Tartu Ülikool
Loodus- ja täppisteaduste valdkond
Tehnoloogiainstituut

Aap Vare

Linna protseduuriline genereerimine

Bakalaureusetöö (12 EAP)
Arvutitehnika eriala

Juhendaja:
Raimond-Hendrik Tunnel, MSc

Tartu 2020

Resümee/Abstract

Linna protseduuriline genereerimine

Protseduuriline genereerimine leiab arvutigraafikas kasutust suurte virtuaalsete maailmade loomises. See lubab väiksemate failidega luua palju andmeid, tekitades seejuures juhuslikkust, mis muudab genereeritavad andmed ettearvamatuks. Antud töö eesmärk on luua ja kirjeldada süsteemi linna protseduuriliseks genereerimiseks. Lahendus võimaldab tekitada juhusliku välimusega linna, mis võtab arvesse kasutaja etteantud parameetreid. Linn sisaldab endas teedesüsteemi, visuaalselt eristatavaid rajoone ja puid. Lisaks antud töö raames välja töötatud lahendusele kirjeldatakse eelnevalt tehtud teadustööd keskkonna protseduurilises genereerimises. Antakse ettepanekuid ka valminud algoritmi edasiarendamiseks.

CERCS: P150: Geomeetria, algebraline topoloogia; P170, Arvutiteadus, arvutusmeetodid, süsteemid, juhtimine; P175: Informaatika, süsteemiteooria

Märksõnad: protseduuriline genereerimine, arvutigraafika, linna genereerimine, teede genereerimine, Unity

Procedural generation of a city

Procedural generation is used in computer graphics to construct large virtual worlds. It allows to create a significant amount of data with a small number of files, producing randomness in the process which makes the generated data unpredictable. The aim of this thesis is to create and describe a system for the procedural generation of a city. The solution allows to create a city with a random appearance that takes into account parameters given by the user. The city includes a road system, visually distinguishable regions, and trees. In addition to the solution developed in this thesis, previously created scientific work in the field of procedural environment generation is described. Proposals for further development of the created algorithm are also given.

CERCS: P150: Geometry, algebraic topology; P170: Computer science, numerical analysis, systems, control; P175: Informatics, systems theory

Keywords: procedural generation, computer graphics, city generation, teede genereerimine, Unity

Sisukord

Jooniste loetelu

Tabelite loetelu

Lühendid, konstandid, mõisted

L-süsteem – paralleelne ümberkirjutamise süsteem ja formaalse grammatika tüüp [1].

Perlini müra – protseduuriline tekstuur, mille abil on võimalik arvutigraafikas luua realistlikke looduslikke mustreid [2].

Produktsioon – produktsioon $(a, \chi) \in P$ kirjutatakse kujul $a \rightarrow \chi$ ja eeldatakse, et iga tähe $a \in V$ korral on olemas vähemalt üks sõna $\chi \in V^*$ nii, et $a \rightarrow \chi$, kus V on tähestik, V^* on kõigi tähestiku V sõnade hulk ja $P \subset V \times V^*$ on lõplik produktsioonide hulk [1].

1 Sissejuhatus

Arvutimängude otstarbeks luuakse üha suuremaid ja detailsemaid virtuaalseid keskkondi. See nõuab omakorda ka suuremaid faile, mis sisaldavad andmeid virtuaalse maailma loomiseks. Andmemahtude kokku hoidmiseks ja suure keskkonna loomiseks kasutavad mitmed mängud protseduurilist genereerimist. Kõige populaarsemad ja hiljutisemad nende seas on „Minecraft" (2009) [3] ja „No Man's Sky" (2016) [4].

Protseduuriline genereerimine on andmetöötluse meetod, millega luuakse andmeid arvuti protseduuriga ehk algoritmiga. Seda kasutatakse arvutigraafikas muuhulgas tekstuuride ja 3D-mudelite tegemisel. Põhiliselt leiab protseduuriline genereerimine kasutust arvutimängudes mängu sisu automaatseks tegemiseks. See võimaldab väiksemate failidega luua suure koguse andmeid, tekitades seejuures juhuslikkust, mis muudab genereeritavad andmed ettearvamatuks[1].

Käesolevas bakalaureusetöös tehti algoritm, mis võimaldab protseduuriliselt genereerida 3D linna võttes arvesse etteantud parameetritena tihedust ja pindala. Linna loomisel on rõhku pandud erinevatele rajoonidele ja sellele, et nad oleksid üksteisest eristatavad. Varem on ka tehtud bakalaureuse- ja magistritöid protseduurilisest genereerimisest [5,6,7]. Seda on tehtud seoses nii loodusliku kui ka tehisliku ehk inimeste loodud keskkonnaga. Antud uurimistöö keskendub rohkem realistlikult planeeritud linna genereerimisele. Tulemus sisaldab teedesüsteemi ja erinevat tüüpi hooneid. Algoritm on realiseeritud kasutades Unity mängumootorit[2] ja C# programmeerimiskeelt. Implementatsiooni lähtekood on avalikult saadaval GitHubi repositooriumis[3]. Algoritmi demonstreeriv rakendus on lisas 1.

Uurimistöö käigus uuritakse erinevaid eksisteerivaid lahendusi sarnastel teemadel ja kasutatakse meetodeid, mis on kasulikud antud eesmärgi täitmiseks. Töö koosneb kuuest peatükist. Teine peatükk räägib kolmest erinevast olemasolevast lahendusest virtuaalse keskkonna protseduuriliseks genereerimiseks. Kolmas peatükk kirjeldab käesoleva töö käigus valminud algoritmi. Neljas peatükk võrdleb valminud generaatorit olemasolevate lahendustega. Viiendas peatükis arutletakse edasiarendamise võimaluste üle.

[1] https://et.wikipedia.org/wiki/Protseduuriline_genereerimine
[2] https://unity.com/
[3] https://github.com/Fa1nt/CityGenerator

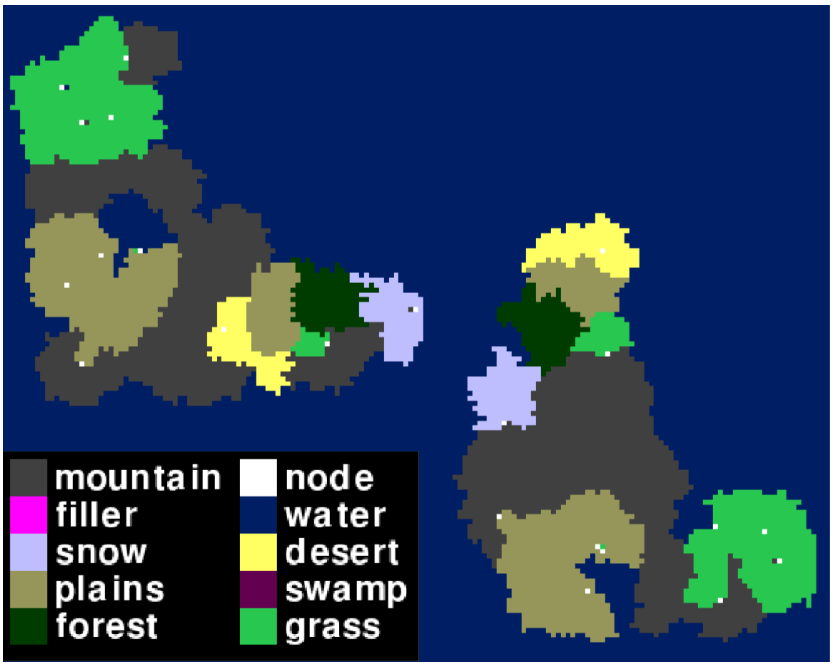
2 Olemasolevad lahendused

Käesolevas peatükis on kirjeldatud olemasolevad lahendused erinevate keskkondade protseduuriliseks genereerimiseks. Selleks, et luua läbimõeldud lahendus käesoleva bakalaureusetöö eesmärgi saavutamiseks, uuriti kolmes teaduslikus artiklis kirjeldatud süsteeme. Nende pealkirjad on „Procedural Generation of Story-Driven Maps" [8], „Procedural City Generator" [9] ja „Procedural Content Generation of Villages and Road System on Arbitrary Terrains" [10]. Järgnevalt on välja toodud aspektid artiklites räägitud generaatoritest, mis seostuvad antud töö raames valminud süsteemiga.

2.1 Procedural Generation of Story-Driven Maps

Matthewsi ja Malloy [8] 2011. aastal avaldatud artiklis on esitletud süsteemi kaartide genereerimiseks põhjaliku loojutustamisega arvutimängude jaoks. See meetod on mõeldud sellise maailma loomiseks, mis koosneb arendaja määratud piirangutega sobivalt paigutatud linnadest. Seda generaatorit nimetatakse artiklis MapGeniks. Antud süsteem koosneb kolmest etapist.

Esimeses etapis loetakse sisse kaardi spetsifikatsioon, mis sisaldab endas kasutaja poolt defineeritud atribuute. Nende hulgas on linnade nimed, ümbritsev maastik, linnade arv, genereeritava kaardi suurus, ruudustiku suurus, maastiku genereerimise sõlmed ja ühendused kaardi sõlmede vahel. Sõlmed on selles lahenduses olulised asukohad kaardil. Need võivad sisaldada linnu, koopaid või muid paiku, mida võib olla vajalik mängumaailmas luua. Esimese etapi tulemuseks on objektid Pythoni koodis, mille põhjal on võimalik luua neile vastavad kujutused kaardil. Teises etapis kasutatakse neid initsialiseeritud objekte mootori loomiseks. See mootor teostab simulatsiooni, kuni süsteem saavutab stabiilse lahenduse. See lahendus koosneb sõlmede nimedest ja nende (x, y) koordinaatidest kaardil, mis rahuldavad kasutaja määratud piiranguid. Kolmandas etapis kasutatakse seda informatsiooni kaardistaja initsialiseerimiseks. See on Pythoni klass, mis paneb paika genereeritud kaardi ülesehituse. Kaardistaja hakkab täitma kaarti ja kontrollib iga sõlme juures piiranguid, et koostatav kaart (vt. joonis 1) vastaks kõigile kasutaja nõuetele.



Joonis 1. MapGeniga genereeritud kaks kontinenti [8].

Sarnaselt käesoleva bakalaureusetöö süsteemiga genereerib MapGen maailma, mis koosneb erinevat sorti piirkondadest. Antud juhul on need piirkonnad maastikutüübid. Sõlmed sisaldavad endas infot selle kohta, missugust maastiku on vaja enda ümbruses tekitada. Sõlmed paigutatakse kaardile esialgu juhuslikult. Pärast seda luuakse igale sõlme jaoks oma füüsiline keha ning kasutatakse füüsikamootorit (ing. k. *physics engine*), et nihutada need sobivamatesse kohtadesse. Kehade nihutamine sõltub kasutaja määratud piirangutest algoritmile. Selle tagajärjel tekib kaart, kus leidub erinevat sorti maastikke juhuslikes, kuid samas loogiliselt paigutatud kohtades.

2.2 Procedural City Generator

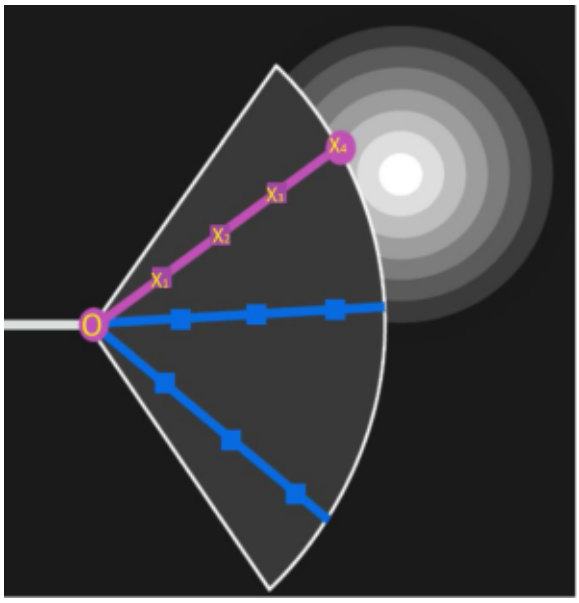
R. Sharma [9] 2016. aastal loodud süsteem genereerib linnu kasutaja piiratud sekkumisega. Selles generaatoris mõjutab autoteede mustrit rahvastiku tihedus. Pildi kujul antakse sisendiks rahvastiku hall-skaala, mille järgi joonistatakse teed.

