TARTU ÜLIKOOL
Arvutiteaduse instituut
Informaatika õppekava

**Kristjan Perli**

# Protseduuriline linnade genereerimine

**Bakalaureusetöö (9 EAP)**

Juhendajad: Ahti Peder, PhD
Raimond-Hendrik Tunnel, MSc

Tartu 2016

## Protseduuriline linnade genereerimine

**Lühikokkuvõte:**

Käesolevas bakalaureusetöös kirjeldatakse eksisteerivaid linnade genereerimise süsteeme ning nende põhjal autori poolt väljatöötatud linnade genereerimise süsteemi. Töö raames valminud süsteem genereerib protseduuriliselt linna ilma kasutajapoolset sisendit vajamata, kuid võimaldab kasutajal linna omaduste muutmiseks interaktiivselt parameetreid muuta. Lisaks võrreldakse autori poolt loodud süsteemi teiste töös kirjeldatud süsteemidega.

**Võtmesõnad:**

Protseduuriline genereerimine, linna genereerimine, arvutigraafika

**CERCS:**

P150: Geomeetria, algebraline topoloogia;

P170, Arvutiteadus, arvutusmeetodid, süsteemid, juhtimine;

P175: Informaatika, süsteemiteooria.

## Procedural generation of cities

**Abstract:**

In this Bachelor's thesis several existing procedural city generation systems are described and a new city generation system based on those is presented. The procedural city generation system presented in this thesis generates cities without needing any input from user but allows the user to change parameters interactively. In addition, a comparsion between existing city generation systems and presented city generation system is given.

**Keywords:**

Procedural generation, city generation, computer graphics

**CERCS:**

P150: Geometry, algebraic topology;

P170: Computer science, numerical analysis, systems, control;

P175: Informatics, systems theory.

# Sisukord

# 1. Sissejuhatus

Tänapäeval kujutatakse arvutimängudes ning muudes graafilistes rakendustes väga suuri või koguni lõpmatuid kolmemõõtmelisi maailmu ning samuti on neist mõne osaks ka linnad. Linnad on väga keerulised kooslused ning sisaldavad väga palju detaile erinevatel tasemetel. Sellest tulenevalt on nende manuaalne modelleerimine väga töömahukas ja kallis protsess, mistõttu oleks tarvilik kasutada protseduurilist genereerimist ehk andmete algoritmilist loomist. Protseduurilise genereerimisega on võimalik etteantud reeglite põhjal genereerida suurel hulgal unikaalseid linnu. Protseduurilist genereerimist kasutatakse veel tekstuuride [1], puude [2] ja maastike [3] genereerimiseks.

Linnade protseduurilist genereerimist on varem käsitletud mitmetes teadustöödes [4,5,6], kuid Eesti teadusmaastikul pole sellega varasemalt tegeldud.

Käesoleva lõputöö eesmärgiks on kirjeldada eksisteerivaid linnade genereerimise süsteeme ning nende põhjal autori poolt väljatöötatud süsteemi. Töö raames valminud süsteem genereerib protseduuriliselt linna ilma kasutajapoolset sisendfaili vajamata. Samas on kasutajal võimalik linna omaduste muutmiseks interaktiivselt parameetreid muuta. Antud süsteem on loodud kasutades Three.js'i[1].

Töö koosneb kuuest peatükist, millest teises kirjeldatakse kolme eksisteerivat linnade genereerimise süsteemi. Kolmandas peatükis kirjeldatakse töö raames valminud linnade genereerimise süsteemi. Neljandas peatükis võrreldakse eelnevates peatükkides kirjeldatud süsteemide poolt genereeritavaid linnu. Töö lõpus pakutakse välja võimalikke töö raames valminud süsteemi edasiarendusvõimalusi.

[1] http://threejs.org/

# 2. Eksisteerivad linnade genereerimise lahendused

Käesolevas peatükis kirjeldatakse kolme eksisteerivat lahendust, milleks on CityEngine [1], CityGen [3] ja genereerija, mis on kirjeldatud artiklis „Real-time Procedural Generation of 'Pseudo Infinite' Cities" [2], mida siin töös nimetame lõpmatute linnade genereerimise süsteemiks. Kõik selles peatükis välja toodud algoritmid ja pildid on pärit eelnimetatud allikatest.

Üldiselt jaotavad erinevad autorid linnade genereerimise probleemi kolmeks eraldiseisvaks osaks:

1) Teedevõrgustiku genereerimine vastavalt maastikule.

2) Teedevõrgustiku pealt kruntide defineerimine ja leidmine.

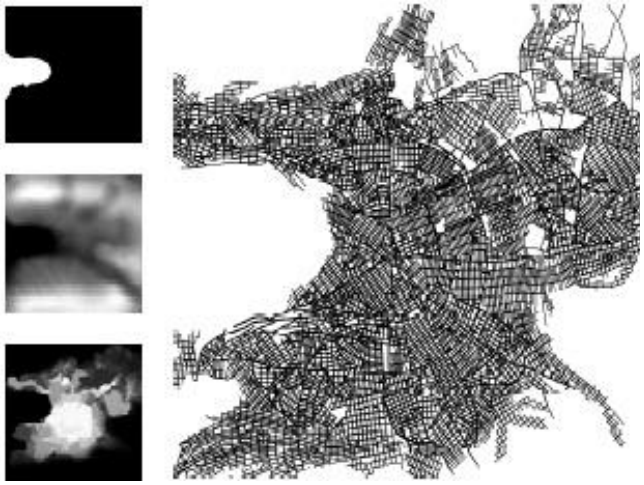
3) Hoonete genereerimine ning paigutamine kruntidele.

Väljatoodud linnade genereerimise süsteeme vaadates keskendub antud töö nendele kolmele alamülesandele vastavates süsteemides. Tuuakse välja olulisemad detailid teistest süsteemidest, mis seostuvad kõige rohkem käesolevas töös välja pakutud süsteemiga.

## 2.1 CityEngine

CityEngine on protseduuriline ja lihtsasti laiendatav linna genereerimise süsteem aastast 2001, mis kasutab rohkesti L-süsteeme [4].

Linna genereerimiseks on kasutajal tarvis sisendiks anda geograafilise kaardi pilte, millelt CityEngine süsteem saab lugeda maastiku kõrgust või veekogusid ning rahvastikutiheduse kaardi pilti, millest sõltub linna paigutus maastikul (vt. joonis 1). Peale selle on võimalik sisendiks anda ka muid statistilist infot edastavate kaartide pilte, millega on võimalik määrata näiteks piirkonnas leiduvate hoonete tüüpi, hoonete kõrgust ning teedevõrgu mustreid.



Joonis 1. Vasakul sisendiks antud vee paigutust kujutav kaart, kõrguskaart ja rahvastikutiheduse kaart. Paremal genereeritud linn.

Põhiteed, mida CityEngine'is nimetatakse kiirteedeks, ühendavad linna suurema populatsiooniga piirkondi. Lisaks on olemas kõrvalteed ehk tänavad, mis pakuvad ligipääsu kiirteedele. Samuti moodustuvad kõrvalteede vahelistele aladele elurajoonid.

Pärast teedevõrgu loomist jaotatakse maastik väiksemateks aladeks, mis on ümbritsetud kõrvalteedega. Väiksemad alad jaotatakse omakorda kruntideks, millele genereeritakse hooned.

Hooneid on kolme tüüpi: pilvelõhkujad, ärihooned ja elumajad. Nende paigutus sõltub sisendandmetest.

### 2.1.1 Teede genereerimine

Teede genereerimiseks kasutab CityEngine süsteem laiendatud parameetrilist L-süsteemi. L-süsteem on formaalne grammatika, milles produktsioonireegleid rakendadakse paralleelselt. L-süsteemiga genereeritakse sõned, milles igale tähele saab määrata toimingu. Näiteks antud süsteemis tähendab täht B tee hargnemist ja täht R tee loomist. Parameetriline L-süsteem töötab *parameetriliste sõnadega*, mis kujutavad endast moodulitest koosnevaid sõnesid [4]. *Moodulid* on tähed, millega on seotud parameetrid.

Antud süsteemis on parameetrilist L-süsteemi laiendatud nii, et moodulite parameetreid ei seataks produktsioonireeglitega, vaid selleks kasutatakse väliseid protseduure. Tänu sellele, et parameetrite arvutamine on eraldatud moodulite järgnevusest, on laiendatud L-süsteemi produktsioonireeglid lihtsamad. Samuti on võimalik teede genereerimisel arvesse võetavaid piiranguid lisada nii, et produktsioonireegleid muutma ei pea.

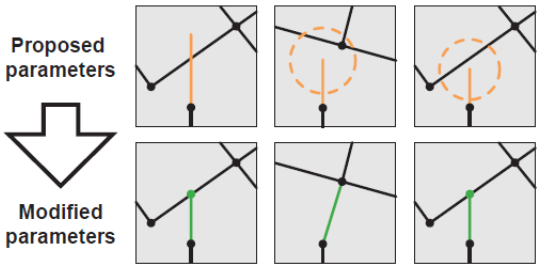
Iga kord kui moodulitega sõnele rakendatakse reegleid, tehakse järgnevad toimingud:

- L-süsteem tagastab järglase ehk produktsiooni reegli paremaks pooleks oleva sõne, mille moodulitel on parameetrid määramata.

- Rakendatakse protseduuri, mis määrab moodulite parameetrid sõltuvalt rahvastikutihedusest ning teedevõrgu mustrist.
- Rakendatakse kohalike piirangute protseduuri, mis muudab parameetrite väärtusi sõltuvalt ümbritsevatest piiravatest teguritest nagu veekogud, pargid ja asukoha kõrgus. Juhul kui parameetritele sobivaid väärtusi ei leita, siis moodul kustutatakse.

Põhitee ühe teelõigu genereerimiseks moodustatakse olemaseoleva teelõigu otspunktist või algpunktist väljuvad kindla pikkusega sirglõigud, mille järgi võetakse rahvastikutiheduse kaardilt teatud hulk väärtuseid. Saadud väärtused jagatakse nende võtmise asukoha ning teelõigu otspunkti vahemaaga. Seejärel liidetakse iga sirglõigu järgi võetud väärtused eraldi kokku ning valitakse suurima väärtuste summaga sirglõik, mille järgi teed kasvatatakse.

