TARTU ÜLIKOOL
Arvutiteaduse instituut
Informaatika õppekava

Andreas Sepp

Protseduuriline lõpmatu maastiku genereerimine

Bakalaureusetöö (9 EAP)

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

Tartu 2016

Protseduuriline lõpmatu maastiku genereerimine

Lühikokkuvõte:

Käesolevas bakalaureusetöös kirjeldatakse müraalgoritme nagu väärtusmüra, Perlini müra ja simpleksmüra ning nende praktilisi rakendusi pseudolõpmatu maastiku genereerimiseks. Töös kirjeldatav autori loodud maastiku genereerimise algoritm genereerib mitmekülgset maastikku, jagades maailma bioomide ehk makroökosüsteemide vahel ning seejärel kujundades neid erineva pinnase, maastiku kuju ning taimkattega. Algoritm on implementeeritud programmeerimiskeeles C# ja mängumootoriga Unity loodud rakendusena.

Võtmesõnad:

Protseduuriline genereerimine, maastiku genereerimine, lõpmatu maastiku genereerimine, müra, väärtusmüra, gradientmüra, Perlini müra, simpleksmüra, bioomide genereerimine, taimede paigutuse genereerimine, arvutigraafika, vokslid, piirnevad kuubikud, Unity.

CERCS:

P150: Geomeetria, algebraline topoloogia;

P170, Arvutiteadus, arvutusmeetodid, süsteemid, juhtimine;

P175: Informaatika, süsteemiteooria.

Procedural infinite terrain generation

Abstract:

This Bachelor's thesis describes several noise algorithms such as value noise, Perlin noise and simplex noise and their practical applications in generation of pseudo-infinite terrains. The described generation algorithm generates diverse landscape by dividing the world between different biomes and then shapes the terrain and determines the ground material and flora depending on the biome. The algorithm is available as an application written in C# using the Unity game engine.

Keywords:

Procedural generation, land generation, terrain generation, infinite terrain generation, volumetric terrain generation, noise, value noise, gradient noise, Perlin noise, simplex noise, biome generation, plant placement generation, computer graphics, voxels, marching cubes, Unity.

CERCS:

P150: Geometry, algebraic topology;

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

P175: Informatics, systems theory.

Sisukord

1. Sissejuhatus

Moodsad arvutigraafika rakendused võimaldavad kujutada järjest detailsemaid virtuaalseid keskkondi. Seetõttu kulub selliste keskkondade disainimiseks üha rohkem aega ning modelleerijate ja rakenduste disainerite töökoormus on pidevalt kasvamas. Kuna rakenduste disainimine on kallis ja ajumahukas, tasub seda protsessi võimalikult palju automatiseerida. Selleks on välja töötatud palju erinevaid protseduurilise genereerimise algoritme, mille abil on võimalik genereerida tekstuure, mudeleid, maastikke ja muusikat kasutades väikest hulka sisendparameetreid [1,2,3].

Eelnevatest kategooriatest üheks mahukamaks võib pidada maastiku genereerimist, sest põhjalik algoritm peab olema suuteline genereerima nii maastiku kuju, erinevaid keskkondi kui ka paigutama pinnasele taimestikku ning muid looduslikke objekte. Maastiku genereerimist võib jagada kaheks: lõplikuks ning lõpmatuks. Lõpliku maastiku korral genereeritakse maastik tavaliselt käitusvälisel ajal, sest see protsess võtab märga-tavalt palju aega. Tavaliselt saab lõpliku maastiku genereerimisel lihtsamini rakendada keerukamaid algoritme nagu maastiku erosiooni [4], jõgede [5] ja linnade genereerimise [6] algoritmid, sest korraga kogu maastikku genereerides on võimalik kasutada algoritme, mis vajavad hea tulemuse saavutamiseks andmeid kogu maastiku kohta. Näiteks jõgede genereerimiseks on vaja teada andmeid maastiku kuju kohta suurel maa-alal, et valida jõele sobiv tee. Sellele vastanduvalt on lõpmatu maastiku genereerimise korral võimalik maastikku genereerida ainult tükikaupa. Seetõttu ei ole lihtne kontrollida, kas mingit osa tüki pinnasest peaks näiteks lähedalasuva veekogu tõttu erodeerima või kas tükil paikneb mingi osa jõest, sest nende määramiseks on vaja teada, mis paikneb genereeritavast tükist väljaspool. Tükikaupa genereerimisel on oluline, et iseseisvalt genereeritud lähestikku paiknevad tükid pärast omavahel kokku sobituksid ning ühegi tüki genereerimine ei tohiks sõltuda teiste tükkide olemasolust. Lisaks peavad lõpmatu maastiku genereerimise algoritmid olema kiired, et seda oleks võimalik reaalajas vastavalt kaamera liikumisele juurde genereerida. Lõpmatu maastiku genereerimine on populariseerunud alates aastast 2009, kui kiirelt ülemaailmseks hitiks kujunenud mäng Minecraft[1] laiema avalikkuse ette jõudis. Pärast seda on tehtud sadu sarnaseid projekte ning üle aastate on ilmunud ka mitmeid lõpmatut maastikku käsitlevaid artikleid [7,8].

[1] https://minecraft.net/en/

Käesolevas bakalaureusetöös tutvustatakse pikemalt kolme arvutigraafikas laialt levinud müraalgoritmi ning nende mitmekülgseid rakendusvõimalusi lõpmatu kolmemõõtmelise maastiku genereerimiseks. Pikemalt käsitletakse ühte maastiku genereerimise algoritmi, mis on loodud kasutades üldlevinud maastiku genereerimise ning autori poolt väljatöötatud võtteid. Kirjeldatav algoritm on võimeline genereerima ruumis muutuva kujuga lõpmatut maastikku ning erineva välimusega makroökosüsteeme ehk bioome neile sobiva pinnase ja taimestikuga. Töös kirjeldatavad bioomide ning taimede paigutuse määramise algoritmid on loodud autori poolt ning teadaolevalt pole neid varem kasutatud. Bakalaureusetööga paralleelselt valmis osalt aine „Arvutigraafika projekt“ raames töös kirjeldatavat algoritmi kasutav rakendus, millest on lähemalt juttu vastavas peatükis.

Töö teises peatükis antakse ülevaade algoritmile püstitatud nõuetest ning kirjeldatakse valminud rakendust. Järgnevas kahes peatükis antakse ülevaade maastiku kirjeldamiseks kasutatavatest andmestruktuuridest ning nende visualiseerimise võimalustest. Viiendas peatükis kirjeldatakse müraalgoritmide eesmärke ning protseduurilises genereerimises kolme enimlevinuma müraalgoritmi tööpõhimõtteid. Kuuendas peatükis näidatakse müraalgoritmide üldlevinud kasutusvõimalusi lihtsa lõpmatu maastiku genereerimiseks. Seitsmendas peatükis käsitletatakse bioome, maastiku jagamist bioomideks ning seejärel maastiku algoritmilist kujundamist vastavalt piirkonna bioomile. Kaheksandas peatükis kirjeldatakse taimede paigutamisega kaasnevaid probleeme ning nende ühte võimalikku lahendust kasutades müraalgoritme. Üheksandas peatükis võrreldakse töö varasemas osas kirjeldatud algoritmi implementeeriva rakenduse nõuetele vastavust, sealhulgas pikemalt peatutakse töökiirusel. Kümnendas peatükis pakutakse välja mõned käesoleva algoritmi edasiarendusvõimalused ning kirjeldatakse nendega kaasnevaid lahendamist vajavaid probleeme. Viimases peatükis antakse ülevaade tehtud tööst, saadud uutest teadmistest ning lõppsõna.

2. Rakenduse ülevaade

Käesoleva bakalaureusetööga paralleelselt valmis osaliselt Tartu Ülikooli Arvutiteaduse instituudi aine „Arvutigraafika projekt“ (MTAT.03.316, 3 EAP)[2] raames järgnevalt kirjeldatavat algoritmi implementeeriv rakendus. Töös kirjeldatakse esinenud probleeme ning nende lahendusi ja põhjendatakse algoritmi koostamisel tehtud valikuid. Algoritmi implementeerimiseks kasutati mängumootorit Unity[3] ja programmeerimiskeelt C#. Valminud rakendus on avalikult kättesaadav versioonihaldustarkvara Git repositooriumist[4] ning bakalaureusetöö lisana iseseisva kompileeritud rakendusena.

Tasub märkida, et täielikult lõpmatu maastiku genereerimine on keerukas ülesanne, sest arvestama peab piisavalt suurte distantside läbimisel tekib arvuti mälus koordinaatide ja muude maastikku kirjeldavate muutujate ületäitumise probleem. Enamasti on selliste probleemide tekkimiseks vajalik kasutaja poolt läbitav distants niivõrd suur, et tavakasutaja selleni ei jõua. Seetõttu rakenduse poolt genereeritav maastik pole küll tehniliselt lõpmatu, kuid probleemide ilmnemiseks peaks kasutaja rakenduses eelnevalt päevi, kui mitte kuid, ühes suunas suurel kiirusel liikuma ja seetõttu võib lugeda rakenduse poolt genereeritava maastiku siiski lõpmatuks.

Töö mahu tõttu jäi rakendusest välja mitme lõime peal töötamise tugi. Seetõttu toimub maastiku genereerimine ja joonistamine ühe lõime peal ning uute maastikutükkide genereerimisel on näha lühiajalisi märgatavaid rakenduse aeglustusi. Autor implementeeris töö käigus väärtus- ning Perlini müra, kuid rakenduses on kasutatud müra arvutamiseks Jasper Flicki poolt loodud teeki, mis sisaldab ka simpleksmüra arvutamist [9].

Enne töö alustamist püstitas töö autor viis nõuet, millele valminud algoritmi poolt genereeritav maastik peab vastama. Nendeks nõueteks on:

- lõpmatu suurus – liikudes mööda vabalt valitud suunda maastikul, on alati võimalik genereerida vajalikud maastikutükid, et liikumist lõpmatult jätkata, kui piiratud arvuandmestruktuuridest tulenevaid probleeme ei arvestata;

[2] https://courses.cs.ut.ee/2016/cg-project/spring/Main/HomePage

[3] https://unity3d.com/

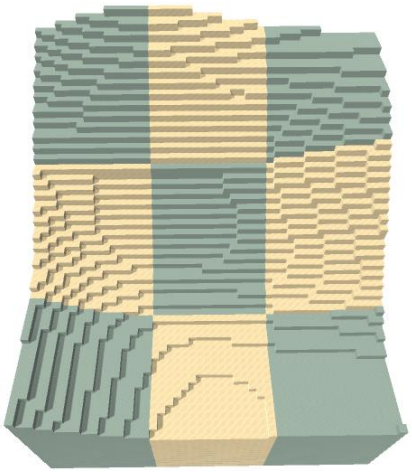
[4] https://bitbucket.org/AndreasGP/procedural-land-generation-with-unity

- determineeritus – andes algoritmile lõpliku hulga sisendparameetreid, on võimalik iga kord samade parameetritega genereerida täpselt samasugune maastik;
- omavahel ühilduv – kõrvuti genereeritud maastikutükid sobivad omavahel kokku;
- kiirus – mõistliku kiirusega maastikul liikudes on reaalajas võimalik eesolevat maastiku genereerida ilma kasutajat häirimata;
- mitmekülgsus – genereeritav maastik on erinevates piirkondades omapärane.

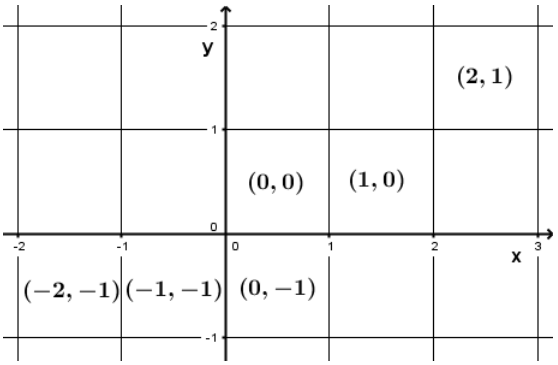
Täpsem ülevaade rakenduse vastavusest püstitatud nõuetele antakse peatükis 9.

3. Maastiku andmestruktuur

Maastiku genereerimise algoritmi kirjeldamiseks on vaja esmalt määrata maastiku hoiustamiseks sobivad andmestruktuurid. Kuna maastik on lõpmatu, ei ole sobilik maastikku hoida vastavalt kasutaja liikumisele rakenduses pidevalt uueneva ühe tükina, vaid mitmete väiksemate tükkidena. Selline hoiustamine võimaldab äärtesse jooksvalt maastikutükke juurde genereerida ja eemaldada. Antud töös käsitletakse maailma kui lapikut kolmemõõtmelist eukleidilist ruumi, kus z-koordinaattelg on suunaga üles. Selles ruumis paikneb maastik, mis koosneb maastikutükkidest. Iga maastikutükk kirjeldab ülevalt vaadates maastikust ühte väljalõigatud tükki. See tähendab, et iga maastikutükk koosneb maastikust alates kõige sügavamast maakihist kuni kõige ülemise kihini maastikutüki pikkuse ja laiuse piires. Selliste maastikutükkide kõrvuti paigutamisel xy-tasandile tekib maastikutükkidest koosnev ruudustik, mis moodustab tervikliku maastiku (vt. joonis 1).



Joonis 1. Vokslitena kujutatud 9 maastikutükki, mis moodustavad antud piirkonna maastiku.



Joonis 2. Maastikutükkide (ruudud) koordinaadid tasandil.

Iga maastikutükki saab kirjeldada selle ruumikoordinaatide kaudu. Kuna iga maastikutükk sisaldab kogu võimalikku maastikku z-telge mööda maastikutüki laiuse ja pikkusega määratud alas, asuvad kõik maastikutükid samal xy-tasandil ning iga maastikutüki asukoha määramiseks piisab ainult tüki x- ja y-koordinaatidest. Nende koordinaatideks on täisarvulised väärtused, mis kirjeldavad mitme maastikutüki kaugusel koordinaattelgede nullpunktist antud maastikutükk kumbagi telge mööda asub (vt. joonis 2). Teisisõnu on igal maastikutükil olemas selle asukohta tasandil määravad Cartesiuse koordinaadid (x_{mt}, y_{mt}), kus $x_{mt}, y_{mt} \in \mathbb{Z}$.

Iga maastikutükk koosneb omakorda kolmemõõtmelisest fikseeritud suurusega massiivist, mille iga väärtus kirjeldab ühe punkti ehk maastikutüki väikseima osa maastikutüüpi, nagu näiteks muru, graniit, liiv, vesi või õhk. Igal maastikutükil on seega olemas kolm mõõtu: laius w_{mt}, pikkus l_{mt} ja kõrgus h_{mt}. Igal punktil on olemas maastikutükisisesed ehk lokaalsed koordinaadid:

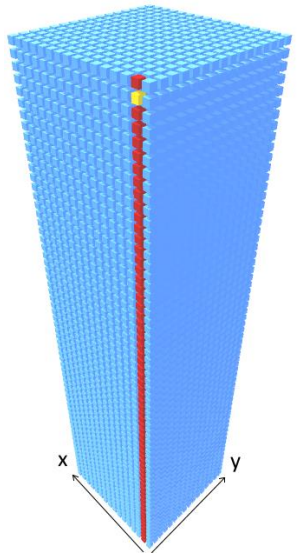
$$(x_{lp}, y_{lp}, z_{lp}),\ \text{kus}\ \begin{matrix} x_{lp} \in \{0, 1, \dots,\ w_{mt} - 1\}, \\ y_{lp} \in \{0, 1, \dots,\ l_{mt} - 1\}, \\ z_{lp} \in \{0, 1, \dots,\ h_{mt} - 1\}. \end{matrix}$$

Lisaks saab iga punkti lokaalsete koordinaatide ning punktile vastava maastikutüki koordinaatide kaudu arvutada punkti unikaalsed globaalsed koordinaadid:

$$(x_{gp}, y_{gp}, z_{gp}) = (x_{lp} + x_{mt} * w_{mt},\ y_{lp} + y_{mt} * l_{mt},\ z_{lp}).$$

Saadud globaalsed koordinaadid vastavad üks-üheselt maastikutüki ühele punktile.

Antud töös käsitletakse lisaks veel maastikutulpi, mis hõlmavad ühe vertikaalse tulba punkte mööda z-telge (vt joonis 3) . Ühel maastikutükil on maastikutulpi $w_{mt} * l_{mt}$ tükki. Iga tulp on tuvastatav selle unikaalsete globaalsete koordinaatide abil, milleks on tulbas asuva suvalise punkti globaalsed x_{gp} ja y_{gp} koordinaadid.



Joonis 3. Maastikutükk (sinine), üks selle maastikutulp (punane) ja üks selles tulbas paiknev punkt (kollane) lokaalsete koordinaatidega (1, 0, 62).

Järgnevalt pakutakse välja paar võimalust, kuidas antud andmestruktuure efektiivsemalt rakendada. Esmalt kirjeldatatakse kahte valikut maastikutükkidele viidete hoiustamiseks. Esimene neist sobib hästi ühe kasutajaga rakendusse ning koosneb fikseeritud suurusega kahemõõtmelisest massiivist, mille nii-öelda keskpunkt sisaldab viidet maastikutükile, kus kasutaja parajasti asub. Kasutaja liikumisel nihutatakse massiivi elemente vastavalt liikumise suunale ja massiivist välja jäävad tükid eemaldatakse ning uued puuduvad tükid genereeritakse liikumisel vabanenud massiivi kohtadele. Selline massiiv ei võimalda eriti mugavalt korraga laadida mitut üksteisest suvalisel kaugusel asuvat maastiku piirkonda. Olukorras, kus see on siiski vajalik,

nagu näiteks mitme kasutajaga rakenduses, tasuks kasutada järgmist lähenemist. Sellisel juhul on sobilik andmestruktuur sõnastik (ing. k. *dictionary*), mille võtmeteks on maastikutükkide koordinaadid ning väärtusteks maastikutükkide viited. Niimoodi on võimalik arvutuslikult kiirelt kontrollida, kas mingi maastikutükk on genereeritud, ning genereerida üksteisest suvalisel kaugusel asuvaid maastikutükke.

Lisaks on kasulik määrata maastikutükkide mõõtmeteks arvu 2 astmed, sest nendega algoritmis tihti tehtavate jagamise[5] ja jäägi[6] leidmise tehted on võimalik lihtsustada kiiremate bitioperatsioonide sooritamiseks. Samuti on mugavam valida maastikutükkidele sama pikkus ja laius, sest siis on maastikutükkidega tegelemisel vaja arvestada ainult kahe mõõduga – pikkuse ja kõrgusega. Eelnevaid punkte arvestades on maastikku kujutavates illustratsioonides ning valminud rakenduses kasutatud maastikutükke suurusega 16 punkti laiust ja pikkust ning 64 punkti kõrgust (vt. joonis 3) ning järgnevalt käsitletakse ainult võrdse pikkuse ja laiusega maastikutükke.

[5] https://stackoverflow.com/questions/17135112/bitwise-division-by-multiples-of-2

[6] https://stackoverflow.com/questions/6670715/mod-of-power-2-on-bitwise-operators

4. Maastiku visualiseerimine

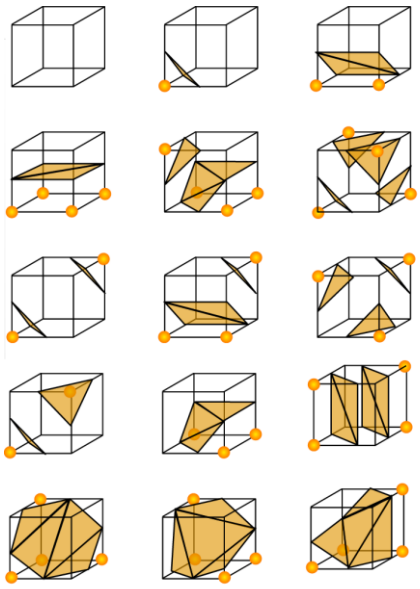
Eelmises peatükis kirjeldatud andmestruktuuriga maastiku visualiseerimiseks on mitmeid võimalusi. Järgnevalt kirjeldatakse neist pikemalt kahte ning kirjeldatakse põgusalt ka teisi võimalusi.

