodavanjem upitnika ispred imena upita.
Sintaksa pravila kojim se prethodnik zamjenjuje s nasljednikom je sljedeća:

lijevi kontekst < *prethodnik* > *desni kontekst* : *uvjet* → *nasljednik* : *vjerojatnost*

Ukoliko su lijevi i/ili desni kontest prisutni u pravilu, u trenutnom stanju moraju biti neposredni susjedi znaka prethodnik s lijeve/desne strane da bi se pravilo izvršilo. Uvjet je logički izraz u kojem se mogu navesti atributi koji pripadaju prethodniku ili nekom od članova konteksta. Pravilo se može izvesti samo ako je uvjet istinit. Nasljednik je znak ili skup znakova čiji su atributi konstante ili atributi koji se pojavljuju s lijeve strane pravila.

Vjerojatnost služi za stohastičko modeliranje, i označava vjerojatnost da će se određeno pravilo izvršiti. Zbroj vjerojatnosti pravila koja imaju jednaku lijevu stranu mora biti jednak 1. U nestohastičkom deterministčkom modeliranju se vjerojatnosti ne navode jer po definiciji imaju vrijednost 1.

Važnost otvorenih L-sustava u modeliranju gradova je upravo u konceptu okoliša koji omogućuje izgradnju modela u kojem odluke o dodavanju križanja i produljivanja cesta donosimo samo na temelju trenutačne ceste, a na temelju položaja drugih cesta i parametara vršimo korekcije ili brisanje novostvorenih cesta.

2. 2. Osnovni elementi modela

Glavni i jedini elementi modela su ceste, koje dijelimo na dvije vrste po funkcijama koje obavljaju:

1. Prometnice (eng. highways)

Zadaća prometnica je povezivanje centara naseljenosti, i tu kategoriju prvenstveno čine ceste namijenjene za veće količine prometa. Prometnice sačinjavaju kostur mreže cesta, i jako je bitno da one budu dobro raspoređene. Faktor koji najviše utječe na njihov raspored je gustoća stanovništva.

2. Ulice (eng. streets)

Ulice se nastavljaju na prometnice i daju nastanjenim područjima izlaz na prometnice. Kod rasporeda ulica glavni faktor nije raspored gustoće stanovništva, jer je mnogo bitniji prevladavajući stil i odnos sa bliskim prometnicama.

Generacija grada počinje s jednom unaprijed zadanom prometnicom i nastavlja se dodavanjem novih cesta. Nove ceste se stvaraju grananjem starih - dodavanjem od jedne do tri ceste kandidata koje počinju gdje originalna cesta završava. Ukoliko su ceste kandidati legalno postavljene, prilagođavaju se drugim cestama u blizini promjenom smjera i dužine i/ili razdvajanjem na više malih cesta. Time njihovo postavljanje postaje

konačno. Proces grananja se prekida ukoliko je cesta u fazi prilagodbe spojena na već postojeću mrežu cesta ili su svi kandidati ilegalno postavljeni.

Zbog razlike u funkciji prometnica i ulica generacija je podijeljena na dvije faze. U prvoj se fazi generira mreža prometnica, i određuju kandidati za ulice. U drugoj fazi se postupno aktiviraju ulice kandidati i obavlja se njihovo grananje, koje traje sve do susreta s nekom od prometnica. U originalnom radu dvije faze generacije nisu bile striktno razdvojene, već implicitno putem velike odgode grananja ulica. Pod pretpostavkom da će grananje prometnica u određenom dijelu mape završiti prije nego što istekne dovoljan broj koraka, rezultati su identični. Nažalost, ispostavilo se da ta pretpostavka nije sasvim točna, što je rezultiralo blokovima ulica koji su naknadno bili presiječeni prometnicama. Neprirodan izgled tako stvorenih križanja je uzrokovao promjenu modela.

2. 3. Model kao L-sustav

Implementacija navedenog modela putem otvorenih L-sustava je moguća, ali ona nije u stanju oponašati sve dijelove modela. Izbor kandidata je složen problem koji je skoro pa nemoguće opisati koristeći formalna pravila L-gramatike zbog njegove kompleksnosti. Upravo zato u gramatici ovog L-sustava postoji posebno pravilo koje enkapsulira grananje, ali ne odgovara formalnom modelu L-sustava. Sva ostala svojstva modela moguće je opisati sljedećom gramatikom:

ω: `R(0) ?I(initRoadAttr, UNASSIGNED)`

p_1: `R(del) : del < 0` → ε

p_2: `R(del) > ?I(roadAttr, state) : state == SUCCESS` →

{

Poziva vanjsku funkciju koja na temelju `roadAttr` stvara skup cesta kandidata u obliku novih `roadAttr` koji ih opisuju. Generira niz znakova `B(1, roadAttr)` za svaku predloženu cestu istog tipa osim zadnje kojoj se pridružuju znakovi `R(0) ?I(roadAttr, UNASSIGNED)`. Ukoliko je cesta čiji se kandidati generiraju prometnica, stvara se i niz znakova `B(HIBERNATION + delay, roadAttr)` koji opisuju ulice koje će nakon kraja generacije prometnica biti aktivirane nakon slučajno odabranog broja koraka delay. `HIBERNATION` je dovoljno velika konstanta (u implementaciji iznosi 1000). }

p_3: `R(del) > ?I(roadAttr, state) : state == FAILURE → ε`

p_4: `B(del, roadAttr) : del > 0 && del < HIBERNATION → B(del - 1, roadAttr)`

p_5: `B(del, roadAttr) : del == 0 → R(del) ?I(roadAttr, UNASSIGNED)`

p_6: `B(del, roadAttr) : del < 0 → ε`

p_7: `R(del) < ?I(roadAttr, state) : del < 0 → ε`

p_8: `?I(roadAttr, state) : state == UNASSIGNED →` {prilagođavanje okolnim cestama i provjera legalnosti} `?I(newRoadAttr, newState)`

p_9: `?I(roadAttr, state) : state != UNASSIGNED → ε`

p_{10}: `B(del, roadAttr) : del >= HIBERNATION && highwayGenerationComplete == true → B(del - HIBERNATION, roadAttr)`

U abededi sustava su tri znaka:
`R(delay)` - simbolička oznaka za cestu čija će legalnost biti provjerena za `delay` koraka;
`?I(roadAttr, state)` - upit je li cesta na položaju `roadAttr` dobro postavljena, `state` označava uspješnost, a moguće promjene su zapisane na mjesto starih podataka o cesti;
`B(delay, roadAttr)` - znak koji označava ogranak ceste i služi razdvajanju faza generacije.

Aksiom sustava je početna prometnica, i nakon provjere je li legalno postavljena započinje grananje. Pravilo p_2 je ranije spomenuto posebno pravilo za grananje koje generira proizvoljan broj znakova. Pravilo p_8 u sebi sadrži provjeru legalnosti i prilagodbu sakrivenu u pozivu upita. Pravila p_4 i p_{10} se brinu za odvojenost faza generacije, pri čemu se u p_{10} koristi varijabla `highwayGenerationComplete`. Ukoliko bi se odlučili za sasvim formalan zapis L-sustava, provjera te varijable zahtijevala bi korištenje upita, ali je zbog jednostavnosti gramatike zadržan ovakav pristup. Sva ostala pravila brinu se za pretvaranje i uništavanje znakova koji se obrađuju.