Kõrvalteede genereerimisel rahvastikutiheduse muutumist ei arvestata, vaid jälgitakse teedevõrgu mustrit. Kõrvalteede kasvamine lõpetatakse, kui jõutakse asukohta, kus rahvastikku ei ole.




Joonis 2. Olemasoleva teega ühendumine.

Kohalike piirangute protseduur teeb kindlaks, kas loodava tee otspunkt asub vees või mujal mittesobivas kohas ning vajadusel muudab loodava tee suunda või pikkust. Seejärel kontrollib funktsioon, kas loodav tee lõikub olemasoleva teega või loodava tee otspunkt on lähedal olemasolevale teele või ristmikule ning ühendab vajadusel loodava tee olemasolevaga (vt. joonis 2).

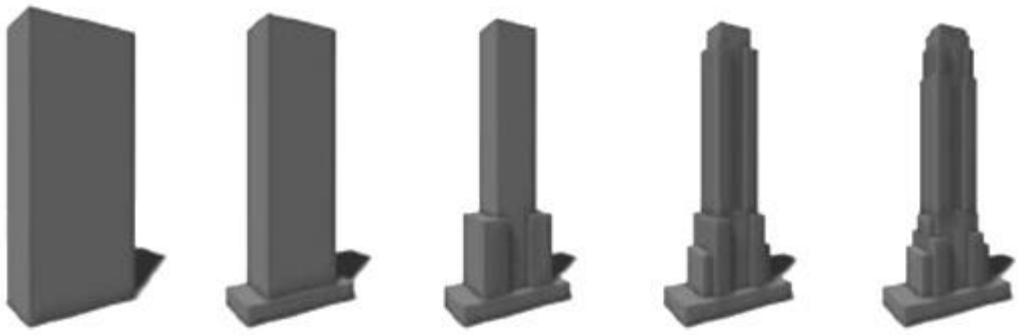
### 2.1.2 Hoonete genereerimine

Kvartaliteks on selles implementatsioonis teedega ümbritsetud alad. Kvartalid jaotatakse omakorda kruntideks kasutades rekursiivset algoritmi. Jaotamisel jagatakse kujundi pikimad ligikaudu paralleelsed küljed kaheks. Algoritm jaotab krunte kuni saadud krundid on etteantud piirist väiksemad (vt. joonis 3). Krundid, mis on liiga väikesed või millel puudub ühendus teega, jäetakse tühjaks.

Joonis 3. Vasakul teedevõrk. Keskel teedevõrku skaleerides saadud kvartalid. Paremal kruntideks jaotatud kvartalid.

Hoone tüüp valitakse sisendandmete põhjal, kus on märgitud hoone tüübid erinevates piirkondades.

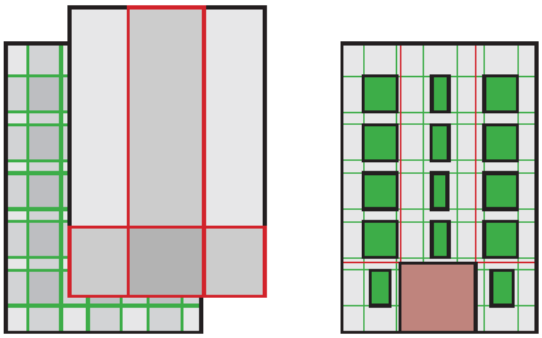
Hoonete loomiseks kasutatakse parameetrilist stohhastilist L-süsteemi, millel on iga hoonetüübi jaoks eraldi produktsioonireeglid. *Stohhastiline L-süsteem* on selline L-süsteem, mille puhul on moodulil rohkem kui üks produktsioonireegel ning iga produktsioonireegli valimiseks on määratud mingi tõenäosus. Antud L-süsteemi mooduliteks on muundamis-, hargnemis-, peatumis- ja eendamismoodul ning selle *L-süsteemi aksioomiks* ehk algolekuks on hulktahukas, mille põhjaks on krundi kujund. Seda hulktahukat töödeldakse mitmel sammul, muutes seda iga kord detailsemaks (vt. joonis 4).



Joonis 4. Hoone geomeetria töötlemise sammud.

Hoonete fassaadide loomiseks kasutatakse käsitsi paika pandud stiilitekstuure (ing. k. *style texture*), millel on märgitud elementide suurused ja korrapärasused. Igast stiilitekstuurist on võimalik luua suurel hulgal erinevaid tekstuure erinevate suurustega fassaadide tarbeks.

Fassaad jaotatakse ruutudeks ning igale ruudule saab määrata tekstuuri. Selleks kasutatakse kihistamist (ing. k. *layering*) ja liitfunktsiooni tehnikat [5]. Erinevad kihid võivad üksteisele mõju avaldada (vt. joonis 5). Seejuures lähtutakse järgnevatest eeldustest:



Joonis 5. Vasakul on kujutatud kaks kihti ruudustikke ning paremal on näha kuidas pealmine kiht mõjutab alumises kihis mõningate ruutude suurust.

- fassaadidel on näha üht või mitut ruudustikku, milles enamik ruute täidab samu ülesandeid;
- osad ruudustiku ruudud mõjutavad ümbritsevate ruutude suurust ja paigutust. Näiteks hoone ukse ruudu kohal paiknevates ruutudes on aknad teistest akendest erineva suurusega;
- ebakorrapärasused ruudustiku struktuuris mõjutavad enamasti tervet rida või veergu.

## 2.2 Lõpmatute linnade genereerija

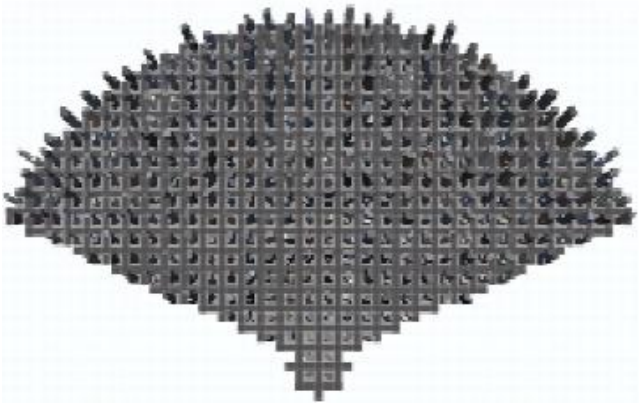
Käesolev süsteem genereerib pseudolõpmatuid linnu, mida on kasutajal võimalik interaktiivselt tänavavaates näha (vt. joonis 6). Linnas ringi liikudes genereeritakse hooneid juurde ning vaateväljast välja jäävad hooned kustutatakse mälust (vt. joonis 7). Teedevõrguks on korrapärane ruudustik, mille igas ruudus on hoone.

Joonis 6. Linna tänavavaade.

### 2.2.1 Hoonete genereerimine

Hoone kuju ja välimus sõltuvad ühest pseudojuhusliku numbrigeneraatori seemnest (ing. k. *seed*). Seeme arvutatakse ruudustiku ruudu asukoha järgi kasutades funktsiooni, mis kujutab ruudu koordinaadid üheks reaalarvuks, et lähestikku paiknevate ruutude seemned oleks võimalikult erinevad. Lähestikku asuvate ruutude seemned peavad olema võimalikult

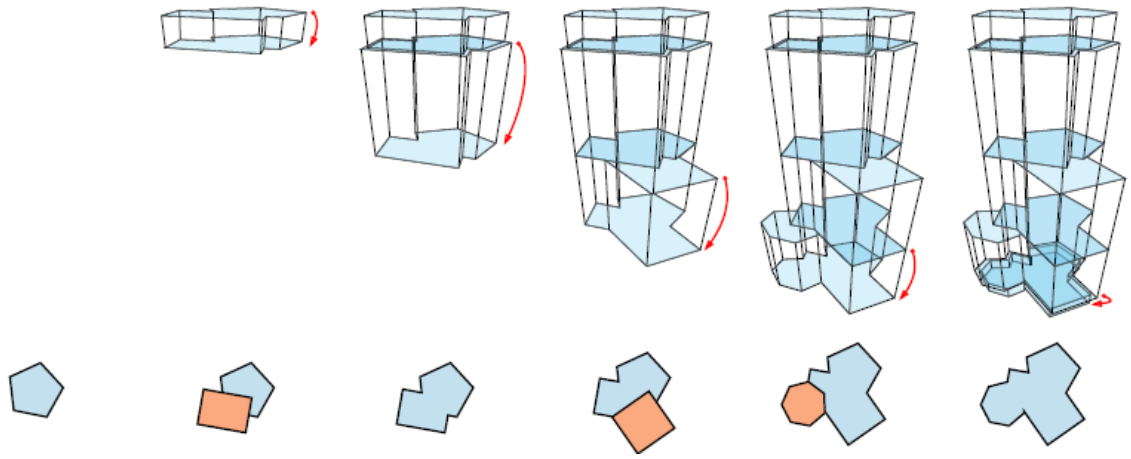
erinevad, kuna selles süsteemis kasutatav numbrigeneraator loob sarnaste seemnete korral sarnased hooned. Samuti on arvutatav seeme sama ruudu asukoha puhul sama ning seetõttu tekib sinna alati ühesugune hoone ka juhul, kui kasutaja vahepeal sellest eemale liigub.



Joonis 7. Hooned genereeritakse ainult vaatevälja.

Põrandaplaanide genereerimiseks kasutatakse iteratiivset algoritmi, mis esimesel sammul genereerib suvalise korrapärase hulknurga või ristküliku ning järgmistel sammudel moodustab ühendi suvalise korrapärase hulknurga või ristkülikuga (vt. joonis 8). Seda protsessi korratakse etteantud arv kordi ning igal sammul saadud tulemus salvestatakse järjendisse. Saadud põrandaplaanid skaleeritakse, et nad mahuks tänavaruutu ära.

Hoone loomiseks kasutatakse põrandaplaanide genereerimise igal sammul saadud hulknurki ning loomine hakkab pihta hoone katusest. Selleks eendatakse (ing. k. *extrude*) teist põrandaplaani kuni kahe korruse kõrguse võrra allasuunas (vt. joonis 8) . Järgnevate hoone sektsioonide loomiseks eendatakse iga järgmise etapi põrandaplaani suvalise kõrguse võrra.



Joonis 8. Alumisel real on kujutatud põrandaplaanide genereerimise järjestikuseid samme. Ülemisel real on näha põrandaplaanide eendamist ja seeläbi hoone loomist.