4.1 Vokslid

Vokslite (ing. k. *voxels*) kasutamine on kõige lihtsam viis maastiku visualiseerimiseks peale maastiku lihtsalt punktidena ruumis esitamist. Vokslite kasutamisel maastiku visualiseerimiseks kujutatakse iga maastikupunkti, mis ei ole tüübiga õhk, kuubina, kui antud punktis ei paikne tühjus (vt. joonis 1) [10]. Seejuures joonistatakse ainult kuupide need tahud, mis ei ole naaberkuupide poolt kaetud. Antud töös visualiseeritakse maastikku kasutades vokslite algoritmi, sest seda on võrreldes järgnevatega lihtsam rakendada. Lisaks on vokslite lihtsa kuju ja nende omavahelise selge visuaalse eraldatuse tõttu lihtsam illustratsioonidelt näha, missugune väärtus maastiku andmestruktuuris vokslile vastaval kohal olla võiks.

4.2 Piirnevate kuubikute algoritm

Piirnevad kuubikud (ing. k. *marching cubes*) on 1987. aastal William Lorenseni ja Harvey Cline'i loodud algoritm, mis teisendab kolmemõõtmelise regulaarse punktivälja kolmnurkadest koosnevaks lihtsasti joonistatavaks ruumiliseks kujundiks [11]. Selline punktiväli on näiteks eelmises peatükis defineeritud maastikutüki andmestruktuur. Algoritmi rakendamiseks peab igal punktivälja punktil olema väärtus, mille abil saab määrata, kas antud punkt asub punktiväljal kujutatud objekti sees või väljaspool. Maastiku korral on objektist väljaspool asuvad punktid need, mille maastikutüüp vastab tühjusele. Algoritm itereerib üle punktivälja vaadates 8 kaupa punkte, mis moodustavad väikseima punktiväljal konstrueeritava kuubi. Iga 8 punkti jaoks genereeritakse nende väärtustele vastav bitimask (ing. k. *bit mask*), kus iga bitt kirjeldab ühe punkti kuuluvust objekti sisemusse. Seejärel kontrollitakse, kas leidub bitimaskile vastav teisendus kolmnurkadeks (vt. joonis 4). Juhul, kui teisendust ei leidu, nende 8 punkti alasse visuaalselt midagi ei lisata. Pärast üle punktivälja itereerimist tekib ühtne ruumiline kujund. Tekkinud kujundi tippudele tekstuuri määramisel vastavalt tipule lähima punkti maastikutüübile saame tulemuseks maastiku visualiseeringu (vt. joonis 5).



Joonis 4. Piirnevate kuubikute algoritmi kolmnurkade paigutuse võimalused. Kollased tipud vastavad objekti sisepunktidele, ülejäänud kuubi tipud objektist väljajäävatele punktidele [33].

Joonis 5. Shamus Youngi poolt genereeritud piirnevaid kuubikuid (v.a. maja) kasutav maastik [31].

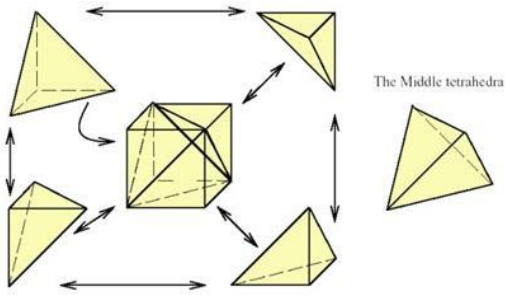
Piirnevate kuubikute algoritmi üheks puuduseks lõpmatu maastiku genereerimisel on vajadus arvutada peale iga maastikutüki väärtuste lisaks ka väärtused ühe punkti võrra maastikutükist väljaspool mööda x- ja y-telge. Kui neid lisaväärtusi ei genereeritaks, ei saaks maastikutüki äärtes algoritmi rakendada, sest seal on kolmnurkade määramiseks vaja kasutada ka punkte väljaspool vaadeldavat maastikutüki punktivälja.

4.3 Muud algoritmid

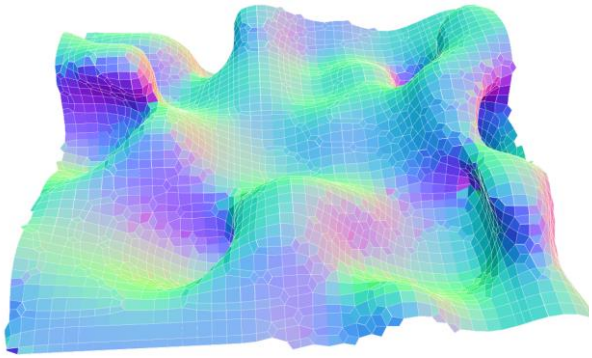
Lisaks vokslitele ja piirnevatele kuubikutele on parema visuaalse tulemuse saavutamiseks veel mitmeid algoritme, millest antakse käesolevas peatükis ainult ülevaatlik kirjeldus.

Üks selline algoritm on piirnevate tetraeedrite algoritm (ing. k. *marching tetrahedra*), mis on piirnevate kuubikute edasiarendus [12]. Iga punktidest moodustuv kuup jagatakse omakorda viieks tetraeedriks (vt. joonis 6), mille nurkade väärtustele vastavate biti-maskide põhjal valitakse genereeritavad kolmnurgad. Piirnevate tetraeedrite algoritmi kasutamine võimaldab genereerida sama punktivälja jaoks detailsema mudeli kui piirnevate kuubikute algoritm.

Samuti võib kasutada maastiku genereerimiseks pinna võrgustiku algoritmi (ing. k. *surface nets*) [13]. Pinna võrgustiku algoritmi poolt visualiseeritud maastik meenutab voks-



Joonis 6. Kuubi jaotamine viieks tetraeedriks [30].



Joonis 7. Pinna võrgustiku algoritmiga genereeritud maastik [13].

litest joonistatud maastikku, kus kuupide nurki on moonutatud, et tulemus sujuvam välja näeks. Täpsemalt analüüsitakse iga punkti jaoks selle ümbruses asuvate punktide väärtusi, et määrata genereeritava voksli tippude nihked (vt. joonis 7).

Täpsemalt on võimalik piirnevate kuubikute, piirnevate tetraeedrite ning pinna võrgustiku algoritmi tulemusi võrrelda Mikola Lysenko poolt loodud interaktiivses veebirakenduses[7].

[7] https://mikolalysenko.github.io/Isosurface/

5. Müraalgoritmid

Arvutigraafikas on müra n-kohaline deterministlik funktsioon f, mis kujutab iga n-kohalise reaalarvude vektori mingiks näivalt juhuslikuks reaalarvuks:

$$f: \mathbb{R}^n \to \mathbb{R} \text{ [14].}$$

Mitmed müraalgoritmid kasutavad mõnel oma sammul väärtuste genereerimiseks pseudojuhuslike arvude generaatorit ning seetõttu on sellistel algoritmidel deterministliku tulemuse saavutamiseks veel lisaks täisarvuline argument, mida kasutatakse arvugeneraatori seemnena (ing. k. *seed*). Sama seemne kasutamine tagab, et arvugeneraator väljastab alati samad pseudojuhuslikud arvud samas järjekorras.

Järgnevad müraalgoritmid on sobilikud lõpmatu maastiku genereerimiseks, sest neil on kaks vajalikku omadust. Esiteks on võimalik kõigi müraalgoritmide väärtust suvalises punktis leida, ilma et oleks vaja eelnevalt teada ühegi teise punkti väärtust. See on kasulik, sest müra kasutamisel maastiku genereerimisel on vaja leida ainult müra väärtused uuritava maastikutüki piires ning nende leidmiseks ei ole vaja mingisugust lisainfot selle maastikutüki ümber paiknevate maastikutükkide kohta. Teiseks on väljavalitud mürad pideva tuletisega [15,16]. Lihtsustatult tähendab see, et lähestikku paiknevate punktide korral on mürade väärtused piisavalt väikese erinevusega, mistõttu iga maastikutüki koordinaatide põhjal genereeritud väärtused sobituvad kokku. Sellest tuleb pikemalt juttu peatükis 6.

Järgnevalt kirjeldatakse kolme erinevat müraalgoritmi, mis on üksteise edasiarendused ning aitavad üksteist paremini mõista, kuigi maastiku genereerimisel kasutatakse neist põhiliselt ainult simpleksmüra. Kõigi müraalgoritmide korral kirjeldatakse kõigepealt niiöelda ühe kihi jaoks väärtuste leidmist mingis punktis. Tavaliselt kasutatakse müraalgoritme mitme kihiga ehk leitakse mitu erinevat väärtust erineva resolutsiooniga kihtidel ning seejärel summeeritakse neist lõplik müra väärtus vaadeldavas punktis. Kihtidest ja nende summeerimisest on pikemalt juttu väärtusmüra peatüki lõpus. Järgnevaid algoritme kirjeldatakse ühe- või kahemõõtmelise ruumi näite põhjal, kuid neid on võimalik üldistada suvalises n-mõõtmelises ruumis rakendamiseks.

Enne müraalgoritmide detailsemat käsitlust on vajalik mõista, mis on nendes algoritmides olulisel kohal olev võrestik. Võrestik (ing. k. *grid*) on tessellatsiooni (ing. k. *tesselation*)

ehk tasandi lõikumatuteks geomeetrilisteks kujunditeks jagamise [17] üldistus suurematesse dimensioonidesse, mis jagab vaadeldava n-dimensioonilise ruumi omavahel lõikumatuteks sama kuju ja suurusega kujunditeks. Näiteks kahemõõtmelises ruumis võib võrestiku koostada tasandit ruutudeks või kolmnurkadeks jagades ning kolmemõõtmelises ruumis kuupideks või tetraeedriteks jagades [16]. Võrestiku moodustavate kujundite nurki nimetatakse võrestiku tippudeks ning igal võrestiku tipul on selle asukohta ruumis määravad koordinaadid.

5.1 Väärtusmüra

Väärtusmüra rakendamiseks on vaja kõigepealt defineerida algoritmis kasutatav võrestik ja võrestiku tippudel defineeritud funktsioon f_{tipp}. Kõige lihtsamaks võrestikuks sobib n-dimensioonilise ruumi ühikhüperkuupidega tessellatsioon. Näiteks ühedimensioonilises ruumis oleks ruum jagatud ühiklõikudeks, kahedimensioonilises ruumis ühikruutudeks jne. Kui ruum on jagatud ühikhüperkuupideks, siis võrestiku tipud moodustavad parajasti kõigi täisarvuliste koordinaatidega ruumipunktide hulga.

Järgmiseks on vaja defineerida võrestiku tippudel, antud juhul n-mõõtmeliste täisarvude vektorite hulgal, funktsioon f_{tipp}, mis kujutab iga võrestiku tipu koordinaadid pseudo-juhuslikuks reaalarvuks:

$$f_{tipp}: \mathbb{Z}^n \to \mathbb{R}.$$

Funktsiooni f_{tipp} poolt määratud väärtused peavad olema ühtlase jaotusega ning funktsiooni tulemuse visualiseerimisel ei tohi märgatavaid mustreid tekkida.

Üks võimalus mürafunktsiooni konstrueerimiseks on kõigepealt genereerida m üksteisest võrdse intervalli võrra erinevat reaalarvu, mille väikseim ja suurim väärtus määravad väärtusmüra minimaalse ja maksimaalse väärtuse. Tavaliselt tagastavad mürafunktsioonid väärtuse lõigus $[-1, 1]$ või $[0, 1]$ [14,18]. Kuna maastiku genereerimisel pole negatiivsetest väärtustest kasu, siis siinkohal eelistatakse lõiku [0, 1]. Seega on siinkohal mõttekas valida väikseimaks ning suurimaks väärtuseks vastavalt 0 ja 1. Näiteks $m = 11$ korral oleks genereeritavatakse arvudeks $\{ 0, 0.1, \dots, 0.9, 1 \}$. Soovituslik on valida $\mathrm{m} \geq 100$, et jagada lõik rohkemateks tükkideks, tekitamaks võrestiku väärtuste hulgas suuremat variatsiooni.

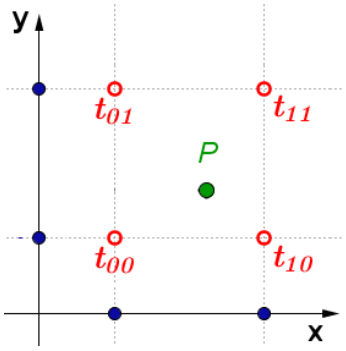
Pärast m järjestikuse arvu genereerimist on vaja leida üks nende arvude pseudojuhuslik permutatsioon ning see muutumatult kogu väärtusmüra kasutamise ajaks massiivi talletada. Sellise permutatsiooni leidmiseks sobib hästi Fisher-Yatesi algoritm, mis arvutab lõpliku hulga suvalise permutatsiooni, kus kõik permutatsioonid on sama tõenäosusega [19]. Kuna mürafunktsioon ise on deterministlik, peab ka funktsioon f_{tipp} olema deterministlik ning seetõttu peaks ka Fisher-Yatesi algoritmi töö olema determineeritud. Selle saavutamiseks piisab, kui kasutada permutatsiooni arvutamisel pseudojuhusliku arvugeneraatori seemnena nn müraseemet, mis tagab, et sama seemnega väärtusmüra korral leitakse alati sama permutatsioon, kui algselt genereeritud m arvu ei muudeta.

Kui massiiv pseudojuhuslike arvudega on genereeritud, saab funktsiooniks f_{tipp} määrata funktsiooni, mis kujutab võrestiku tipu koordinaadid pseudojuhuslikuks arvuks massiivis. Funktsiooni valikul peab arvestama, et telgedega paralleelsetel sirgetel paiknevad võrestiku tipud ei hakkaks mingi väikse intervalli tagant tulemuseks samu või lähedasi reaalarve andma [18]. Vastasel korral on võrestiku tippude väärtuste visualiseerimisel selgelt näha korduvad mustrid, kuid soovitav tulemus peab olema võimalikult juhusliku väljanägemisega. Jasper Flick pakub välja võimaluse kasutada permuteeritud väärtustega massiivist väärtuste võtmist korduvalt, et tekitada arvude näiline juhuslik jaotus [18]. Näiteks leidmaks kahemõõtmelises ruumis võrestiku tipu (x, y) f_{tipp} väärtust, võiks kasutada Jasper Flicki funktsioonile sarnast funktsiooni

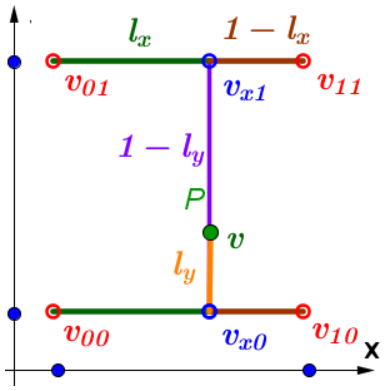
$$f(x, y) = perm[(perm[x \bmod m] + y) \bmod m],$$

kus $perm[i]$ tähistab permutatsiooni massiivi i'ndat elementi. Analoogiliselt saab seda funktsiooni täiendada nii, et kujutada suvaline n-dimensiooniline vektor üheks reaalarvuks. Selle funktsiooni väärtused hakkavad iga m ühiku järel mööda x- ja y-telge korduma, kuid $m \geq 100$ valimisel korduvaid mustreid märgata ei ole.

Vaatleme nüüd kahekohalise väärtusmüra funktsiooni väärtuse leidmist punktis $P = (x, y)$ Jasper Flicki implementatsiooni baasil [18]. Kuna võrestiku tippudeks on täisarvuliste koordinaatidega vektorite hulk, saab punkti P koordinaate lähima täisarvuni alla ümardades leida võrestiku tipu t_{00}, mis on punktile P lähim sellest väiksemate või võrdsete koordinaatidega võrestiku tipp. Seejärel saab leida punkti P ümbritsevad ülejäänud kolm võrestiku tippu, liites t_{00}-le vektorid $(1, 0), (0,1)$ ja $(1, 1)$ ning saades vastavalt võrestiku tipud t_{10}, t_{01} ja t_{11} (vt. joonis 8). Analoogiliselt saab leida suvalises n-



Joonis 8. Väärtusmüra võrestiku tippude ja P paiknemine kahemõõtmelise ruumi korral.



Joonis 9. Võrestiku tippude väärtuste bilineaarne interpoleerimine.

mõõtmelises ruumis 2^n võrestiku tippu, mis moodustavad otsitava punkti ümber ühikhüperkuubi.

Pärast lähimate võrestiku tippude t_{00}, t_{10}, t_{01} ja t_{11} leidmist arvutatakse funktsiooni f_{tipp} kasutades neile vastavad väärtused v_{00}, v_{10}, v_{01} ja v_{11}. Seejärel interpoleeritakse teada-olevatest väärtustest otsitav väärtus punktis P.

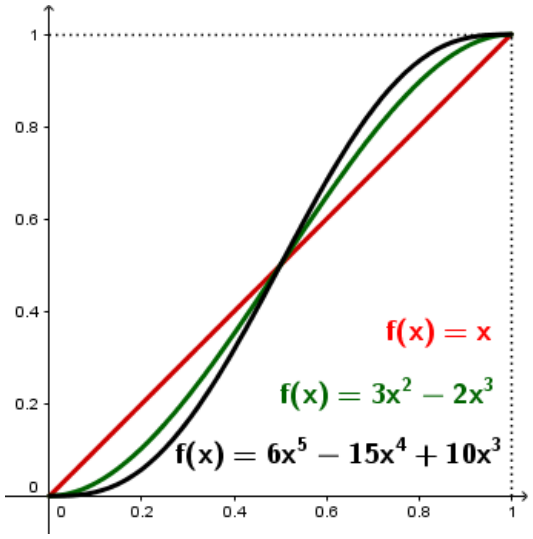
Väärtuste interpoleerimiseks kasutatakse segunemisfunktsiooni (ing. k. *blending function*). Segunemisfunktsioon on monotoonselt kasvav funktsioon $f_{seg}: [0,1] \to [0,1]$, mida kasutatakse arvutamaks iga leitud võrestiku tipu t_{ij} väärtuse osakaalu punkti P väärtusesse. Kõige lihtsam segunemisfunktsioon on $f_{seg}(x) = x$. Sellise lineaarse segunemisfunktsiooni kasutamisel võrestiku tippude piiridel pole mürafunktsiooni tuletis pidev. Lihtsustatult tähendab see, et võrestiku tippude piiridel olevad väärtused on väga märgatava üleminekuga (vt. joonis 11). Seetõttu on sobilik valida segunemisfunktsioon, mille esimest ja teist järku tuletis punktides 0 ja 1 oleks 0 [18]. Täpsem põhjendus, miks esimest järku nulltuletisest otspunktides ei piisa, on leitav Ken Perlini artiklis [15]. Selliseks segunemisfunktsiooniks sobib viienda astme polünoom [16]

$$f_{seg}(x) = 6x^5 - 15x^4 + 10x^3.$$

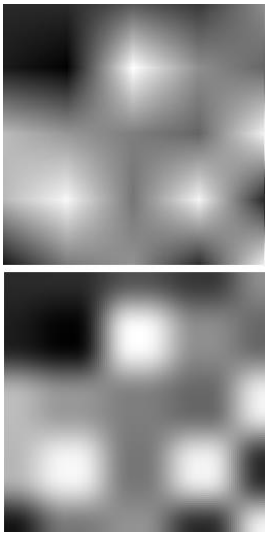
Tõepoolest, antud funktsiooni esimest ja teist järku tuletis punktides 0 ja 1 on 0:

$$f'_{seg}(x) = 30x^4 - 15x^4 + 30x^2, \quad f'_{seg}(0) = 0, \quad f'_{seg}(1) = 0,$$

$$f''_{seg}(x) = 120x^3 - 180x^3 + 60x, \quad f''_{seg}(0) = 0, \quad f''_{seg}(1) = 0.$$



Joonis 10. Punktides 0 ja 1 nullindat, esimest ja teist järku nulltuletistega segunemisfunktsioonid [16].



Joonis 11. Ühekihiline 2D väärtusmüra kasutades lineaarset (üleval) ja teist järku nulltuletisega (all) segunemis-funktsiooni.

Pärast segunemisfunktsiooni valimist, leiame väärtuse punktis $P = (x, y)$. Selleks leiame esiteks kaalu l_x , mis kirjeldab tippude t_{10} ja t_{11} kaugust punktist P x- teljel:

$$l_x = \frac{(x - x_{t_{00}})}{(x_{t_{10}} - x_{t_{00}})} \in [0, 1].$$

Seejärel on võimalik leida kaks interpoleeritud väärtust x-teljel (vt. joonis 9):

$$v_{x0} = v_{00} * f_{seg}(l_x) + v_{10} * (1 - f_{seg}(l_x)),$$

$$v_{x1} = v_{01} * f_{seg}(l_{x)} + v_{11} * (1 - f_{seg}(l_x)),$$

Järgmisena leitakse kaal l_y, mis kirjeldab v_{x0} (muuhulgas ka t_{00} ja t_{10}) kaugust punktist P y-teljel:

$$l_y = \frac{(y - x_{t_{00}})}{(y_{t_{01}} - x_{t_{00}})}$$

ning seejärel v_{x0} ja v_{x1} kaalude põhjal otsitav väärtus punktis P:

$$v = v_{x0} * f_{seg}(l_y) + v_{x1} * (1 - f_{seg}(l_y)).$$

Analoogiliselt on võimalik lõppväärtuse interpoleerimist mööda telgi laiendada suvalise n-dimensioonilise ruumi juhule, kus iga lisanduv dimensioon lisab juurde ühe sammu interpoleerimisi mööda lisanduva dimensiooni telge.