Procedural City Generatori süsteem loob primaarsed ja sekundaarsed teed. Primaarsed teed on mõeldud ühendusteks tihedalt asustatud alade vahele. Pildi kujul antakse sisendiks rahvastiku tiheduse hall-skaala, mille järgi joonistatakse kiirteede võrgustik. Valge värv, mille hall-skaala väärtus on 1, kujutab kõrge rahvaarvuga ala ja must värv, mille hall-skaala väärtus on 0, kujutab inimesteta ala. Alguses otsib süsteem alasid, millel on hall-skaala väärtus 1 ja

lisab need alad rahvastiku sõlmede alla. Rahvastiku sõlmed on sektorid, mida on vaja kiirteedega ühendada.

Kiirteede genereerimist alustatakse ühest rahvastiku sõlmest. Need ulatuvad ka teistesse asustatud aladesse, mis on hall-skaalal nähtavad. Kiirteed levivad kiirtega, mida joonestatakse radiaalselt ettemääratud raadiustega iga tee lõpust. Kiire pealt võetakse punktid X kaugusel D ja arvutatakse SUM kujul

$$SUM = \sum_{i=1}^{n} (X_i / D_i) .$$

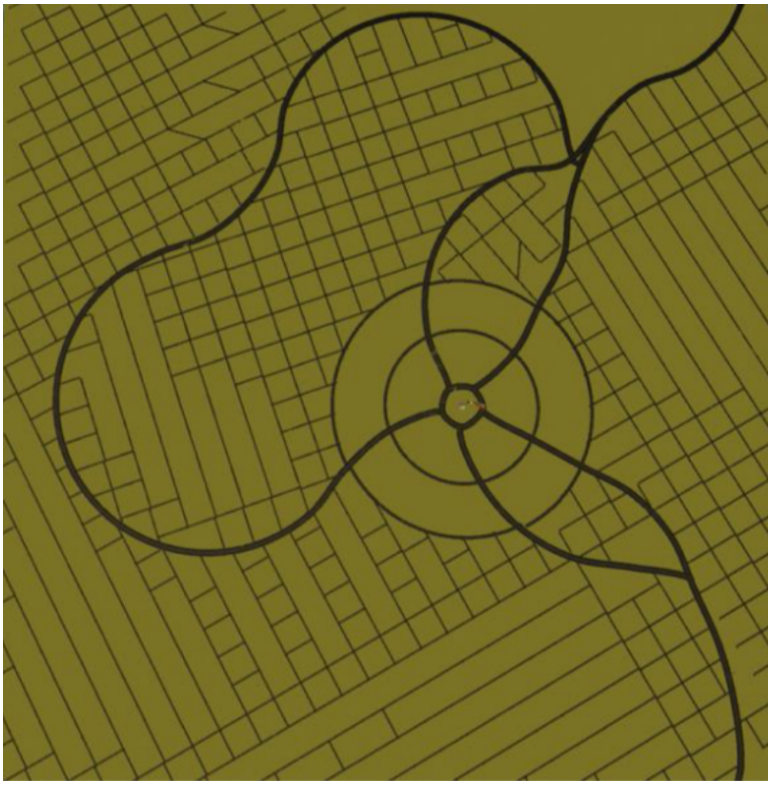


Joonis 2. Tee lõpust väljuvad kiired ja nende peal asuvad punktid, mille põhjal arvutatakse SUM [9].

Seda protsessi korratakse iga kiire peal (vt. joonis 2). Kõige suurema SUM väärtusega kiire suunas jätkatakse tee genereerimist ja edasine teede genereerimine kulgeb samalaadselt.

Tänavad, mis jäävad kiirteede vahele, on sekundaarsed teed. Need on mõeldud kiirteedele ligipääsu tagamiseks ja transpordiks kohalike tihedalt asustatud alade vahel. Sekundaarseid teid genereeritakse kasutades L-süsteemi. See on süsteem sõne ümberkirjutamiseks. Seda initsialiseeritakse esialgse sõnega koos produktsioonireeglitega, mis määravad igale sõnes esinevale sümbolile toimingu [1].

Sharma loodud algoritm saavutab sarnased eesmärgid antud bakalaureusetööga, sest tegemist on linna generaatoriga. Artiklis kirjeldatud teedesüsteemis on olemas nii laiad primaarsed kui ka peenemad sekundaarsed teed (vt. joonis 3). Hall-skaala abil tihedamate asustuste määramine tagab piisavalt juhuslikkust.



Joonis 3. City Generatori teedevõrgustik [9].

Samas ei pruugi ainult sellest piisata, kui on vaja tekitada realistliku rajoonide paigutusega linna. L-süsteemiga genereeritud teedevõrgustik on hea lahendus, sest see tagab piisavalt ühtlase mustri, ent samal ajal tekitab erinevate pikkustega tänavaid. Peateede süsteem ulatub üle piisavalt suure ala linnas. Tänu kiirte meetodile ei ole peateed rangelt sirged, vaid kulgevad pikkade kurvidega, nagu on kohane kõrgema kiirusega kiirteedele.

2.3 Procedural Content Generation of Villages and Road System on Arbitrary Terrains

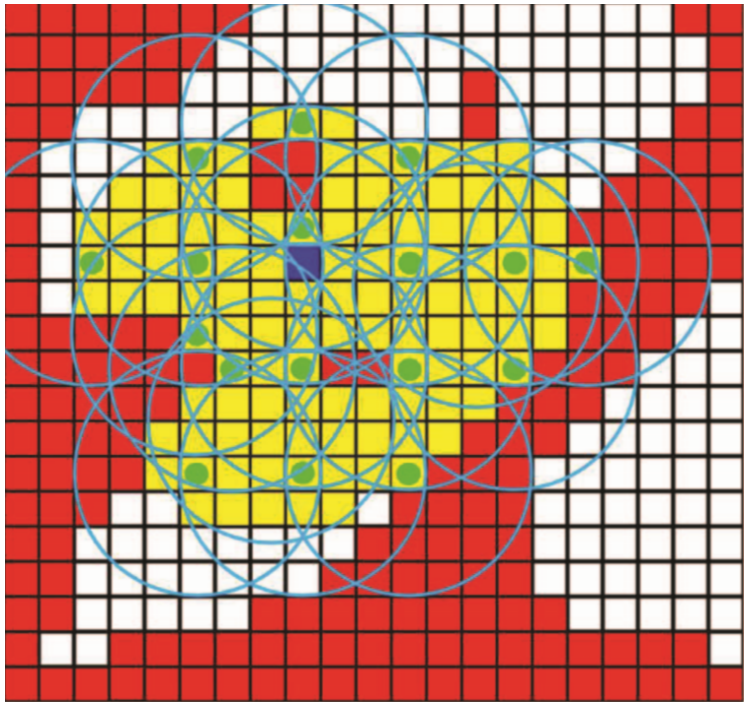
Mizdali ja Pozzeri [10] tehtud külade ja teede protseduuriline generaator hindab olemasoleva maastiku põhjal, millised kohad sobivad kõige paremini asulate rajamiseks. Selle järgi luuakse sõlmed ja iga sõlme järgi tehakse külad. Selline süsteem võimaldab muuta külade asukohti sõlmede muutmisega. Sealjuures rajatakse autoteed A* algoritmiga[4].

Mizdali ja Pozzeri tehtud lahendus koosneb kolmest etapist. See võtab sisendiks protseduurilise meetodiga loodud maastiku. Kõigepealt analüüsitakse maastiku andes igale maastiku sõlmele väärtuse. See näitab, kui hästi sobib sõlm linna genereerimise

[4] https://en.wikipedia.org/wiki/A*_search_algorithm

alguspunktiks. Seejuures defineeritakse ka millised sõlmed milliste alguspunktide juurde kuuluvad. Teiseks valitakse sõlmed, kus genereeritakse asulad ja luuakse teesüsteem, mis neid ühendab. Teesüsteemi loomiseks ühendatakse punkt ühes külas punktiga teises külas. Pärast seda ühendatakse teised külad olemasoleva teega. Kolmandaks paigutatakse küladesse hooned, seejuures valides kohad, kuhu on võimalik hooneid rajada.

Sõlmed, kuhu paigutatakse külad, leitakse etteantud maastiku põhjal (vt. joonis 4). Nende hulgast valitakse välja ettemääratud arv sobilikke kohti C, mis asuvad üksteisest piisavalt kaugel. Selleks tehakse ümber sõlme N ring raadiusega R, mille seest valitakse välja võimalikult kaugel asuvad punktid, mille ümber tehakse omakorda samasugused ringid. Sõlm N jääb samas uute ringide sisse. Siis loetakse, kui palju on olemas aktsepteeritavaid sõlmi, mis jäävad kolme või enama ringi sisse. Kui see arv on väiksem kui C, siis alustatakse taaskord tsüklit tehes ringid ümber viimastele sõlmedele ja lugedes sobivate punktide arvu uuesti. Sellise tsükli tulemuseks on algoritm, mis võimaldab paigutada külasid kasutaja määratud suuruse ja tihedusega muutes R ja C väärtusi.



Joonis 4. Kollased ruudud on valitud ala. Valged ruudud tähistavad kasutatavat, punased mittekasutatavat ala. Rohelised täpid on laiendatud ringide keskpunktid [10].

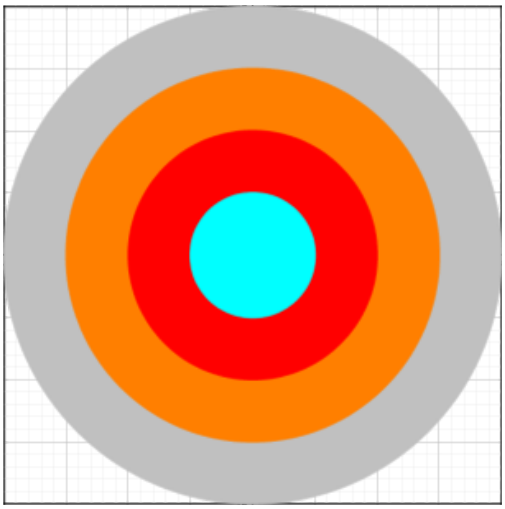
Sõlmede põhjal kaardi erinevate osade paigutamine on hea viis erisuguste piirkondade tekitamiseks. Samas käesolevas töös ei tegeleta maastiku genereerimisega, mistõttu ei ole võimalik linna rajoone selle alusel paika panna. A* algoritm on lihtne moodus teede rajamiseks oluliste punktide vahel. See sisuliselt leiab lühima võimaliku tee sõlmede vahel.

3 Loodud algoritm

Töö käigus valminud algoritm genereerib virtuaalset linna, mis koosneb neljast põhilisest osast. Need on:

1) primaarsed teed,
2) sekundaarsed teed,
3) hooned,
4) puud.