Eendamise käigus tekitatakse ka tekstuuri kaardistamise koordinaadid, jagades fassaadi laiuse hoonele määratud akna laiusega ning hoone sektsiooni pikkuse korruse kõrgusega hoone iga sektsiooni puhul eraldi. Erinevaid tekstuure on 10.

## 2.3 CityGen

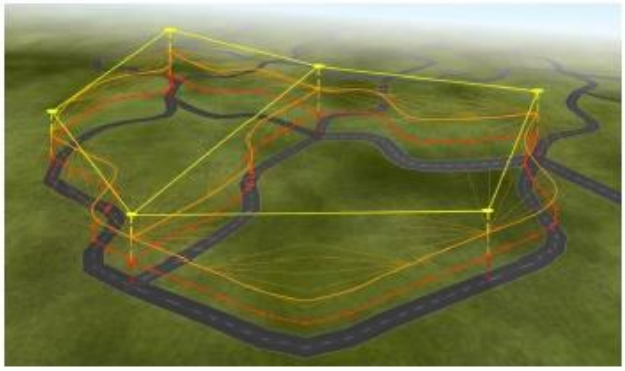
CityGen on interaktiivne protseduurilise linna genereerimise süsteem aastast 2007. Kasutajal on võimalik genereerimise ajal muuta erinevaid parameetreid.

Linna genereerimine on jaotatud kolmeks etapiks:

1. põhiteede genereerimine;
2. väiksemate teede genereerimine;
3. hoonete genereerimine.

### 2.3.1 Põhiteede genereerimine

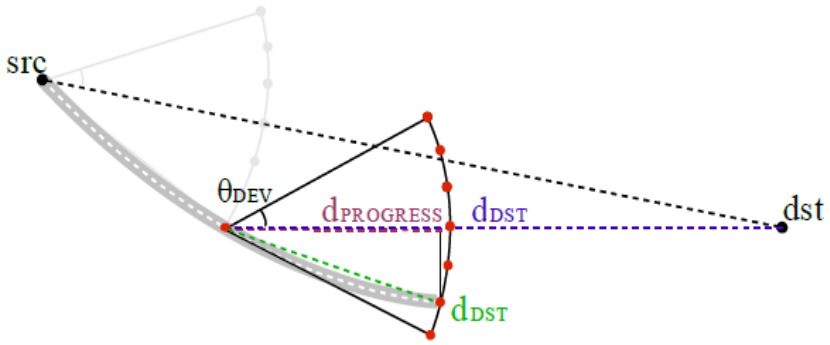
Teedevõrgustikku kujutatakse suunamata graafidena ning teedevõrgustiku andmeid külgnevusmaatriksina.



Joonis 9. Kollasega on kujutatud põhiteede esimene graaf, punasega teine graaf ning oranziga hulk valitud punkte ja interpoleeritud joon.

Põhiteede tarbeks on kaks graafi, millest esimese tippudeks on põhiteede ristmikud, ning teise tippudeks on lisaks ristmikutele ka kõik punktid, mida ristmikke ühendavad teed läbivad (vt. joonis 9).

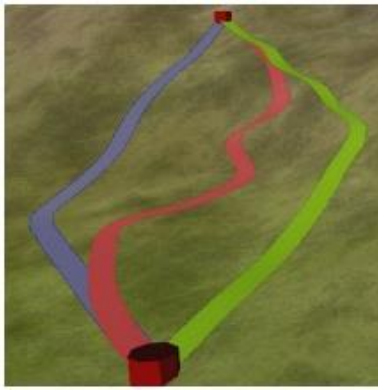
Esimese graafi tippe on võimalik rakenduses interaktiivselt juurde lisada, kustutada ja liigutada, et muuta teede paigutust. Teise graafi tipud ja servad arvutatake süsteemi poolt.




Joonis 10. Teede loomise algoritmi samm. Vasakpoolne punane punkt on hetkepunkt ning parempoolsed punktid on potentsiaalsed valitavad punktid.

Tee loomiseks kahe põhiteede graafi tipu vahel hakkab algoritm mõlemast tipust üheaegselt vastu liikuma. Igal sammul valitakse etteantud arv punkte P, kuhu tee võib edasi liikuda. Punktid valitakse kindla vahemaa kauguselt ning loodava teelõigu nurk peab tee teise otspunkti suhtes jääma etteantud vahemikku (vt. joonis 10). Algoritm lõpetab, kui mõlemast tipust lähtuvad tööjärjed kohtuvad.

Igal sammul liikumissuuna valikuks valitud punktidest kõige sobivama väljavalimiseks on välja toodud kolm strateegiat (vt. joonis 11):



Joonis 11. Sinisega kujutatud madalaima kõrguse strateegia, punasega väikseima kõrguse erinevuse strateegia ja rohelisega võrdse kõrguse erinevuse strateegia.

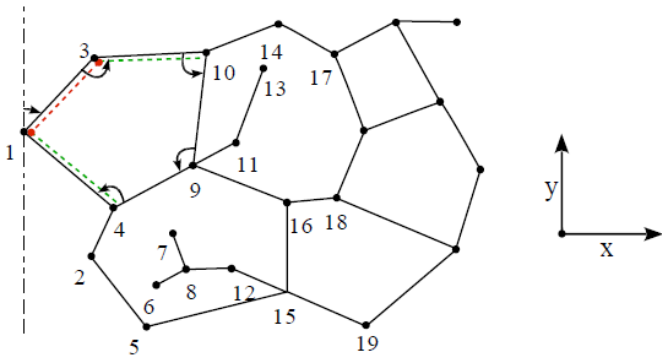
- madalaima kõrguse strateegia – valitakse kõige madalamal asuv punkt hulgast P;
- väikseima kõrguse erinevuse strateegia – valitakse selline punkt hulgast P, mille kõrguse erinevus hetkepunktiga võrreldes on kõige väiksem;
- võrdse kõrguse erinevuse strateegia – iga potentsiaalse valitava punkti $(x, y) \in P$ ja hetkepunkti $(x_0, y_0)$ puhul arvutatakse nende kõrguse erinevuse $|h(x, y) - h(x_0, y_0)|$ ja lõpppunkti $(x_p, y_p)$ poole liigutud teepikkuse $d_{progress}$ (vt. joonis 10) suhe. Samuti arvutatakse hetkepunkti $(x_0, y_0)$ ja lõpppunkti $(x_p, y_p)$ kõrguste erinevuse $|h(x_0,\ y_0) - h(x_p, y_p)|$ ja samade tippude vahemaa $d_{DST}$ suhe. Lõpuks valitakse hulgast P punkt, mille puhul on arvutatud suhete vahe minimaalne. $min\left\{\left|\frac{|h(x,y)-h(x_0,y_0)|}{dProgress} - \frac{|h(x_0,y_0)-h(x_p,y_p)|}{dDST}\right| : (x, y) \in P\right\}$.

Kõikide väljavalitud punktide põhjal luuakse neid interpoleeriv *Catmull Rom* splain [5]. Saadud splainist on võimalik täpsemaid teelõike eraldada, mis lisatakse ristmikke ühendavate teede graafi. Splain on lõigul $[a, b]$ määratud funktsioon, mis teatava lahutuse $a = x_0 < x_1 < \ldots < x_n = b$ korral on igas vahemikus $(x_k, x_{k+1})$ esitatav polünoomina. Enamasti nõutakse täiendavalt veel funktsiooni ja selle teatavat järku tuletise pidevust jaotuspunktides $x_k$ [8].

### 2.3.2 Kõrvalteede genereerimine

Kõrvalteed teenindavad rajoonisisest piirkonda pakkudes ligipääsu põhiteedele. Rajoonideks on maa-alad, mis on põhiteedega ümbritsetud. Igal rajoonil on eraldi teedevõrgustiku graaf kõrvalteede jaoks.

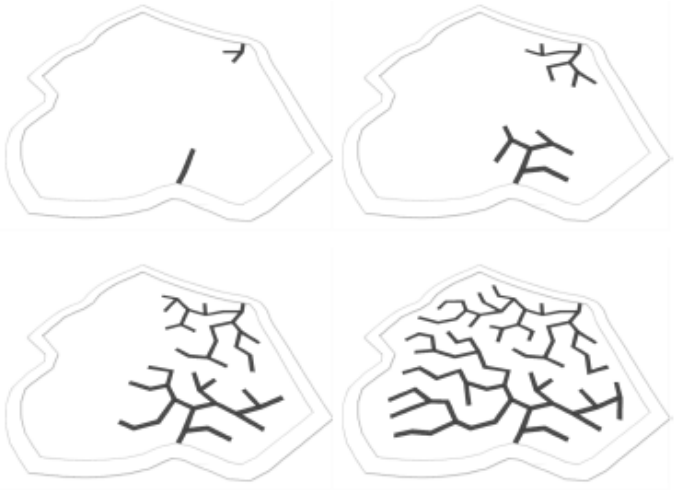
Kõrvalteede genereerimiseks on esmalt vaja põhiteede graafist eraldada rajoonid. Kuna rajoonid avalduvad põhiteede graafis tsüklitena, siis on tarvis graafist tsüklid eraldada.




Joonis 12. Graafist tsüklite eraldamise algoritmi illustratsioon.

CityGen süsteemis kasutatakse graafist tsüklite eraldamiseks algoritmi (vt. joonis 12), mis kõigepealt sorteerib põhiteede graafi kõik tipud x-koordinaatide järgi kasvavalt. Seejärel hakkab algoritm graafi läbima, alustades kõige väiksema x-koordinaadiga tipust ning liikudes graafi servi mööda päripäeva. Kui jõutakse juba läbitud tippu, on tsükkel leitud. Leitud tsükli esimene serv eemaldatakse graafist ning samuti eemaldatakse ka kõik oksad, mis selle serva otspunktidega ühendatud on. Peale seda valitakse järele jäänud tippudest uuesti kõige väiksema x-koordinaadiga tipp ning korratakse sama protseduuri, kuni kõik tsüklid on leitud.

Eraldatud tsüklid grupeeritakse okstega, mida tsüklid ümbritsevad ning sisestatakse seejärel rajoonide järjendisse. Igal rajoonil on omaette teedevõrk koos ümbritseva teega ning hulk parameetreid teede loomise kontrollimiseks.