Funkcionalnost modela je pomoću L-sustava razdvojena u nekoliko dijelova:
1. Pozivanje grananja i briga o mreži cesta.
2. Stvaranje cesta kandidata.
3. Provjera legalnosti i prilagodba okolnim cestama.

O prvom dijelu se brine sam L-sustav, dok se dijelovi 2 i 3 trebaju izvesti zasebnim funkcijama. U drugom dijelu se grananjem pokušavaju ostvariti globalni ciljevi neovisno o okolnim cestama. Suprotno tome, primjereno ime za treći dio je lokalna prilagodba, budući da mijenja parametre cesta tako da se bolje poklapaju sa svojim okruženjem.

3. Globalni ciljevi i grananje

3. 1. Izbor cesta za grananje

Pri svakom grananju potrebno je stvoriti jednu ili više cesta kandidata koje nastavljaju proces generacije. Za svaku se cestu na njezinom kraju određuju područja u kojima je dozvoljeno stvoriti novu cestu. Primjer dozvoljenih prostora za postavljanje cesta označen je na slici 3. 1. svijetlo plavom bojom. Unutar tih prostora se postavljaju probne ceste slučajno odabrane duljine i pronalazi se najpovoljniji smjer s obzirom na vrijednosti

Slika 3. 1 Primjer dozvoljenih područja grananja

funkcija koje opisuju utjecaj gustoće stanovništva i nagiba terena u slučaju prometnica ili izgled susjednih ulica u slučaju ulica. Na temelju odnosa najboljih vrijednosti se odlučuje koje će ceste biti izabrane kao ceste kandidati.

3. 2. Utjecaj gustoće stanovništva

Gustoća stanovništva ima najveći utjecaj na prometnice, dok je njen utjecaj na ulice zanemariv. Osnovni cilj koji generator pokušava ostvariti postavljanjem prometnica je

povezivanje centara populacije. Najveći problem prilikom ostvarenja tog cilja je to što je njegov opis neprikladan za direktnu primjenu kao kriterij prilikom grananja. Pronalaženje centara populacije je trivijalan problem za ljudski vid, ali je prva prepreka koju generator mora savladati.

Pretpostavljajući raspodjelu gustoće stanovništva koja nema oštre prijelaze, vjerojatno je da će ceste kandidati koje imaju najveći porast gustoće stanovništva na svojem putu voditi prema područjima najveće naseljenosti. Funkcija koja evaluira ceste uzorkuje gustoću stanovništva i vrijednosti dobivene uzorkovanjem podijeljene s udaljenošću od mjesta gdje su uzete do početka ceste pribrajaju vrijednosti funkcije. Broj točaka i udaljenost između točaka ovise o dužini najkraće promatrane ceste i o predefiniranim konstantama. Ukoliko je uzorkovanje loše napravljeno, moguće je da kratke ili duge ceste imaju zanemarivu prednost koja će se očitovati postepenim skraćivanjem ili produživanjem prosječne duljine ceste.

Još jedno važno pitanje za grananje je koliko će se cesta kandidata prihvatiti. Prema ranije navedenim kriterijima centri populacije su mjesta kojima teže prometnice, i zato je očekivano da će ona postati križanja. U grananju se taj efekt emulira prihvaćanjem više cesta ako su njihove vrijednosti približno jednake, što je očekivano upravo u lokalnim centrima populacije.

3. 3. Utjecaj nagiba

Nagib je kao i gustoća stanovništva važan faktor za prometnice, ali ne na isti način kao i gustoća. Gustoća je kao kriterij neophodna, a nagib može samo smanjiti njezin utjecaj zbog težine izgradnje ceste. Funkcija koja opisuje utjecaj nagiba izabire niz točaka na cesti i računa nagib između parova susjednih točaka. Zbroj prosječnog i maksimalnog nagiba pomnožen s konstantnim težinskim faktorom je iznos u postocima za koji će vrijednost funkcije koja opisuje utjecaj gustoće naseljenosti biti umanjena. Na taj način nagib može utjecati na to koja će cesta biti dodana bez mogućnosti pojave slučaja u kojemu odabir isključivo ovisi o nagibu.

3. 4. Grananje ulica

Za razliku od grananja prometnica, grananje ulica ima većinom estetski faktor. Prilikom stvaranja ulica vrijedi da su počeci ulica na prometnici na kojoj su stvorene jednako udaljeni i da su sve ulice koje izviru s iste strane prometnice jednake dužine.
Grananje pokušava očuvati takav izgled blokova ograničavanjem smjerova u kojima se ulice šire i prilagođavanjem dužina ulica. Uz svaku stvorenu ulicu je pridružen omjer, veličina koja govori koliki je odnos između širine i dužine bloka koji ulice formiraju. Ulice stvorene ispred zadržavaju istu veličinu i omjer kao i originalna, dok one na stranama imaju veličinu jednaku originalnoj pomnoženoj s omjerom, a njihov omjer je recipročan originalnome. Na taj se način prilikom grananja očuvava odnos između širine i dužine blokova.

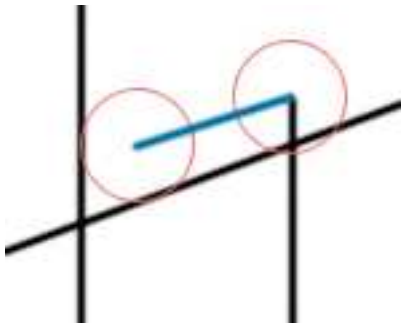
4. Lokalne prilagodbe

4. 1. Utjecaj vode

U slučaju da je dio ceste postavljen na terenu označenom kao voda, cestu je potrebno uništiti ili prepraviti tako da se ne sudara s vodom. Prepravljanje se radi samo za prometnice i sastoji se od određivanja smjera s najvećom udaljenošću od početka ceste do pojavljivanja vode i prepravljanja prometnice tako da slijedi taj smjer i zaustavlja se pred vodom. Na taj način se dobivaju ceste koje prate oblik obale, što je česta pojava u gradovima s izlazom na vodu. Ulice se ne prepravljaju na ovaj način jer bi male varijacije nastale zbog kvalitete ulaznih podataka imale prevelik učinak.

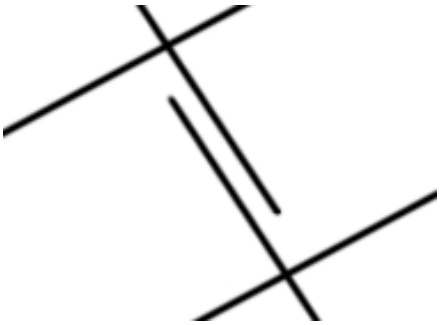
4. 2. Brisanje sličnih cesta

Postupak generacije nije ni blizu savršenog te je potreban set korekcija koji se brine da nastala mreža cesta odgovara izgledu pravog grada. Najosnovnija korekcija je brisanje suvišnih cesta. Ukoliko su oba kraja ceste dovoljno blizu neke već postojeće ceste, novonastala cesta je suvišna. Primjer takve situacije se može vidjeti na slici 4. 1 na kojoj je prikazana cesta koju je potrebno izbrisati.



Sl. 4.1 Novostvorena cesta, označena plavom bojom, preblizu je cesti ispod nje, što opisuju krugovi koji označavaju radijus provjere.