Algoritm alustab hall-skaala järgi sõlmede genereerimisega heledamates kohtades. Need sõlmed on punktid, mis ühendatakse laiade teedega, luues põhiteede süsteemi. Pärast seda genereeritakse üle linnaala peenemad radade hulgad. Seeläbi valmib tänavate võrgustik. Hooned genereeritakse olemasolevate teede vahel asuvatesse kvartalitesse. Tihedamalt asustatud piirkonnad sisaldavad rohkem äri- ja büroohooneid. Kesklinnast väljapoole minnes esineb ette rohkem elamupiirkondi ja kaugel linna servade lähedal ka tööstushooneid (vt. joonis 5). Algoritmil on kolm konfigureeritavat parameetrit, millega saab mõjutada linna välimust. Need on linna piirala kõrgus ja laius ning rahvaarv. Kõrgus ja laius määravad ala, mille piiride sisse genereeritakse linn. Rahvaarv paneb paika, kui tihedalt paigutatakse hooned.



Joonis 5. Ligikaudne kujutus rajoonide paiknemisest. Sinine tähistab pilvelõhkujaid, punane kaubapindadega hooneid, oranž korteried, hall eramaju. Tähistama ala sisse jäävad tööstushooned. Mustad piirjooned on linna piirid.

3.1 Primaarsed teed

Enne primaarsete teede tekitamist loob algoritm sõlmed, mis hiljem omavahel ühendatakse. Selleks, et sõlmed satuksid suvalistesse kohtadesse, on kasutatud Perlini müra[5]. Selles mürast valitakse välja heledamad kohad. Nende koordinaatidele tekitatakse sõlmepunktid. Selleks, et need punktid ilmuksid juhuslikesse paikadesse, on kasutatud pseudojuhusliku numbrigeneraatori seemet (ing. k. *seed*). Seda genereerib .NET meetod nimega `Random.Range`, mis tagastab arvu etteantud vahemikus.

Selleks, et luua teedevõrgustik, ühendatakse tekitatud sõlmed üksikute teedega (vt. joonis 6). Iga sõlme puhul on vaja kindlaks teha, millise teise sõlmega tuleb see ühendada. Antud algoritm teeb seda kauguse alusel. Iga sõlme juures arvutatakse välja, kui kaugel see asub teistest olemasolevatest sõlmedest. Teega ühendamiseks valitakse kõige lähemal asuv punkt.

Joonis 6. Kolm näidet protseduuriliselt genereeritud peateede võrgustikust.

Selline lahendus peateede loomiseks on tehtud Sharma algoritmi eeskujul. Selles töös ei ole implementeeritud kiirtega levitamise meetodit, mis võimaldaks kurvidega teid luua. Sellepärast on antud töös primaarsete teede süsteem pigem lihtsam sirgetest teedest koosnev võrgustik, mis loob otsesed ühendused sõlmede vahel.

3.2 Sekundaarsed teed

Sekundaarsed teed on kitsamad võrreldes eelnevalt genereeritud peateedega. Need täidavad palju suurema osa linnaalast. Sekundaarsete teede genereerimine koosneb kahest etapist. Esimesena tekitatakse linnaalale ülevalt keskelt alustades esialgne teedesüsteem

[5] https://et.glosbe.com/et/en/Perlini%20m%C3%BCra

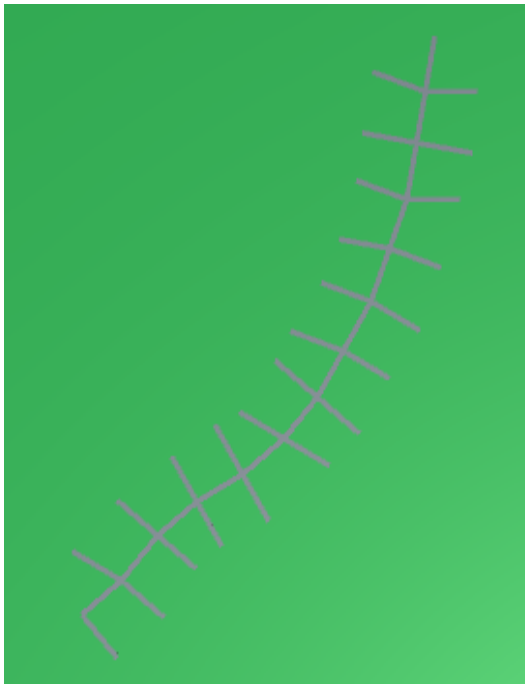
L-süsteemiga. See pikeneb, kuni jõuab linna piirini. Teises etapis laiendatakse olemasolevat teedevõrku edasi, kuni taaskord jõutakse piirideni. Tulemuseks on teedega täidetud ala.

L-süsteemiga genereeritakse tee sõnega ".F[-F][+F]X". Produktsioonireegel on “X → .F[-F][+F]X”. Tähiste funktsioonid on välja toodud tabelis Tabel 1.

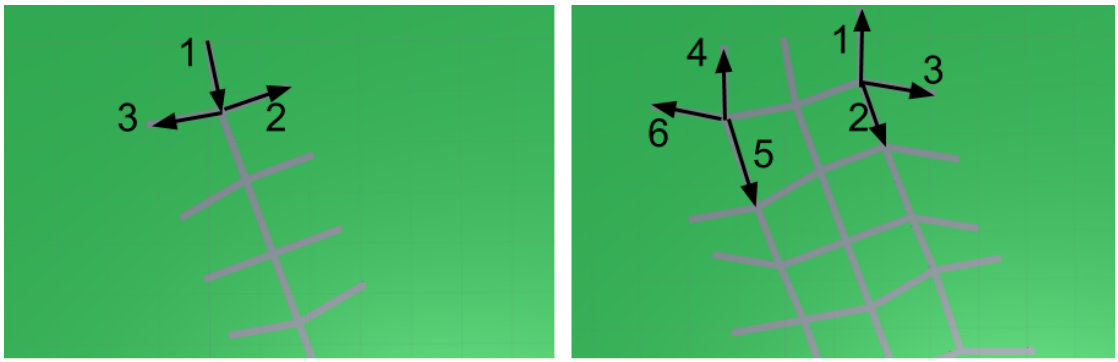
Tabel 1. L-süsteemi tähemärkide funktsioonid.

Tähis	**Funktsioon**
F	Liigub edasi
+	Pöörab ±90 kraadi paremale
-	Pöörab ±90 kraadi vasakule
[	Salvestab asukoha ja nurga
]	Läheb tagasi salvestatud kohta
.	Muudab nurka

Selliselt defineeritud sõne, produktsioonireegli ja funktsioonidega L-süsteem genereerib esialgse lihtsa teedevõrgu, mida on pärastpoole võimalik laiendada (vt. joonis 7). Kuigi vaikimisi tekitaks see süsteem tänavaid 90-kraadiste nurkadega, on siinkohal jälle kasutatud pseudojuhuslikku numbrigeneraatorit `Random.Range` meetodi näol. Sümbolite ‘+’, ‘-’ ja ‘.’ korral lisatakse tee pöördenurgale -10, 0 või 10 kraadi olenevalt sellest, millise arvu numbrigeneraator nende seast valib. Igal nurgal on ⅓ võimalus valituks osutumiseks. Selliselt muudetakse nurka ka teises etapis rajatavate teede puhul. Tee pöördenurga valimisel on kasutatud vahemiku asemel diskreetset valikut, sest see tagab osade tänavate sirgelt kulgemise. Samal ajal on siiski ka hea võimalus, et tänav kaldub pärast mõnda ristmiku kõrvale.

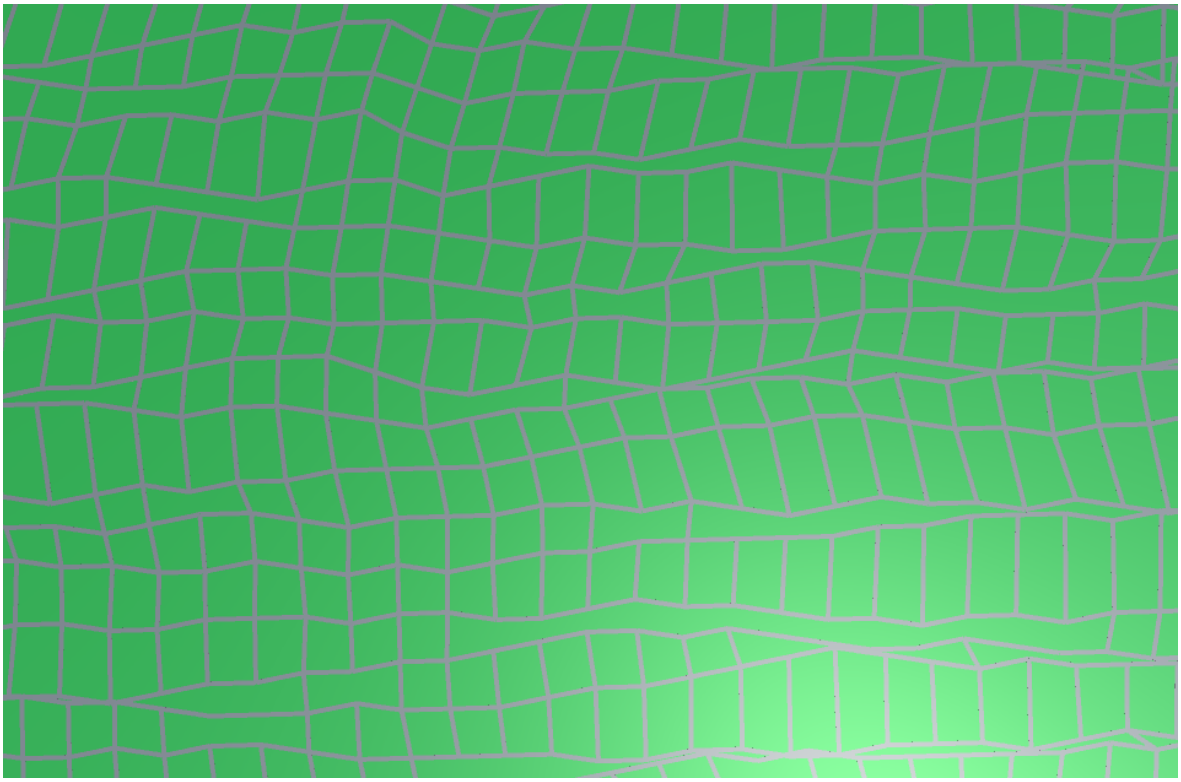


Joonis 7. L-süsteemiga genereeritud teedevõrk.



Joonis 8. Sekundaarsete teede süsteemi loomisprotsess. Vasakul pool on tähistatud esialgsete teede tekitamise järjekord. Paremal pool on tähistatud esialgsete teede pikenduste tekitamise järjekord.

Teises etapis jätkatakse teedevõrgu suurendamist otsast ühendamata teede arvelt (vt. joonis 8). Selliste tänavate otsad salvestatakse listi esimeses etapis. Kõigepealt luuakse uus tänav suunaga üles, siis pikendatakse kõrvale ning viimaks ühendatakse ristmik allpool olemasoleva ühendamata tupiktänavaga. Nagu esimese etapi puhul, kulgeb teede pikendamine ülevalt alla.