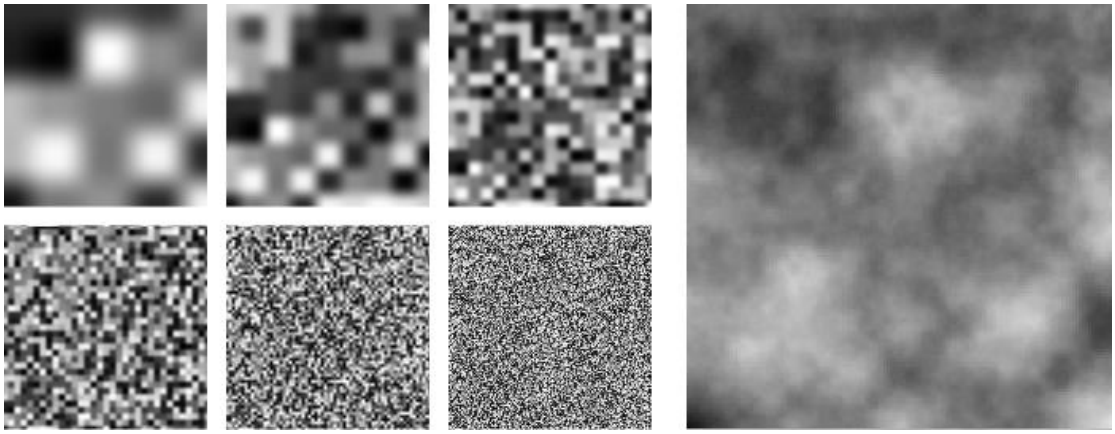
Kui visualiseerida sellisel viisil väärtuse leidmist pildina, kus pikslite koordinaate kasutatatakse punktide koordinaatidena, siis on üpris selgelt näha, kus asuvad võrestiku tipud (vt. joonis 11). Tulemuse korrapärasuse peitmiseks ning mitmekülgsuse lisamiseks arvutatakse tavaliselt iga punkti jaoks mitu väärtust mitmel võrestikul, mille punktidevaheline kaugus erineb, ning tulemused summeeritakse. Igal erineval võrestikul väärtuse leidmist nimetatakse kihi väärtuse leidmiseks. Mitmest kihist koosneva tulemuse kirjeldamiseks on mitmeid parameetreid. Järgnevalt esitletakse üldlevinud müra parameetreid, toetudes Jasper Flicki implementatsioonile [18]. Esimene neist on kihtide arv, mis määrab, mitu erinevat kihti väärtusmüra on kokku liidetud. Teine parameeter on algsagedus (ing. k. *frequency*), mille pöördväärtus kirjeldab, kui suur on võrestiku tippude vaheline kaugus väärtusmüra esimesel kihil. Senises näites paiknesid võrestiku tipud täisarvulistel koordinaatidel, ehk seega oli algsageduseks $\frac{1}{1} = 1$. Kui esimese kihina oleks kasutatud võrestikku, mille tippudevaheline kaugus oli 0.5, oleks algsagedus olnud $\frac{1}{0.5} = 2$. Kolmas parameeter on lakunaarsus (ing. k. *lacunarity*) ehk järgmise kihi lünklikuse kordaja. Lakunaarsus kirjeldab, kui palju iga järgneva kihi võrestiku tippude vaheline kaugus muutub võrreldes eelmisega. Selle teguriga korrutatakse alates teisest kihist eelmise kihi sagedust, et saada uue kihi sagedus. See tähendab, et näiteks algsagedusega $\frac{1}{4}$ ning lakunaarsusteguriga 2, on teise kihi sageduseks $\frac{1}{4} * 2 = \frac{1}{2}$, kolmanda kihi sageduseks $\frac{1}{2} * 2 = 1$ jne. Viimane parameeter on püsivus (ing. k. *persistence*), mis määrab iga kihi mõju suuruse lõpptulemusele. Püsivustegur kirjeldab, mitu korda vähem iga järgnev kiht lõpptulemust mõjutab võrreldes sellele eelneva kihiga. Näiteks kolme kihi ja püsivusteguri 0.5 korral on kõikide kihtide kaalude kogusumma $\sum_{i=1}^{3} 0.5^i$ ning iga kihi kaal oleks sellele vastava kogusumma liikme ning kogusumma suhe. Seega nende kolme kihi kaalud oleksid vastavalt

$$\frac{0.5}{0.5 + 0.5 * 0.5 + 0.5 * 0.5 * 0.5} \approx 0.571,$$

$$\frac{0.5 * 0.5}{0.5 + 0.5 * 0.5 + 0.5 * 0.5 * 0.5} \approx 0.286,$$

$$\frac{0.5 * 0.5 * 0.5}{0.5 + 0.5 * 0.5 + \ 0.5 * 0.5 * 0.5} \approx 0.143,$$

kogusummaga 1.



Joonis 12. Kuus üksikkihti väärtusmüra ning nendest kombineeritud müra algsagedusega $\frac{1}{32}$, lakunaarsusteguriga 2 ja püsivusteguriga 0.5.

Üldistatud kujul avaldub kihi kaal kujul

$$f_{kaal}(kiht) = \frac{püsivus^{kiht}}{\sum_{i=1}^{kihte} püsivus^{i}}.$$

Kombineerides mitu kihti, saame tulemuseks pildi, mis meenutab pehmeid pilvi. Jooniselt võib näha, et erinevates lõpptulemuse piirkondades on kõige rohkem näha kõige väiksema sagedusega ning kõige suurema kaaluga kihi väärtusi (vt. joonis 12 üleval vasakul), kuid lähemalt vaadates on näha ka teiste kihtide mõju.

5.2 Gradientmüra

Kuigi väärtusmüra näeb välja sujuv ning juhuslik, on siiski näha, et võrestikuna on kasutatud ruudustikku. See on ebasobilik, kui eesmärgiks on saavutada kaootilisema ning looduslikuma välimusega müra. Selle parandamiseks sobib väärtusmüra asemel kasutada gradientmüra, mille tulemusest pole enam võimalik võrestikku märgatavalt eraldada.

Gradientmüra on väärtusmürale üldjoontes sarnane müra, kuid igale võrestiku tipule määratakse skalaarse väärtuste asemel pseudojuhuslik gradient. Gradient on mitme muutuja funktsiooni f kiireima kasvamise suunda ja kiirust antud punktis iseloomustav vektor [20]. Lihtsustatult määratakse n-dimensioonilises ruumis igale võrestiku tipule n-kohaline vektor g (tähistatakse ka ∇f) ning müra väärtus leitakse gradientide abil moodustatud funktsiooni kaudu. Järgnevalt kirjeldatakse kahte enimlevinumat gradientmüra ning nende eeliseid väärtusmüra ees.

5.2.1 Perlini müra

Perlini müra oli esimene gradientmüra algoritm, mille töötas välja 1983. aastal Ken Perlin [14]. Perlin sai selle loomise eest 1997. aastal Ameerika Filmikunsti ja -teaduste Akadeemialt tehnilise saavutuse Oscari ning see algoritm leiab tänapäevani väga laialdast kasutust filmikunstis eriefektide tegemisel [14,21].

Perlini müra kasutab sarnaselt väärtusmürale võrestikuna ruumi jaotamist hüperkuupideks, mille tipud moodustavad lihtsaimal juhul täisarvuliste koordinaatidega vektorite hulga.

Enne Perlini müra kasutamist defineeritakse funktsioon g_{tipp}, mis seab igale võrestiku tipule vastavusse pseudojuhusliku gradiendi:

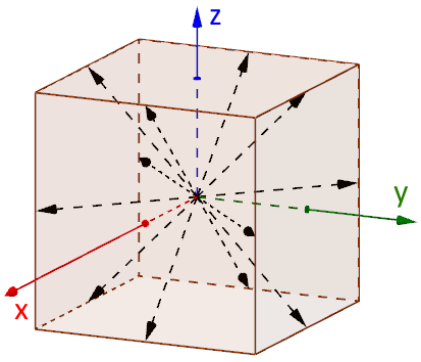
$$g_{tipp}\colon \mathbb{R}^n \to \mathbb{R}^n.$$

Põhimõtteliselt pole vahet, millised gradiendid funktsioon täpselt määrab, kuni need on ühtlase jaotusega ruumis, vältimaks gradientide soodumust mingis kindlas suunas [16]. Kasulik on valida gradientideks võrdse pikkusega vektorid, mis on jaotatud ruumis ühtlaselt. Sellisel juhul on lihtne veenduda, et vektorid on tõepoolest ühtlase jaotusega. Kolmemõõtmelise müra jaoks soovitab Ken Perlin valida vektorid koordinaatide nullpunktist sellise kuubi servade keskpunktidesse, mille servapikkus on 2 ja mille keskpunkt asub koordinaatide nullpunktis (vt. joonis 13) [15]. Sellisel juhul on võimalike gradientide hulk järgnev:

$$g_1 = (1,1,0),\ g_2 = (1,-1,0),\ g_3 = (-1,1,0),\ g_4 = (-1,-1,0),$$

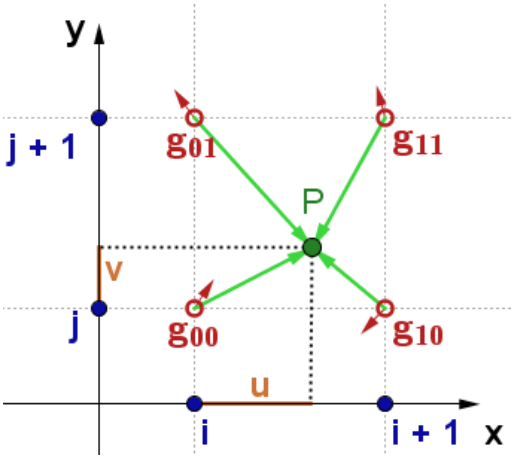
$$g_5 = (1,0,1),\ g_6 = (1,0,-1),\ g_7 = (-1,0,1),\ g_8 = (-1,0,-1),$$

$$g_9 = (0,1,1),\ g_{10} = (0,1,-1),\ g_{11} = (0,-1,1),\ g_{12} = (0,-1,-1).$$

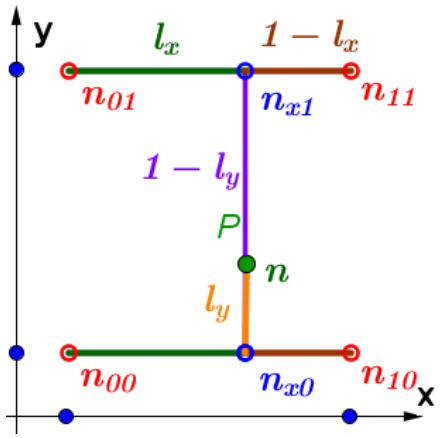


Joonis 13. Gradientide üks võimalik jaotus kolmemõõtmelises ruumis.

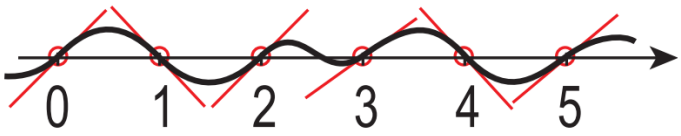
Soovituslik on valida gradiendid, mille koordinaadid on arvud 1, 0 või -1, sest algoritmi hilisemal sammul gradiente sisaldav skalaarkorrutis on võimalik lihtsustada kujule, mis ei nõua korrutamist ja jagamist, ning koosneb ainult liitmisest ja lahutamisest [16]. Analoogiliselt saab võimalike gradientide hulka laiendada suuremate dimensioonide peale, kus saab võtta ruumi mõõtmele vastava hüperkuubi, paigutada see koordi-



Joonis 14. Perlini müra väärtuse leidmiseks kahemõõtmelise ruumi punktis P vajalikud suurused.



Joonis 15. Perlini müra väärtuse leidmiseks vajalikud suurused tasandil.



Joonis 16. Ühemõõtmelise Perlini müra korral konstrueeritav funktsioon [16].

naatide nullpunkti ning valida võimalikeks gradientideks hüperkuubi servade keskpunktide kaudu saadud vektorite hulga.

Kahes dimensioonis soovitab Stefan Gustavson valida 8 või 16 gradienti, mis on ühtlaselt jaotatud ühikringi peal, sest analoogiliselt kuubi servade keskpunktide gradientide jaotusele oleks kahemõõtmelises ruumis gradiente ainult 4 [16].

Kui võimalike gradientide hulk vaadeldavas mõõtmes on defineeritud, võib sarnaselt väärtusmüra funktsioonile f_{tipp} konstrueerida funktsiooni g_{tipp}, mis kujutab võrestiku koordinaadid täisarvuks ning valib saadud arvu jäägiklassi põhjal gradientide massiivi vastava elemendi.

Ühemõõtmelises ruumis on igale tipule määratavad gradiendid ühekohalised vektorid ehk skalaarväärtused, mis kirjeldavad funktsiooni tuletist antud tipus. Seejärel konstrueeritakse funktsioon, mille väärtus igal võrestiku tipul oleks 0 ning tuletis oleks tipule vastav gradient (vt. joonis 16) [16]. Väärtuse leidmiseks suvalises punktis leitakse konstrueeritud funktsiooni väärtus. Analoogiliselt konstrueeritakse ka kõrgemates dimensioonides funktsioon, mille väärtus on 0 igal võrestiku tipul ning tuletis tipu kohal on võrdne tipu gradiendiga.

Järgnevalt vaatleme ühe kihi Perlini müra väärtuse leidmist kahedimensioonilise ruumi punktis $P(x, y)$ Stefan Gustavsoni näite baasil [16]. Kõigepealt leitakse punktile P neli lähimat võrestiku tippu. Selleks ümardatatakse P koordinaadid alla, et leida üks võrestiku tipp t_{00} = (i, j) = ($\lfloor x \rfloor$, $\lfloor y \rfloor$). Seejärel leitakse leitud tipu põhjal ka ülejäänud 3 tippu t_{10}, t_{01} ja t_{11} ning funktsiooni g_{tipp} kasutades neile vastavad gradiendid (vt. joonis 14):

$$g_{00} = g_{tipp}(t_{00}) = g_{tipp}((i,j)), \qquad g_{01} = g_{tipp}(t_{01}) = g_{tipp}((i,j+1)),$$

$$g_{10} = g_{tipp}(t_{10}) = g_{tipp}((i+1,j)), \qquad g_{11} = g_{tipp}(t_{11}) = g_{tipp}((i+1,j+1)).$$

Järgmisena leitakse iga tipu t_{00}, t_{10}, t_{01} ja t_{11} jaoks vektor sellest tipust punkti P (vt. joonis 14 rohelisi vektoreid). Seejärel arvutatakse skalaarväärtused n_{00}, n_{10}, n_{01} ja n_{11}, leides iga tipu gradiendi ning tipust punkti P tõmmatud vektorite skalaarkorrutise [16]:

$$u = x - i, \qquad v = y - j$$

$$n_{00} = g_{00} \cdot \begin{bmatrix} u \\ v \end{bmatrix}, \qquad n_{10} = g_{10} \cdot \begin{bmatrix} u-1 \\ v \end{bmatrix},$$

$$n_{01} = g_{01} \cdot \begin{bmatrix} u \\ v-1 \end{bmatrix}, \qquad n_{11} = g_{11} \cdot \begin{bmatrix} u-1 \\ v-1 \end{bmatrix}.$$

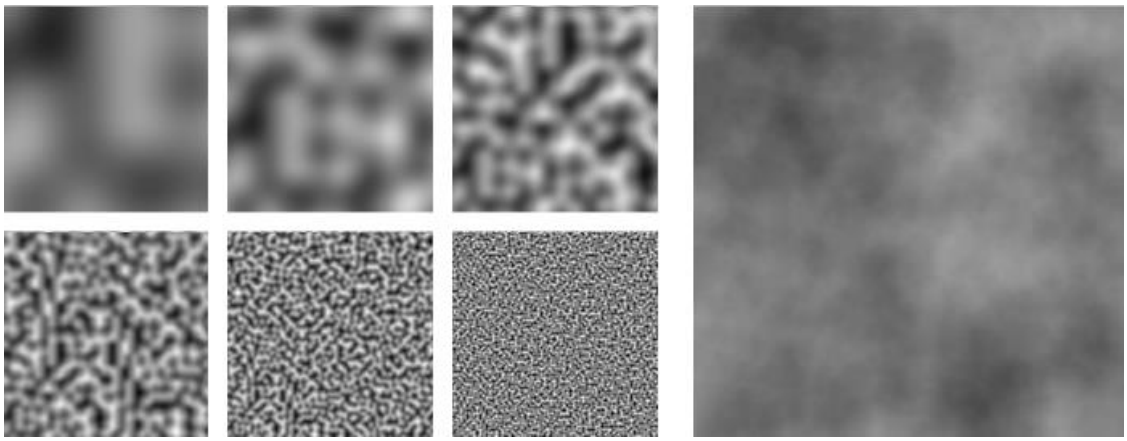
Pärast nelja skalaarväärtuse leidmist interpoleeritakse saadud väärtused üheks väärtuseks, arvestades iga väärtuse juures talle vastava võrestiku punkti kaugust punktist P. Selleks kasutatakse jällegi otspunktides teist järku nulltuletisega segunemisfunktsiooni. Kõigepealt leitakse analoogiliselt väärtusmürale tippude t_{00} ja t_{01} kauguse kaal l_x x-teljel punktist P (vt. joonis 15):

$$l_x = \frac{(x - x_{t_{00}})}{(x_{t_{10}} - x_{t_{00}})} \in [0,1].$$

Seejärel kasutatakse saadud kaalu segunemisfunktsioonis, et leida kaks punktiga P x-teljel paralleelselt paiknevat väärtust

$$n_{x0} = n_{00} * \left(1 - f_{seg}(l_x)\right) + n_{10} * f_{seg}(l_x),$$

$$n_{x1} = n_{01} * \left(1 - f_{seg}(l_x)\right) + n_{11} * f_{seg}(l_x).$$



Joonis 17. Kuus üksikkihti Perlini müra ning nendest kombineeritud müra algsagedusega 1/64, lakunaarsusteguriga 2 ja püsivusteguriga 0.5.

Järgmisena leitakse kauguse kaal l_y, mis kirjeldab n_{x0} kaugust punktist P y-teljel:

$$l_y = \frac{(y - x_{t_{00}})}{(y_{t_{01}} - x_{t_{00}})},$$

ning seejärel otsitav väärtus punktis P:

$$n = n_{x0} * \left(1 - f_{seg}(l_y)\right) + n_{x1} * f_{seg}(l_y).$$

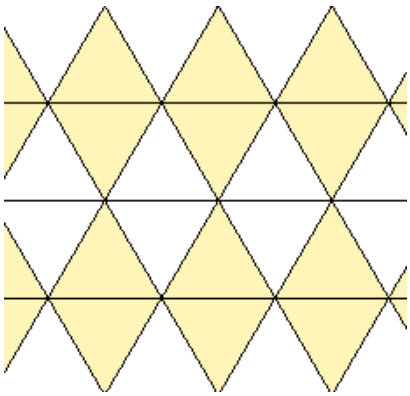
Analoogiliselt on võimalik algoritmi laiendada suvalise n-mõõtmelise ruumi juhule, kus sarnaselt väärtusmürale laienedes suuremasse dimensiooni, lisatakse algoritmi juurde interpoleerimised lisanduval teljel [16].

Nii nagu väärtusmüra korral, ei ole üks kiht Perlini müra eriti huvitava väljanägemisega, ning seetõttu kombineeritakse lõpptulemusse mitu kihti erinevate parameetritega, kus-juures kasutatavad parameetrid on samad, mis väärtusmüral (vt. joonis 17).