Problem s ovakvim brisanjem je nedosljednost koju to može uzrokovati. Na slici nakon brisanja ceste označene plavom bojom ostaje dio ceste koji se ne nastavlja. Makar se takve ceste javljaju u gradovima, nisu jako učestale. Iako se čini da bi jednostavno rješenje bilo brisanje dijela koji nije prikladan, to mijenja prirodu funkcije lokalne prilagodbe i povećava njezinu složenost jer mora utvrđivati kako su ostale ceste povezane. Primjer situacije u kojoj funkcija neće dobro reagirati nalazi se na slici 4. 2.



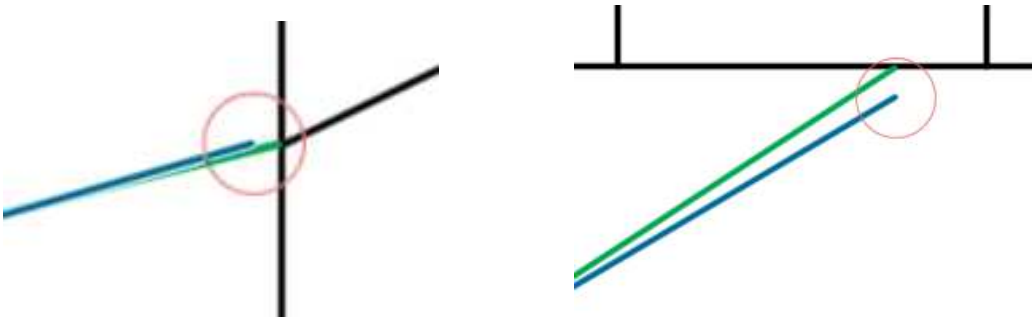
Sl. 4. 2 Primjer slučaja koji je teško popraviti

U ovakvoj situaciji, funkcija nije u stanju prepoznati potrebu za spajanjem ove dvije ceste bez da primijeni kriterije koji bi uzrokovali spajanje i u sličnim, ali ispravnim situacijama. Najbolja korekcija bi uključivala promjenu obje ceste, što funkcija ne radi zbog mogućnosti stvaranja većih problema u izgledu grada.

4. 3. Nadovezivanje na postojeće ceste

Proces grananja završava kad se novonastala cesta spoji na neku od već postojećih. Nadovezivanje ima velik utjecaj na izgled mreže cesta i zato postoje mnoge provjere koje osiguravaju da će se ono obaviti na najbolji mogući način. Postupci nadovezivanja su isti za prometnice i ulice, ali se razlikuju udaljenosti unutar kojih je dozvoljeno primijeniti te postupke.

Najjednostavnije nadovezivanje se sastoji u pripajanju novonastale ceste križanju kao što je to prikazano na slici 4. 3. a). Ukoliko nema križanja unutar dozvoljene udaljenosti od kraja novonastale ceste, pokušava se pronaći dovoljno bliska cesta na koju će novonastala biti povezana, primjer na slici 4. 3. b). Ako svi postupci nadovezivanja propadnu, cesta se ne mijenja i kreće faza određivanja presjecišta.



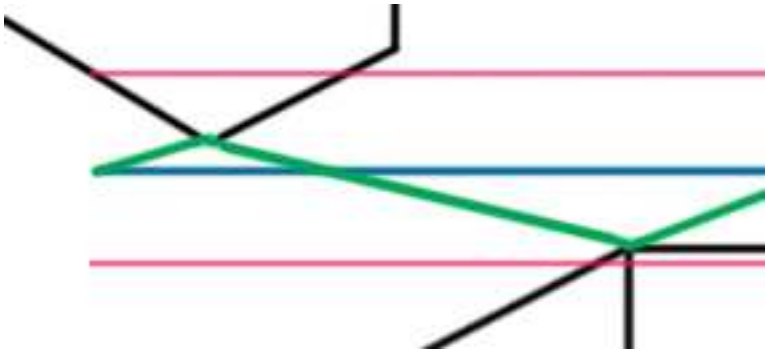
a) b)

Sl. 4. 3 Primjeri nadovezivanja cesta a) na križanje b) na cestu
Cesta prije pripajanja je označena plavom bojom, a nakon zelenom.

4. 4. Prilagođavanje križanjima na putu

Sve ranije navedene metode popravljanja mijenjaju izgled ceste pomicanjem krajnje točke, što je dobar postupak za kratke ceste. U slučaju dugih cesta, ponekad je potrebno raditi promjene koje prilagođavaju cestu križanjima i cestama koje sreće na putu. Za takve slučajeve se primjenjuje postupak prikazan na slici 4. 4 koji razbija cestu u niz segmenata koji odgovaraju putanji. Originalna cesta je označena plavom bojom, a crvenom bojom je na slici označena udaljenost unutar koje se promatraju križanja i završeci cesta.

Ukoliko je neki kraj ceste dovoljno blizu, cesta se razbija na fragmente koji čine izlomljenu liniju koja počinje i završava na istim mjestima kao i originalna cesta.
Taj postupak se ponavlja za svaki od fragmenata, s ograničenjem da nije dozvoljeno koristiti kraj ceste koji je već bio upotrijebljen u bilo kojem dijelu procesa. Primjer izgleda ceste nakon što je proces završen označen je zelenom bojom na slici 4. 4.



Slika 4. 4 Primjer prilagodbe dužih cesta

4. 5. Presijecanje cesta

Nakon što su obavljene sve prilagodbe koje je moguće napraviti, stvara se lista cesta koje se sijeku s cestom koja se obrađuje. Svaka od tih cesta se razdvaja na dvije zasebne ceste za potrebe kasnijih izračuna, dok se cesta koja se obrađuje siječe onoliko puta koliko je potrebno. Segment ceste koji je najbliži kraju se posebno obrađuje kako bi se utvrdilo je li njegova veličina manja od najmanje dopuštene. U slučaju da je, segment se briše i zaustavlja se daljnje grananje.