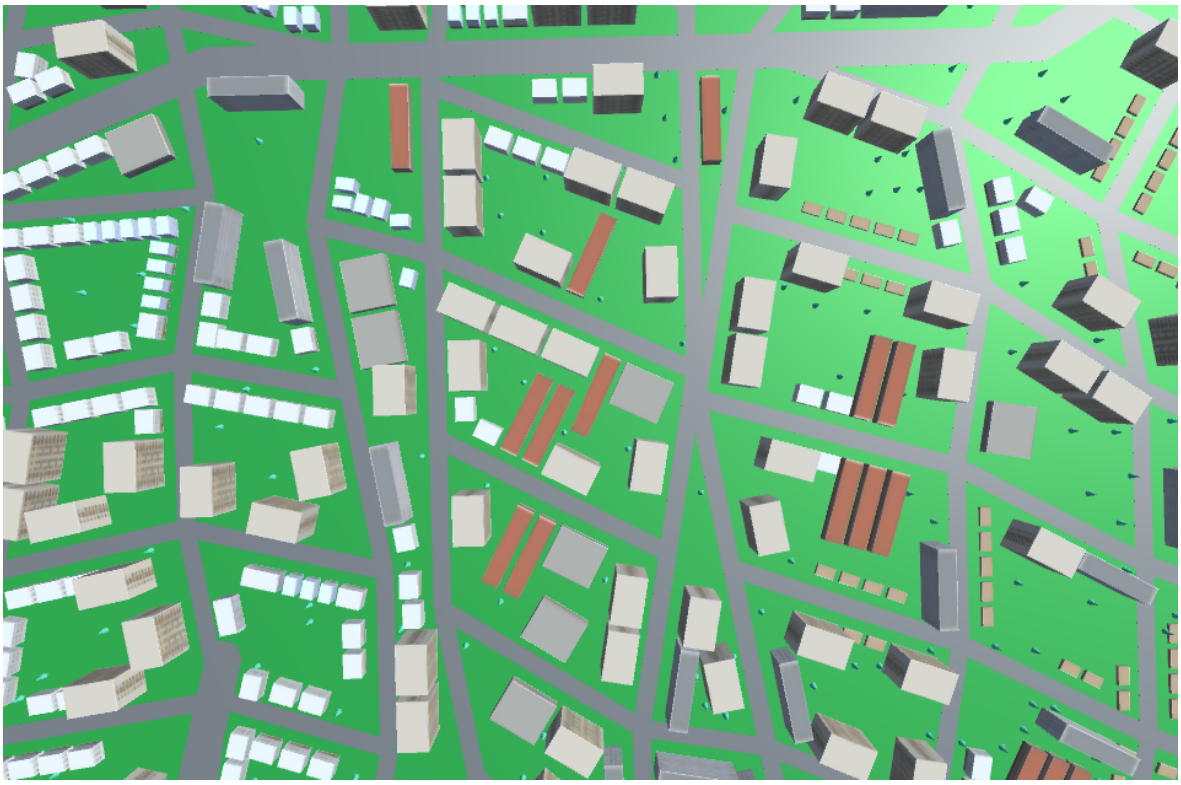
Joonis 9. Lõplik sekundaarsete teede süsteem.

Sekundaarsete teede süsteemi tulemus sisaldab erinevate pikkustega tänavaid, mis paigutatakse üle kogu linnaala erinevate pöördenurkadega (vt. joonis 9). Selleks, et vältida liiga lühikeste tänavate tekkimist, on määratud minimaalne lubatud tee pikkus. See tagab piisavalt suurte kvartalite loomise, mille sisse saab paigutada hooneid.

3.3 Hooned

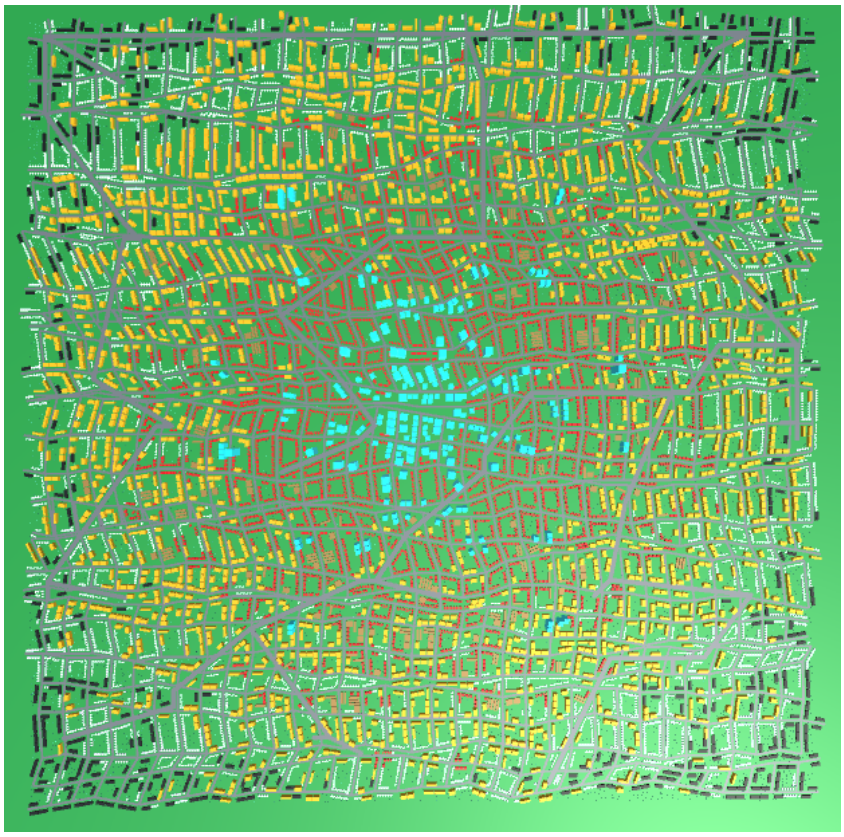
Hoonete paigutus sõltub eelnevalt genereeritud tänavatest. Hooned paigutatakse tekkinud kvartalitesse teede äärde. Hooned on selles süsteemis ette määratud, mis tähendab, et algoritm nende mõõtmeid ei muuda. Antud töö raames on loodud 12 erinevat hoonet. Igaühte neist tekitatakse linna erineval määral olenevalt Perlini mürast ja kaugusest linna keskpunktist. Iga hoone välimus, tüüp ja tekitamise väärtuste vahemik on välja toodud lisas 2.

Iga tee juures tekitatakse hooned mõlemale poole tänavat. Algoritm püüab neid paigutada ettemääratud kaugustega tee äärest ja eelnevast hoonest nii kaua, kuni rohkem antud teelõik saab otsa. Selleks, et hooned ei tekiks üksteise sisse, kontrollitakse enne kohale panemist selle kokkupõrget teiste kehadega kasutades `Physics.CheckBox` meetodit. On ka võimalik, et selle paika panemise töö käigus satuvad ehitised tee peale. Selle jaoks on eraldi kontroll pärast hoonete paigutamise tsükli töö lõppu. See vaatab üle kõik olemasolevad teed ja kontrollib, kas nendega puutub kokku objekt sildiga (ing. k. *tag*) “Building”. Kui see vastab tõele, siis selle sildiga objekt kustutatakse.



Joonis 10. Hoonete paigutus teede vahel.

Selleks, et luua realistlik linnapilt, on vaja erinevat sorti hooned paigutada nii, et kujuneksid erinevad rajoonid. Käesolev süsteem kasutab selle eesmärgi saavutamiseks hoonete paigutamist Perlini müra ja kauguse järgi keskpunktist (vt. joonis 10). Kauguse põhjal hinnatakse, millised hooned sobivad rohkem linna keskele ja millised rohkem linna äärde paigutamiseks. Linna keskelt väljapoole liikudes muutuvad hooned algoritmi kohaselt kahanevas järjekorras järgmiselt: kõrghooned, ärihooned, institutsionaalsed hooned, kortermajad, eramajad, tööstushooned. Selleks, et rajoonid ei oleks liiga tehislikud ja oleks rohkem segunemist erinevat sorti hoonete vahel, on kasutatud ka Perlini müra. Kaugusel ja Perlini müral on kummalgi maksimaalne väärtus 10. Nende väärtuste summat kasutatakse hoone tüübi määramiseks.



Joonis 11. Rajoonide paiknemine linnas. Sinised on pilvelõhkujad, punased kaubapindadega hooned, oranžid korterid, pruunid institutsioonilised hooned, valged eramajad, mustad tööstushooned.

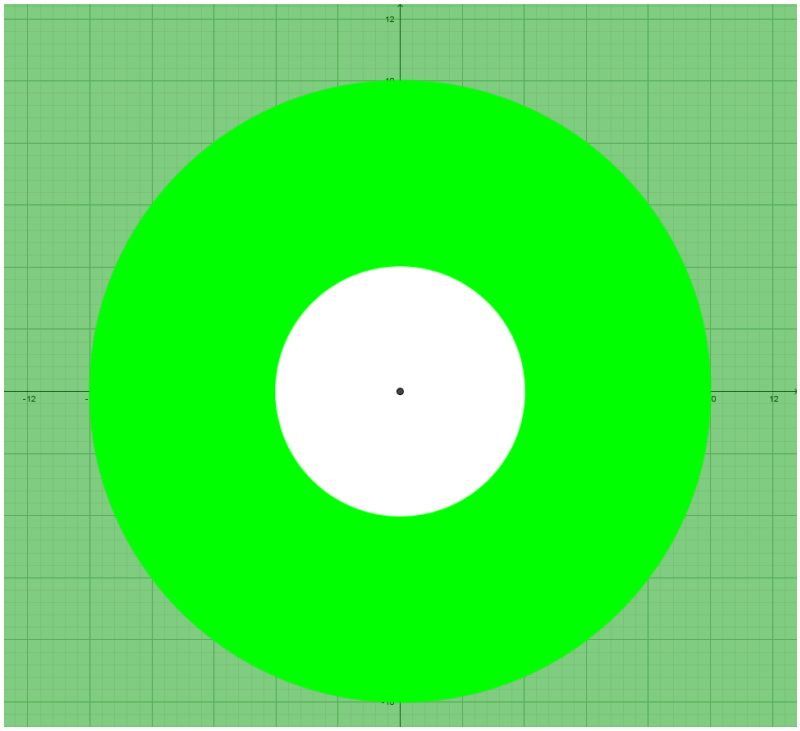
Käesolevas töös genereeritakse hoonete paigutus, mis oleneb nii kaugusest linna keskpunktist kui ka juhuslikust Perlini mürast (vt. joonis 11). Selle tulemuseks on linn, mille rajoonide muutus on märgatav keskelt linna piirideni liikudes. Järgmisena täidetakse järelejäänud haljasala puudega.

3.4 Puud

Peale hoonete ja teede loob linna generaator ka puid. Puude tihedus kasvab linnas keskelt väljapoole liikudes. Nende genereerimiseks on kasutatud sama põhimõtet, millega luuakse selles süsteemis hooneid.