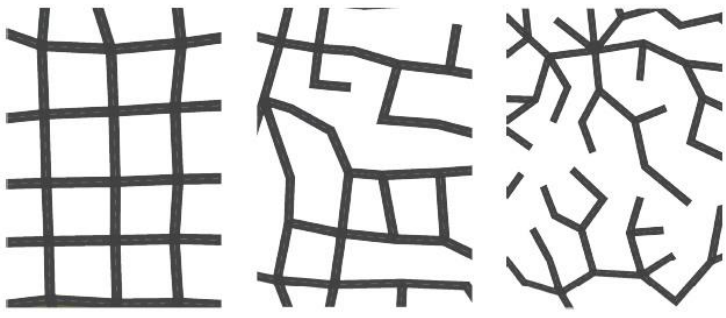


Joonis 13. Kõrvalteede kasvamine rajoonis.


Joonis 14. Olemasolevate teedega ühendumise kontroll.

Kõrvalteid kasvatatakse L-süsteemi abil alustades punktist, mis valitakse rajooni pikimate külgede keskpunkti järgi (vt. joonis 13). Tee kasvamisel arvestatakse lähedal asuvate tippude ja teelõikudega ning teatud tingimustel ühendatakse tee olemasoleva teega või lõpetatakse vastava teelõigu kasvatamine (vt. joonis 14).

Igal rajoonil eraldi olevad teede genereerimise kontrollparameetrid nagu lõigu pikkus, hargnemise määr, teiste tippudega ühendumise suurim kaugus ja ühendumise tõenäosus võimaldavad luua rajoonikaupa erineva mustriga teedevõrke (vt. joonis 15).



Joonis 15. Erinevate parameetritega loodud teedevõrgud.

Olemasolevate teedega ühendumiseks vaadeldakse tippe, mis jäävad loodavast teest parameetriga määratud raadiusesse (vt. joonis 14). Kui vaadeldavasse piirkonda jääb mitu tippu, siis valitakse neist loodava tee algpunktile kõige lähem punkt.

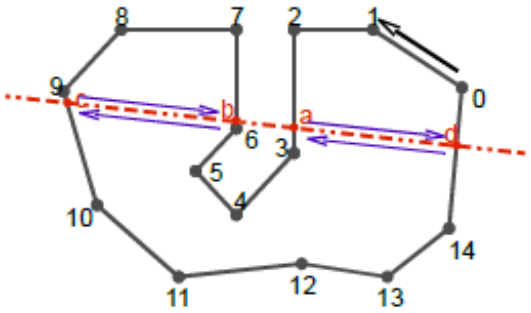
### 2.3.3 Hoonete genereerimine

Kvartalid on antud implementatsiooni puhul kõrvalteedega piiratud alad ning kujutavad seega hulknurki. Neid hulknurki vähendades jäetakse tee ning hoonete vahele kõnnitee jaoks ruumi.

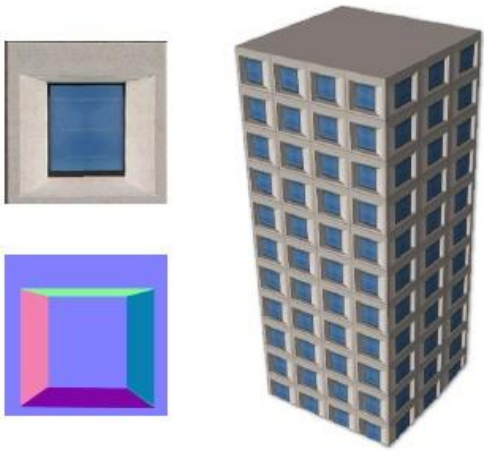
Kvartal jaotatakse edasi kruntideks ning selleks kasutatakse rekursiivset jagamise algoritmi (vt. joonis 16), mis iga jaotuse korral teeb neli sammu:

1. valib ala pikima külje ning arvutab külje hälbega keskpunktist ristsirge. *Külje hälbega keskpunkt* on selline punkt, mis on juhusliku suuruse võrra külje keskpunktist külje ühe või teise otspunkti suunas liigutatud;
2. arvutab leitud joone ja kvartali hulknurga lõikepunktid;
3. töötleb lõikepunkte ning loob graafi hoidmaks tekkinud lõike, mis jäävad kvartali hulknurga sisse;
4. liidab lõikumiste graafi servad ja kvartali hulknurka kujutava graafi ning eraldab saadud graafist tsüklid, mida on võimalik omakorda edasi jaotada.

CityGen süsteemis täiendati seda algoritmi nii, et teedega piirnevad maa-alad jaotatakse esimesena ning selle tulemusel on krundid enamasti teega risti. Samuti paiknevad ka reaalelus linnades krundid enamasti teega risti. Kruntidele, millel ligipääs teele puudub, hooneid ei genereerita.




Joonis 16. Kvartali jaotamine



Joonis 17. Tekstuur, normaalikaart ning hoone.

Erinevatel kruntidel on parameetrite hulgas märge, mis tüüpi hoone sinna luuakse. Hoone tüübist sõltub hoone suurus ning paigutus krundil.

Hoone põrandaplaaniks on vähendatud krundi hulknurk ning hoone loomiseks eendatakse põrandaplaani üles, et luua ruumiline kujund. Kuna loodavate hoonete kujundid on väga lihtsad, siis on detailsuse suurendamiseks tekstuuridele lisaks kasutusel ka normaalikaart (ing. k. *normal map*) (vt. joonis 17). *Normaalikaart* on pilt, mille igas pikslis hoitakse suunda. Normaalikaardiga mõjutatakse valgustuse arvutamist, et kujund tunduks keerulisema kujuga ilma geomeetriat muutmata.

# 3. Endaloodud linnade genereerimise süsteem

Käesolevas töös implementeeritud süsteem on loodud eesmärgiga, et linnad genereeritaks ilma kasutajapoolseid sisendandmeid vajamata. Kasutajal on samas võimalik muuta parameetreid ning genereerida teistsuguste omadustega linnu.

Esmalt genereeritakse maastik, milleks kasutatakse kaastudengi loodud generaatori [7] modifikatsiooni. Käesoleva töö autori poolt maastiku generaatorile tehtud modifikatsioonideks on maastikule antava värvi muutmine ning veekogude eristamine. Maastikuga seotud kasutaja muudetavateks parameetriteks on maastiku minimaalse ja maksimaalse veekogude protsendid, mis määravad genereeritava maastiku veekogude osakaalu.

Seejärel arvutatakse kolm juhuslikku punkti, mis ei asu veekogus ning millega määratakse rahvastikutihedus. Peale seda alustatakse põhiteede genereerimisega ühest varasemalt arvutatud punktist. Edasi genereeritakse kõrvalteed, mida kasvatatakse põhiteedest. Teedevõrgu omaduste mõjutamiseks saab kasutaja eraldi muuta loodavate põhiteede ja kõrvalteede lõikude arve ning põhiteede erinevate hargnemissuundade tõenäosust.

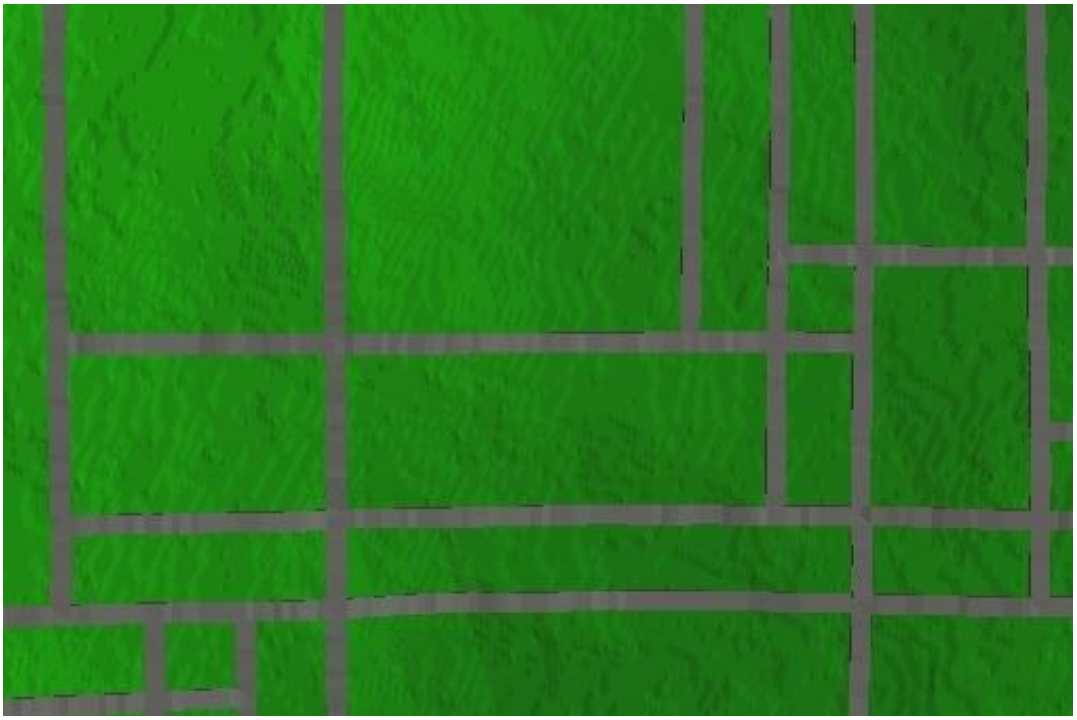
Pärast teedevõrgustiku genereerimist eraldatakse sealt kvartalid ning kvartalitest omakorda krundid. Kruntidele genereeritakse hooned. Hoonete maksimaalset kõrgust on kasutajal võimalik parameetritega muuta.