5. Implementacija

5. 1. Korištene tehnologije

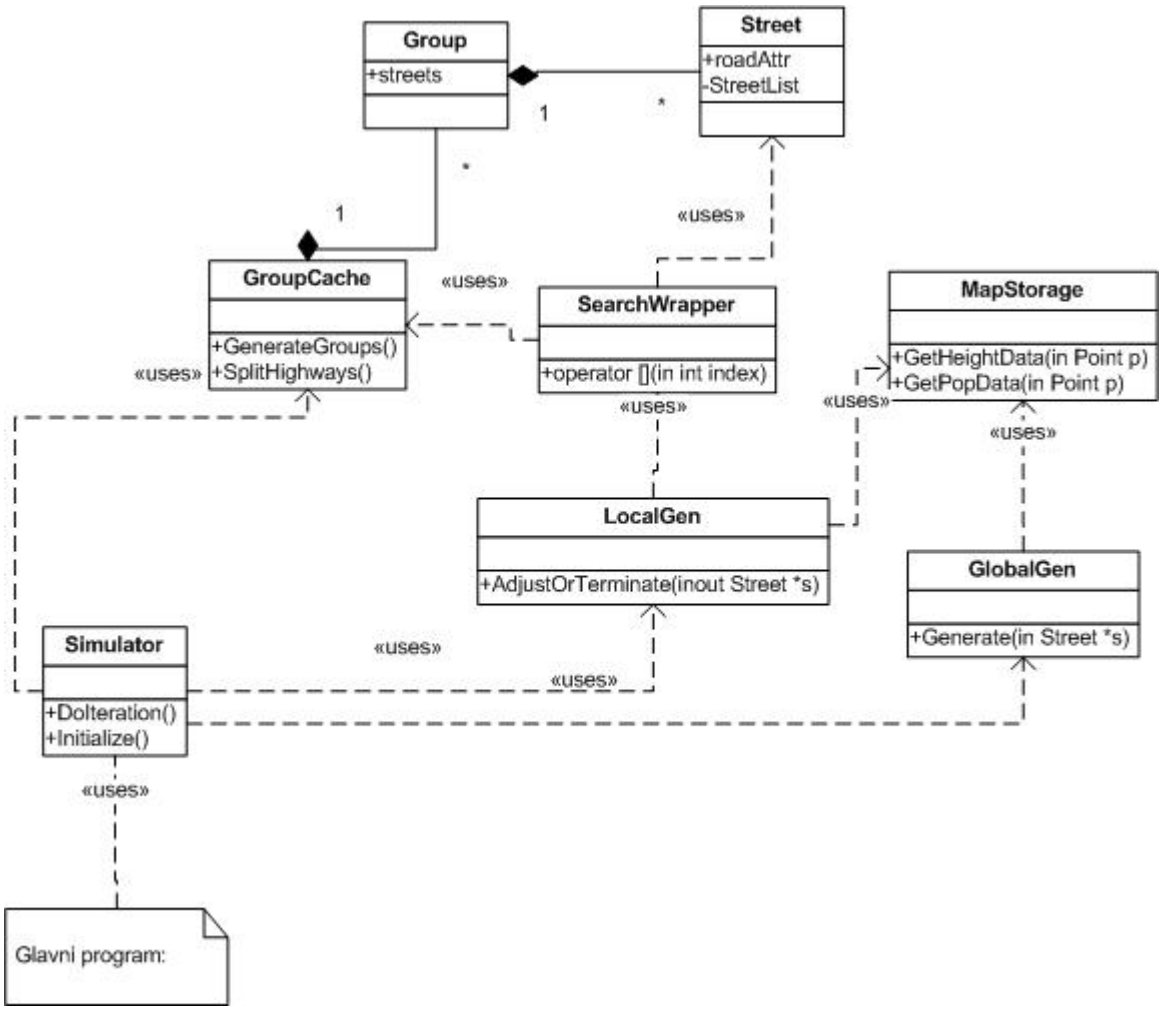
Program CityGen je napisan u programskom jeziku C++ i preveden koristeći Microsoftov Visual Studio 2003. Za iscrtavanje rezultata je korišten OpenGL API, a za učitavanje ulaznih podataka biblioteka Magick++.

OpenGL je biblioteka funkcija korištena pri iscrtavanju 2D i 3D objekata. Stvorio ju je Silicon Graphics 1992., a njezin razvoj i proširivanje su nastavile mnoge druge tvrtke. Njezina namjena je omogućavanje programiranja neovisnog o specifičnostima grafičkog sklopovlja koje se koristi za izračunavanje i prikazivanje rezultata. Programer koristi isti set funkcija neovisno o operacijskom sustavu za koji piše program, ali se implementacija tih funkcija razlikuje tako da najbolje iskoristi mogućnosti operacijskog sustava i sklopovlja. Ukoliko sklopovlje ne podržava funkcije koje se koriste, OpenGL ih emulira. Za ovaj rad je odabran zbog jednostavnosti korištenja. [9]

ImageMagick je programski paket namijenjen manipulaciji slika koji podržava velik broj različitih formata i naprednih funkcija. Moguće ga je koristiti samostalno, ali i kao ekstenziju programskih jezika putem biblioteka za pojedine jezike. Magick++ je ImageMagick biblioteka za C++ koja sadrži funkcije i tipove podataka napravljene za pristup slikovnih datoteka.

5. 2. Arhitektura

Prilikom izrade programa korišten je objektno orijentiran pristup, ali nije primjenjen u cijelosti. Primjerice, klasu koja predstavlja ceste bi bilo moguće implementirati kao apstraktnu klasu koju nasljeđuju ulice i prometnice, ali to nije učinjeno zbog vrlo čestog korištenja te klase koje bi dovelo do smanjenja efikasnosti. Načelo koje je najbolje primjenjeno je enkapsulacija, budući da se klase većinom koriste kako bi sakrile detalje vezane uz korištenje vanjskih biblioteka ili specifične izračune. Najvažnije klase, metode i njihovi parametri su prikazani na klasnom UML dijagramu na slici 5. 1.



Slika 5.1 Klasni UML dijagram

Najvažnija klasa je `Simulator`, koja je ime dobila po činjenici da simulira L-sustav opisan u poglavlju 2.3. Funkcionalnost te klase je smještena u statičkim funkcijama, budući da nije praktično paralelno generirati više gradova. Glavni program inicijalizira klasu pomoću metode `Initialize`, koja kao parametar prima prometnicu iz koje počinje grananje. Ta metoda postavlja aksiom i ostale interne varijable bitne za rad L-sustava. Nakon poziva metode `Initialize` dozvoljeno je pozvati metodu `DoIteration` koja primjenjuje pravila na niz znakova L-sustava koji je trenutačno zapamćen. Interno se koriste dva spremnika znakova koji izmjenjuju uloge kao trenutno i sljedeće stanje. `DoIteration` u ovisnosti o pravilu koje primijeni poziva metode klasa `GlobalGen` ili `LocalGen` i na temelju njihovih rezultata dodaje nove znakove ili briše stare. Kad je iteracija završena, `DoIteration` provjerava je li stanje u kojem se našao L-sustav takvo da daljnja primjena pravila ne mijenja stanje sustava. Ukoliko je, to označava kraj faze koji se javlja glavnom programu putem povratne vrijednosti metode `DoIteration`. Kraj

stvaranja mreže prometnica uzrokuje poziv metode `GenerateGroups` klase `GroupCache`, koja stvara grupe namijenjene ubrzavanju lokalne prilagodbe, o kojima će biti rečeno više u poglavlju 5.3.

Podaci o cestama pamte se u klasi `Street`, koja ima dvije namjene. Objekti te klase sadrže podatke o cesti poput početka, kraja, je li ulica ili prometnica, etc. dok statički element `StreetList` sadrži podatke o svim cestama koje su uspješno završile fazu lokalne prilagodbe i postale dio mreže cesta. Klasa `Simulator` koristi statičku metodu `Report` kako bi prijavila da je neka instanca klase `Street` dodana u mrežu cesta. Veliku važnost imaju i metode koje se koriste za određivanje odnosa među cestama i prepravljanje cesta. Za svaku vrstu transformacije postoji posebna metoda kako bi se osiguralo da metode klasa `LocalGen` i `GlobalGen` ne ovise o internom zapisu podataka klase `Street`. Među atributima klase se nalaze unaprijed izračunati parametri koji se koriste u metodama kako bi se ubrzalo izvođenje izračuna.