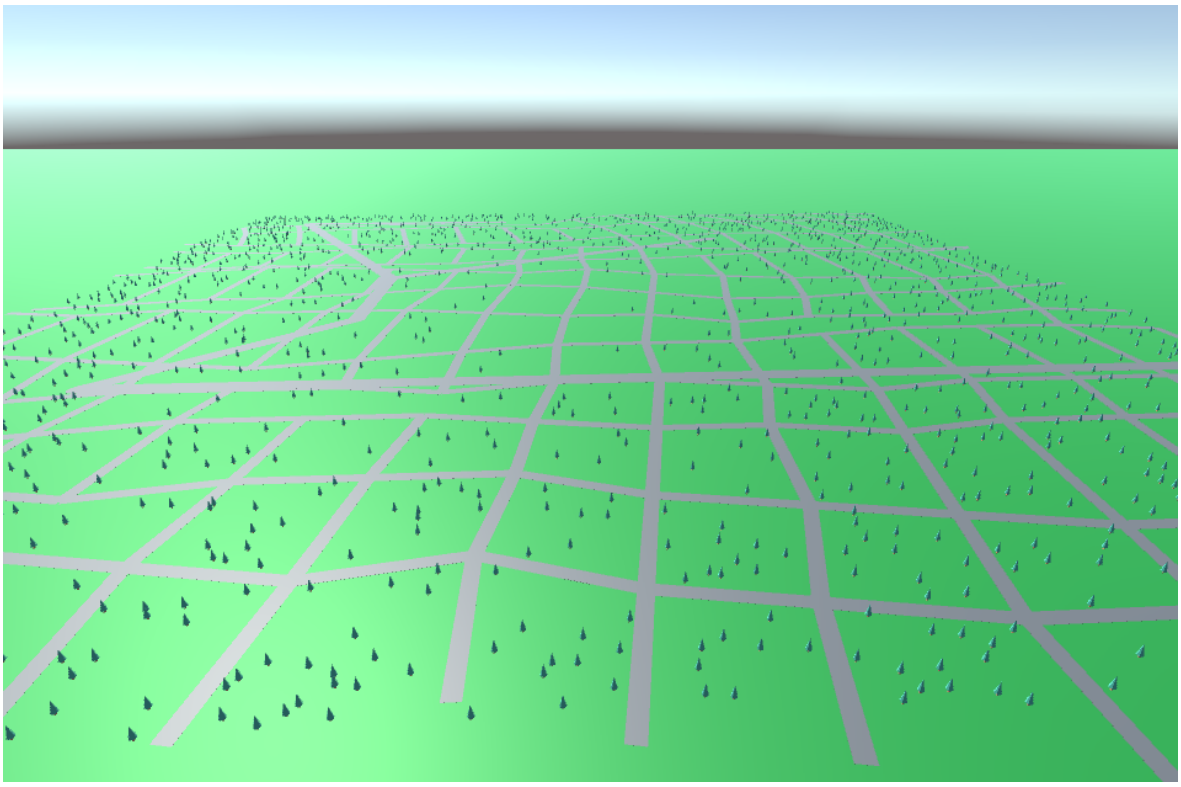
Puid tekitatakse tihedamalt, kui Perlini müra väärtus on väiksem ja vaadeldav asukoht on kaugel linna keskpunktist. Siinkohal on jälle kasutatud hall-skaala ja kauguse väärtuse

summat, et määrata objektide paigutus. Kui summa on alla 3-e, on puid tihedalt. Üle selle väärtuse ilmub neid hõredalt ja üle 10-e ei teki puid üldse (vt. joonis 12).



Joonis 12. Puude paiknemise tsoonid. Valges alas ei tekitata puid, helerohelises tekitatakse hõredalt, tumerohelises läbipaistvas alas tekitatakse tihedalt.

Selleks, et puud ei satuks tehisliku väljanägemisega ruudustiku laadsesse paigutusse, on koordinaatidele liidetud `Random.Range` meetodiga genereeritud väärtus vahemikus [-10, 10].



Joonis 13. Puude asetus.

Tulemuseks on puude paigutus, mis on tihedam linna piiride läheduses ja hõreneb keskele liikudes (vt. joonis 13). Nagu hoonete ja teede puhul, genereeritakse puid linna piiridesse jääva ala sisse. Nende määramiseks on kasutajal võimalik muuta parameetreid.

3.5 Muudetavad parameetrid

Linna generaator võimaldab kasutajal määrata linna pindala ja rahvaarvu, millega saab mõjutada linna välimust. Need mõjutavad eelkõige hoonete tekitamist. Lisas 3 on toodud näited erinevate parameetritega genereeritud linnadest.

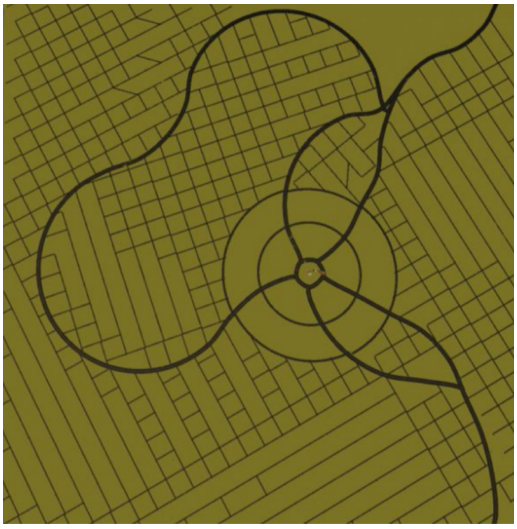
Linna üldine kuju on alati kandiline risttahukas. Selle pindala muutmiseks on võimalik defineerida pikkuse ja laiuse väärtused. Rahvaarvu parameetri muutmine võimaldab muuta linna asustust hõredamaks või tihedamaks. See väärtus võib olla nullist sajani. Kui rahvaarvu väärtus on sada, tähendab see seda, et linn on nii tihedalt asustatud kui võimalik. Kõik võimalikud hooned, mis on võimalik tekitada, on olemas, kui rahvaarv on 100. Kui väärtus on null, ei ole linnas üldse inimesi, mis tähendab, et pole ka ehitisi. Kui arv jääb nulli ja saja vahemikku, siis see on tõenäosus, mis on igal hoonel tekkimiseks.

4 Tulemuse analüüs

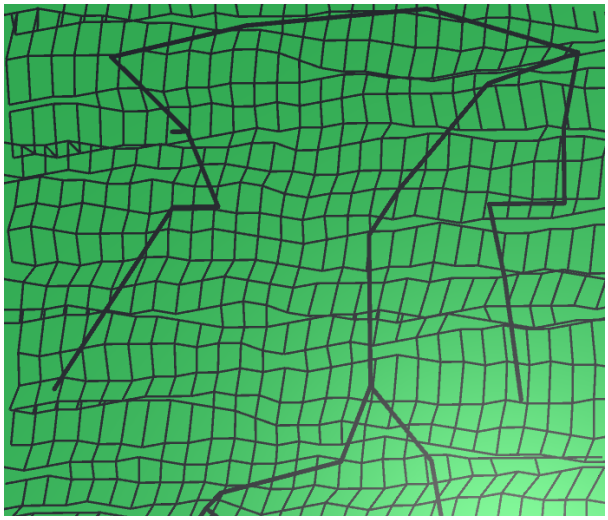
Selles peatükis on võrreldud bakalaureusetöö käigus valminud linna generaatorit teiste varem kirjeldatud teiste autorite loodud protseduuriliste generaatoritega. Antud töö käigus tehtud süsteemi eesmärk on protseduuriliselt genereerida eristatavate loogiliselt paigutatud rajoonidega. Selle linna genereerimiseks ei ole tingimata vaja kasutajapoolset sisendit. Samas on kasutajal võimalik mõjutada linna pindala ja tihedust, muutes neile vastavaid parameetreid. Kuna antud töö keskendub põhiliselt protseduuriliselt teede ja hoonete genereerimisele, siis keskendub ka nende võrdlemisele see tulemuste analüüs.

4.1 Teede genereerimine

Sarnaselt City Generatoriga, on antud töö raames teed genereeritud hall-skaala abil. Mõlemas süsteemis on loodud sõlmed hall-skaala heledatele aladele, mis ühendatakse laiade põhiteedega. Sharma süsteem genereerib pikemate kurvidega kiirteed (vt. joonis 14). Samas antud töö raames loodud süsteem genereerib kitsamate pöördenurkadega peateed (vt. joonis 15).



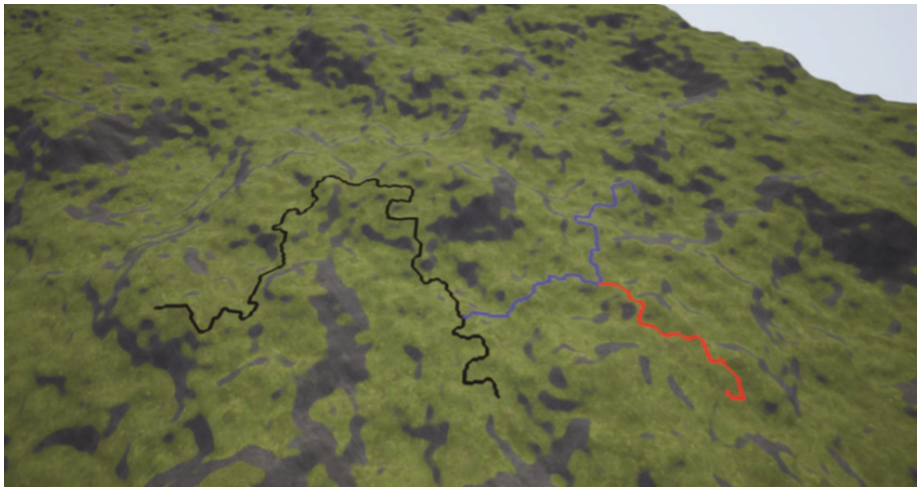
Joonis 14. Sharma protseduurilise linna generaatori teedevõrgustik [9].



Joonis 15. Käesolevas töös valminud teedevõrgustik.

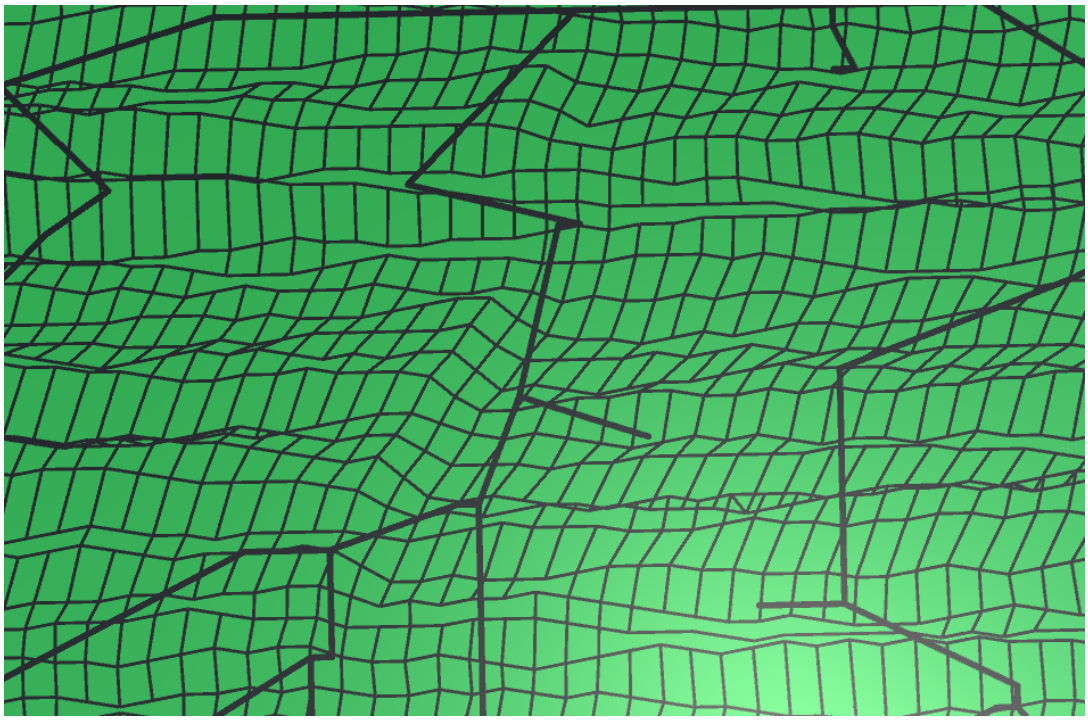
Algoritm, mille järgi sõlmed ühendatakse, sarnaneb A* algoritmiga, mida kasutab Mizdali ja Pozzeri loodud külade ja teede protseduuriline generaator (vt. joonis 16). Erinevalt tollest

algoritmist, ei ole antud töö raames loodud süsteemis arvestatud maastikuga, sest tegemist on täiesti tasase pinnaga. Külade ja teede generaator samas ei tekita üldse sekundaarsete teede võrgustikku.



Joonis 16. Mizdali ja Pozzeri loodud generaatoris rajatud teed [10].