## 3.1 Põhiteede genereerimine

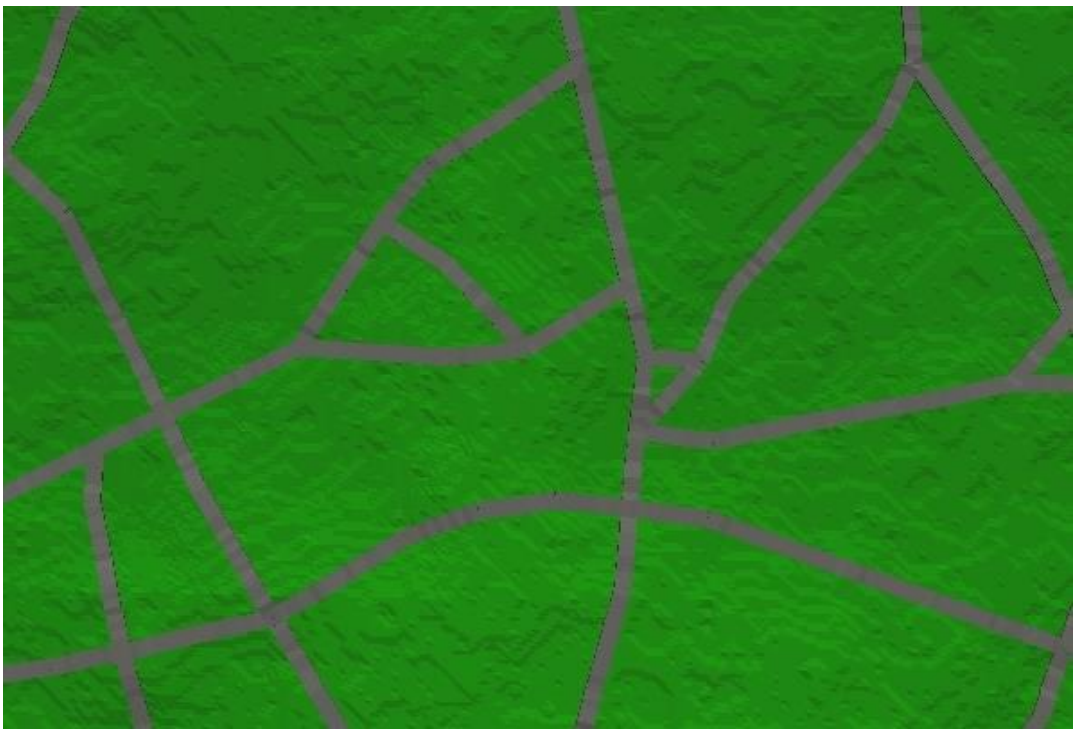
Teedevõrgustikku kujutatakse suunamata graafina ning teedevõrgustiku andmeid külgnevusmaatriksina nagu CityGen'is, kuid käesolevas implementatsioonis on kasutuses ainult üks graaf. Rakenduses on graafi külgnevusstruktuuri hoidmiseks sõnastik (ing. k. *dictionary*), kus võtmeteks on tipud ning võtmetele vastavateks väärtusteks tipust väljuvate kaarte loetelu sisaldavad järjendid.

Põhiteede genereerimist alustatakse ühest linnakeskuse punktist ning kasvatatakse esimesel sammul neljas suunas korraga, moodustades ristmiku. Seejärel kasvatatakse teedevõrgustikku iteratiivselt edasi saadud teelõikude otspunktidest. Igal sammul võetakse kõigepealt arvesse rahvastiku tihedust, et selgitada, kas on vajadust tee hargnemist kontrollida. Tänu sellele on väiksema rahvastikutihedusega piirkondades vähem hargnevad põhiteed. Kui tee hargnemist kontrollitakse, siis määratakse, kas tee hargneb vasakule, paremale, mõlemas suunas või jaguneb kaheks. Tee kasvab igal sammul ka otse suunas välja

arvatud juhtudel, kus veekogu pole võimalik ületada ning veekogu vältimiseks vajalik pöördenurk on liiga suur või toimus tee jagunemine kaheks. Iga kasvava teelõigu alguspunkt, pöördenurk ja tüüp antakse edasi teelõigu genereerimise algoritmile.



Joonis 18. Põhiteed kõrge rahvastikutihedusega piirkonnas.



Joonis 19. Juhuslikult kasvavad teed.

Kõrge rahvastikutihedusega piirkondades kasvavad hargnevad teed täisnurga all kasvatatava tee suhtes ning otsesuunas loodava teelõigu pöördenurk on 0 kraadi. Sedasi on loodud teededvõrgustik sarnane ruudustikule (vt. joonis 18).

Madalama rahvastikutihedusega piirkondades on hargnevate teede pöördenurgale lisatud juhuslik nurk -5 ja 5 kraadi vahel ning otsesuunas loodava tee pöördenurgale lisatakse juhuslik nurk -20 ja 20 kraadi vahel. Teedevõrgustik on madalama rahvastikutihedusega piirkondades kurvilisem ning ebakorrapärasem kui kõrge rahvastikutihedusega piirkonnas (vt. joonis 19).

## 3.2 Kõrvalteede genereerimine

Kõrvalteed on kitsamad kui põhiteed ning kõrvalteede puhul on iga teelõik lühem kui põhiteede puhul. Samuti ei saa kõrvalteed kasvades luua sildu.

Kõrvalteede genereerimiseks kasutatakse samasugust algoritmi nagu põhiteede genereerimiseks, kuid mõningad kasvamisreeglid on erinevad. Kõrvalteed hargnevad ainult täisnurga all ja tee ei saa jaguneda kaheks (vt. joonis 20). Tee otsesuunas kasvamine ei ole

Joonis 20. Kõrvalteed kasvavad peamiselt ruudustikuna.

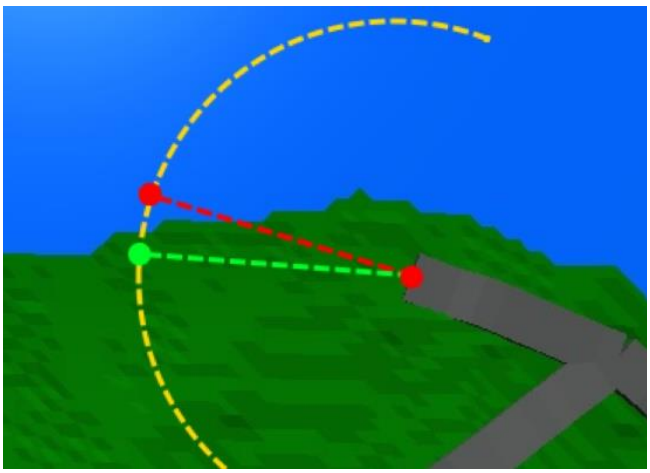
kindel, vaid sõltub rahvastikutihedusest antud piirkonnas ning eraldi konstandiga määratud tõenäosusest.

Esimese sammuna proovitakse kasvatada kõrvalteid kõikidest põhiteede tippudest, mis ei ole silla otspunktid ega hargnemiskohad ehk ristmikud. Seejärel kasvatatakse kõrvalteid edasi uutest loodud tippudest, kuni kõrvalteede arv jõuab parameetriga määratud maksimumini või kasv lõpeb.

Kui kõrvalteed on loodud, siis genereeritakse sama algoritmiga lisaks juurde ka hulk nähtamatuid kõrvalteid, et luua krunte ka piirkonda, mida nähtavalt teed ei ümbritse.

## 3.3 Teelõigu genereerimise algoritm

Teelõigu loomiseks on vaja algtippu, teelõigu suunda ning tee tüüpi. Tee tüübiks on kas põhitee või kõrvaltee ning sellega määratakse loodava teelõigu pikkus ja silla loomise võimalus. Alguses leitakse loodava teelõigu otsale vastava tipu asukoht, liikudes etteantud tipust teelõigu pikkuse konstandi võrra antud suunas. Seejärel kontrollitakse, kas saadud tipu asukoht jääb veekogusse.



Joonis 21. Alguses leitud teelõigu ots jäi veekogusse. Kollane punktiirjoon tähistab teelõigu uue otsa võimalikke asukohti.

Kui loodava teelõigu ots jääb veekogusse, siis kontrollitakse kõigepealt, kas teelõigu suunda kuni 90 kraadi võrra vasakule või paremale muutes jääks teelõigu ots endiselt veekogusse (vt. joonis 21). Juhul kui teelõigu ots jääb veekogusse ning tee tüübiks on põhitee, tuleb kontrollida, kas on võimalik sild luua. Selleks proovitakse teelõigu pikkust suurendada sammu kaupa kuni silla maksimaalse pikkuse konstandini. Igal sammul kontrollitakse punkte, mis jäävad teelõigu algpunktist teelõigu pikkuse kaugusele ja suunaga -30 ja 30 kraadi vahele. Kui leitakse punkt, mis veekogus ei asu, siis kontrollitakse, et selle läheduses ühtegi silla otspunkti ei leiduks, et vältida liiga lähestikku asuvate sildade loomist.

Kui teelõigu ots veekogusse ei jää, siis tee loomist jätkatakse. Selleks kontrollitakse, kas teelõigu otsa lähedal leidub olemasolevaid tippe. Juhul kui leidub teelõigu otsale lähedal asuvaid tippe valitakse neist lähim ning määratakse see teelõigu uueks otsaks. Seejärel kontrollitakse, et teelõik ei lõikuks ühegi olemasoleva teelõiguga. Lõpuks lisatakse teelõik külgnevusmaatriksisse.

Joonis 22. Teed järgivad veekogu piiri.

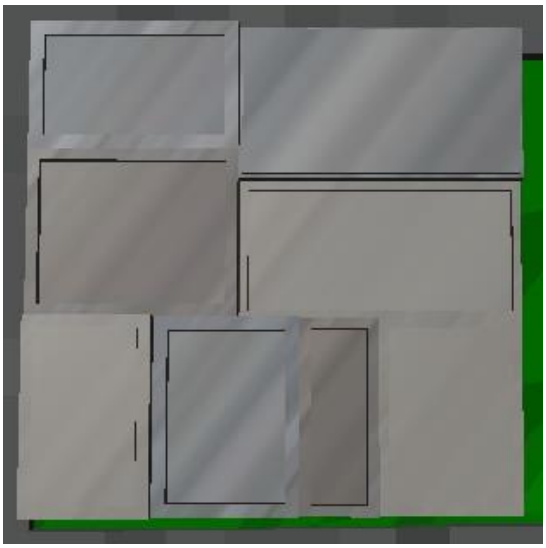
Teelõigu algne nurk lisatakse nurkade järjendisse ja teelõigu algse nurga ning lõpliku nurga vahe lisatakse nurkade muutude järjendisse. Kui teelõik muutis veekogu tõttu suunda, siis nurkade järjendisse lisatud nurk erineb teelõigu tegelikust nurgast. Tänu sellele kasvab selle teelõigu otsatipust uus tee edasi mööda veekogu äärt (vt. joonis 22).

## 3.4 Kruntide määramine

Krundid on alad, millele luuakse hooned. Kruntide määramiseks on tarvis kõigepealt leida kvartalid. Kvartalid on teedega ümbritsetud alad ehk teedevõrgustiku graafis on nad tsüklid, mis omakorda ühtegi teist tsüklit ei sisalda. Kvartalite leidmiseks tuleb seega graafist eraldada tsüklid.