Klasa `GlobalGen` stvara ceste kandidate u ovisnosti o vrsti ceste i pravilu koje se koristi. Pravilo određuje u kojim se područjima traže ceste koje služe kao nastavci. Jedina javna metoda je `Generate`, koja u spremnike koje prima kao parametre dostavlja pokazivače na instance klase `Street` koje opisuju ceste kandidate. Funkcije za uzorkovanje podatke dobivaju pozivima metoda klase `MapStorage`. Sva funkcionalnost u klasi `GlobalGen` je statička jer je ona namijenjena prvenstveno grupiranju funkcija koje se koriste za istu svrhu. U slučaju da je potrebno proširiti funkcionalnost ili dodati nove načine generacije, potrebno je samo promijeniti klasu `GlobalGen` ili ju naslijediti i nadjačati metodu `Generate`.

Unutar klase `LocalGen` se nalazi serija provjera vezanih uz lokalnu prilagodbu koje se pokreću pozivom metode `AdjustOrTerminate`. Ove provjere su najzahtjevniji dio generacije, te se zbog toga koristi klasa `SearchWrapper`. Stvaranjem instance klase `SearchWrapper` se za određenu cestu određuje skup cesta koje mogu utjecati na njezino postavljanje. Metoda `AdjustOrTerminate` stvara instancu klase `SearchWrapper` za cestu koju obrađuje i ponaša se kao da je cijela mreža cesta sastavljena isključivo od cesta dohvatljivih putem `SearchWrapper`-ovog operatora „[ ]“. Sve konstante koje određuju

način na koji se nadovezuju ceste dohvaćaju se iz klase `C`, koja se brine da su veličine konstanti takve da odgovaraju veličinama na koordinatnom sustavu mape.

Ulaznim podacima se pristupa putem klase `MapStorage`, čije metode `GetHeightData` i `GetPopData` vraćaju vrijednosti ulaznih podataka na koordinatama prenesenim kao parametar. Inicijalizacijom te klase se u njezine statičke elemente pohranjuju veličine ulaznih podataka i sami podaci. Unutar inicijalizacije je sadržana sva manipulacija slikama izvedena korištenjem Magick++ biblioteke. Budući da su podaci dobiveni iz slika takvi da su vrijednosti poznate samo za točke označene cijelim brojevima i da za točke za koje trebamo saznati vrijednosti to ograničenje ne vrijedi, unutar metoda za dohvaćanje se bilinearno interpoliraju nepoznate vrijednosti.

5. 3. Grupiranje cesta i efikasnost

U procesu lokalne prilagodbe je potrebno pronaći sve ceste ili krajeve cesta koji su na određenoj udaljenosti od ceste koju obrađujemo. Najjednostavniji način za učiniti takvo što je usporediti trenutnu cestu sa svakom drugom cestom koja je već postavljena. Koristeći takav pristup, za n-tu cestu moramo pregledati (n-1) cesta. Ako pretpostavimo da će sveukupno biti postavljeno n cesta, broj cesta koje će biti pregledane iznosi:

$$N_{pregledavanja} = \sum_{i=1}^{n} (i-1) = \frac{n(n-1)}{2} = O(n^2)$$

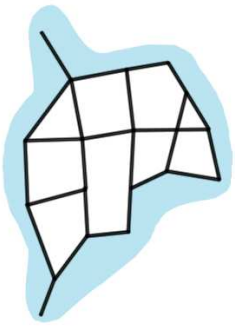
Operacije koje se vrše u lokalnoj prilagodbi su skupe i ponavljaju se za svaki od različitih postupaka prepravljanja. Uz tako visok konstantni faktor i kvadratnu ovisnost, nije moguće napraviti veliku mrežu cesta koristeći jednostavan pristup. Srećom, mreža prometnica je prilično rijetka i jednostavni pristup omogućava generiranje iste u kratkom vremenu. Nakon što je ona generirana, potrebno je optimirati način na koji se izvršava lokalna prilagodba ulica tako da njezina uporaba bude prihvatljivo brza.

Jedini način da se ubrza postupak je korištenje algoritma koji bi za proizvoljno odabranu cestu odredio set cesta kojima bi moglo biti potrebno prilagoditi cestu. Drugim riječima, potreban je algoritam koji u manje od N koraka, gdje je N ukupan broj cesta,

odlučuje koje ceste nije potrebno uzeti u obzir. Ceste koje nije potrebno uzeti u obzir su one ceste kojima su obje točke udaljene od ceste koju se provjerava za više od neke konstantne vrijednosti. Nije nužno da algoritam primjenjuje točno taj kriterij, ali se efikasnost algoritma mjeri i time koliki je dio skupa stvarno potrebno uzeti u obzir.

Smišljanje algoritma koji rješava taj problem u općem slučaju je teško, i stoga je u ovom radu primijenjen pristup koji uzima u obzir prirodu problema. Ignorirajući razlike izazvane zbog gustoće stanovništva, moguće je pretpostaviti da će grananje cestama podijeliti prostor virtualnog grada na područja koja će biti mala naspram veličine prostora koji grad zauzima. Ukoliko pretpostavimo da na grananje ulica unutar nekog područja ograđenog prometnicama ne utječu ceste izvan tog područja, možemo koristiti jednostavni pristup samo na tom području. Ako je očekivan broj cesta koje sačinjavaju jedno područje značajno manji od ukupnog broja cesta, moguće je pretpostaviti da je maksimalni broj cesta koji će se pojaviti unutar jednog područja ovisan samo o odnosu prosječne dužine ceste, minimalne dužine ulice i prosječnog broja cesta koji sačinjavaju područje zbog činjenice da se dovoljno slične ulice uklanjaju. Uz te pretpostavke moguće je zaključiti da će pretraživanje samo unutar ograđenih područja uz dovoljno dobru podjelu prostora biti brže od jednostavnog pristupa.

Nedostatak tog pristupa je mogućnost pojavljivanja područja koja su mnogo veća od prosjeka. Ukoliko je površina bilo kojeg od tih područja usporediva sa cjelokupnom površinom, vjerojatno je da će proporcionalno mnogo cesta potpasti pod to područje, što vraća kompleksnost algoritma na kvadratnu, makar s mnogo manjim konstantnim faktorom. Nažalost, priroda problema osigurava da uvijek postoji barem jedno takvo područje – obrub grada, prikazan na slici 5. 2. Performanse algoritma zato jako ovise o tome je li mreža prometnica dovoljno razgranata i da li ceste zatvaraju područja ili se granaju po volji.



Slika 5. 2 Primjer veličine područja na rubu grada

Raspoređivanje u područja je implementirano kroz metodu `GenerateGroups` klase `GroupCache`. U sklopu te metode se svakoj cesti koja ima neoznačenu stranu dodaje nova oznaka grupe i obavlja pretraga koja utvrđuje koje druge ceste pripadaju toj grupi. Postupak dodatno komplicira činjenica da je za ceste bitno koja grupa je lijevo, a koja desno prema smjeru u kojem su originalno bile postavljene. Naime, ulice kandidati se referenciraju na određenu stranu cesta iz kojih su potekle kako bi utvrdile svoju grupu. Složenost postupka stvaranja grupa je O(N log N), gdje je N broj cesta, zato što je svaka cesta obrađena jednom pri čemu se za nju utvrđuju susjedi, što se radi u O(log N) vremenu zahvaljujući balansiranom binarnom stablu u koje se stavljaju krajevi cesta. U binarnom stablu se poredak određuje usporedbom X koordinata ili Y koordinata ako su X koordinate jednake.