Sekundaarsed kitsad teed genereeritakse käesoleva töö algoritmis osaliselt L-süsteemiga. City Generator kasutab samuti L-süsteemi teevõrgustiku loomiseks, aga läheneb sellele teistmoodi. Selles artiklis kirjeldatud algoritm rajab väiksemaid teid sektorite sisse, mis on ümber piiratud primaarsete teede ja maastikuga. Tänu sellele on tolles süsteemis primaarsed ja sekundaarsed teed paremini ühendatud üksteisega. Käesoleva töö raames loodud süsteemis ei tee kumbi teevõrgustik teineteisest välja. Samas genereerib antud töös loodud algoritm suurema varieeruvusega sekundaarseid teid (vt. joonis 17).

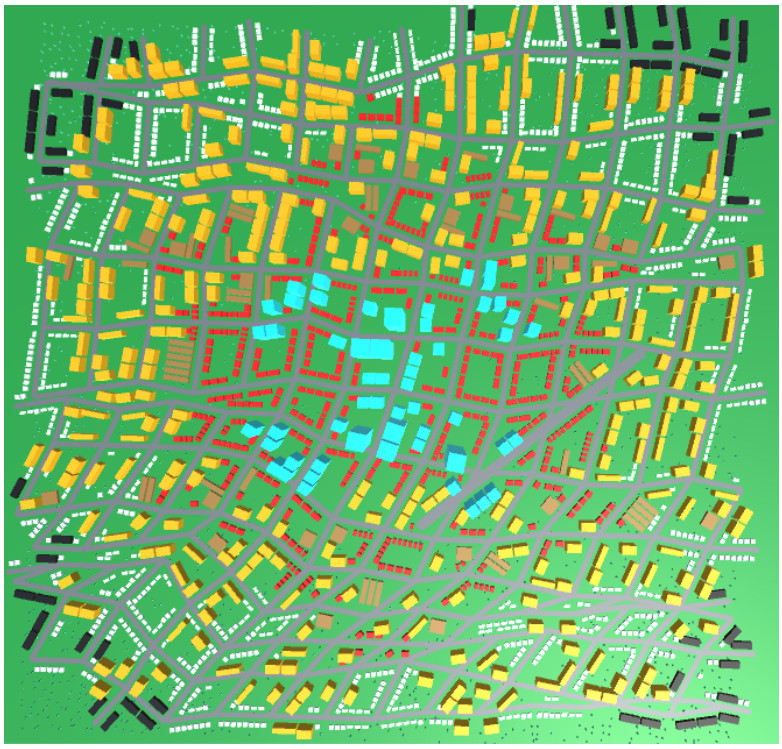


Joonis 17. Käesolevas töös valminud teedevõrgustik.

Käesoleva bakalaureusetöö generaatori tulemuses on teedevõrk vähem korrapärasem teede pikkuste ja pöördenurkade poolest. Matthewsi ja Malloy loodud süsteem ei loo teedesüsteeme, aga tekitab erineva otstarbega piirkondi.

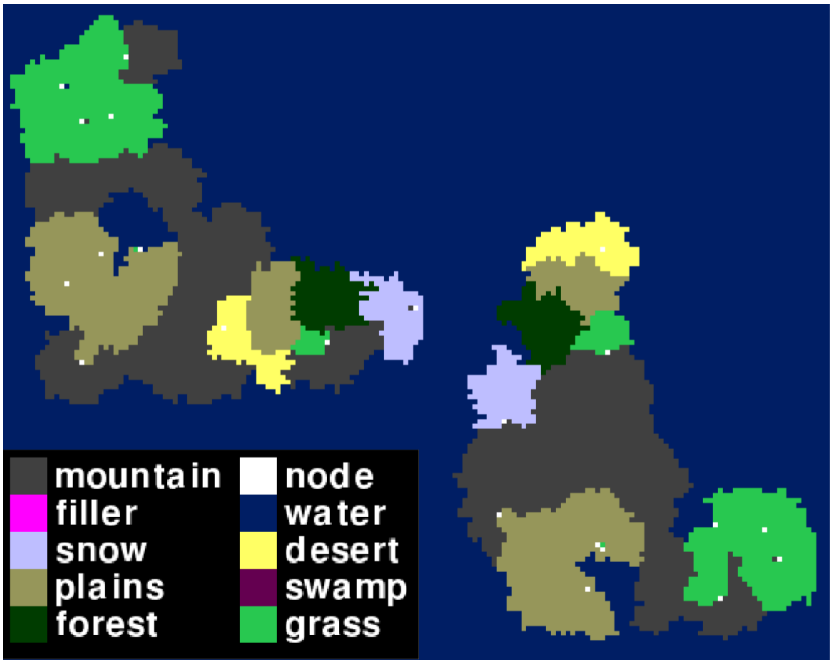
4.2 Hoonete paigutus

Erinevate piirkondade genereerimine on oluline osa MapGeni süsteemist, nagu ka selle bakalaureusetöö puhul. Samas ei ole antud töös loodud algoritm võrdlemisi nii keeruline. Ei ole kasutatud füüsikamootorit, et määrata erisuguste rajoonide paiknemist. MapGen nõuab ka kasutajalt palju rohkem erinevaid sisendandmeid. See teeb antud töö raames valminud süsteemi kasutamise lihtsamaks ja kiiremaks, aga MapGen on rohkem kohandatav parajasti vajaliku eesmärgi jaoks mõnes arvutimängus. Käesolevas töös loodud süsteem loob rajoonid, mis üldiselt muutuvad kaardi keskelt väljapoole liikudes (vt. joonis 18).



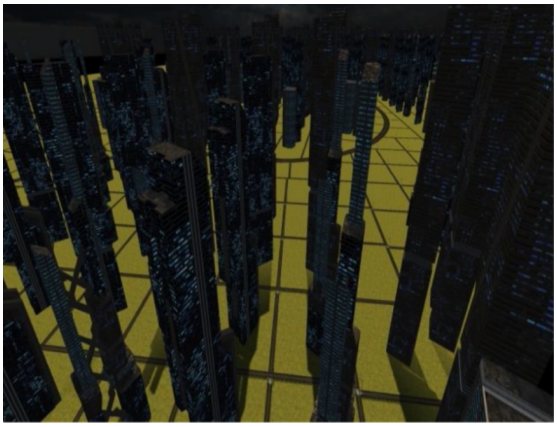
Joonis 18. Piirkondade paiknemine käesolevas töös loodud süsteemis.

MapGen võtab piirkondade genereerimisel arvesse palju rohkem erinevaid parameetreid. Seetõttu on see võimeline looma rohkem keeruliste ja mitmekesiste paigutustega kaarte (vt. joonis 19).



Joonis 19. Piirkondade paiknemine MapGeni süsteemis [8].

City Generator on kõige sarnasem selle bakalaureusetöö käigus valminud generaatoriga. See tegeleb samuti kaasaegse linna protseduurilise genereerimisega (vt. joonis 20). Kuigi teede genereerimises on nendel töödel sarnasusi, ei ole City Generatori hoonete paigutus nii täpne, kui antud bakalaureusetöös (vt. joonis 21). Hooned on pigem üksluise välimusega pilvelõhkujad ja ei ole eristatavaid rajoone. Hooned ei ole pööratud tänavate poole ja ei asu korrapäraselt tänavate ääres.



Joonis 20. Hooned City Generatori süsteemis [9].

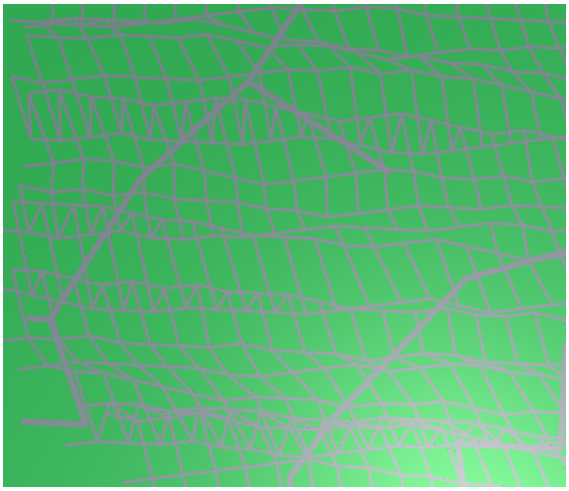
Joonis 21. Hooned käesoleva töö süsteemis.

Külade ja teede generaator käsitleb jällegi pigem üksteisega sarnanevate majade paigutamist. Kuna selle süsteemi puhul on tegemist ebatasase pinnaga, sõltub hoonete paigutus maastikust. Maastik määrab, kuhu tekitatakse hõredamalt ja tihedamalt asustatud külad.

5 Võimalused edasiarendamiseks

Antud töö raames valminud generaatorit on võimalik detailsemaks muuta. Saaks implementeerida rohkem linnas esinevate objektide genereerimist, näiteks lambipostid, kõnniteed, pargid, väljakud ja parklad. Praegusel kujul genereerib töö raames valminud lahendus protseduuriliselt hoonete loogilise paigutuse, aga ei mõjuta ühtegi hoonet ennast. Oleks võimalik ka protseduuriliselt genereerida erinevate kujudega hooneid, et luua rohkem variatsiooni nende vahel.

Algoritmi tööd oleks võimalik parendada. Praeguses seisus ei hooli primaarsete ja sekundaarsete teede võrgustikud teineteisest (vt. joonis 22). Sellepärast ei pruugi käesolevas implementatsioonis primaarsed teed hästi ühenduda ülejäänud teedevõrguga. Laiad peateed võivad tihtipeale lõppeda tupikus. Sekundaarsed teed võivad praeguses implementatsioonis ka sattuda segamini puntrasse. Ühest küljest võib see tekitada huvitavama teedevõrgustiku. Samas võib see teinekord liiga segane välja näha. Liiga keeruliste ühendustega teevõrk tekitaks väljakutseid siis, kui oleks vaja luua linnale liikluskorraldus.



Joonis 22. Võimalikud teedemustrid.

Liikuvatest autodest ja inimestest koosnev liikluse loomiseks on vaja pinda, mille peal saaks liikuda. Antud töö raames loodud süsteemis on teed omaette mänguobjektid, millele on võimalik tekitada NavMesh Surface. Selle komponendiga objektide peal saavad liikuda NavMesh Agentiga mänguobjektid, mis oleksid sõidukid ja inimesed. Liikuvatele objektidele oleks vaja juhuslikult genereerida erinevad sihtpunktid iga kord, kui olemasolevateni on kohale jõutud. Selleks, et sõidukid ei kokku ei põrkaks üksteisega, on vaja liikluspiirangute loogika implementeerida. Valgusfooridega oleks seda kõige lihtsam teha, sest sel juhul saab konkreetselt aja järgi paika panna, kus liigub ristmikul liiklus ja kus mitte. Ilma valgusfoorideta tuleks välja arendada dünaamiliste objektide jaoks keerulisem tehisintellekt, mis tuvastaks teisi liiklejaid ümberringi ja teeks selle põhjal otsuseid ise.

6 Kokkuvõte

Käesolev töö kirjeldas virtuaalse linna protseduurilise genereerimise probleemi. Töös uuriti kolmes teadusartiklis käsitletud süsteemi keskkonna protseduuriliseks genereerimiseks. Nende põhjal loodi uus algoritm.

Antud töös loodi süsteem, mis võimaldab protseduuriliselt genereerida linna. See ei nõua kasutajapoolset sisendit. Samas on olemas parameetrid, millega on võimalik muuta linna omadusi. Süsteem sisaldab endas primaarsete ja sekundaarsete teede võrgustikke, hooneid ja puid. Neid tekitatakse, kasutades juhuslikku Perlini müra, L-süsteemi ja ettemääratud parameetreid. Linn genereeritakse tasasele pinnale.