Selleks, et graafist oleks lihtsam tsükleid eraldada, eemaldatakse graafist oksad, kustutades tippe, millest väljub ainult üks teelõik. Korratakse seda protsessi, kuni selliseid tippe enam ei leidu.

Tsüklite leidmiseks valitakse töödeldud graafist selliseid tipud, millest väljub vähemalt kolm teelõiku, kuna igas tsüklis leidub vähemalt üks tipp, millest väljub kolm teelõiku. Igast valitud tipust liigutakse rekursiivselt teelõike pidi, kuni jõutakse algse tipuni tagasi. Rekursiooni sügavusele on pandud ka piir ehk kui läbitud teelõikude arv jõuab lubatud piirini, siis tsüklit ei leitud. Algse tipuni jõudes kontrollitakse, et leitud tsüklis ei sisalduks omakorda tsüklit. Selleks tehakse kindlaks, kas tsüklis leidub tipp, millest teedevõrgustiku graafis väljub kaar mõnda teise samasse tsüklisse kuuluvasse tippu, mis ei paikne tsüklis antud tipu kõrval. Juhul kui selline tipp leidub, siis seda tsüklit tsüklite järjendisse ei lisata, kuna selles sisaldub omakorda teine tsükkel. Kui sellist tippu ei leidu, siis lisatakse tsükkel tsüklite järjendisse.



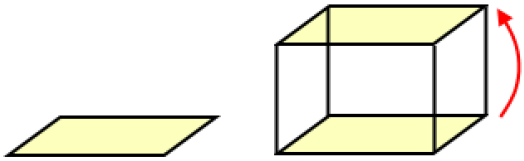
Joonis 23. Kvartali rekursiivse jaotamisega saadud krundid.

Kvartalitest kruntide eraldamiseks on rakenduses kasutatud hulknurga rekursiivset jaotamist, mis baseerub CityEngine'is ja CityGen'is kasutatud algoritmidel (vt. joonis 23).

Krundid, mille ükski nurk ei asu teeserval, jäetakse tühjaks.

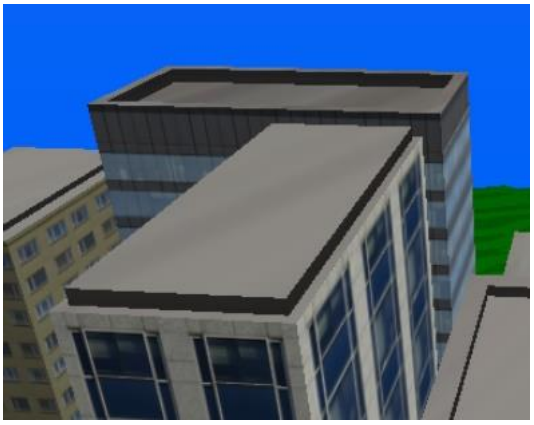
## 3.5 Hoonete genereerimine

Hoone kõrgus määratakse juhuslikult, kuid kõrge rahvastikutihedusega piirkonnas on minimaalne kõrgus suurem kui väikse rahvastikutihedusega piirkonnas. Samuti on kõrge rahvastikutihedusega piirkonnas võimalus, et tekib tavalisest tunduvalt kõrgem hoone. Rahvastikutiheduse vähenedes suureneb ka hoone genereerimata jätmise võimalus. Sel viisil on genereeritava linna siluett sarnane päris linnade siluettidele.



Joonis 24. Hoone kujundi eendamine üles suunas.

Hoonetel käsitletakse eraldi hoone esimest korrust, keskosa, viimast korrust ja katust. Hoone põrandaplaan on krundi hulknurga vähendatud kuju. Esimese sammuna luuakse hoone esimene korrus. Selleks eendatakse hoone põrandaplaani üles suunas (vt. joonis 24). Järgmiste sammudena luuakse samal viisil hoone keskosa ning viimane korrus. Viimasena genereeritakse hoonele katus.

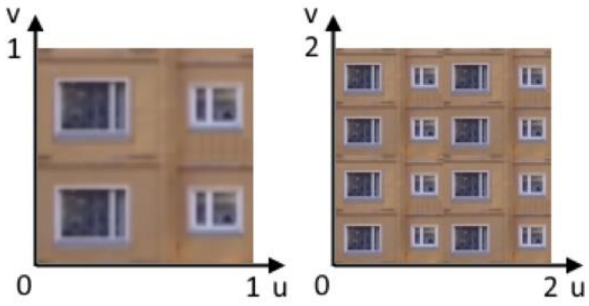


Joonis 25. Alla ja üles eendatud lamedad katused.

Joonis 26. Valik kasutatud tekstuure, alumisel real esimese korruse tekstuurid ning ülemisel real viimase korruse tekstuurid.

Madalamatele ristküliku kujulistele hoonetele genereeritakse suure tõenäosusega peale viilkatus. Selleks leitakse hoone kahe vastastikuse külje keskpunktid ning tõstetakse neid juhusliku suuruse võrra. Ülejäänud hoonetele genereeritakse lame katus, millel on kas kõrgendatud servad või kõrgendatud keskosa. Lameda katuse loomiseks võetakse esialgu hoone põrandaplaani kuju, vähendatakse seda juhusliku suuruse võrra ning lahutatakse

vähendatud kujund esialgsest kujundist. Seejärel eendatakse vähendatud kujundit üles või alla (vt. joonis 25).




Joonis 27. Tekstuuri koordinaadid. Ühel tekstuuril kujutatud korraga kahte korrust.

Hoonetele antakse detailne välimus kasutatades tekstuure. Hoone esimese kahe korruse, keskosa ja viimase kahe korruse jaoks on kasutusel erinevad tekstuurid (vt. joonis 26). Hoone iga külje jaoks määratakse tekstuuri koordinaadid, mille järgi tekstuurilt külje igale punktile vastav värvus võetakse. Tekstuurid peaksid hoonel paigutuma korrektselt ning korduma vertikaalselt sõltuvalt korruste arvust ning horisontaalselt sõltuvalt hoone külje laiusest. Selleks korrutatakse hoone küljele määratud tekstuuri v-koordinaati hoone korruste arvust poolega ning u-koordinaati hoone külje laiuse ja korruse kõrguse suhtega (vt. joonis 27). Sedasi tagatakse ka, et tekstuuri kõrguse ja laiuse suhe oleks sama.

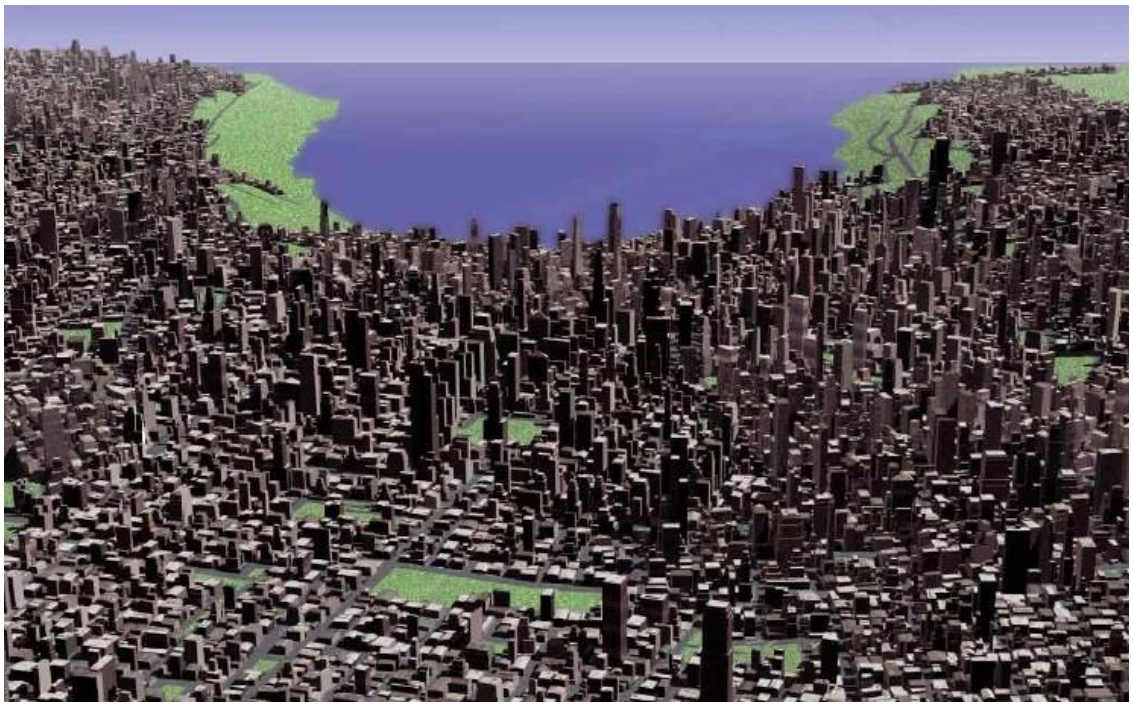
# 4. Tulemuse võrdlus

Käesolevas peatükis tuuakse välja erinevused eelmistes peatükkides kirjeldatud linnade genereerimise süsteemide poolt genereeritavate linnadega. Kuna lõpmatute linnade genereerimise süsteemi puhul on maapind tasane ning teedevõrgustikuks korrapärane ruudustik, siis selles peatükis kirjeldatakse antud süsteemi puhul ainult hooneid.