Rezultat procesa raspoređivanja su grupe, koje su u programu predstavljene putem klase `Group` čije instance sadrže pokazivače na ceste koje su članovi grupe. Grupe koristi klasa `SearchWrapper` kad joj se tijekom inicijalizacije pridruži ulica. Ukoliko u ulici nisu zapisani podaci o grupi u koju pripada, ta će informacija biti dohvaćena iz ceste iz koje je nastala grananjem. Pozivi operatora „`[]`" i `size()` te instance će dohvaćati podatke zapisane u grupi. U slučaju prometnica, `SearchWrapper` pristupa cjelokupnom popisu cesta, i vraća odgovarajuće podatke.

5. 4. Stvaranje i reaktiviranje ulica kandidata

Tijekom prve faze generacije mreže cesta se za svaku napravljenu prometnicu stvaraju ulice kandidati. Prije nego što stvaranje započne, za svaku se stranu prometnice određuje omjer dužine i širine bloka ulica. Sljedeći bitan podatak je razmak između pojedinih ulica, koji se slučajno određuje i onda prilagođava duljini prometnice kako bi se dobio cijeli broj postavljenih ulica. Na temelju omjera i razmaka između ulica se za svaku stranu određuje početna dužina ulice, i postavljaju ulice kandidati. Pozicije gdje počinju ulice kandidati se pamte u klasi `GroupCache`. Prema pravilima L-sustava, nastavak grananja ulica kandidata nije dozvoljen sve do kraja generacije prometnica.

Nakon što je grananje prometnica gotovo, one se razbijaju prema zapamćenim pozicijama putem metode `SplitHighways` klase `GroupCache`. Sljedeći korak je postepena aktivacija ulica kandidata i njihovo grananje. Ukoliko bi se sve ulice kandidati aktivirale paralelno, ne bi postojalo niti jedno susjedstvo u kojem se pojavljuje pravilnost u blokovima ulica jer bi ulice različitih prometnica nametale vlastite stilove. Da bi se doskočilo ovom problemu, uvedena je postepena aktivacija ulica kandidata. Na taj način ranije aktivirane ulice imaju vremena razgranati se i ustanoviti dominirajući stil susjedstva.

Broj koraka do aktivacije je određen slučajnom varijablom s velikim rasponom, što je dobro iz više razloga. Prvi razlog je nedostatak potrebe za praćenjem u kojoj grupi je aktivirana koja ulica. Slučajan izbor osigurava da je vjerojatno da će jedan stil ulica dominirati. Naravno, to znači da će se pojavljivati i suprotni slučajevi, ali to je drugi razlog zašto je slučajan izbor dobar. Slučajevi u kojima se paralelno aktivira više ulica su poželjni u malim količinama jer doprinose raznolikosti, a priroda slučajne distribucije osigurava da će slučajevi s najviše paralelnih aktivacija koji najgore izgledaju biti zanemarivo vjerojatni. Treći razlog zašto je slučajna aktivacija dobra ima veze s kaotičnom prirodom stvaranja mreže prometnica. Naime, moguća je situacija sa slike 6. 3, gdje prometnice sačinjnavaju područje za koje nije vjerojatno da će biti potpuno ispunjeno ulicama ako grananje krene samo sa jedne strane. Ukoliko grananje ulica koje su prve aktivirane nije u stanju zaobići način na koji su postavljene prometnice, prije ili kasnije će se aktivirati neki drugi set ulica kandidata koji će osigurati ispravan rad.

Slika 5. 3. Primjer područja koje je teško popuniti

Problem korištenja različitih stilova za različite prometnice u istom području je mogućnost da se prometnice neće poklapati. Proces lokalne prilagodbe zajedno s rastavljanjem prometnica osigurava da je vjerojatno da se to neće dogoditi. Ključan detalj pri rastavljanju prometnica je da se sve prometnice paralelno rastavljaju. Zahvaljujući tome, u procesu lokalne prilagodbe se u razmatranje uzimaju i mjesta gdje će tek kasnije nastati križanje dodavanjem ulice. Provjera sličnih cesta sprječava postavljanje većine renundatnih ulica, dok mali broj ulica koje nisu uhvaćene čine susjedstva zanimljivijima.

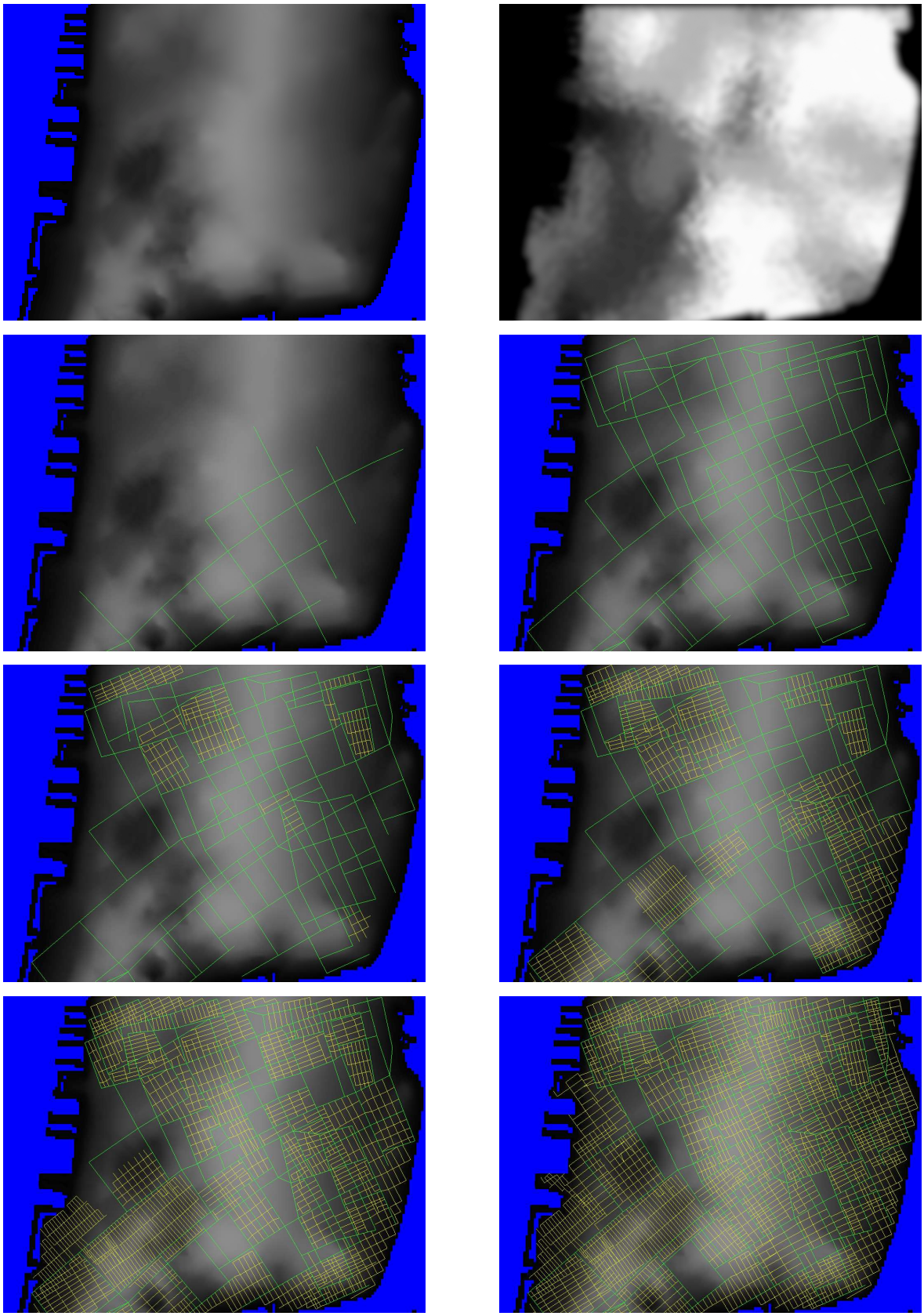
5. 5. Funkcije za iscrtavanje i interakciju s korisnikom