Valminud algoritmi võrreldi teadusartiklites kirjeldatud lahendustega. Käesolevas töös tehtud protseduuriline generaator loob võrdlemisi mitmekesise, samas lihtsama teedesüsteemi. Rajoonide genereerimine on ka pigem lihtsakoeline, kuid sellegipoolest võimaldab tekitada töö eesmärgiks vajaliku paigutuse. Antud töö käigus loodud süsteem ei nõua nii palju kasutajapoolseid sisendeid kui Matthewsi ja Malloy generaator.

Vaatamata eesmärgi saavutamisele on võimalik töö raames valminud algoritmi paremaks teha. Edasiarendamise võimaluste hulgas on välja toodud detailid ja algoritmilised aspektid, mida saaks parendada.

Viited

[1] A. Lindenmayer ja P. Prusinkiewicz. The algorithmic beauty of plants. Springer Science & Business Media, 2012.

[2] K. Perlin, „An Image Snythesizer", ACM Siggraph Computer Graphics, vol. 19, nr. 3, lk. 287-296, juuli 1985.

[3] Gilbert B. 'Minecraft' has been quietly dominating for over 10 years, and now has 112 million players every month. Business Insider, 2019. https://www.businessinsider.com/minecraft-monthly-player-number-microsoft-2019-9 (03.05.2020)

[4] Strickland D. No Man's Sky was one of Steam's most-played games four years in a row, 2020. https://www.tweaktown.com/news/69692/mans-sky-one-steams-played-games-four-years-row/index.html (03.05.2020)

[5] K. Perli, 2016, „Protseduuriline linnade genereerimine" https://comserv.cs.ut.ee/ati_thesis/datasheet.php?id=53791&year=2016

[6] M. Janno, 2018, „Procedural Generation of 2D Creatures" https://comserv.cs.ut.ee/ati_thesis/datasheet.php?id=61897&year=2018&language=en

[7] A. Sepp, 2018, „Infinite Procedural Infrastructured World Generation" https://comserv.cs.ut.ee/ati_thesis/datasheet.php?id=62135&year=2018&language=en

[8] E. A. Matthews ja B. A. Malloy. Procedural generation of story-driven maps. 2011 16th International Conference on Computer Games (CGAMES). Louisville, 2011, 107-112.

[9] R. Sharma. Procedural City Generator. 2016 International Conference System Modeling & Advancement in Research Trends (SMART). Moradabad, 2016, 213-217.

[10] T. Boelter Mizdal ja C. T. Pozzer. Procedural Content Generation of Villages and Road System on Arbitrary Terrains. 2018 17th Brazilian Symposium on Computer Games and Digital Entertainment (SBGames). Foz do Iguaçu, Brazil, 2018, 205-211.

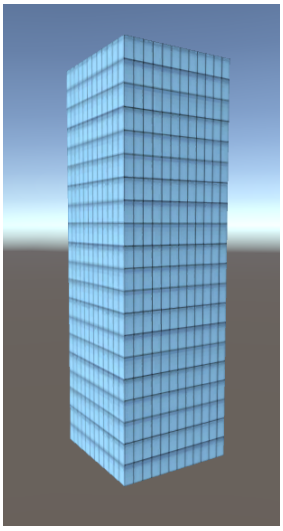
Lisad

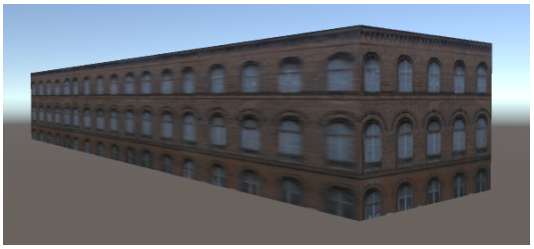
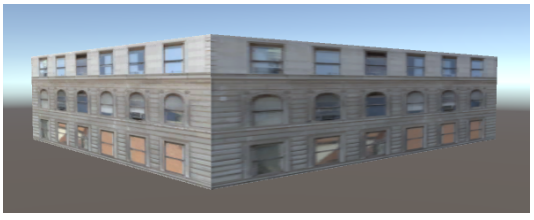
Lisa 1. Demonstreeriv rakendus

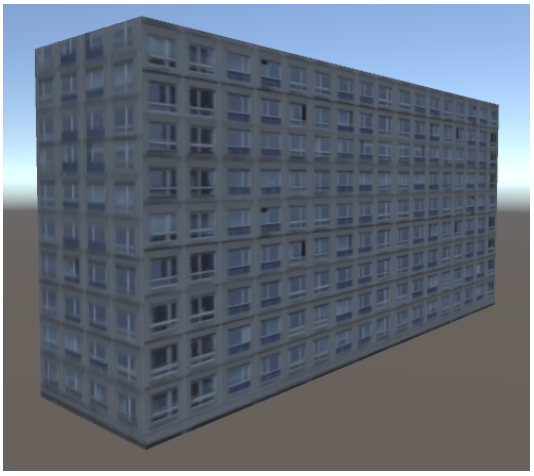
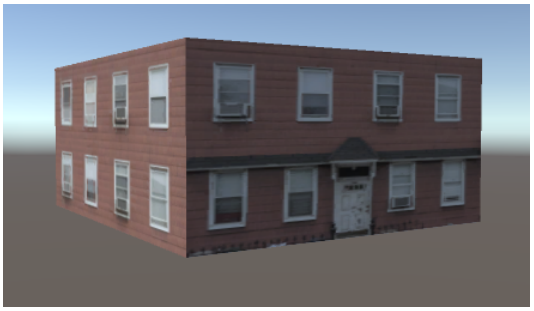
Demonstreeriv rakendus asub käesoleva tööga kaasas olevas ZIP-failis „citygendemo”.

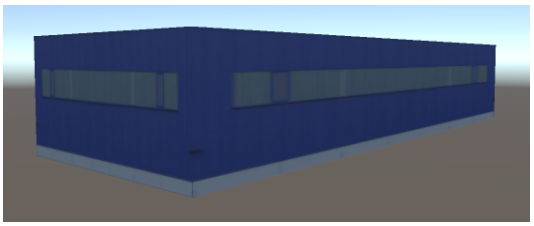
Lisa 2. Hoonete tabel

Tabel 2. Hoonete välimused, tüübid, nende esinemise väärtuste vahemikud.

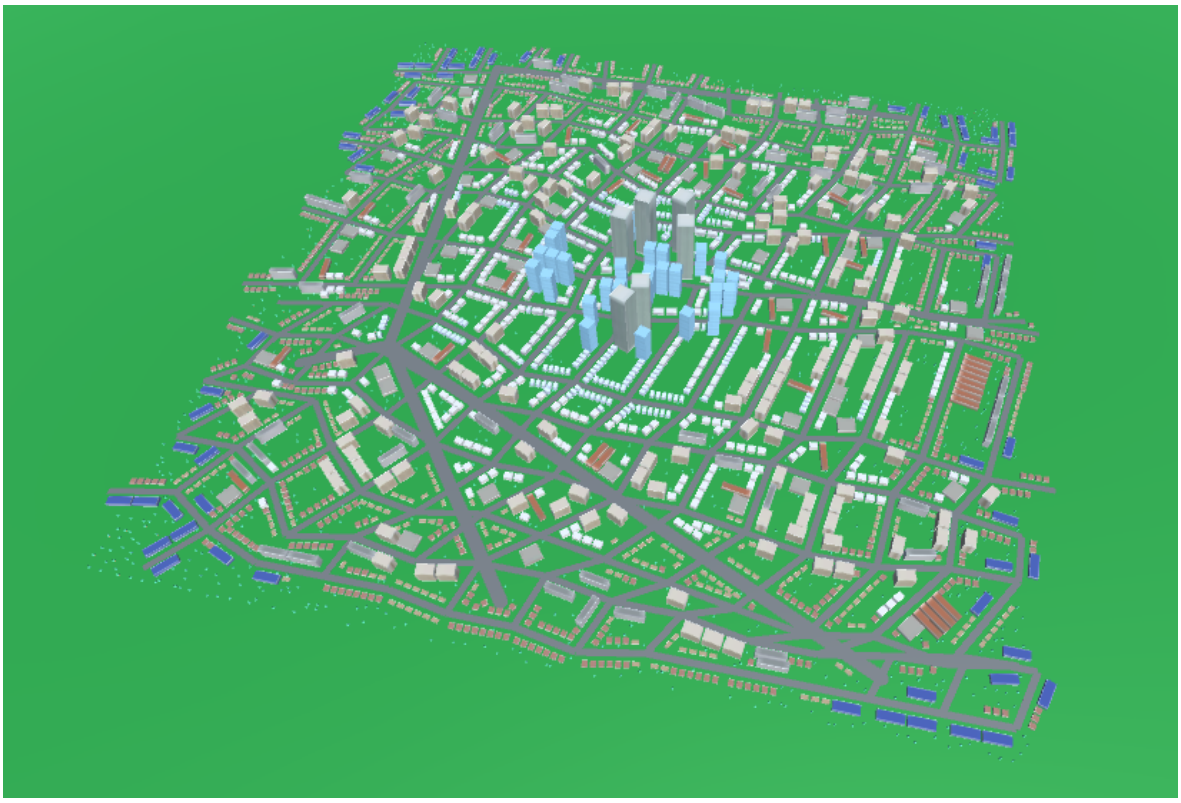
Hoone välimus	Tüüp	Perlini müra ja keskpunktist kauguse summa vahemik, mille juures tekitatakse
	Pilvelõhkuja	[14, 16)
	Pilvelõhkuja	[13, 14)
	Äripinnaga hoone	[12, 13)

	Äripinnaga hoone	[10, 12)
	Äripinnaga hoone	[8.5, 10)
	Institutsiooniline hoone	[8.3, 8.5)
	Institutsiooniline hoone	[8, 8.3)

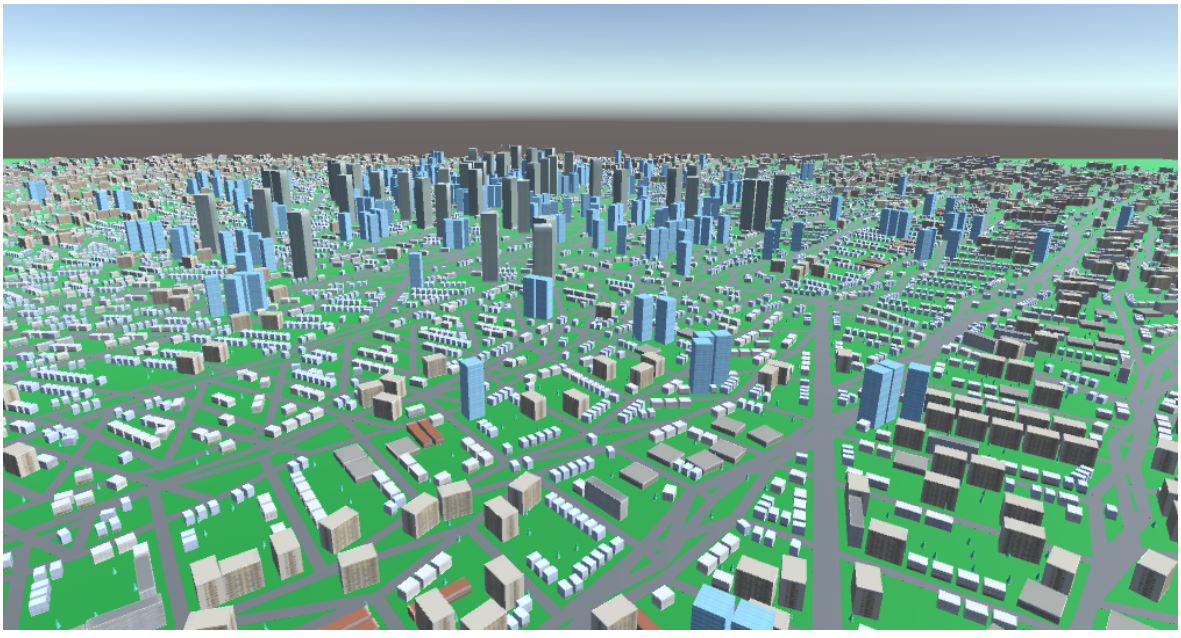
	Kortermaja	[6, 8)
	Kortermaja	[5, 6)
	Elumaja	[4, 5)
	Elumaja	[2, 4)