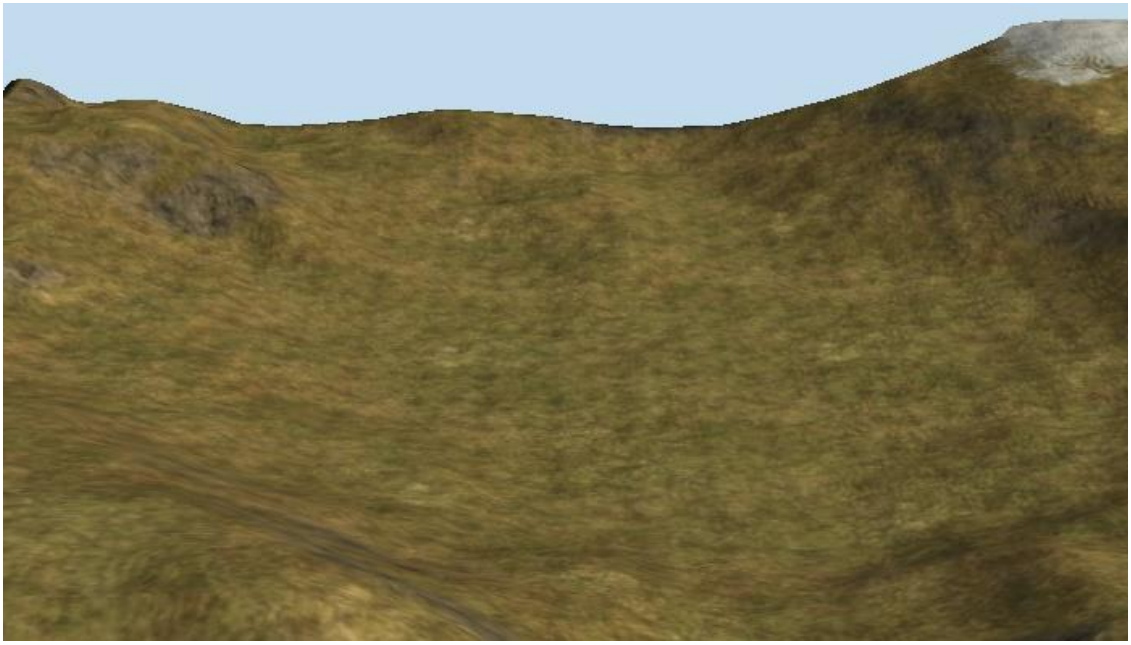
## 4.1 Maastik

Töö raames valminud süsteemi puhul on maastik üsna tasane ning seetõttu teede kulgemist ei mõjuta. Maastikust võivad teatud osa katta ka veekogud, millega loodavad teed arvestavad (vt. joonis 28).

Joonis 28. Töö raames valminud süsteemi poolt genereeritud linn.



Joonis 29. CityEngine maastik ja sellele genereeritud linn.



Joonis 30. CityGen maastik.

CityEngine süsteemis on maastik määratud kasutaja poolt sisendiks antud piltidega, mis määravad maastiku pinnareljeefi ja veekogud (vt. joonis 29).

CityGen süsteemis on maastik loodud kõrguskaardi järgi, mis on pildina kaasas. Veekogusid selles süsteemis ei leidu (vt. joonis 30).

Erinevalt teistest väljatoodud algoritmidest ei vaja käesolevas töös loodud süsteem maastiku loomiseks pilti. Selle asemel genereeritakse kasutajale maastik vastavalt tema poolt määratud veekogude tõenäosuse parameetritega. Seega on käesolev lahendus teiste lahendustega võrreldes parem kasutajatele, kes soovivad kiiresti geneerida linna juhuslikule maastikule.

## 4.2 Teedevõrgustik

Käesolevas töös valminud süsteemis on genereeritav teedevõrgustik (vt. joonis 31) sarnane CityEngine süsteemis genereeritava teedevõrguga (vt. joonis 32) ning teede genereerimiseks kasutatud algoritm sarnaneb CityGen'is kõrvalteede genereerimiseks kasutatud algoritmile. Mõlemas süsteemis arvestatakse teede genereerimisel rahvastikutihedusega ja veekogudega ning mõlemad algoritmid suudavad genereerida sildu üle veekogude.

Joonis 31. Töö raames valminud süsteemi teedevõrgustik.

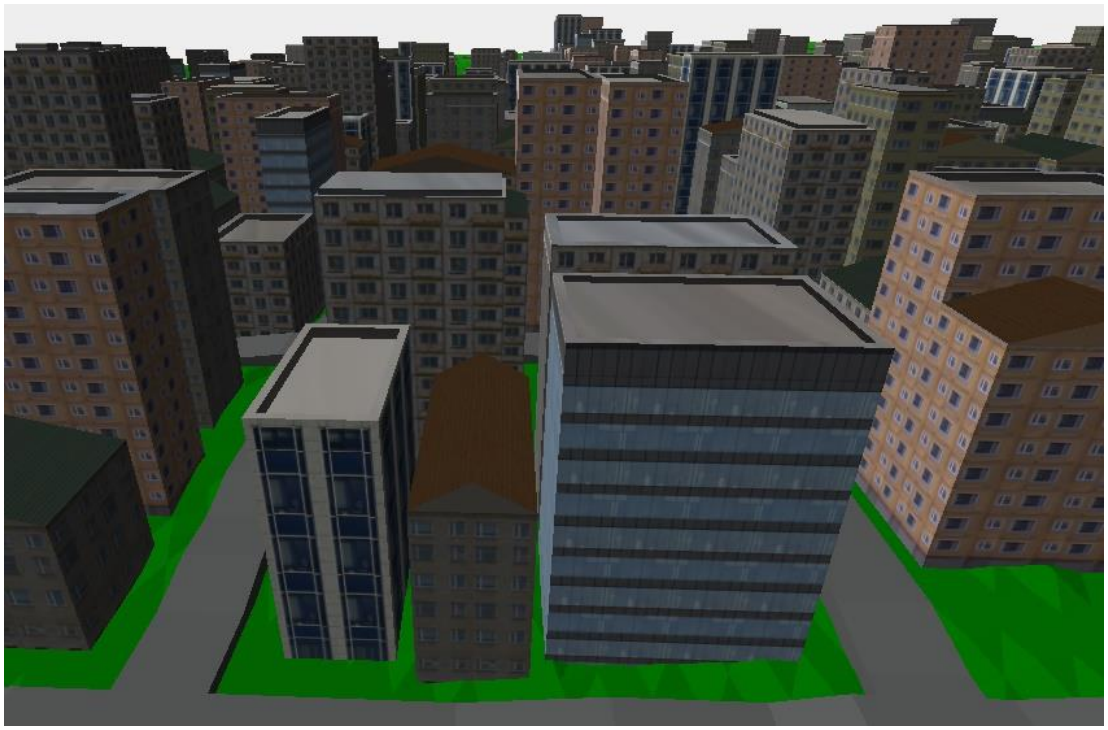
Joonis 32. CityEngine süsteemi teedevõrgustik.

Joonis 33. CityGen süsteemi teedevõrgustik.

CityGen süsteemis veekogusid ei leidu ning rahvastikutihedusega teede genereerimisel ei arvestata (vt. joonis 33).

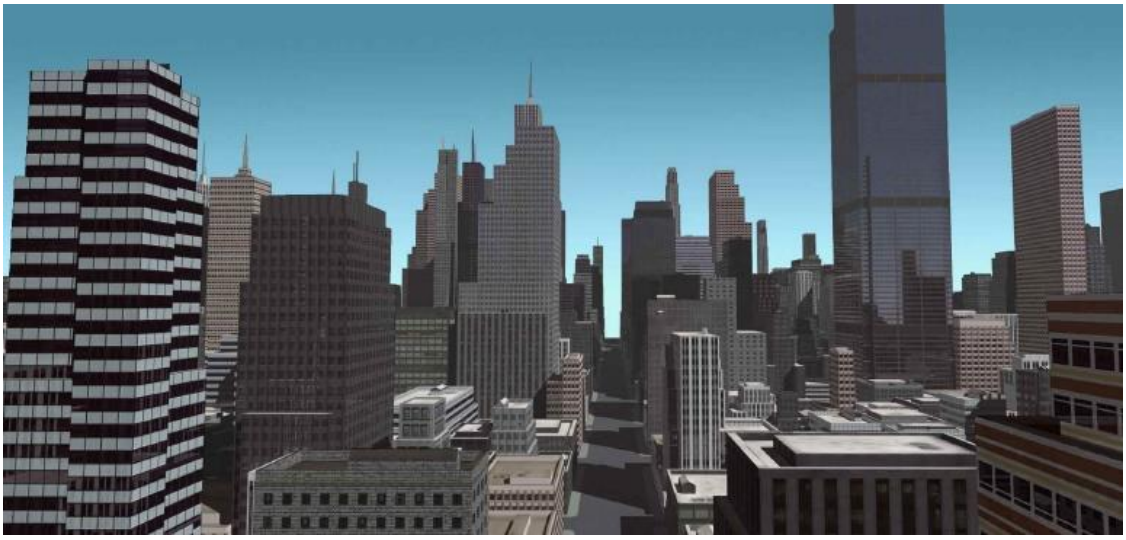
## 4.3 Hooned

Töö raames valminud süsteemis on hooned lihtsa geomeetriaga, kuid hoonetele on genereeritud erinevad katused (vt. joonis 34). Samuti on rakendusega kaasas hulk tekstuure, et hoonete fassaadid varieeruks.



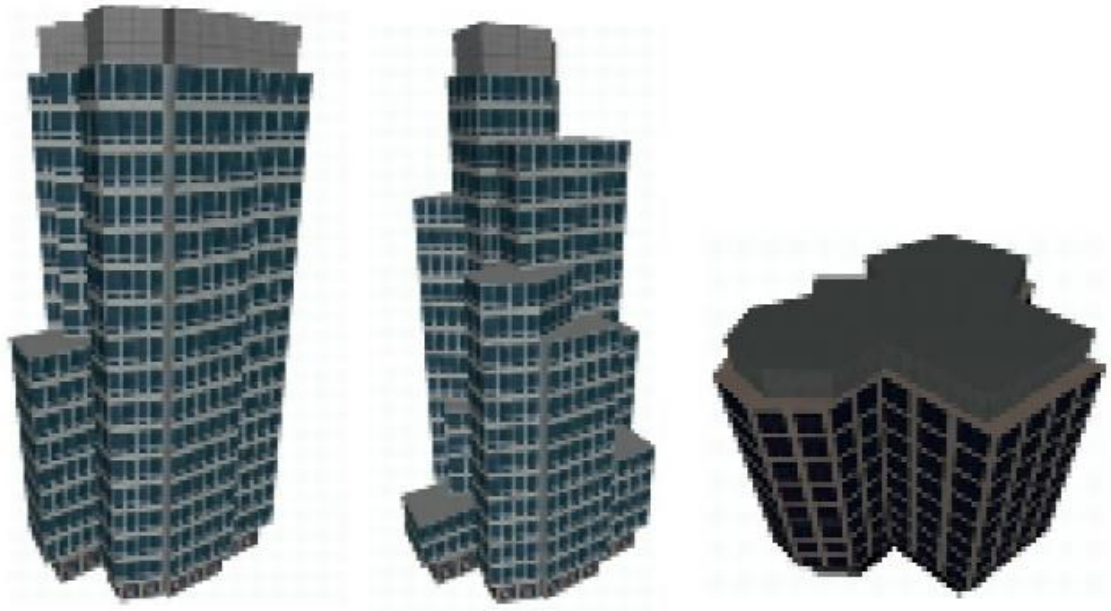
Joonis 34. Töö raames valminud süsteemis esinevad hooned.