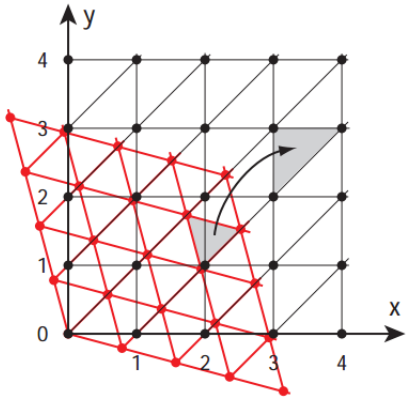
5.2.2 Simpleksmüra

Simpleksmüra on 2001. aastal Ken Perlini poolt välja pakutud Perlini müra edasiarendus. Stefan Gustavson toob välja järgnevad simpleksmüra eelised Perlini müra ees:

- simpleksmüral on väiksem arvutuslik keerukus;
- kui Perlini müra keerukus N dimensiooni korral on $O(2^N)$, siis simpleksmüra keerukus on $O(N^2)$;
- simpleksmüral pole märgatavaid tehislikke jooni;
- simpleksmüra on lihtne riistvara tasemel implementeerida [16].



Joonis 18. Üks võimalus tasandi jaotamine võrdkülgseteks kolmnurkadest. Kõrvuti paiknevad kolmnurgad moodustavad rombid.



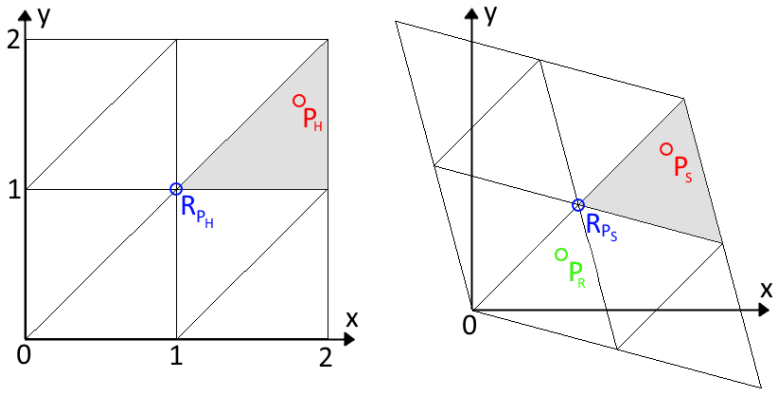
Joonis 19. Kahemõõtmelist võrdkülgsete kolmnurkadega kaetud ruumi on võimalik teisendada ühikruutudega kaetud ruumiks [16].

Kuigi antud töös kirjeldatav algoritm piirdub ainult kahemõõtmelise simpleksmüraga, tasub teada, et erinevalt eelnevatest müradest on simpleksmüra rakendamine teatud viisidel alates kolmandast dimensioonist patenteeritud [22].

Simpleksmüra töötab kohati sarnaselt Perlini mürale, aga jaotab ruumi teistmoodi tükkideks. Simpleksmüra algoritmis jaotatakse n-dimensiooniline ruum tükkideks selles dimensioonis kõige väiksema tippude arvuga korrapäraste kujundite, millega saab ruumi täita, ehk korrapäraste simpleksite abil [16]. Simpleks on n-mõõtmelise vektorruumi niisuguse $(n + 1)$-punktilise hulga ehk simpleksi tippude hulga kumer kate, mille kõik tipud ei paikne mingis $(n - 1)$-mõõtmelises alamruumis [20]. Ühemõõtmelises dimensioonis on simpleksiks lõik. Kahemõõtmelises dimensioonis on korrapäraseks simpleksiks võrdkülgne kolmnurk (vt. joonis 18) ning kolmemõõtmelises ruumis on simpleksiks korrapärane tetraeeder ehk nelitahukas, mille külgedeks on võrdkülgsed kolmnurgad.

Jooniselt 18 on näha, et võrdkülgseteks kolmnurkadeks jagatud kahemõõtmelist ruumi võib vaadelda ka kui rombidest ehk n-ö moonutatud ruutudest koosnevat ruumi. Kui sellist ruumi kindlal viisil moonutada, on võimalik rombid teisendada ruumi katvateks ühikruutudeks, kus iga ühikruut vastab kahele algse võrestiku võrdkülgsele kolmnurgale (vt. joonis 19).

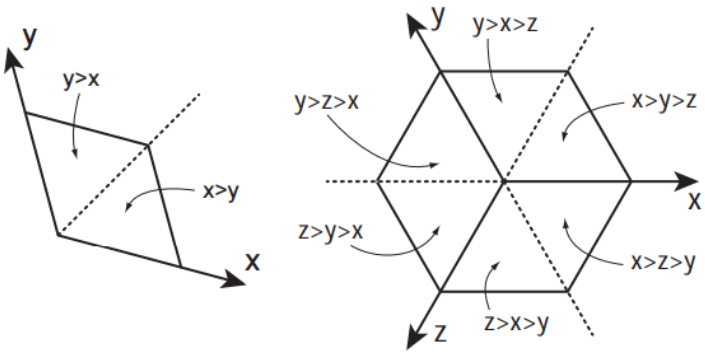
Stefan Gustavson kirjutab simpleksite omaduste kohta järgnevat [16]. Kolmemõõtmelises ruumis on korrapäraseks simpleksiks korrapärane tetraeeder ning need moodustavad kuue kaupa analoogiliselt rombidele moonutatud kuubi, mis on mööda peadiagonaali kokku surutud. Ka sellisel juhul on võimalik ruumi moonutada nii, et iga ühikkuup ruumis vastab



Joonis 20. Otsitavat punkti sisaldava simpleksi leidmiseks vajalikud punktid ruumides H (vasakul) ja S (paremal).

algse võrestiku kuuele tetraeedrile. Üldistades selgub, et iga suvaline n-mõõtmeline ruum on võimalik jagada osadeks $n + 1$ nurgaga simpleksi abil. Need kujundid jaotavad $n!$ kaupa ruumi pärast moonutamist n-dimensioonilisteks hüperkuupideks. Lisaks on võimalik vastupidiselt hüperkuupidega jagatud ruum moonutada üheselt tagasi simpleksitega jagatud ruumiks, kus igale hüperkuubile vastab $n!$ simpleksit. Täpsemad ruumi moonutamise detailid on leitavad Jasper Flicki simpleksmüra juhendis ning siinkohal nende detailidesse ei laskuta, sest need ei ole simpleksmüra põhimõtetest arusaamiseks niivõrd olulised [9]. Defineerime siinkohal vaid n-kohalised funktsioonid, mille abil on võimalik teisendada koordinaate simpleksite ruumist hüperkuupide ruumi ja vastupidi: $f_{S\to H}\colon \mathbb{R}^n \to \mathbb{R}^n$ ja $f_{H\to S}\colon \mathbb{R}^n \to \mathbb{R}^n$.

Järgnevalt uurime simpleksmüra väärtuse leidmist punktis $P_S = (x_S, y_S)$ Stefan Gustavsoni näite baasil [16]. Punkt P_S paikneb simpleksitega jagatud ruumis S. Antud juhul on selleks ruumiks tasand, mis on kaetud võrdkülgsete kolmnurkadega. Esimeseks eesmärgiks on teisendada P_S koordinaadid moonutatud, kahemõõtmelistest hüperkuupidest ehk ruutudest koosneva ruumi H koordinaatideks $P_H = (x_H, y_H) = f_{S\to H}(x_S, y_S)$. Seejärel saab P_H koordinaate sarnaselt väärtusmürale lähima täisarvuni alla ümardades leida, mitmenda ruudu sees otsitav punkt mööda iga telge asub: $R_{P_H} = (\lfloor x_H \rfloor, \lfloor y_H \rfloor)$. Kuna iga moonutatud ruumi ruut vastab kahest kolmnurgast koosnevale rombile ruumis S, saame leitud koordinaadid tagasi ruumi S teisendades punkti P_S sisaldava rombi tasandi nullpunktile lähima tipu $R_{P_S} = f_{H\to S}(R_{P_H})$ (vt. joonis 20). Järgmise sammuna on vaja määrata, kumma rombi moodustava kolmnurga sees asub punkt P_S. Selleks saame teisendada P_S koordinaadid tasandil ümber nii, et leitud rombi tipp oleks uue punkti jaoks

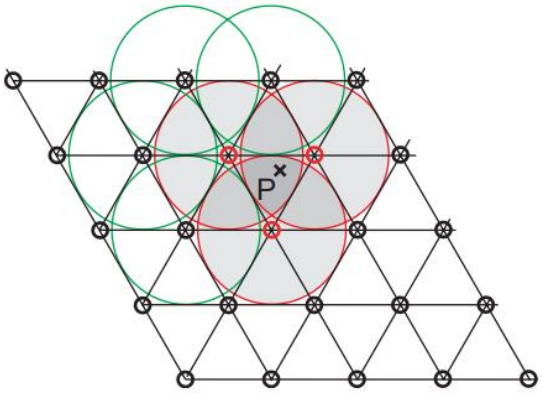


Joonis 21. Punkti sisaldava simpleksi valimine $n!$ võimaliku simpleksi seast kasutades punkti P_R koordinaatide võrdlemist 2D ja 3D ruumis [16].

nullpunkt: $P_R = P_S - R_{P_S}$. Selle tulemusena saadud koordinaadid ei ole mõjutatud rombi paiknemisest tasandi nullpunkti suhtes, vaid ainult rombi lokaalse tasandi suhtes. Nüüd on x_{P_R} ja y_{P_R} koordinaate võrreldes võimalik otsustada, kas punkt P_S paikneb alumise või ülemise kolmnurga sees (vt. joonis 21). Analoogiliselt on võimalik ka kolmemõõtmelises ruumis määrata, millise simpleksi sisse punkt kuulub. Täpsemalt saadakse sellisel juhul kolmekohaline punkt P_R ning igale ruumi H kuubile vastab ruumis S kuuest tetraeedrist koosnev nelitahukas. Seejärel saab punkti koordinaatide x_{P_R}, y_{P_R} ja z_{P_R} põhjal otsustada, millisesse kuuest simpleksist otsitav punkt kuulub (vt. joonis 21). Lisaks on võimalik algoritmi laiendada suurematele n-mõõtmelistele dimensioonidele, kus kontrollitakse leitud P_R koordinaatide suhet, et määrata, millisesse simpleksisse $(n!)$-st võimalikust simpleksist punkt P_S kuulub.

Pärast punktile P_S vastava simpleksi ning selle nullpunktile lähima tipu $S_0 = R_{P_S}$ leidmist, saame leida simpleksi ülejäänud tipud. Ülejäänud tipud on määratavad kasutades simpleksi tippu S_0 eelmisel sammul leitud koordinaatide suhteid. Algoritmi implementeerides on mõttekas iga erineva mõõtmega müra jaoks genereerida sõnastik, kus võtmeks on ühele koordinaatide järjestussuhtele vastav arv (näiteks $y > x > z$ korral 213) ning väärtuseks n n-kohalist vektorit, mis punkti S_0 koordinaatidele liidetuna annavad koos S_0 endaga vastava simpleksi $n + 1$ tippu $S_i, i \in \{0, \dots, n\}$. Sõnastiku täpsed väärtused on võimalik ise arvutada või kasutada olemasolevaid sõnastikke ning siinkohal nende genereerimist ei käsitleta [16,18].

Pärast punkti P_S sisaldava simpleksi kõigi tippude leidmist, saab Perlini mürast tuttavat funktsiooni g_{tipp} kasutades leida igale simpleksi tipule vastava gradiendi g_i. Erinevalt



Joonis 22. Simpleksite tippudele rakendatavate kaalufunktsioonide mõjualad võrestikul [16].

väärtusmürast ning Perlini mürast ei ole seekord võimalik väärtust otsitavas punktis teadaolevaid väärtusi mööda telgi interpoleerides leida, sest simpleksite tipud ei paikne enam regulaarsel ruudustikul.

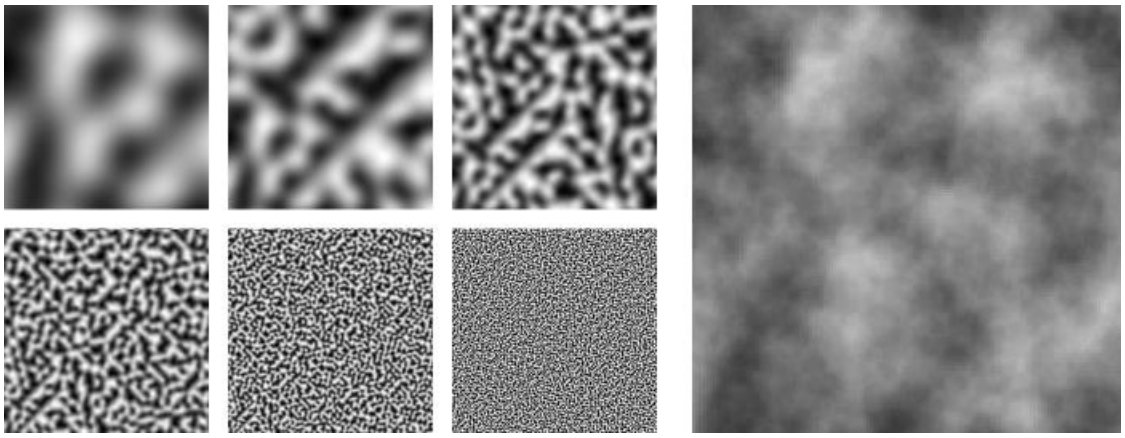
Järgnevalt uurime simpleksväärtuse müra leidmist punktis P Jasper Flicki implementatsiooni näitel [9]. Esmalt leitakse igast simpleksi tipust S_i punkti P vektor v_i ning leitud vektori v_i ja tipule vastava gradiendi g_i skalaarkorrutis s_i. Järgmise sammuna arvutatakse iga simpleksi tipu S_i jaoks selle kaugus l_i punktist P. Selle tulemusena on leitud igale simpleksi tipule vastav skalaarväärtus s_i ning selle kaugus l_i otsitavast punktist. Järgmisena valitakse dimensioonile sobiv mittekasvav kaalufunktsioon $f_k: [0, \infty) \rightarrow [0, 1]$, mille korral $f_k(0) = 1$. Seejärel leitakse simpleksmüra väärtus punktis P kasutades valemit:

$$f_{simp}(P) = \sum_{i=0}^{n} s_i \, f_k\left(\frac{l_i}{h}\right).$$

Kaalufunktsioon konstrueeritakse nii, et kui simpleksi tipu kaugus punktist P on suurem kui simpleksi kõrgus h, on funktsiooni kujutiseks 0. Näiteks kahemõõtmelises ruumis on simpleksi kõrgus võrdkülgse kolmnurga kõrgus, kolmemõõtmelises ruumis regulaarse tetraeedri kõrgus. See tähendab, et iga punkti väärtus sõltub ainult seda ümbritseva simpleksi tippudest, sest ülejäänud tippude korral oleks kaalufunktsiooni kujutiseks 0 (vt. joonis 22). Lõigus $[0, 1]$ võib kaalufunktsioonina kasutada sarnaselt eelnevatele segunemisfunktsioonidele polünoomi, mille korral $f_k(0) = 1$, $f_k(1) = 0$ ja $f_k'(0) = f_k''(0) = f_k'(1) = f_k''(1) = 0$. Selleks polünoomiks sobib näiteks $(1 - x^2)^3$ [9]. Seega oleks lõplik kaalufunktsioon järgnev:

$$f_k(x) = \begin{cases} (1 - x^2)^3, & \text{kui } x < 1, \\ 0, & kui\ x \geq 1. \end{cases}$$

Tasub märkida, et hetkel f_{simp} tagastatavad väärtused ei asu lõigus [-1, 1], vaid on kaalufunktsioonist sõltuvate piiridega. Seetõttu peab välja arvutama f_{simp} maksimaalse väärtuse ning tulemust sellega jagama, et saada sarnaselt teistele müradele alati väärtused



Joonis 23. Kuus üksikkihti simpleksmüra ning nendest kombineeritud müra algsagedusega 1/64, lakunaarsusteguriga 2 ja püsivusteguriga 0.5.

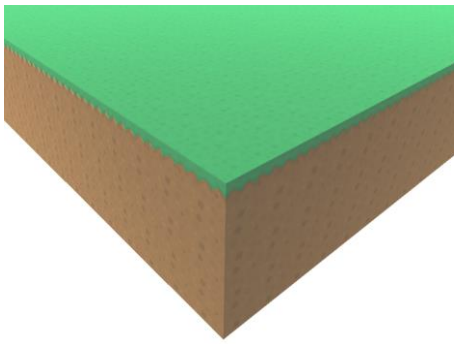
lõigus [-1, 1]. Kahemõõtmelise simpleksmüra korral on maksimaalseks väärtuseks $\frac{2916*\sqrt{2}}{125} \approx 33$ ning kolmemõõtmelise simpleksmüra korral $\frac{8192*\sqrt{2}}{375} \approx 30.9$ [9]. Antud arvude leidmine on detailselt kirjeldatud Jasper Flicki juhendis ning siinkohal nende leidmist ei käsitleta [9].

Sarnaselt väärtus- ja Perlini mürale rakendatakse simpleksmüra tavaliselt mitme kihina ning kasutatakse juba varasemalt kirjeldatud parameetreid (vt. joonis 23).

Simpleksmüra eeliseks väärtusmüra ees on see, et simpleksmüra kasutamisel ei teki märgatavaid väärtusmüra võrestiku paiknevusest tulenevaid ruudukujulisi piirkondi ning seetõttu on tulemus kaootilisema väljanägemisega. Kuna antud töös kasutatakse mürasid ainult kahe ja kolmemõõtmelisena, ei ole ajaliselt Perlini ja simpleksmüra väärtuste leidmiseks kuluval ajal suurt vahet. Simpleksmüra on siiski Perlini mürast eelistatum, kuna on tagatud, et väärtuste visualiseerimisel ei teki tehislikke jooni, mis Perlini müras mõnikord esinevad.

6. Lihtsa maastiku genereerimine

Eelmises peatükis kirjeldatud müraalgoritmid on käesolevas ja järgnevates peatükkides käsitletava maastiku genereerimise algoritmi aluseks. Kuna simpleksmüra on väärtusmürast sujuvam ning sellelt ei paista erinevalt Perlini mürast kunagi võrestik välja, kasutatakse järgnevalt müra kasutatavatel algoritmi sammudel just simpleksmüra. Kuna maastiku genereerimise algoritm peab olema deterministlik ning suuteline genereerima rohkem kui ühte maastikku, on igal maastikul seda määrav täisarvuline seeme. Seda seemet kasutatakse maastiku genereerimise algoritmi kõigi pseudojuhuslike arvugeneraatorite, sealhulgas ka müraalgoritmide, seemnena. Seemne kasutamine tagab, et juhul kui algoritmi teisi parameetreid ei muudeta, on ühe seemnega alati võimalik genereerida täpselt samasugune maastik.



Joonis 24. Tasane maastik lävendiga $h = 3$.