Budući da je namjena CityGen-a samo prikazivanje rezultata rada, sučelje je minimalističko i u cijelosti podređeno funkcionalnosti. Kostur programa je izgrađen koristeći WinAPI programsko sučelje zbog dobre interakcije s OpenGL-om i jednostavnog hvatanja unosa s tipkovnice. Podržano je pregledavanje rezultata s proizvoljnim uvećanjem i različitim prikazima ulaznih podataka i nastale mreže cesta. Moguće je i pratiti stvaranje mreže cesta pri kojem se nakon svakog dodavanja novih cesta osvježava prikaz. Dvije glavne funkcije za iscrtavanje su `DrawBackground` i `DrawRoadmap`. Ispis koje one vrše ovisi o trenutnim postavkama. `DrawBackground` projicira neke od ulaznih podataka kao pozadinu ili ne iscrtava ništa, dok `DrawRoadmap` prikazuje mrežu cesta koja je proizvedena do te točke. `DrawBackground` koristi napredne OpenGL funkcije za iscrtavanje velikih količina podataka, a `DrawRoadmap` koristi jednostavno iscrtavanje linija.

6. Rezultati

6. 1. Usporedba s originalnim radom

Zbog jako malog broja detalja iznesenog u originalnom radu, vrlo je teško procijeniti koliko je ovaj rad uspješan ili zapravo sličan originalnome. U originalnom radu je naglasak bio na generaciji uzoraka u gradovima koji su efikasno izgledali ali su mogli biti napravljeni i bez ikakve proceduralnosti. Od tih uzoraka, planirane blokove je moguće emulirati s ovim radom promjenom nekoliko konstanti, dok bi za monocentrične gradove bile potrebne samo manje preinake za koje je ostavljen prostor u obliku pravila grananja. Makar su ti uzorci dobra demonstracija fleksibilnosti i snage sustava koji se koristi, njihova implementacija i dobiveni rezultati nisu pretjerano zanimljivi. Umjesto toga, u ovom je radu najviše pažnje posvećeno grananju temeljenom na ulaznim podacima i namještanju tog grananja kako bi dalo što bolje rezultate. Tijekom tog procesa su razvijeni neki kriteriji koji nisu prisutni ili bar spomenuti u originalnom radu, poput prilagodbe ceste na putu do odredišta, a koji su jako bitni za dobar izgled grada. Primjer ostvarenih rezultata nalazi se na slici 6. 1.



Slika 6.1 Primjer generacije jedne mreže cesta. U prvom redu su visinska mapa i mapa gustoće populacije, a u redovima nakon nje različite faze generacije.

6. 2. Određivanje konstanti

Problematičan dio izrade CityGen-a je bilo utvrđivanje vrijednosti različitih konstanti koje su bitne za rad algoritma, budući da ih nije moguće izdvojiti statističkom analizom stvarnih gradova. Zbog toga je moguće je da su neke od konstanti koje se koriste u programu pogrešne apsolutne veličine, ali njihov relativan odnos je takav da dobro opisuje odnose koji se pojavljuju u stvarnim gradovima. To narušava mogućnost korištenja CityGen-a za pokušaj rekonstrukcije postojećih radova, ali je u tom slučaju potrebno promijeniti samo konverziju koju obavlja metoda `InitSizeValues()` klase `C`.

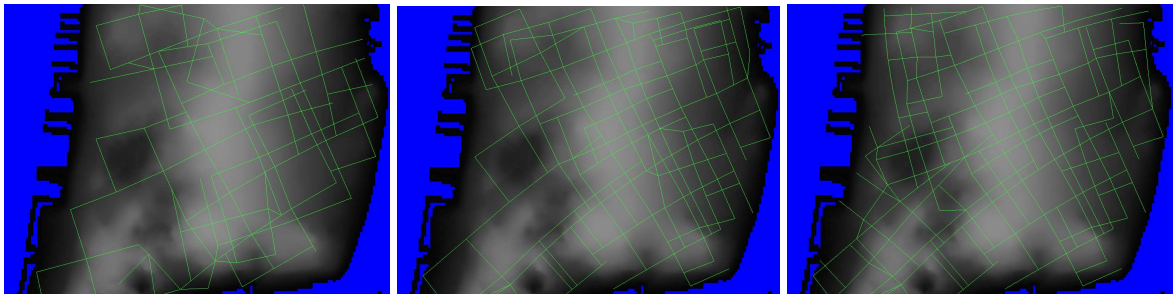
6. 3. Performanse

Vrlo je teško objektivno ocijeniti kakva je brzina ovog rada u usporedbi s originalnim, budući da je brzina generacije dana samo za jedan slučaj čiji je rezultat mnogo manji od onog koji CityGen proizvodi. Za taj slučaj, u kojem je moguće pretpostaviti prisutnost oko dvije tisuće cesta, generacija traje 10 sekundi. Prosječni slučaj CityGen-a sadrži oko osam tisuća cesta, traje oko dvije minute i 35 sekundi, i uključuje paralelno iscrtavanje novostvorenih cesta. Ako pretpostavimo kvadratnu složenost u ovisnosti o broju cesta, CityGen ima performanse jednako dobre kao i originalni rad. Prava složenost je vjerojatno manja od te procjene, a razlike u performansama proizlaze iz implementacije L-sustava. Unutar ovog rada je implementiran simulator koji doslovno interpretira proces prepisivanja niza znakova i zbog kontekstne ovisnosti koristi dva spremnika. Taj simulator je moguće optimirati tako da izvršava manje operacija, što su autori originalnog rada vjerojatno i učinili. Najbolje rješenje bi bilo vrlo udaljeno od direktne primjene L-sustava, zadržavajući bit pretvorbi s korištenjem drugačijih struktura podataka za određivanje kad je moguće primijeniti koje pravilo.

6. 4. Upotreba

U praksi nije vjerojatno da će CityGen sustav iz prvog pokušaja dati zadovoljavajući izgled grada, te je potrebno nekoliko pokušaja s preinakama ulaznih podataka i sitnim promjenama parametara kako bi se poboljšao rezultat. Najveću razliku u izgledu proizvode promjene parametara vezanih uz grananje i mijenjanje raspona u kojem

se slučajno mijenja veličina ceste. Dobar primjer utjecaja parametara su rezultati dobiveni promjenom parametra PEER_PRESSURE. Njegova namjena je regulacija grananja i čak i vrlo male promjene imaju kao posljedicu velike razlike u izgledu. Na slici 6. 2 se nalaze slike mreže prometnica nastale na temelju ulaznih podataka korištenih na slici 6. 1 s različitim vrijednostima parametra PEER_PRESSURE. Za stvaranje prve slike je korišten parametar manji od običnog, što uzrokuje manje grananja i rezultat koji je besmislen. Druga slika prikazuje grananje s preporučenim parametrom kod kojeg se pojavljuje razlika u gustoći prometnica u ovisnosti o gustoći naseljenosti. Za treću sliku je korištena veća vrijednost, što ne utječe na pravilnost izgleda mreže, ali uklanja utjecaj gustoće stanovništva.



a) b) c)

Slika 6. 2 Izgled mreže cesta s različitim vrijednostima parametra PEER_PRESSURE: a) 1.25; b) 1.35; c) 1.45.