CityEngine süsteemis on hooned genereeritud L-süsteemiga ning nende kuju on keerulisem (vt. joonis 35).



Joonis 35. CityEngine hooned.

Lõpmatute linnade genereerimise süsteemi [5] hooned on samuti keerukama kujuga (vt. joonis 36).



Joonis 36. Lõpmatute linnade genereerimise süsteemi hooned.

CityGen süsteemi genereeritud hooned on töö raames valminud süsteemi hoonetega sarnased (vt. joonis 37). Hoonete kuju tuleneb krundi kujust, mida on eendatud üles suunas ning hoonete katused on lamedad. Tänu normaalikaartide kasutamisele koos tekstuuridega tunduvad hooned väga detailsed. Samas on hooned paigutatud üksteisega kokku ilma vahesid jätmata.

Töö raames loodud süsteemiga võrreldes on teiste süsteemide hooned detailsemad. CityGen süsteemi genereeritud hoonete kuju on autori poolt loodud süsteemi genereeritud hoonetega võrreleldes veel lihtsam, kuid tänu normaalkaartide kasutamisele tundub, et hooned on näiliselt keerulisema kujuga.

Joonis 37. CityGen hooned.

# 5. Edasiarendusvõimalused

Töö raames valminud linnade genereerimise süsteemis kasutatavad maastikud on üpris tasapinnalised ning seetõttu ei olnud teede genereerimisel teelõigu suuna määramiseks tarvis maastiku kõrgusega arvestada. Mägise maastiku puhul oleks võimalik luua tunneleid ning teede genereerimisel tuleks kindlasti arvestada maastiku kõrguse muutumisega, et loodavad teed oleksid realistlikumad.

Üheks töö raames valminud süsteemi puuduseks on teedevõrgustiku detailsem visuaalne kujutamine. Teedel puudub tekstuur ning tee servades ei ole kõnniteid, mis linnades tavaliselt olemas on. Kuna teid genereeritakse antud süsteemis lõigu kaupa, siis teelõikude otsad ei ühti alati omavahel. Sellisel juhul oleks võimalik CityGen süsteemile sarnaselt teelõikude punktide põhjal neid interpoleeriv splain luua ning sealt täpsemad teelõigud eraldada.

Hoonete kujud on käesoleva töö raames valminud süsteemis üsna lihtsakoelised. Hoonetele keerulisema kuju andmiseks oleks võimalik kasutada töös kirjeldatud eksisteerivates süsteemides kasutatavaid hoonete genereerimise algoritme. Visuaalse detailsuse suurendamiseks võiks tekstuuridele lisaks kasutada normaalikaarti nagu CityGen süsteemis.

# 6. Kokkuvõte

Käesolev töö kirjeldas linnade protseduurilise genereerimise probleemi. Referatiivselt kirjeldati kolme lahendust. Töö põhitulemusena pakuti välja teistest linna genereerimise süsteemidest inspireeritud töö käigus valminud süsteem.

Autori poolt loodud süsteem genereerib linna ilma kasutajapoolseid sisendandmeid vajamata ning võimaldab samas kasutajal interaktiivselt muuta parameetreid, mis mõjutavad genereeritava linna erinevaid omadusi. Genereeritakse vastavalt rahvastiku tihedusele varieeruv teedevõrgustik, mis on jaotatud põhi- ja kõrvalteedeks. Üle veekogude genereeritakse sildu. Teevõrgustiku põhjal luuakse krundid, millesse genereeritakse ida-eurooplaslikud hooned. Hoonetele välimuse andmiseks paigutatakse neile tekstuurid.

Töös on toodud ka võrdlus teiste algoritmidega. Võrdlusest lähtub, et autori poolt töö raames valminud süsteem genereerib lihtsakoelisemaid hooneid ja teevõrgustik on üpriski sarnane kahes eksisteerinud süsteemis genereeritavate teedevõrgustikega. Sellegipoolest on töös esitatud süsteemi lihtsam kasutada, sest erinevalt CityEngine süsteemist ei vaja see kasutajapoolset sisendfaili maastiku loomiseks ning samuti ei ole teedevõrgustiku loomiseks kasutajal tarvis tippe lisada nagu CityGen süsteemis.

Kui autor peaks süsteemi uuesti kirjutama, siis kasutaks ta mõnda teist platvormi, kuna suurte linnade korral jääb praeguse süsteemi jõudlus kehvaks.

# 7. Kasutatud materjalid


[1] Ken Perlin, "An Image Snythesizer," *ACM Siggraph Computer Graphics*, vol. 19, nr. 3, lk. 287-296, juuli 1985.

[2] Raimond-Hendrik Tunnel, "Protseduuriline puude genereerimine," Tartu Ülikool, Tartu, bakalaureusetöö 2012.

[3] Andreas Sepp, "Protseduuriline lõpmatu maastiku genereerimine," Tartu Ülikool, Tartu, bakalaureusetöö 2016.

[4] Yoav IH Parish and Pascal Müller, "Procedural modeling of cities," in *Computer Graphics and Interactive Techniques*, Los Angeles, 2001.

[5] Stefan Greuter, "Real-time procedural generation of 'pseudo infinite'cities.," in *Computer graphics and interactive techniques*, 2003, lk. 87-94.

[6] McCabe H Kelly G, "Citygen: An interactive system for procedural city generation," in *Game Design and Technology*, 2007, lk. 8-16.

[7] Lindenmayer A Prusinkiewicz P, *The algorithmic beauty of plants*.: Springer Science & Business Media, 2012.

[8] D.S., Musgrave, F.K., Peachey, D., Perlin, K. and Worley, S Ebert, *Texturing and Modeling: A Procedural Approach*, 2nd ed.: Academic Press, 1998.

[9] Rom R Catmull E, "A class of local interpolating splines," in *Computer Aided Geometric Design*, märts 1974, lk. 317-326.

[10] Ülo Kaasik, *Matemaatikaleksikon*, 3rd ed. Tartu: Atlex, 2003.

[11] Andreas Sepp. (2016) Midpoint Displacement Algorithm (MDA). [Võrgumaterjal]. http://morsakabi.com/veeb/ (vaadatud 12.05.2016)

# Lisad

## I. Terminid

| | |
|---|---|
| Interpoleerimine<br><br>Funktsiooni väärtuse hindamine piirkonnas, mille mõnes punktis on funktsiooni või selle tuletise väärtused teada [8]. | Interpolation<br><br>Estimating the value of a function in an area in which some of the data points or their derivatives are known. |
| L-süsteem<br><br>Paralleelne ümberkirjutamise süsteem ja formaalne grammatika [4]. | L-system<br><br>Parallel rewriting system and a type of formal grammar. |
| Normaalikaart<br><br>Normaalikaart on pilt, mille igas pikslist hoitakse suunda, mida nimetatakse normaaliks. Normaalikaardiga lisatakse objektidele detaile ilma hulknurki lisamata. | Normal map<br><br>Image that stores a direction at each pixel. These directions are called normals. Normal maps are used to add details to objects without using more polygons. |

| Produktsioon | Production |
| --- | --- |
| Produktsioon $(a,\ \chi) \in P$ kirjutatakse $a \rightarrow \chi$ ja on eeldatud, et iga tähe $a \in V$ korral on vähemalt üks sõna $\chi \in V^*$ nii, et eksisteerib vastav $a \rightarrow \chi$, kui hulk $V$ on süsteemi tähestik, $V^*$ kõigi tähestiku $V$ poolt moodustatud sõnade hulk ning $P \subset V \times V^*$ on lõplik produktsioonide hulk. Kui ühtegi sellist produktsiooni ei ole hulgas $P$, siis eeldatakse, et seal eksisteerib samasusprodukt sioon $a \rightarrow a$. | A production $(a,\ \chi) \in P$ is written as $a \rightarrow \chi$ and it is assumed that for any letter $a \in V$ there is at least one word $\chi \in V^*$ such that $a \rightarrow \chi$, where $V$ denotes an alphabet, $V^*$ is the set of all words over $V$ and $P \subset V \times V^*$ is a finite set of productions. If no production is explicitly specified for a given letter $a \in V$, the identity production $a \rightarrow a$ is assumed to belong to the set of productions $P$ [7]. |
| Splain | Spline |
| Lõigul $[a, b]$ määratud funktsioon, mis teatava lahutuse $a = x_0 < x_1 < \ldots < x_n = b$ korral on igas vahemikus $(x_k, x_{k+1})$ esitatav polünoomina. Enamasti nõutakse täiendavalt veel funktsiooni ja selle teatavat järku tuletise pidevust jaotuspunktides $x_k$ [8]. | Spline is a function on an interval $[a, b]$ composed of $n$ subintervals $[x_k, x_{k+1}]$ with $a = x_0 < x_1 < \ldots < x_n = b$. Spline is required to be both continuous and continuously differentiable at the interior points $x_k$. |

## II. Litsents

**Lihtlitsents lõputöö reprodutseerimiseks ja lõputöö üldsusele kättesaadavaks tegemiseks**

Mina, **Kristjan Perli**,

1. annan Tartu Ülikoolile tasuta loa (lihtlitsentsi) enda loodud teose
   **Protseduuriline linnade genereerimine**,
   mille juhendajad on Ahti Peder ja Raimond-Hendrik Tunnel,
   1.1. reprodutseerimiseks säilitamise ja üldsusele kättesaadavaks tegemise eesmärgil, sealhulgas digitaalarhiivi DSpace-is lisamise eesmärgil kuni autoriõiguse kehtivuse tähtaja lõppemiseni;
   1.2. üldsusele kättesaadavaks tegemiseks Tartu Ülikooli veebikeskkonna kaudu, sealhulgas digitaalarhiivi DSpace´i kaudu kuni autoriõiguse kehtivuse tähtaja lõppemiseni.
2. olen teadlik, et punktis 1 nimetatud õigused jäävad alles ka autorile.
3. kinnitan, et lihtlitsentsi andmisega ei rikuta teiste isikute intellektuaalomandi ega isikuandmete kaitse seadusest tulenevaid õigusi.

Tartus, **14.05.2016**