Esiteks vaatleme aga kõige lihtsama maastiku genereerimist. Selleks defineeritakse kolmekohaline funktsioon f_M, mis määrab iga maastikutüki iga punkti globaalsete koordinaatide põhjal sellele maastikutüübi M:

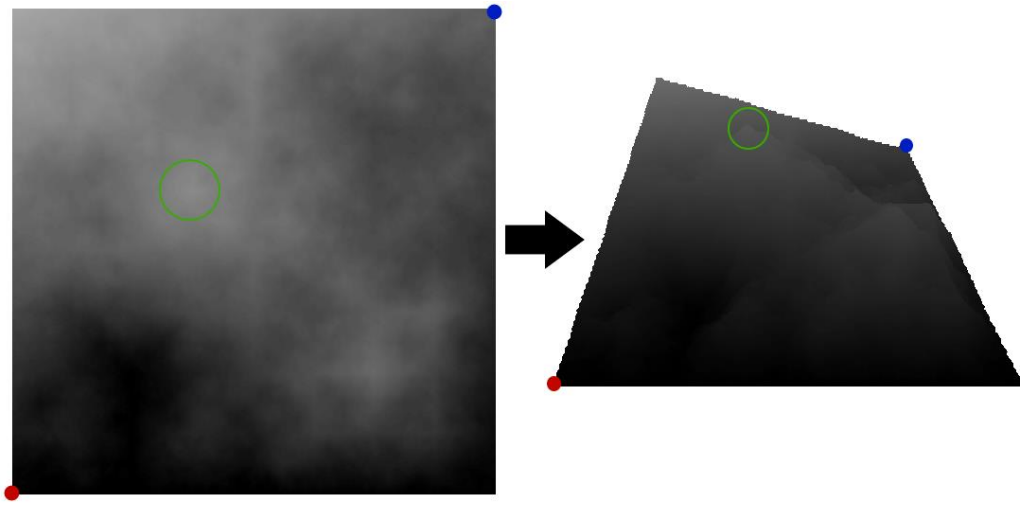
$$f_M\colon \mathbb{Z}^3 \to M.$$

Lihtsuse huvides olgu maastikutüüpe algul kolm: $M = \{\text{Õhk}, \text{Muru}, \text{Muld}\}$. Kõige lihtsam on genereerida tasast maastikku, mille korral punkti kõrgus- ehk z-koordinaadi ja lävendi h põhjal otsustatakse, missugune maastikutüüp antud punktis asub (vt. joonis 24):

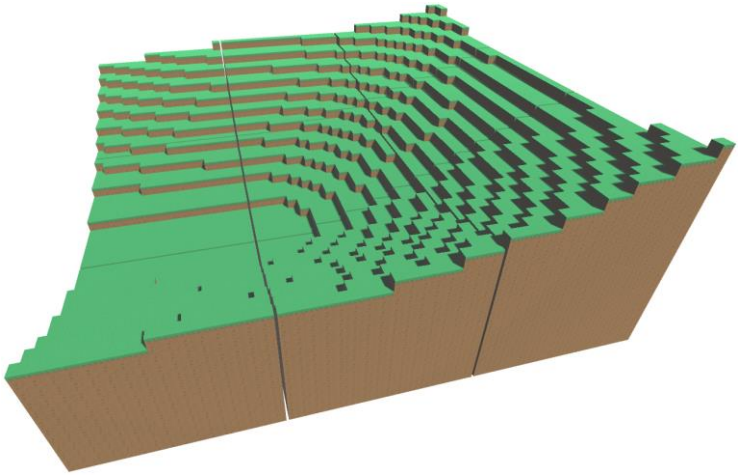
$$f_M(x_{gv}, y_{gv}, z_{gv}) = \begin{cases} \text{Muru}, & \text{kui } z_{gv} = h \\ \text{Muld}, & \text{kui } z_{gv} < h \\ \text{Õhk}, & \text{kui } z_{gv} > h. \end{cases}$$

Selle tulemusena saadav maastik on üpris igava väljanägemisega ning mitte eriti realistlik. Selle parandamiseks tuleb funktsiooni f_M täiendada nii, et igal pool ei oleks samasugune tasane maastik. Sobiva tulemuse saavutamiseks sobib hästi kõrguskaardi (ing. k. *heightmap*) kasutamine. Kõrguskaart on maatriks, mida tavaliselt kujutatakse halltoonides pildina. Iga maatriksi element määrab maastiku kõrguse maatriksi reale ja veerule vastavas maastikutulbas (vt. joonis 25). Kõrguskaardi kasutamine on üldlevinud viis maastiku

kujutamiseks ning seda kasutavad väga paljud arvutimängud nagu näiteks Battlefield 2[8], Cities: Skylines[9] jpt.



Joonis 25. Kõrguskaart ja sellega genereeritud maastik. Mõned regioonid kõrguskaardil on märgitud, et neid maastikult paremini leida.



Joonis 26. 3×3 eraldatud maastikutükki, mis on genereeritud, kasutades kahe-mõõtmelist simpleksmüra kõrguskaardina. $h_{max} = 20$. Simpleksmüra jaoks kasutati kolme kihti algsagedusega 1/64, lakunaarsusteguriga 2 ning püsivusteguriga 0.5.

Lõpmatu maastiku jaoks on vaja lõpmatut kõrguskaarti, et igale maastikutulbale vastaks üks-üheselt üks kõrguskaardi väärtus. Seetõttu sobivad kõrguskaarti genereerimiseks hästi müraalgoritmid, sest müraga on võimalik leida iga maastikutulba globaalseid koordinaate kasutades müra väärtus, mida vastava tulba kõrgusena kasutada. Lisaks ei pea muretsema,

[8] http://www.battlefield.com/battlefield-2/

[9] http://www.citiesskylines.com/

et kõrvuti sõltumatult genereeritud maastikutükid kokku ei sobiks, sest müraalgoritmide sujuvus tagab, et maastikutükkide piirides arvutavad väärtused omavahel kokku sobituks. Kuna simpleksmüra tagastab väärtuse lõigus $[0, 1]$, on vaja saadavaid müra väärtusi korrutada konstandiga h_{max}, mis määrab maksimaalse maastikutulba kõrguse. Seega uueks maastiku genereerimise funktsiooniks sobib funktsioon

$$f_M(x_{gp}, y_{gp}, z_{gp}) =$$

$$= \begin{cases} \text{Muru}, & \text{kui } z_{gp} \leq f_{simp}(x_{gp}, y_{gp}) * h_{max} \wedge f_M(x_{gp}, y_{gp}, z_{gp} + 1) = \text{Õhk}, \\ \text{Muld}, & \text{kui } z_{gp} \leq f_{simp}(x_{gp}, y_{gp}) * h_{max} \wedge f_M(x_{gp}, y_{gp}, z_{gp} + 1) \neq \text{Õhk}, \\ \text{Õhk}, & \text{kui } z_{gp} > f_{simp}(x_{gp}, y_{gp}) * h_{max}. \end{cases}$$

Katsetamisel selgus, et piisavalt hea tulemuse annab simpleksmüra vähemalt kolme kihi ning piisavalt väikese sagedusega, et künkad oleksid jaotunud piisavalt suurte vahemaade peale (vt. joonis 26).

7. Bioomid

Maastikule mitmekesisuse andmiseks võiks eelmises peatükis genereeritud maastikule anda rohkem eripära. Üks võimalus selleks on modifitseerida maastiku genereerimise algoritmi nii, et analoogiliselt pärismaailmaga on erinevates piirkondades erinev pinnasekate ning maastiku kuju. Järgnevalt uurime, kuidas selliseid muudatusi makroökosüsteemide ehk bioomide tasemel teha.

Bioomid (ing k. *biomes*) on makroökosüsteemid, mis on geograafiliselt piiritletud alad sealse kliima ja domineeriva taimkatte järgi [23]. California Ülikooli Paleontoloogiamuuseum (UCMP) jagab bioomid kuueks suuremaks grupiks: magevee-, soolvee-, kõrbe-, metsa-, rohumaa- ja tundrabioomideks [24].

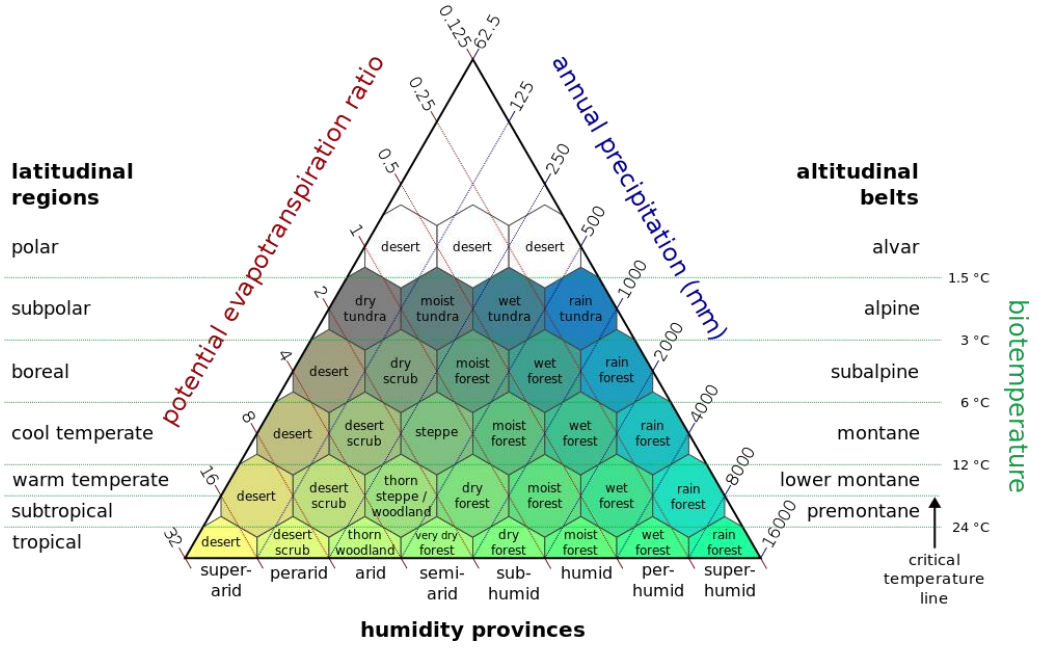
UCMP jaotab need kuus suuremat gruppi veel omakorda väiksemateks:

- mageveebioomide floora ja fauna on kohastunud madalasoolsusega keskkonnaga ning selle alla kuuluvad jõed, järved, tiigid ning muud märgalad;
- soolveebioomid on kõige suuremad ning katavad ligi kolm neljandikku Maa pinnast ja selle alla kuuluvad ookeanid, korallrahud ning jõesuudmed;
- kõrbebioomid on ökosüsteemid, kus sademete hulk jääb alla 50 cm aastas ning need jaotatakse neljaks: kuumad ja kuivad kõrbed, poolkuivad kõrbed, rannikukõrbed ning külmad kõrbed;
- metsabioomid on puukattega bioomid, mis jaotatakse kolmeks: troopiline mets, parasvöötme metsad ning boreaalsed metsad ehk taigad;
- rohumaabioomid on bioomid, kus domineerivaks flooraks puude ja suurte põõsaste asemel on rohttaimed. Need jaotatakse savannideks ning parasvöötme rohumaadeks, mis omakorda jagatakse preeriateks ning steppideks;
- tundrabioomid on kõige külmemad bioomid ning sarnaselt kõrbetele liiga raskete eluoludega enamuste liikide jaoks. Tundrabioomid jaotatakse arktilisteks ja mägisteks.

Järgnevalt tutvume kahe võimalusega, kuidas on võimalik mõned neist lisada maastiku genereerimise algoritmi.

7.1 Bioomide parameetrid

Bioomide määramiseks maastikule on vaja eelnevalt määrata teatavad arvutuslikud parameetreid. 1947. aastal avaldas botaanik ja kliimateadlane Leslie Holdridge artikli, kus ta kirjeldas bioomide klassifitseerimist kolme parameetri abil [25]. Nendeks parameetriteks oli aastane sademete hulk, keskmine temperatuur ning vee aurustumise suhe sademetega (vt. joonis 27). Sellest lähtuvalt kasutatakse siinkohal bioomide klassifitseerimiseks sarnast, kuid lihtsustatud lahendust. Täpsemalt kasutatakse nendest kolmest parameetrist kahte – sademete hulka ning temperatuuri, mis on väärtused lõigus $[0, 1]$. Väärtus 0 tähendab sademete korral sademete puudumist ning temperatuuri korral madalat temperatuuri ning väärtus 1 vastavalt rohket sademete hulka ja kõrget temperatuuri.



Joonis 27. Holdridge'i bioomide klassifikatsioon parameetrite põhjal [32].

Lisaks sademete hulgale ning temperatuurile määrati igale töös implementeeritud bioomile järgnevad parameetrid:

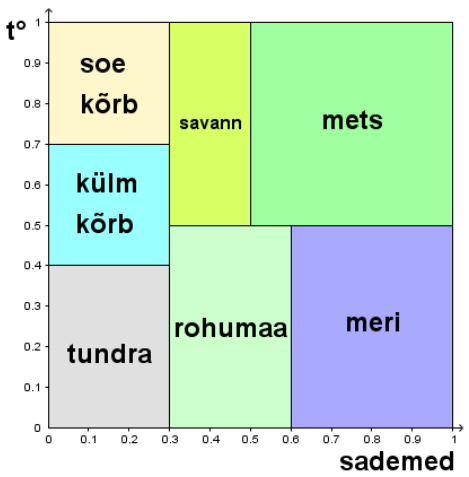
- bioomi nimi;
- pinnasekate;
- bioomile sobivate sademete hulka kirjeldav poollõik $[x_s, y_s)$ või lõik $[x_s, y_s] \subseteq [0, 1]$;
- bioomile sobivat temperatuuri kirjeldav poollõik $[x_t, y_t)$ või lõik $[x_t, y_t] \subseteq [0, 1]$;

- võimendustegur v lõigus $[0, 1]$, millega korrutatakse üldise maastiku genereerimisel kasutatava kõrguskaardi müra väärtust, et muuta mõned bioomid teistest tasasemaks;
- segunemistegur s, mida kasutatakse bioomide piiraladel erinevate bioomide ühendamiseks;
- taimede genereerimise sagedus f_{taimed}.

Töö autor valis eelnevas peatükis kirjeldatud bioomidest välja seitse erikategooriatesse kuuluvat bioomi, mis implementeeriti valminud rakendusse. Väljavalitud bioomideks on soe kõrb, külm kõrb, tundra, savann, parasvöötme rohumaa, parasvöötme mets ning veebioome esindav meri. Need bioomid on lähemalt nähtavad kõigi järgnevate peatükkide illustratsioonides. Järgnevalt on tabelis 1 ära toodud nende bioomide parameetrite väärtused. Bioomide sademete ja temperatuuri väärtuste määramisel on lähtutud

Tabel 1. Implementeeritud bioomide omaduste tabel.

Bioom	**Pinnase-kate**	**Sademed**	**Tempera-tuur**	**Võimendus-tegur**	**Segunemis-tegur**	**Taimede sagedus**
Soe kõrb	liiv	$[0, 0.3)$	$[0.7, 1]$	0.3	10	0.018
Külm kõrb	lumi	$[0, 0.3)$	$[0.4, 0.7)$	0.1	1	0.018
Tundra	pruun muru/muld	$[0, 0.3)$	$[0, 0.4)$	0.8	1	0.000
Savann	kollakas muru	$[0.3, 0.5)$	$[0.5, 1]$	0.4	1	0.050
Rohumaa	roheline muru	$[0.3, 0.6)$	$[0, 0.5)$	0.4	1	0.500
Mets	roheline muru	$[0.5, 1]$	$[0.5, 1]$	0.5	1	0.190
Meri	vesi	$[0.6, 1]$	$[0, 0.5)$	0	1000	0.000



Joonis 28. Valitud bioomide jaotus sademete ja temperatuuri põhjal.

Holdridge'i bioomide klassifikatsioonist. Ülejäänud väärtused on määratud jooksvalt algoritmi implementeerides, et saavutada meeldiv maastik. Nende seitsme bioomi sademed ja temperatuuride võimalikud paarid katavad kogu võimaliku sademete ja temperatuuri väärtuste ala (vt. joonis 28). Seega igale sademete ja temperatuuri väärtuste paarile vastab üks nendest seitsmest bioomist.

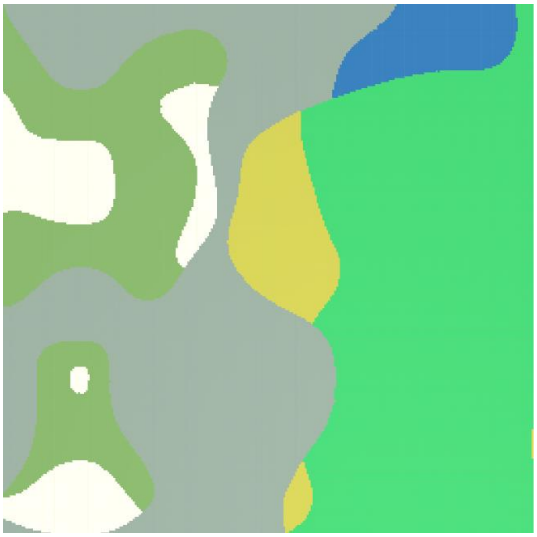
7.2 Bioomide määramine

Pärast võimalike genereeritavate bioomide defineerimist, on vaja leida viis, kuidas genereeritav maastik bioomide vahel ära jagada. Bioome on sobilik määrata maastikutulpade tasemel ehk algoritm võiks määrata igale maastikutulbale ühe seitsmest bioomist. Maastikutulpade tasemel genereerimine on mõttekam, kui iga punkti jaoks omaette väärtuste genereerimine, sest enamasti kasutajad maastiku sisse niikuinii ei näe (v.a. üksikud kaljud) ja seetõttu ei oma vertikaalselt erinevate bioomide genereerimine eriti mõtet. Samuti võtaks iga punkti jaoks bioomide määramine kümneid kordi kauem aega kui maastikutulpadele bioomide määramine. Seega on vaja konstrueerida funktsioon määramaks igas maastikutulbas paiknevat bioomi:

$$f_{bioom} : \mathbb{Z}^2 \rightarrow B.$$

See funktsioon kujutab iga maastikutulba globaalsed x_{gp} ja y_{gp} koordinaadid üheks eelmise peatükis kirjeldatud bioomide hulga B elemendiks.

On loomulik, et üks bioom ei kata ainult väikest, ühe maastikutulbaga piiratud ala, vaid paljudest maastikutulpadest koosnevat osa maastikust. Seetõttu on bioomide määramisel oluline, et lõpptulemusena tekiksid maastikule suured bioomid, seejuures arvestades, et nende määramine peaks käima ainult maastikutulba koordinaatide kaudu naabermaastikutulpadest ja –tükkidest sõltumata.



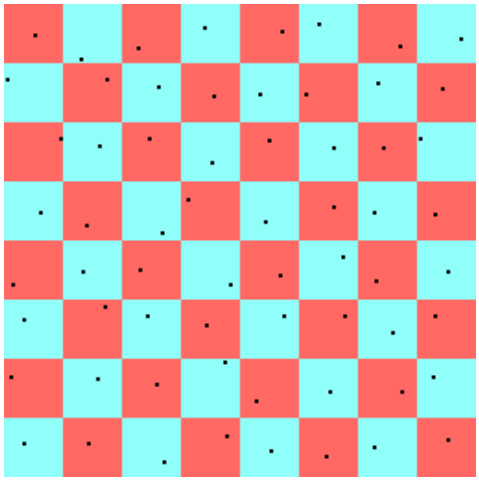
Joonis 29. Bioomid, mis on määratud arvutades iga tulba jaoks kaks erinevat simpleksmüra väärtust, mida on kasutatud temperatuuri ja sademetena.

Joonis 30. Bioomid, mis on saadud, kasutades bioomitükkide koordinaate temperatuuri ja sademete määramiseks.

Järgnevalt pakub autor välja kaks võimalust, kuidas maastikku bioomide vahel jagada. Esimene neist on üldlevinud lihtne lahendus, mille põhjal autor töötas välja sobivad täiendused, et algse lahenduse kvaliteeti parandada.

Üldlevinud lahendus maastiku bioomideks jagamiseks on arvutada iga maastikutulba koordinaatide kaudu kaks erinevat simpleksmüra väärtust, mida kasutatakse temperatuuri ja sademete hulga väärtusena. Erinevad väärtused saab genereerida nihutades teise väärtuse arvutamiseks simpleksmüra argumente suure konstandi võrra. Nihe tagab, et saadud kaks väärtust on omavahel sõltumatud, kuid siiski arvutatavad kasutades sama seemnega simpleksmüra. Seejärel kontrollitakse bioomide tabelist, millisele bioomile leitud sademete hulk ja temperatuur vastab ning antud tulbale määratakse vastav bioom. Selle tulemusena tekivad bioomid, millel on küll üldjoontes sujuvad piirid, kuid kuna temperatuuri ja sadmete müra on omavahel sõltumatud, tekivad osadesse kohtadesse järsud ristsuunalised üleminekud (vt. joonis 29, nt. meri ja muru). Seetõttu näevad bioomide kujud liialt tehislikud välja ja sel põhjusel ei ole maastikutulba koordinaatide kaudu sademete ja temperatuuri määramine soovituslik.

Teine, autori arvates paremaid tulemusi andev võimalus bioomide määramiseks on kõigepealt jagada maastik sarnaselt maastikutükkidele bioomitükkideks. Üks bioomitükk on ülevalt vaadates ruudukujuline tükk, mis sarnaselt maastikutükkidele moodustab xy-tasandil ruudustiku, mis katab kogu tasandi. On oluline märkida, et bioomitüki laius l_{bioom}



Joonis 31. Osa bioomitükkide tasandist, koos bioomitükkidele märgitud keskpunktidega.

ei pea maastikutükkide laiuse ega pikkusega ühtima, kuid peab olema täisarvuline. Bioomitüki ja maastikutüki laiuse sõltumatus võimaldab hiljem bioomitüki laiust vastavalt vajadusele muuta, ilma et oleks vaja maastikutükkide mõõtmeid samuti muuta. Analoogiliselt maastikutükkidele on ka bioomitükkidel olemas koordinaadid. Neid koordinaate võib kasutada simpleksmüra argumentidena, et arvutada igale bioomitükile vastav sademete hulk ning temperatuur. Selle tulemusena tekiksid kandilised bioomid (vt. joonis 30).