Općenito, program pokazuje jako veliku osjetljivost na vrijednosti ulaznih podataka, parametara i poziciju početne prometnice. Najveći problem s finim podešavanjem izgleda je nedostatak lokalne kontrole rezultata. Svaka promjena utječe na izgled cijele mreže i nije moguće izolirati dio kako bi se samo njega promijenilo. Taj nedostatak proizlazi iz prirode algoritma i nemoguće ga je popraviti na razini generiranja mreže prometnica. Jedini način da se zaobiđe taj problem bi uključivao sučelje putem kojeg bi korisnik mogao između faza mijenjati izgled prometnica i ručno korigirati nepoželjne dijelove. Generaciju bi bilo moguće nastaviti na isti način, a u fazi generiranja ulica bi se područja nezadovoljavajućeg izgleda mogla ponovno generirati s različitim parametrima. Takav dodatak bi bio zanimljiva ekstenzija ovog rada, ali je upitno koliko bi takve korekcije uopće bile primjetljive nakon dodavanja zgrada. Za program je bitno da

radi dobro u prosječnom slučaju, a mali nedostaci su ili neprimjetni ili doprinose uvjerljivosti razbijajući monoton izgled.

6. 5. Daljnji razvoj

Funkcionalnost CityGen-a je samo dio one ostvarene u originalnom radu, i prirodan nastavak razvoja bi uključio pokrivanje cjelokupne generacije grada. Nakon faze generacije mreže cesta slijedi podjela blokova na parcele za zgrade, za koju je većina potrebne funkcionalnosti ostvarena u programu. Proces generacije grupa je moguće još jednom primijeniti na završenu mrežu cesta kako bi se izdvojili blokovi, a algoritam podjele na parcele se oslanja na pronalaženje približno paralelnih bridova, za što su svi potrebni izračuni već implementirani u programu. Faze koje slijede nakon te uključuju generaciju zgrada i pročelja zgrada, ali su one problematične zbog velike složenosti procesa izgradnje koji nije detaljno opisan u originalnom radu. Srećom, ovaj rad nije jedini koji se bavi proceduralnom generacijom zgrada, postoje drugi radovi koji postižu bolje rezultate i daju više uvida u njihovo stvaranje, od kojih se ističe [10].

Zaključak

Rezultati potvrđuju tezu originalnog rada da je moguće konstruirati grad proširenjem metoda obično rezerviranih za prikazivanje biljaka, ali i pokazuju koliko je teško odrediti oblik grada za koji se može reći da odgovara stvarnome i pravila koja ga opisuju. Najveće ograničenje ove metode je priroda gradova koje generira. Planski izgrađene gradove, koji uključuju veliku većinu nedavno nastalih gradova ili predgrađa, nije moguće generirati jer ne podliježu obrascima koji se pojavljuju spontanom izgradnjom gradova već su napravljeni s određenom funkcijom na umu, koju je vrlo teško opisati korištenim postupcima grananja.

Implementacija metode pati od manjih nedostataka, ali zadovoljava osnovnu namjenu – izgleda uvjerljivo. Kombinacija ove metode s nekim od postupaka proceduralnog modeliranja zgrada i terena može proizvesti virtualne krajolike koji svoju primjenu nalaze u filmskoj industriji i industriji igara. Korisnost je veća u industriji igara upravo zbog svojstva algoritma da se proizvedeni sadržaj razlikuje u mnogo detalja sa zanemarivim promjenama u ulaznim parametrima. Na taj način je moguće pružiti korisniku drugačije iskustvo svaki put kad pokrene igru.

Na apstraktnijem nivou, koncept grananja koje formira zatvorene petlje može se iskoristiti za modeliranje mnogih prirodnih sustava. Primjerice, slična bi se metoda mogla iskoristiti pri modeliranju riječnih sustava, gdje je grananje ograničeno geografskim faktorima a rijeke se, kao i ceste, prestaju prostirati tek kad se spoje na dio postojećeg sustava. Saznanja stečena rješavanjem ovog problema se mogu upotrijebiti za efikasnije modeliranje i implementaciju sličnih sustava.

Literatura

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

[2] J. Olsen. Realtime Procedural Terrain Generation. www.oddlabs.com/download/terrain_generation.pdf, svibanj 2008.

[3] C. Alexander, S. Ishikawa, M. Silverstein, M. Jacobson, I. Fiksdahl-King and S. Angel. A Pattern Language. Oxford University Press, New York, 1977.

[4] http://www.vision.ee.ethz.ch/~pmueller/wiki/CityEngine/Front, svibanj 2008.

[5] http://forums.introversion.co.uk/introversion/viewtopic.php?t=1132, svibanj 2008.

[6] http://seamless.usgs.gov, travanj 2008.

[7] http://home2.nyc.gov/html/dcp/pdf/census/mappl1.pdf, travanj 2008.

[8] P. Prusinkiewicz, A. Lindenmayer. The Algorithmic Beauty of Plants. Springer-Verlag, 1990.

[9] D. Shreiner, M. Woo, J. Neider, T. Davis. OpenGL Programming Guide: The Official Guide to Learning OpenGL, Version 1.4, Fourth Edition. Addison-Wesley, 2003

[10] P. Wonka, M. Wimmer, F. Sillion, W. Ribarsky. Instant architecture. ACM Transactions on Graphics (TOG), v.22 n.3, July 2003

Postupci proceduralnog modeliranja gradova

Sažetak

Proceduralna generacija gradova je postupak u kojem se korištenjem tehnika preuzetih iz algoritamske generacije biljaka pokušava stvoriti virtualni grad prilagođen zadanim ulaznim podacima, koji uključuju podatke o naseljenosti, visinske mape i mape vodenih površina.. Proceduralna generacija gradova originalno je objavljena u radu Parisha i Müllera „Procedural Modeling of Cities“ gdje je bila izvedena korištenjem L-sustava. U ovom radu načinjena je implementacija proceduralne generacije gradova koja stvara i prikazuje mogući raspored mreže cesta. Generacija mreže je razdvojena na faze tijekom kojih se generiraju različite vrste cesta, što omogućuje korištenje naprednih postupaka prilagodbe i drugačijih struktura podataka od originalnog rada.

Ključne riječi: L-sustavi, razvoj gradova, arhitektura, modeliranje

Methods for procedural modeling of cities

Abstract

Procedural generation of cities is a technique inspired by algorithmic botany which attempts to create a virtual city based on various input data. The data includes elevation, population density and water maps. The L-system based technique was originally published in Pascal and Müller's paper „Procedural Modeling of Cities“. The implementation of the technique described in this thesis generates and displays a city road map. The generation process is broken up into phases according to road type, allowing the implementation of different data structures and enhanced local constraints.

Keywords: L-systems, city development, architecture, modeling