	Tööstushoone	[0, 2)

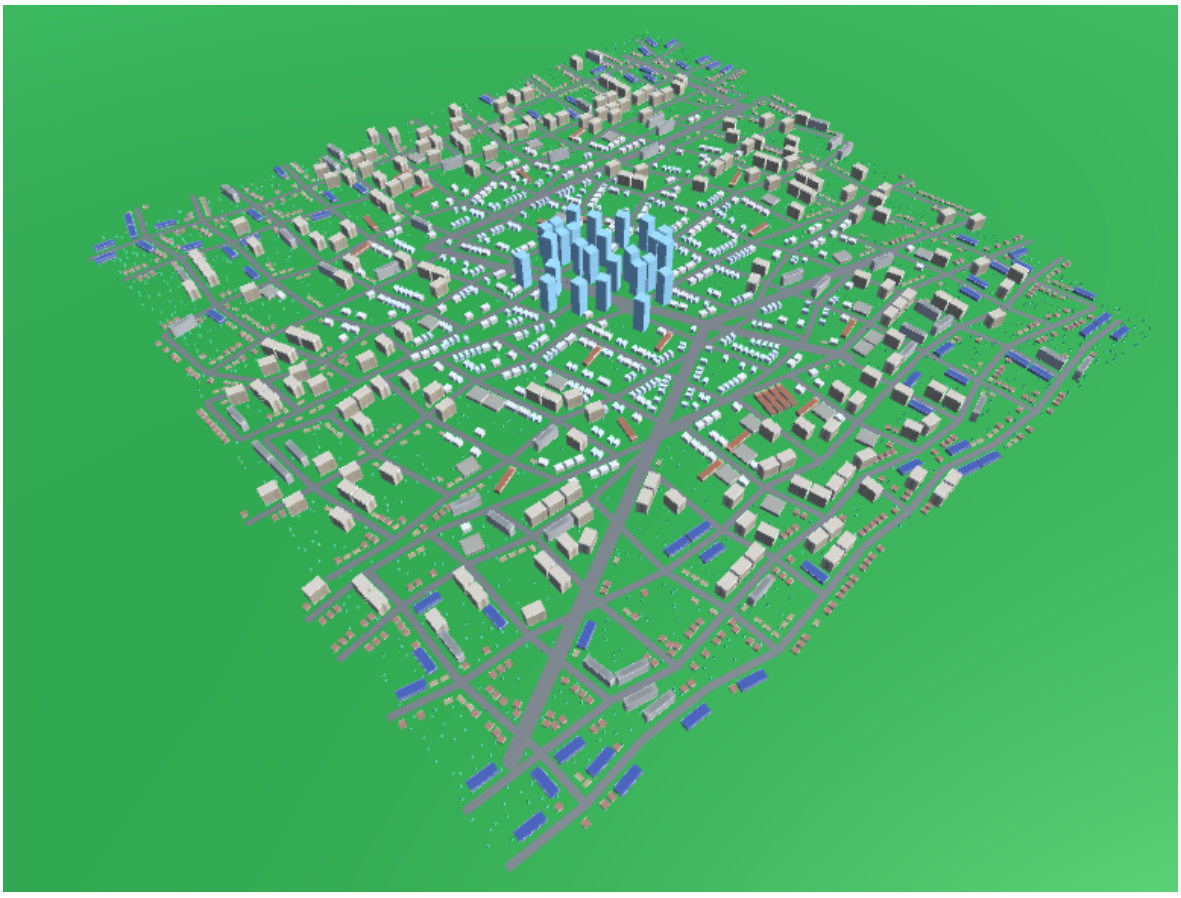
Lisa 3. Pildid erinevate pindalade ja rahvaarvudega linnadest



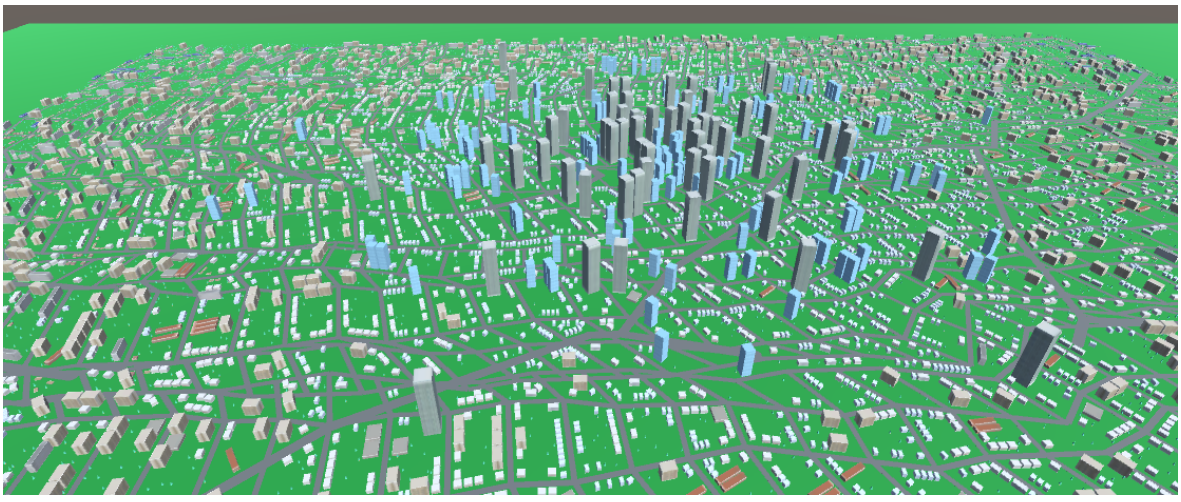
Joonis 23. Laius: 20. Kõrgus: 20. Tihedus: 100.



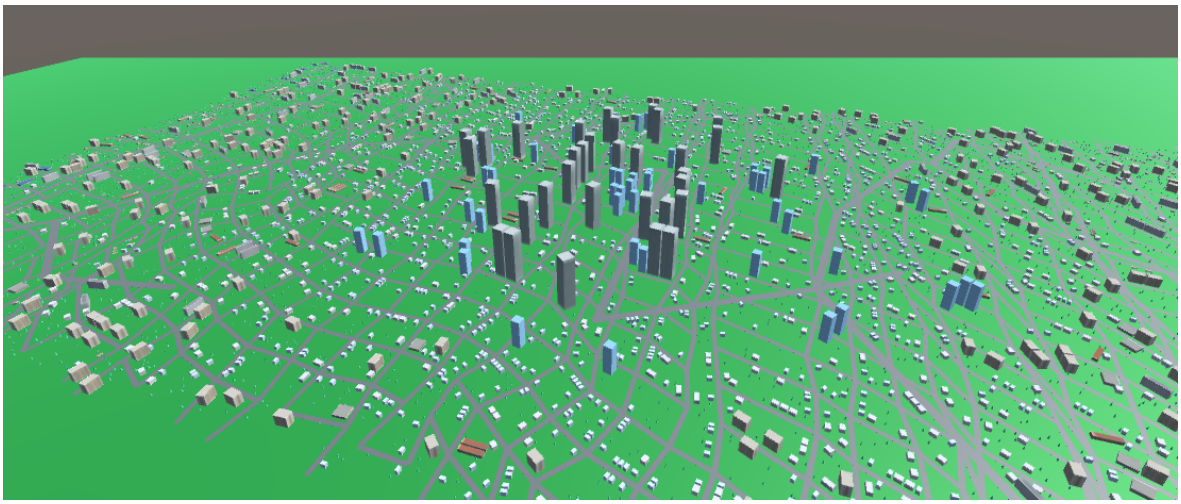
Joonis 24. Laius: 80. Kõrgus: 80. Tihedus: 85.



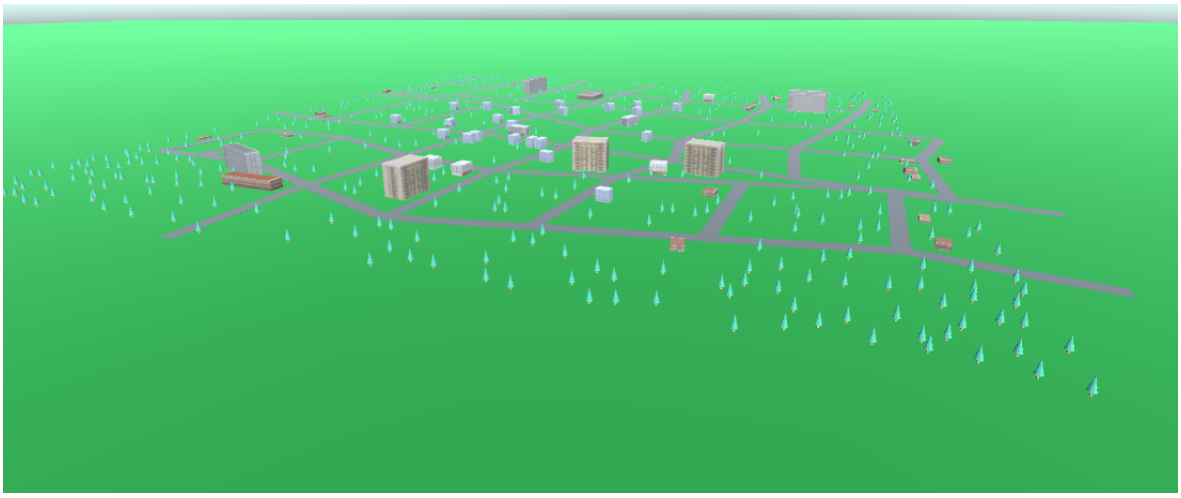
Joonis 25. Laius: 20. Kõrgus: 20. Tihedus: 70.



Joonis 26. Laius: 70. Kõrgus: 40. Tihedus: 50.



Joonis 27. Laius: 30. Kõrgus: 60. Tihedus: 25.



Joonis 28. Laius: 10. Kõrgus: 10. Tihedus: 10.

Lihtlitsents lõputöö reprodutseerimiseks ja üldsusele kättesaadavaks tegemiseks

Mina, Aap Vare

1. annan Tartu Ülikoolile tasuta loa (lihtlitsentsi) minu loodud teose

   “Linna protseduuriline genereerimine”

   mille juhendaja on Raimond-Hendrik Tunnel

   reprodutseerimiseks eesmärgiga seda säilitada, sealhulgas lisada digitaalarhiivi DSpace kuni autoriõiguse kehtivuse lõppemiseni.

2. Annan Tartu Ülikoolile loa teha punktis 1 nimetatud teos üldsusele kättesaadavaks Tartu Ülikooli veebikeskkonna, sealhulgas digitaalarhiivi DSpace’i kaudu Creative Commonsi litsentsiga CC BY NC ND 3.0, mis lubab autorile viidates teost reprodutseerida, levitada ja üldsusele suunata ning keelab luua tuletatud teost ja kasutada teost ärieesmärgil, kuni autoriõiguse kehtivuse lõppemiseni.

3. Olen teadlik, et punktides 1 ja 2 nimetatud õigused jäävad alles ka autorile.

4. Kinnitan, et lihtlitsentsi andmisega ei riku ma teiste isikute intellektuaalomandi ega isikuandmete kaitse õigusaktidest tulenevaid õigusi.

Aap Vare

20.05.2020