Järgnevalt uurime, kuidas hetkel veel kandilisi bioomidevahelisi piire vähem korrapärasteks muuta. Selle saavutamiseks saab igale bioomitükile määrata bioomitükisisese nn keskpunkti lokaalsete koordinaatidega (vt. joonis 31):

$$(x_{lbk}, y_{lbk}), \qquad x_{lbk}, y_{lbk} \in \{\, 0, 1, \dots, l_{bioom} - 1 \,\}.$$

Siinjuures keskpunkt ei tähenda, et punkt paikneb bioomitüki keskel, vaid et see on järgnevalt kirjeldatava algoritmi töö tulemusena tekkiva bioomirakku keskpunktiks (vt. joonis 33) ning võib paikneda suvalisel kohal bioomitüki sees.

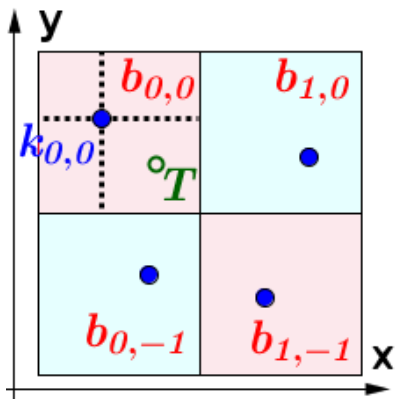
Igale keskpunktile saab määrata ühe bioomi, kui genereerida analoogiliselt eelnevale kaks simpleksmüra väärtust, kasutades keskpunkti koordinaate müra argumentidena. Seejärel saab leitud väärtusi kasutada sademete ja temperatuurina, et valida bioomide tabelist sobiv bioom.

Algoritmi determineeritusest tingitult peab keskpunkti määramine olema deterministlik. Selle tagamiseks sobib sarnaselt müraalgoritmidele enne bioomitükkide keskpunktide arvutamist leida seemet ja Fisher-Yates'i algoritmi kasutades permutatsioon arvudest $0, 1, \dots, l_{bioom} - 1$ ning hoiustada see massiivi. Seejärel saab analoogiliselt väärtusmüra funktsioonile f_{tipp} bioomitüki koordinaadid teisendada üheks reaalarvuks ning võtta sellele vastava indeksiga massiivi elemendi.

Pärast maailma jaotamist bioomitükkideks on võimalik arvutuslikult leida igale maastikutulbale vastav bioomitükk jagades maastikutulba koordinaadid bioomitüki pikkusega ning ümardades tulemuse allapoole lähima täisarvuni:

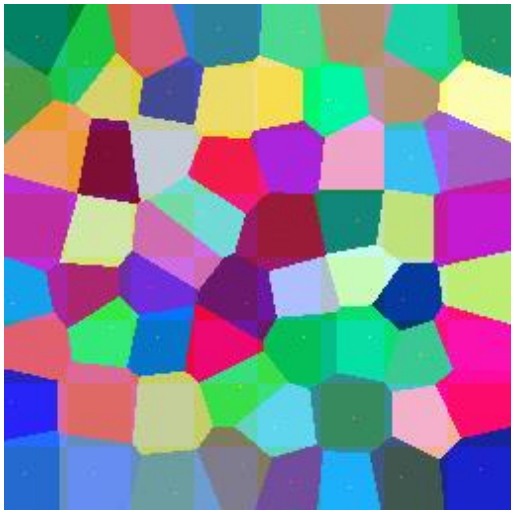
$$f_{bioomitükk}(x_{gp}, y_{gp}) = \left(\left\lfloor \frac{x_{gp}}{l_{bioom}} \right\rfloor, \left\lfloor \frac{y_{gp}}{l_{bioom}} \right\rfloor\right).$$

Järgnevalt kirjeldatakse algoritmi, mis leiab igale maastikutulbale lähima bioomitüki keskpunkti. Seejärel määratakse igale maastikutulbale lähimale keskpunktile vastav bioom. Määratud bioom ei pruugi ühtida maastikutulbale vastava bioomitüki bioomiga. Keskpunktide pseudojuhuslik paiknevus bioomitükkide sees tagab, et tekkivad bioomidevahelised piirid ei paikne enam alati x- ja y-teljega paralleelselt (vt. joonis 33).

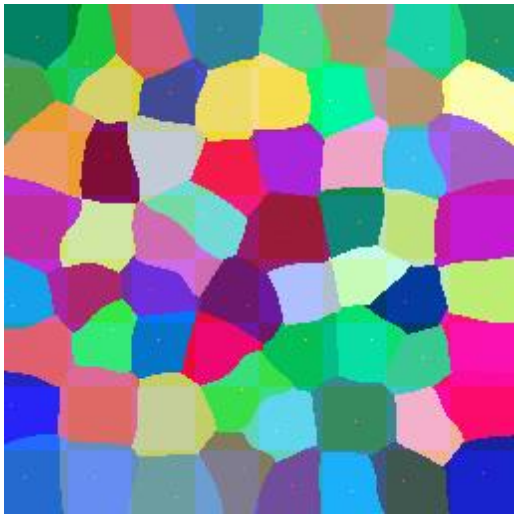


Joonis 32. Vaadeldava maastikutulba T (roheline) ja seda sisaldava bioomitüki $b_{0,0}$ keskpunkti $k_{0,0}$ põhjal määratud neli lähima keskpunktiga bioomitüki kandidaati.

Lähima keskpunkti leidmiseks on vaja esmalt leida vaadeldavat maastikutulpa T sisaldava bioomitüki $b_{0,0}$ keskpunkt $k_{0,0}$. Selgub, et maastikutulba T ja keskpunkti $k_{0,0}$ omavahelist paiknevust võrreldes, on võimalik valida veel 3 bioomitükki nii, et üks neljast vaadeldavast bioomitükist kindlasti sisaldab tulbale T lähimat keskpunkti. Kui $k_{0,0}$ asub x-koordinaadi poolest tulbast T vasakul, siis ei saa lähima keskpunktiga bioomitükk kindlasti asuda tükist $b_{0,0}$ vasakul, sest $k_{0,0}$ on kindlasti nende keskpunktidest lähemal. Vastupidiselt aga, kui $k_{0,0}$ asub x-koordinaadi poolest maastikutulbast paremal, ei saa lähima keskpunktiga bioomitükk asuda tükist $b_{0,0}$ paremal. Analoogiliselt y-koordinaatide erinevusi võrreldes jääbki $b_{0,0}$ vahetutest naabrites alles ainult kolm bioomitükki. Näiteks joonisel 32 näidatud $k_{0,0}$ ja T paiknemise korral jäävad lähima keskpunkti kandidaatideks tükid $b_{0,0}, b_{1,0}, b_{0,-1}$ ja $b_{1,-1}$. Pärast maastikutulbale nelja võimaliku lähima keskpunktiga bioomitüki leidmist saab nende keskpunktide kaugusi maastikutulbast T võrreldes määrata, milline keskpunkt ning bioomitükk tegelikult tulbale lähim on. Pärast lähima keskpunkti leidmist määratakse tulba T bioomiks lähimale keskpunktile vastav bioom. Tähistades igale keskpunktile lähimaid tulpi omaette värviga, saame tulemuseks Voronoi diagrammi (vt. joonis 33). Kuna bioomi määramine ei käi enam sellele vastava



Joonis 33. Värvides kõik maastikutulbad neile lähima keskpunkti bioomitükile omistatud suvalise värviga tekib Voronoi diagramm, mille tükke nimetatatakse antud töös bioomirakkudeks.

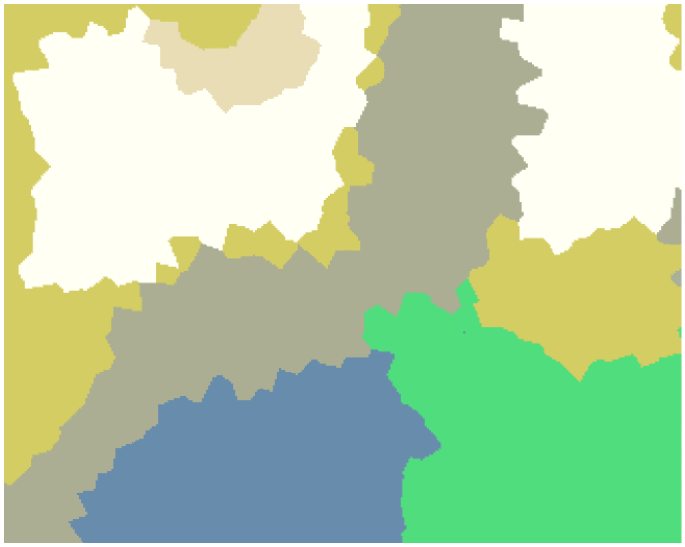


Joonis 34. Simpleksmüraga hägustatud piiridega bioomirakud.

bioomitüki, vaid lähima keskpunkti põhjal, siis järgnevalt räägitakse lähima keskpunktiga piiratud alast kui bioomirakust. Seejuures vastab igale maastikutulbale parajasti üks bioomirakk, mis määrab bioomi selles tulbas.

Kuna maastiku genereerimine toimib maastikutüki kaupa, ei ole mõttekas iga maastikutulba lähimate bioomitükkide keskpunkte korduvalt arvutada. Seetõttu on otstarbekas enne tulpadele bioomide määramise alustamist leida kõigi genereeritava maastikutükiga lõikuvate bioomitükkide keskpunktid ning nende bioomitükkide vahetute naabrite keskpunktid ja need ajutiselt sõnastikku hoiustada. Seejärel saab genereeritava tüki maastikutulpadele bioome määrates leitud keskpunkte korduvalt kasutada, ilma et neid uuesti arvutama peaks.

Valikuliselt võib algoritmi veelgi täiendada, et vähendada tekkivate bioomirakkude äärte korrapärasust. Selleks saab muuta algoritmi lähimat keskpunkti leidvat osa. Selle asemel, et üle uuritavate bioomitükkide keskpunktide itereerides lihtsalt kontrollida keskpunkti kaugust uuritavast maastikutulbast, võib kaugusele liita kahemõõtmelise simpleksmüra väärtuse vaadeldava maastikutulba koordinaatide põhjal. See tähendab, et alati ei valita iga maastikutulba jaoks lähimat keskpunkti, vaid sõltuvalt müra väärtusele võidakse valida ka mõni kaugem keskpunkt (vt. joonis 34). Selgus, et lõpptulemusse selline täiustus eriti palju



Joonis 35. Bioomide määramine suurel maastiku alal kasutades temperatuuri ja sademete hulga jaoks simpleksmüra järgnevate parameetritega: kihte 3, sagedus 0.005, lakunaarsus 2, püsivus 0.5.

visuaalset väärtust juurde ei andnud ning programmi kiiruse huvides on rakenduses simpleksmüraga rakkude piiride hägustamise funktsionaalsus välja lülitatud.

Kasutades sademete ja temperatuuri arvutamiseks kasutatava müra jaoks piisavalt väikest sagedust, satuvad kõrvuti paiknevatesse bioomirakkudesse sarnased müra väärtused. Selle tulemusena paiknevad bioomid enamasti üle mitme bioomiraku ning tekivad suured, omapärase kujuga bioomid (vt. joonis 35).

7.3 Bioomidega maastiku kujundamine

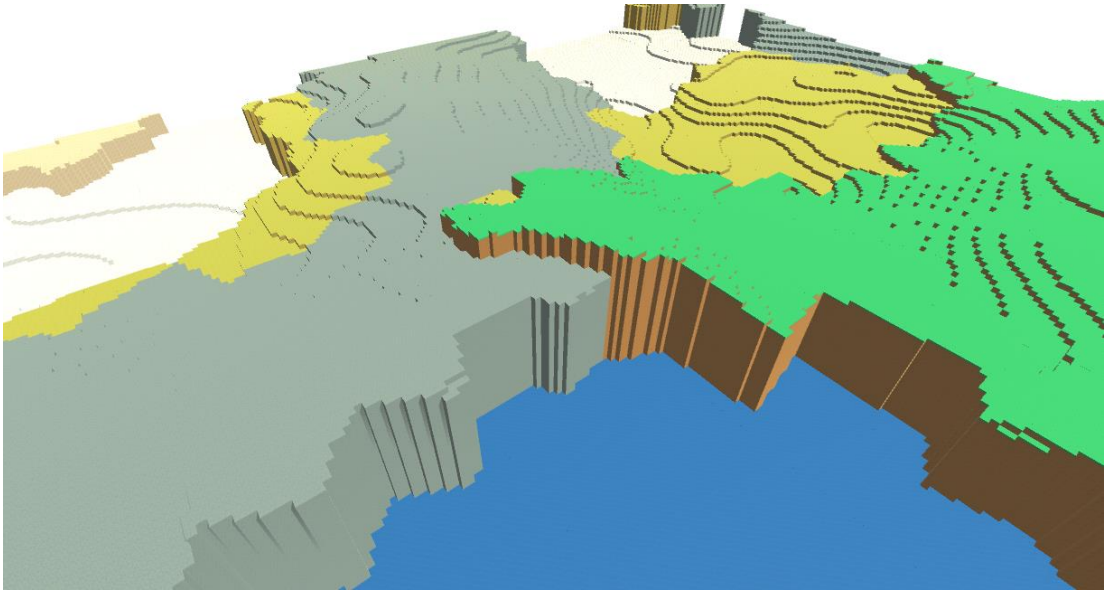
Eelmises alapeatükis kirjeldati algoritmi, millega saab igale maastikutulbale määrata bioomi. Siinkohal pakub autor välja paar võimalust, kuidas maastikku vastavalt määratud bioomidele algoritmiliselt kujundada.

Kõige lihtsam muudatus võrreldes tavalise kõrguskaardiga maastiku genereerimisega on asendada muru ja muld vaadeldava maastikutulba bioomi pinnasetüübiga. Järgmise sammuna võib enne iga maastikutulba genereerimist sellele vastava kõrguskaardi väärtust korrutada tulba bioomi võimendusteguriga v. Selle tulemusena on erinevates bioomides maastik erineva künklikuse ja erineva maksimaalse kõrgusega (vt. joonis 36).

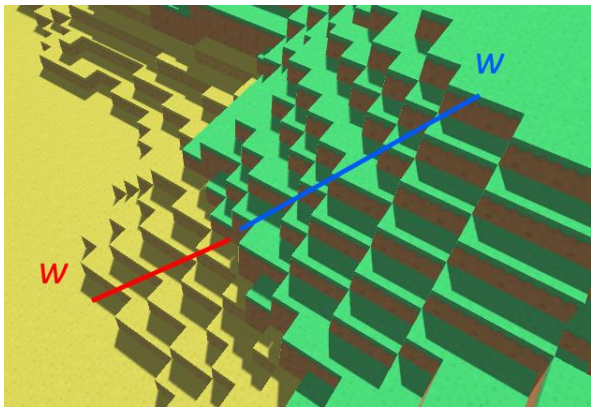
Jooniselt 36 on näha, et selle tulemusena on erineva võimendusteguriga bioomide vaheline üleminek väga järsk ja ebaloomulik. Selle parandamiseks peab bioomide piiride lähistel arvutama uue võimendusteguri, mis sõltub lähimate keskpunktide kaugusest ja neile vastavatest võimendusteguritest. Näiteks lihtsaimal juhul võiks olla kahe bioomi b_1 ja b_2 piiril võimendusteguriks $v = \frac{v_{b_1} + v_{b_2}}{2}$ ning eemaldudes piirist ühe bioomi suunas, muutuks võimendustegur lineaarselt seal oleva bioomi võimendusteguriks alates fikseeritud konstantsest kaugusest w (vt. joonis 37). Sarnaselt võiks olla juhul, kui ristuvad kolm või enam bioomi, ristumiskohas võimendusteguriks nende bioomide võimendustegurite aritmeetiline keskmine. Seega üldistatult võiks iga maastikutulba bioomiga B jaoks käituda järgnevalt:

- kui tulp asub lähimast bioomide piirist kaugemal kui w, rakendada tulbale tulba bioomi B võimendustegurit;
- kui tulp asub n-le bioomile lähemal kui w, arvutada uus võimendustegur B võimendusteguri ja nende n võimendusteguri kaudu.

Vaatleme võimendusteguri leidmist suvalises maastikutulbas T. Sujuvate üleminekute lisamiseks on kõigepealt vaja määrata konstant w, mis määrab mitme punkti kauguselt bioomidevahelisest piirist kohandatud üleminekut rakendama hakatakse. Soovituslik on valida $w \leq \frac{l_{bioom}}{2}$, sest sellisel juhul piisab selleks, et määrata, kas lähedal on veel teisi bioome, ainult vaadeldava maastikutulba bioomitüki vahetute naabrite kontrollimisest.



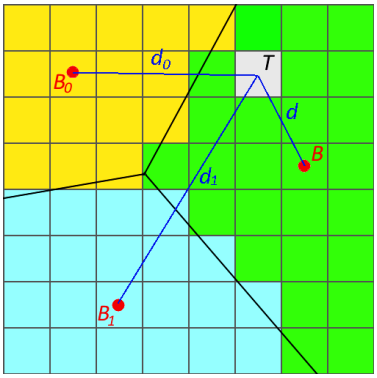
Joonis 36. Maastik, kus igale maastikutulbale on määratud pinnas ning kõrguskaardi arvutamisel on kasutatud ka bioomi võimendustegurit.



Joonis 37. Kahe erineva bioomi vaheline interpoleeritud võimendusteguriga üleminek. $w = 6, l_{bioom} = 12$

Pärast w määramist tuleb tulpa T sisaldava bioomitüki B koordinaatide kaudu leida bioomitüki naabertükid. Naabertükkide abil on seejärel võimalik määrata, millised bioomid T lähedal paiknevad ning kui kaugel on nende keskpunktid tulbast T. Bioomide määramise peatükis soovitas autor, et maastikutüki genereerimisel hoiustataks ajutiselt kõik genereeritava maastikutükiga lõikuvad bioomitükid ning nende vahetud naabrid koos neid puudutavate andmetega, et neid korduvalt kasutada. Selle tulemusena sisalduvad hoiustatud bioomitükkide seas ka meile huvipakkuvad bioomitükid. Järgmisena itereeritakse üle naaberbioomitükkide ning kontrollitakse, kas nende bioom erineb tulbas T olevast bioomist. Kui ükski bioom ei erine, siis järelikult asub tulp T kuskil bioomi keskel ning seal võib rakendada lihtsalt vastava bioomi enda võimendustegurit. Kui siiski leidub bioomitükke $B_i, i \in \{0, 1, \dots, m\}$, mille bioom erineb tulbas T olevast, peab arvutama uue võimendusteguri.

Uue võimendusteguri arvutamiseks fikseeritakse kõigepealt tulpa T sisaldava bioomi B võimendustegur v_B, keskpunkt k_B ning keskpunkti kaugus tulbast T: $d = |B - k_b|$.



Joonis 38. Võimendusteguri arvutamiseks vajalikud suurused.

Järgmiseks leitakse iga ümbritseva bioomi $B_i, i \in \{0, 1, \dots, m\}$ keskpunkti k_i kaugus tulbast T, mida tähistatakse sümboliga d_i (vt joonis 38). Seejärel filtreeritakse bioomide B_i seast välja need, mille korral $|d_i - d| \geq w$, sest need bioomid ei mõjuta tulba T võimendustegurit. Ütleme, et alles jääb $n \leq m$ bioomi ning tähistame neid kasutades sümboleid $B_i, i \in \{0, 1, \dots, n-1\}$, sealjuures olgu B_i tulbale T lähemal kui B_{i+1} iga i korral. Selle tulemusena on meil teada tulbale T lähimad kauguse kasvades sorteeritud bioomid, mis mõjutavad T võimendustegurit.

Arvutame nüüd iteratiivselt võimendusteguri tulbas T. Selleks fikseerime esiteks võimendusteguri $v \coloneqq v_B$. Seejärel rakendame võimendustegurile v järgnevat protseduuri n korda iga $i \in \{0, 1, \dots, n-1\}$ jaoks:

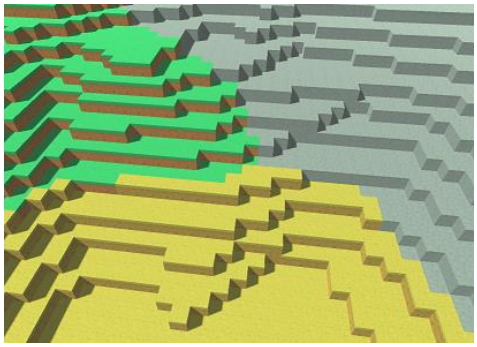
- arvutame bioomide B ja $B_j, j \in \{0, 1, \dots, i\}$ võimendustegurite aritmeetiliste keskmise:

$$\overline{v_i} = \frac{v_B + \sum_{j=0}^{i} v_j}{i+1};$$

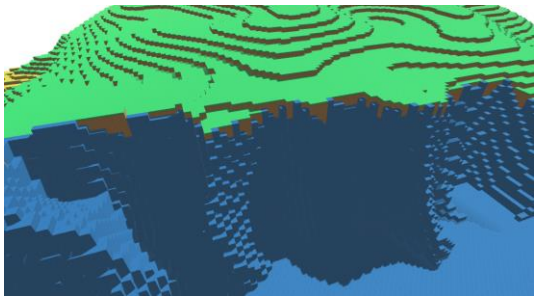
- uuendame senist võimendustegurit interpoleerides uue väärtuse võimendusteguri v ja keskmise võimendusteguri $\overline{v_i}$ põhjal kasutades kaaluna $k = \frac{|d_i - d|}{w} \in [0, 1]$:

$$v \coloneqq \overline{v_i} * (1 - k) + v * k.$$

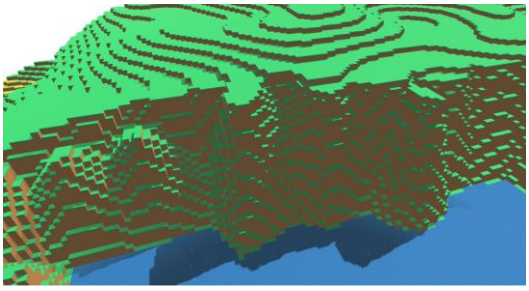
Peale itereerimise lõppu on saadud v võimendustegur tulbas T. Algoritmi esimene samm arvutab $i + 1$ bioomi korral võimendusteguri, mis saavutataks, kui T asuks $i + 1$ bioomi ristumispunktis. Antud juhul on selleks $i + 1$ bioomide võimendustegurite keskmine. Teise sammuna arvutatakse uus interpoleerimistegur kasutades i bioomi põhjal leitud võimendustegurit. Kaal k on selline, et kui T asub bioomide B_i ja B piiril, siis on kaalu väärtuseks 0 ning kui T asub bioomi piirist ligikaudu kaugusel w, on kaalu väärtuseks 1. See tähendab, et kui T asub bioomide piiril, määratakse võimendusteguriks bioomide võimendustegurite keskmine, kuid eemaldumisel bioomide piirist läheneb võimendustegur bioomi B võimendustegurile. Autori arvates annab selline algoritm võimendustegurite arvutamiseks sobiva tulemuse (vt. joonis 39).



Joonis 39. Kolme erineva võimendusteguriga bioomi bioomidevaheline interpoleeritud üleminek.



Joonis 40. Mere problemaatiline üleminek maismaaks.



Joonis 41. Segunemisteguri rakendamisel tekkiv mere üleminek maismaaks.

Paraku ei sobi antud lahendus veel kõigil juhtudel. Näiteks vee ning maa vahelisel piiril ei tohiks veepiir maastiku võimendusteguriga koos tõusta (vt. joonis 40). Seetõttu on igale bioomile lisaks määratud segunemistegur s, mis määrab, kui palju antud bioom ennast piiridel teistest bioomidest mõjutada laseb. Segunemistegur 1 tähendab, et bioom käitub tavaliselt, suur arv aga tähendab, et bioom laseb ennast teistest bioomidest vähem mõjutada. Selle implementeerimiseks, muudetakse eelneva segunemisalgoritmi esimest sammu. Aritmeetilise keskmise $\overline{v_i}$ asemel arvutatakse kaalutud keskmine, kus kaaludena kasutatakse iga bioomi segunemistegurit:

$$v_{kaalutud_i} = \frac{v_B * s_B + \sum_{j=0}^{i} v_j * s_j}{s_B + \sum_{j=0}^{i} s_j}.$$

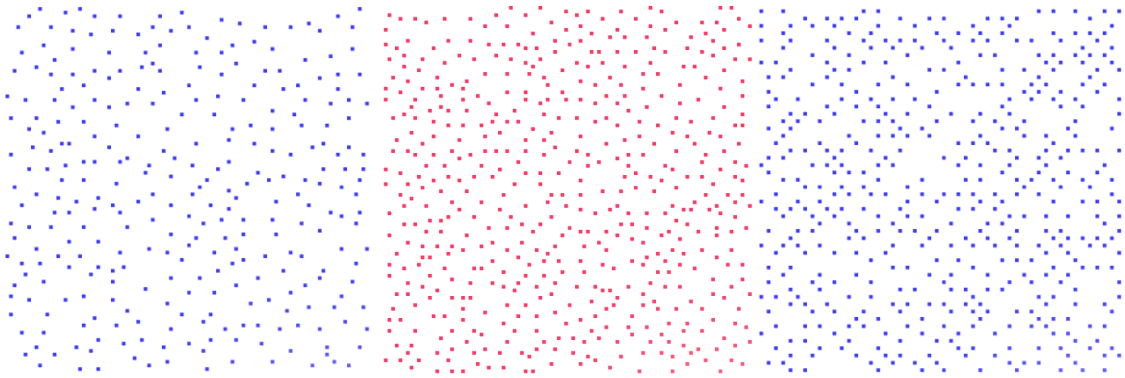
Selle tulemusena on merele väga suure segunemisteguri määramise korral teiste bioomide mõju olematu ning meri püsib sobivalt tasane (vt. joonis 41).

8. Taimede paigutuse genereerimine

Seni genereeritav maastik on küll mitmekülgse väljanägemisega pinnakatte ja kõrgendike poolest, kuid näeb siiski tühi välja. Selle parandamiseks saab algoritmi täiendada, lisades maastikule vastavalt bioomile sobivat taimestikku. Nagu ülejäänud algoritmgi, peab ka taimede valiku ja paigutuse genereerimine olema deterministlik. Antud töös genereeritakse põhiliselt puid, kuid sekka on lisatud ka väiksemaid rohttaimi rohumaa- ning savannibioomile. Taimede asukohad genereeritakse maastikutulpade täpsusega ehk maksimaalselt saab ühes maastikutulbas paikneda üks taim. See taim paigutakse vaadeldava tulba kõige ülemise sellise punkti peale, mis ei ole tüübiga Õhk.

Taimede paigutuse genereerimisega kaasneb mitmeid probleeme. Esiteks peab algoritm paigutama taimed maastikutulpadele nii, et tulemusena ei jääks näha selget mustrit, mille põhjal taimede asukohad jaotati. Vastasel korral näevad genereeritud taimed välja nagu istanduses mustri järgi istutatud taimed, kuid töö eesmärgiks on genereerida metsikut taimestikku. Teiseks peab arvestama, et puude paigutamisel ei satuks puud kõrvuti paiknevatele maastikutulpadele, sest puud on enamasti horisontaalselt laiemad, kui üks maastikutulp. Puude suurust arvestamata jättes juhtub aeg-ajalt, et kahe puu mudelid on osaliselt üksteise sees, mis ei näe hea välja.

Mõlema kirjeldatud probleemi lahenduseks sobib taimede asukohtade määramiseks kasutada simpleksmüra autori väljatöötatud viisil. Nimelt on võimalik iga maastikutulba koordinaatide kaudu leida sellele vastava kahemõõtmelise simpleksmüra väärtus ning seda tulpa vahetult ümbritsevatele 8 maastikutulbale vastavat müra väärtust. Kui keskmisele tulbale vastav väärtus on ümbritsetavatest suurem ehk tegu on n-ö lokaalse maksimumiga, siis on antud maastikutulp taime paigutamiseks sobilik. Selline lähenemine tagab, et kahe taime asukoha vahel on alati vähemalt üks tühi maastikutulp, sest kõrvuti paiknevates tulpades ei saa lokaalne maksimum korraga paikneda. Lisaks, kuna simpleksmüra kasutab võrestikuna erinevalt väärtusmürast ja Perlini mürast koordinaatteljestikuga mitteparalleelselt ruumijaotust, siis ei moodusta simpleksmüra lokaalsed maksimumid märgatavat korrapärast mustrit (vt. joonis 42). Lisaks on võimalik muuta simpleksmüra sagedust, mis muudab lokaalsete maksimumide esinemissagedust. See võimaldab erinevatele bioomidele anda erinevad taimede esinemissagedused. Näiteks kõrbe korral on sagedus väga madal, sest taimi esineb harva, rohumaal aga väga kõrge, sest rohttaimi

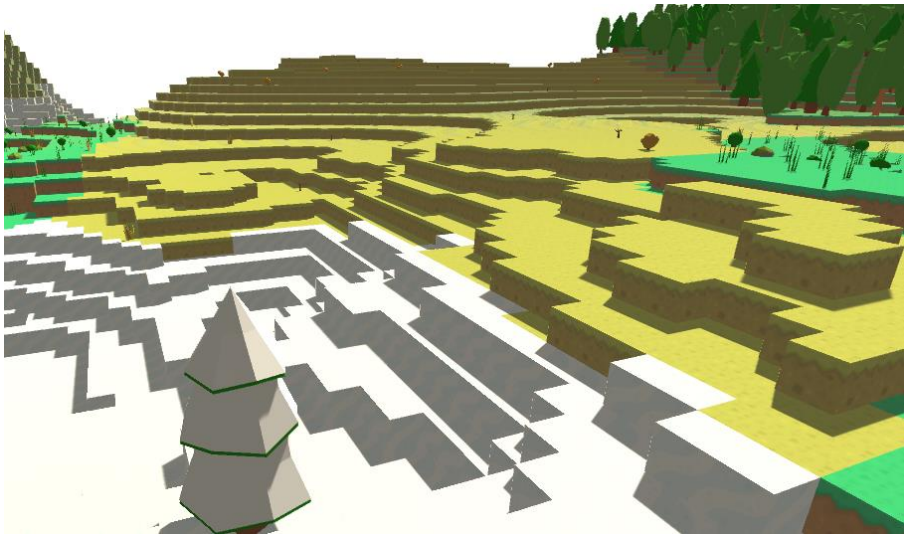


Joonis 42. Ühe kihi simpleksmüra lokaalsed maksimumid sagedusega 0.16 (vasakul) ja 0.22 (keskmisel) ning ühe kihi väärtusmüra lokaalsed maksimumid sagedusega 0.5 (paremal). Nagu näha, on väärtusmüra lokaalsete maksimumide jaotus korrapärasem kui simpleksmüra jaotus. Sarnane jaotus on ka Perlini müral.

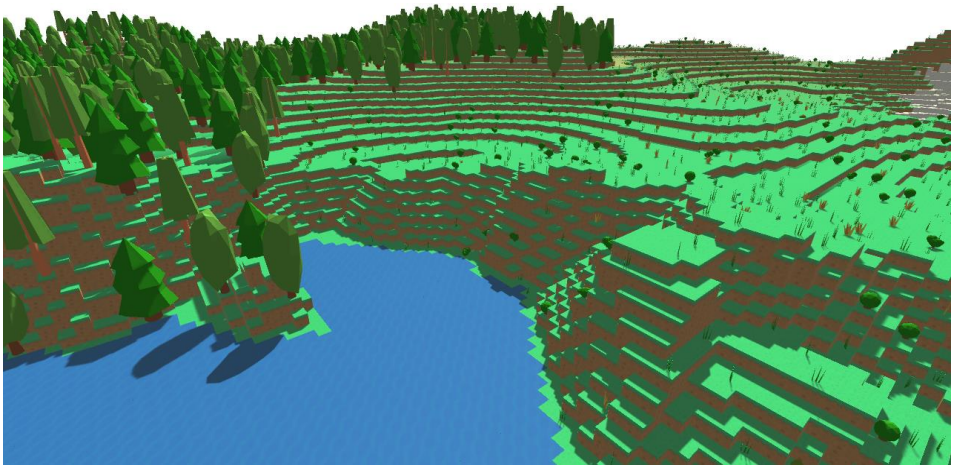
esineb tihkelt. Täpsemad väärtused kõigi bioomide jaoks on välja toodud bioomide omaduste peatükis 7.1.

Taimeliigi valimiseks maastikutulbas luuakse iga bioomi jaoks enne maastiku genereerimist massiiv, mis sisaldab vastava bioomi kõikvõimalikke taimi ning nende esinemissagedust p_i. Seejärel kujutatakse taimede paigutuseks sobilike maastikutulpade koordinaadid sarnaselt väärtusmüra funktsioonile f_{tipp} üheks reaalarvuks r. Siis jagatakse r maksimaalse võimaliku väärtusega, mida koordinaatide kujutus anda võib, et saada tõenäosustegur $p = \frac{r}{r_{max}} \in [0,1]$. Tulemuseks saadud tõenäosustegurit p kasutatakse vastava bioomi taimede esinemissageduste hulgast taime valimiseks ning valitud taime mudel lisatakse maastikule. Viimase sammuna võib taimedele omapära lisamiseks r põhjal taime suurust natukene, näiteks kuni 20% muuta ja taime $(r \bmod 360°)$ kraadi võrra ümber z-telje pöörata, et igal taimel oleks omaette pseudojuhuslik pööre ruumis.

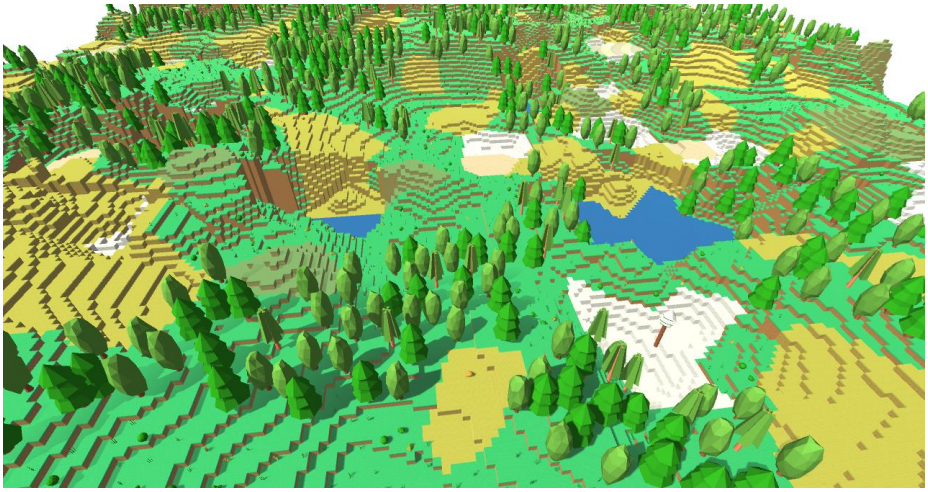
Nüüd saab iga maastikutulpa genereerides kontrollida, kas tulp on taime paigutamiseks sobilik ning seejärel sobivuse korral deterministlikult sinna ka taime valida. Implementeerides erineva taimkatte kõigile bioomidele, saame lõpptulemuseks visuaalselt päris põneva maastiku (vt. joonised 43, 44, 45).



Joonis 43. Külma kõrbe ning tundra hõre taimkate.



Joonis 44. Metsa ja rohumaa bioomi taimestik.

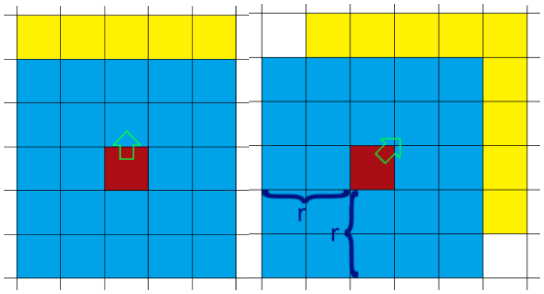


Joonis 45. Kõikvõimalikud implementeeritud bioomid väikesel kujul koos taimkattega.

9. Nõuetele vastavus

Töös kirjeldatud algoritmi implementeerivale rakendusele püstitati viis nõuet, mida kõiki rakendus ka rahuldab.

Esiteks on algoritm suuteline genereerima teoreetiliselt lõpmatu suurusega maastikke, sest maastikutükkide genereerimiseks kasutati müraalgoritme, mida on võimalik rakendada suvalises punktis maastiku genereerimiseks. Seetõttu on teoreetiliselt võimalik genereerida ükskõik kus paiknev maastikutükk selle koordinaatide paari (x_{mt}, y_{mt}) ja maastiku seemne abil. Teiseks on kõik maastiku genereerimise algoritmi sammud ühe seemne abil genereeritud pseudojuhuslike väärtuste tõttu deterministliku tulemusega ning seetõttu saadakse sama seemne kasutamisel alati täpselt sama maastik. Kolmandaks on maastikutükid omavahel ühilduvad, sest peatükkides 6 ja 7.3 välja töötatud lahendus tagab, et lähestikku paiknevad maastikutulbad oleksid samasuguse kõrgusega. Neljandaks on genereeritav maastik mitmekülgne, sest genereeritakse mitmeid erinevaid bioome ehk makroökosüsteeme ning neile vastavat taimestikku. Samuti on üpris lihtsa vaevaga võimalik rakendusse genereeritavaid bioome juurde lisada.



Joonis 46. Kasutaja liikumisel ühelt tükilt (punaselt) teisele, on raadiuse 2 korral vaja juurde genereerida kas 5 (vasakul) või 9 (paremal) maastikutükki.

Viimasena püstitati rakendusele nõue maastikku piisavalt kiiresti genereerida. Testides rakendust Intel i5 4460 3.20GHz protsessoriga arvuti peal, genereeriti 1000 maastikutükki 38,4 sekundiga ehk ühe maastikutüki genereerimiseks kulus keskmiselt 38,4 millisekundit, millest umbes kolmandik kulus tüki vokslitest koosneva mudeli ehitamiseks ning ülejäänud aeg andmete genereerimiseks. Kui kasutaja liigub rakenduses kiirusega 4 vokslit sekundis, kulub ühe maastikutüki läbimiseks $\frac{16}{4} = 4$ sekundit ning seetõttu on iga nelja sekundi tagant vaja juurde genereerida mingi hulk maastikutükke. Juurde genereeritavate tükkide arv sõltub rakenduses määratud raadiusest. Näiteks raadiuse 2 korral on juurde vaja genereerida maksimaalselt üheksa tükki (vt. joonis 46). Rakendust testides selgus, et mõistlik raadius on 7, mis tähendab, et korraga on genereeritud maastikutükke $(1 + 7 * 2)^2 = 225$.

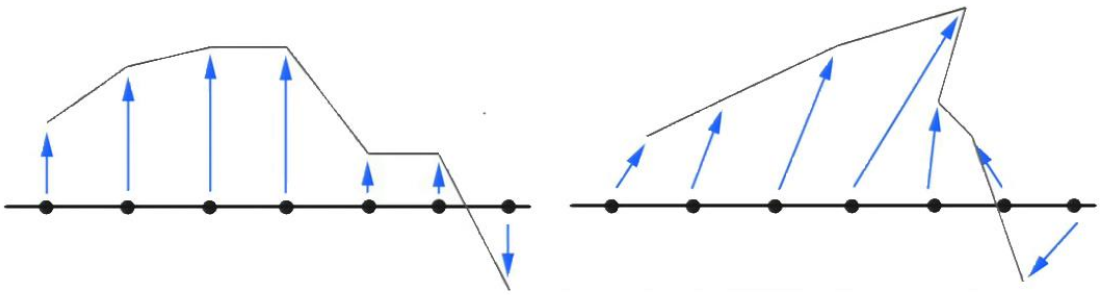
Joonisele 46 toetudes selgub, et liikudes ühelt maastikutükilt teisele, on juurde vaja genereerida $2r + 1$ kuni $4r + 1$ ehk $r = 7$ korral 15 kuni 31 maastikutükki, mis võtaks aega vastavalt $38.4 * 15 = 576$ kuni 38.4 * 29 = 1190 millisekundit. Seega oleks $r = 7$ ja $v = 4\ vokslit/s$ korral võimalik liikumiskiirust või raadiust veelgi suurendada. Rakenduse kasutamisel aeglasema protsessoriga arvutil on võimalik raadiust vähendada, kuni tükkide genereerimisele kuluv aeg on väiksem kui maastikutüki läbimiseks kuluv aeg.

10. Edasiarendusvõimalused

Järgnevalt on pakutud välja paar lõpmatu maastiku genereerija edasiarendusvõimalust, mis bakalaureusetöö mahu tõttu töö skoopi ei mahtunud.

Kuna maastiku genereerimisel kasutati kõrguskaardi ideel põhinevat kahemõõtmelist müra, ei ole võimalik senise algoritmiga genereerida ühtki eendit või muusuguseid maastikuosasid, kus mingi osa maastikust paikneb teise osa peal, vaid ainult vertikaalselt tõstetud maastikku. Järgnevalt pakutakse välja paar võimalust eendite genereerimiseks.

Üks võimalus eendite genereerimiseks on kõrguskaardi lahenduse edasiarendamine, kus otse vertikaalselt maastikutulba kõrguskaardi väärtuse võrra tõstmise asemel kasutatakse teist kahemõõtmelist müra määramaks, mis suunas kõrguskaardi väärtusi kallutada, et tekitada ka eenduvaid maastiku osi (vt. joonis 47). Sellist lahendust kasutab näiteks 2009. aastal valminud mäng Halo Wars [26].



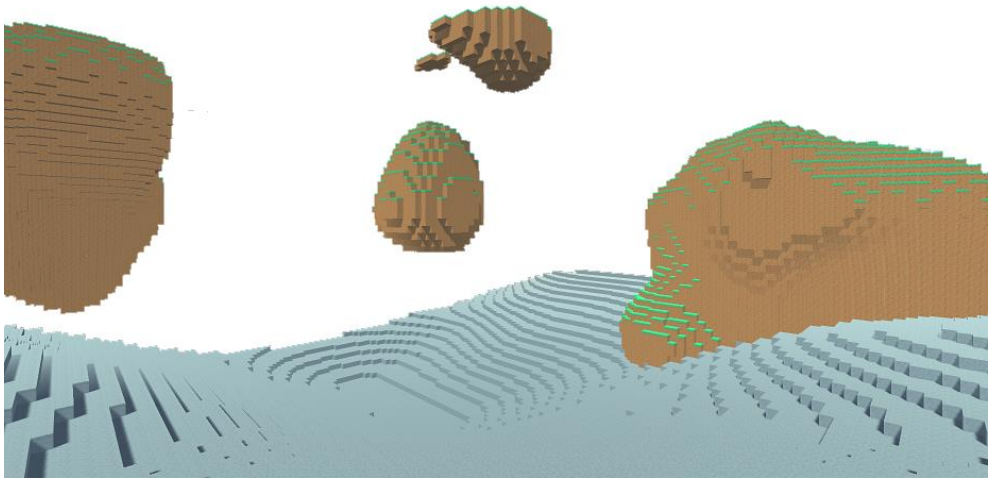
Joonis 47. Tavalise kõrguskaardiga (vasakul) ja kallutatud kõrguskaardiga (paremal) tekkida saav maastikukuju [26].

Teine võimalus huvitavamate maastikuvormide tekitamiseks on lisaks kõrguskaardil põhinevale maastiku genereerimisele kasutusele võtta kolmemõõtmeline müra. Kõrguskaardiga genereeritud maastik katab ainult mingi alumise osa maastikutüki sisemusest, kuid võiks ka ülejäänud tühja ruumi osaliselt täita. Selle täitmiseks sobib itereerida üle tühjaks jäänud maastikutüki punktide ning arvutada iga punkti globaalsete koordinaatide põhjal selles punktis olev kolmemõõtmelise simpleksmüra väärtus. Järgnevalt peab väärtuse põhjal otsustama, kas antud punkti paigutada nähtav osa maastikust või jätta see tühjaks. Selle tegemiseks sobib kasutada lävendit l, millest väiksemate müra väärtustega punktid märgitakse vastava maastikutulba pinnasetüübiks

ning suuremad jäetakse tühjaks. Näiteks bioomideta maastiku korral, kus $M = \{\text{Õhk}, \text{Muru}, \text{Muld}\}$, sobib 3D müraga genereeritava maastiku jaoks funktsioon

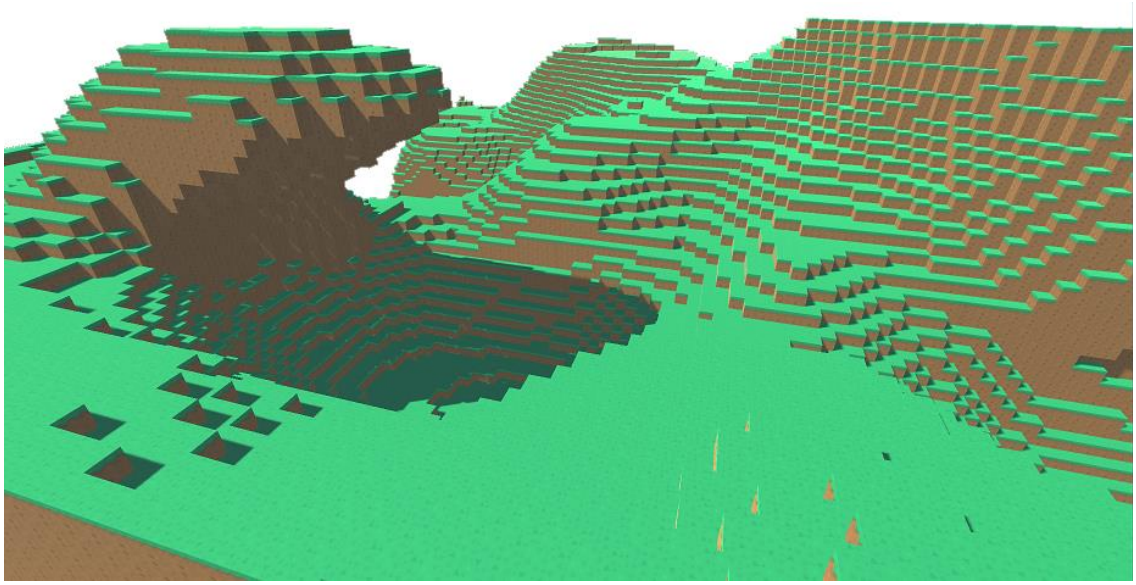
$$f_M(x_{gp}, y_{gp}, z_{gp}) = \begin{cases} \text{Muru}, & \text{kui } f_{simp}(x_{gp}, y_{gp}, z_{gp}) < l \wedge f_M(x_{gp}, y_{gp}, z_{gp}+1) = \text{Tühi}, \\ \text{Muld}, & \text{kui } f_{simp}(x_{gp}, y_{gp}, z_{gp}) < l \wedge f_M(x_{gp}, y_{gp}, z_{gp}+1) \neq \text{Tühi}, \\ \text{Tühi}, & \text{kui } f_{simp}(x_{gp}, y_{gp}, z_{gp}) \geq l. \end{cases}$$

Selle tulemusena tekib jällegi väga palju hõljuvaid ümaraid struktuure, sest hetkel ei taga miski, et 3D müraga tekkivad tükid asuksid vähemalt mingil määral maapinnal (vt. joonis 48). Tulemust saab parandada, asendades konstantse lävendi l lävendiga l_z, mis sõltub



Joonis 48. Tavalise kõrguskaardiga (hall kivi) genereeritud maastikule 3D müraga peale genereeritud maastik (muld ja muru).

punkti z- ehk kõrguskoordinaadist: $l_z = 1 - \frac{z_{gv}}{h_{mt}} \in [0, 1]$. Lävendit kõrguse kasvades lineaarselt vähendades väheneb hõljuvate maastikuosade hulk märgatavalt ning kolmemõõtmelise müraga lisanduvad maastikuosad esinevad põhiliselt just kõrguskaardiga genereeritud maastiku kohal. Muutuva lävendi lisamise tulemusena tekib maastikule rohkem huvitavamaid mägesid, kuid siiski võib aeg-ajalt esineda ka hõljuvaid struktuure (vt. joonis 49).



Joonis 49. Muutuva kolmemõõtmelise müra lävendi rakendamise tulemusena tekkiv maastik.

Tasub märkida, et kaasates maastiku genereerimisse kolmemõõtmelise müra, on vaja teha ka suuri muudatusi senises bioomide määramise ning bioomidevahelise ülemineku algoritmis, sest saadud maastikku on raskem bioomide piiridel kokku sobitada. Otsustades kolmemõõtmelist müra siiski kasutada, peab samuti arvestama sellega, et maastiku kuju määramiseks genereeritavaid 3D müra väärtusi on mitukümmend korda rohkem kui ainult kõrguskaardi müra kasutades, mis võib algoritmi töökiirust kordades vähendada. See on eriti märgatav juhul, kui maailm on jagatud väiksemateks tükkideks kui käesolevas töös. Kiiruse mõnevõrra parandamiseks on sellisel juhul sobilik arvutada müra väärtused mitte iga maastikupunkti, vaid iga teise jaoks ning vahele jäänud väärtused ümbritsetavate kaudu interpoleerida, mis hoiab müra väärtuste leidmisele kuluvat aega mitmeid kordi kokku.

11. Kokkuvõte

Käesoleva töö eesmärgiks oli tutvustada arvutigraafika valdkonnas laialt levinud müraalgoritme ning nende rakendamisvõimalusi lõpmatu maastiku generaatori loomisel. Lisaks maastiku genereerimise algoritmi teoreetilisele kirjeldusele valmis koostöös ainega „Arvutigraafika projekt“ kirjeldatud algoritmi suuremas osas implementeeriv rakendus.

Kirjeldatud algoritm võtab seemneks ühe täisarvulise väärtuse ning genereerib selle põhjal tüki kaupa maastikku. Iga tüki jaoks genereeritakse simpleksmüraga kõrguskaart, mida kasutatakse alusmaastikuna. Järgmisena jagatakse maailm deterministlikult erinevate bioomide ehk makroökosüsteemide vahel erikujulisteks tükkideks. Selleks kasutatakse autori väljamõeldud lahendust, mis jagab maastiku Voronoi diagrammi rakkudeks ning määrab igale rakule simpleksmüra kasutades bioomi. Seejärel modifitseeritakse maastikualuse kuju ning pinnasetüüpi vastavalt piirkonna bioomile ning kõrvuti paiknevad bioomid ühendatakse sujuvate üleminekutega. Viimase sammuna lisatakse maastikule bioomile omane taimestik koos sinna sobiva taimestikutihedusega, kasutades selleks autori loodud lahendust. Töös kirjeldatud maastiku genereerija on täielikult determineeritud ning genereerib muutmata kujul sama seemet kasutades alati samasuguse maastiku olenemata genereerimise alguspunktist. Lisaks müradele ning maastiku genereerimise algoritmile pakuti töös erinevaid võimalusi virtuaalsete maastike visualiseerimiseks ning kirjeldati mitmeid töös esitatud algoritmi edasiarendusvõimalusi, mis teeksid genereeritava maastiku veelgi mitmekülgsemaks ja maastiku avastamise huvitavamaks.

Töö teema valik oli ajendatud autori personaalsest huvist arvutigraafika ja protseduurilise genereerimise vastu ning autor plaanib valminud algoritmi tulevikus edasi arendada, asendades töös kasutatava vokslite algoritmi pinna võrgustiku algoritmiga ning suurendades maastikutükkide punktide hulka. Lisaks soovib autor mingil kujul implementeerida maastiku genereerimisse kolmemõõtmelise müra, mis võimaldaks genereerida unikaalsete vormidega mägesid ning koopaid. Suurema eesmärgina soovib autor välja mõelda lahenduse, kuidas lisada tüki kaupa jõgede ning hoonete deterministlik genereerimine ja kuidas täiendada bioomide määramise algoritmi nii, et tihedamini satuks kokku sarnaste kliimatingimustega bioomid. Autor on valminud tööga üldjoontes rahul, kuid järele mõeldes eelistanuks töö meediumina kasutada Microsoft Wordi asemel TeX tarkvara.

12. Kasutatud materjalid

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

[2] Shamus Young. (2009, mai) Procedural City. [Võrgumaterjal]. http://www.shamusyoung.com/twentysidedtale/?p=3220 (vaadatud 12.05.2016)

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

[4] Jacob Olsen, "Realtime Procedural Terrain Generation," Department of Mathematics and Computer Science, University of Southern Denmark, 2004.

[5] Remco Huijser, Jeroen Dobbe, Willem F. Bronsvoort ja Rafael Bidarra, "Procedural Natural Systems for Game Level Design," *Brazilian Symposium on Games and Digital Entertainment*, Rio de Janeiro, 2010, lk. 189-198.

[6] Daniel Michelson De Carli, Fernando Belivacqua, Cesar Tadeu Pozzer ja Marcos Cordeiro d'Ornellas, "A Survey of Procedural Content Generation Techniques Suitable to Game Development," *Brazil Symposium on Games and Digital Enterainment*, Rio de Janeiro, 2011, lk. 26-35.

[7] Fernando Bevilacqua, Cesar Tadeu Pozzer, d'Ornellas, ja Marcos Cordeiro, "Charack: Tool for Real-Time Generation of Pseudo-Infinite Virtual Worlds for 3D Games," *Brazilian Symposium on Games and Digital Entertainment (SBGAMES)*, Rio de Janeiro, 2009, lk. 111-120.

[8] Aitor et al., "Volumetric Virtual Worlds with Layered Terrain Generation," *International Conference on Cyberworlds*, Yokohama, 2013, lk. 20-27.

[9] Jasper Flick. (2014) Simplex Noise. [Võrgumaterjal]. http://catlikecoding.com/unity/tutorials/simplex-noise/ (vaadatud 12.05.2016)

[10] John F. Hughes, Andries Van Dam, James D. Foley ja Steven K. Feiner, *Computer graphics: principles and practice*, kolmas trükk. Pearson Education, 2013.

[11] William E. Lorensen ja Harvey E. Cline, "Marching cubes: a high resolution 3D surface construction algorithm," *SIGGRAPH*, 1987.

[12] Akio Doi and Akio Koide, "An Efficent Method of Triangulating Equi-Valued Surfaces by Using Tetrahedral Cells," *IEICE Transactions on Information and*

Systems, vol. E74-D, nr. 1, lk. 214-224, jaanuar 1991.

[13] Mikola Lysenko. (2012, juuli) Smooth Voxel Terrain. [Võrgumaterjal]. https://0fps.net/2012/07/12/smooth-voxel-terrain-part-2/ (vaadatud 12.05.2016)

[14] Ken Perlin. (1999, detsember) GDCHardCore. [veebis]. http://www.noisemachine.com/talk1/ (vaadatud 12.05.2016)

[15] Ken Perlin, "Improving Noise," *SIGGRAPH*, 2002, lk. 681-682.

[16] Stefan Gustavson, "Simplex Noise Demystified," Linköping University, 2012.

[17] Clifford A. Pickover, *The Math Book: From Pythagoras to the 57th Dimension, 250 Milestones in the History of Mathematics*, esimene trükk. Sterling, 2009.

[18] Jasper Flick. (2014) Noise. [Võrgumaterjal]. http://catlikecoding.com/unity/tutorials/noise/ (vaadatud 12.05.2016)

[19] Ronald Aylmer Fisher ja Frank Yates, *Statistical Tables for Biological, Agricultural and Medical Research*, kuues trükk. Longman Group Ltd, 1963.

[20] Ülo Kaasik, *Matemaatikaleksikon*, kolmas trükk. Tartu, Eesti: Atlex, 2003.

[21] Ken Perlin. Ken's Academy Award. [Võrgumaterjal]. http://mrl.nyu.edu/~perlin/doc/oscar.html (vaadatud 12.05.2016)

[22] USPTO. (2005, märts) Standard for perlin noise. [Võrgumaterjal]. https://www.google.com/patents/US6867776 (vaadatud 12.05.2016)

[23] Neil A. Campbell and Jane B. Reece, *Biology*. Redwood City: Pearson, 2005.

[24] University of California Museum of Paleontology. (2007) The World's Biomes. [veebis]. http://www.ucmp.berkeley.edu/exhibits/biomes/index.php (vaadatud 12.05.2016)

[25] Leslie R. Holdridge, "Determination of World Plant Formations From Simple Climatic Data," *Science*, vol. 105, nr. 2727, lk. 367-368, aprill 1947.

[26] Colt MCAnlis. (2009) HALO WARS: The Terrain of Next-Gen. Presentation. [veebis]. http://www.gdcvault.com/play/1277/HALO-WARS-The-Terrain-of (vaadatud 12.05.2016)

[27] Heikki Vallaste. (2016, mai) IT ja sidetehnika seletav sõnaraamat. [veebis]. http://www.vallaste.ee/index.asp (vaadatud 12.05.2016)

[28] Eric W. Weisstein. Hypercube. [Võrgumaterjal]. http://mathworld.wolfram.com/Hypercube.html (vaadatud 12.05.2016)

[29] Frank W. Warner, *Foundations of differentiable manifolds and Lie gorups*, esimene trükk. New York, Ameerika Ühendriigid: Springer, 1983.

[30] Ömer Cengiz. Scientific Visualization and 3D Volume Rendering. [Võrgumaterjal]. http://www.byclb.com/TR/Tutorials/volume_rendering/ (vaadatud 12.05.2016)

[31] Shamus Young. (2012, mai) Project Octant. [Võrgumaterjal]. http://www.shamusyoung.com/twentysidedtale/?p=16058 (vaadatud 12.05.2016)

[32] Peter Halasz. (2007, märts) WikiMedia: Holdridge Life Zone Classification Scheme. [Võrgumaterjal]. https://commons.wikimedia.org/wiki/File:Lifezones_Pengo.svg (vaadatud 12.05.2016)

[33] Jean-Marie Favreau. (2006, oktoober) WikiMedia: Marching Cubes. [Võrgumaterjal]. https://commons.wikimedia.org/wiki/File:MarchingCubes.svg (vaadatud 12.05.2016)

Lisad

I. Terminid

Bilineaarne interpoleerimine Interpoleerimise täiendus kahekohaliste funktsioonide interpoleerimiseks kahemõõtmelise korrapärase ruudustiku peal.	Bilinear interpolation An extension of linear interpolation for interpolating functions of two variables on a rectilinear 2D grid.
Bioom Makroökosüsteemid, mis on määratud nende dominantse taimestiku, kliima ja seal elavate organismite kaudu.	Biomes The world's areas classified according to the predominant vegetation, climate and organisms living in it [24].
Bitimask Kahendsüsteemi väärtuste muster, mida rakendatatakse mingile väärtusele, kasutades bitioperatsioone [27].	Bit mask A pattern of binary values on which bitwise operations can be used to extract bit specific booleans.
Bitioperatsioon Tehted, mida tehakse individuaalsete bittide tasandil. Bitioperatsioonid on näiteks täiend, konjunktsioon.	Bitwise operation Operations, which are done at the level of individual bits. For example bitwise operations are NOT and AND.
Cartesiuse koordinaatsüsteem Koordinaatsüsteem, mis määrab üheselt iga punkti asukoha ruumis numbriliste koordinaatidega. Samuti tuntud kui ka Descartes'i koordinaatsüsteem.	Cartesian coordinate system An coordinate system that uniquely assigns numeric coordinates for each point in space. Also known as Descartes's coordinate system.

Eukleidiline ruum Lõplikumõõtmeline vektorruum üle reaalarvude korpuse, kus vektoritel on defineeritud skalaarkorrutis [20].	Euclidean space Finite dimensional vector space over a real numbers field, in which dot product is defined.
Hüperkuup Kolmemõõtmelise kuubi üldistus n-dimensiooni. Näiteks 1D ruumis on hüperkuubiks lõik, 2D ruumis ruut.	Hypercube A generalization of a 3-dimensional cube to n-dimensions [28]. For example 1D hyper-cube is a line, a 2D hypercube is a square.
Interpoleerimine Funktsiooni väärtuse hindamine piir-konnas, mille mõnes punktis on funk-tsiooni või selle tuletise väärtused teada [20].	Interpolation Estimating the value of a function in an area in which some of the data points or their derivatives are known.
Kumer hulk Vektorruumi hulk H, mille iga kahe punkti $x, y \in H$ korral sisaldab H neid punkte ühendavat lõikku [20].	Convex set A subset H of vector space which contains all the segments that can be constructed from any 2 points in the set H.
Kumer kate Vektorruumi fikseeritud alamhulka H sisaldav minimaalne kumer hulk [20].	Convex hull Minimal convex set that contains a vector space subset H.
Kõrguskaart Maatriks, mida tavaliselt kujutatakse halltoonides rasterpildina. Iga maatriksi element kirjeldab maastiku või muu vaadeldava objekti kõrgust antud reas ja veerus.	Heightmap A matrix that is usually portrayed as a raster image. Each matrix element describes the terrain's height at the given row and column position.

Simpleks n-mõõtmelise vektorruumi niisuguse $(n+1)$-punktilise hulga ehk simpleksi tippude hulga kumer kate, mille kõik tipud ei paikne mingis $(n-1)$-mõõtmelises alamruumis [20].	Simplex A convex hull of a set of $(n+1)$ elements in a n-dimensional vector space such that all the elements are not located in a $(n-1)$-dimensional subspace.
Sujuvus Funktsioon on sujuvusklassiga C^n, kui funktsioonil on $\forall i \in \{0, 1, \dots, n\}$ korral pidev i-ndat järku tuletis. Antud töö kontekstis nimetatakse funktsiooni sujuvaks, kui vähemalt selle esimest järku tuletis on pidev.	Smoothness A function is said to be of smoothness class C^n when the function has continuous i-th derivative for $\forall i \in \{0, 1, \dots, n\}$ [29]. In current context a function is said to be smooth if it has at least continuous 1st derivative.
Sõnastik Andmestruktuur, mis koosneb võtmetest ja väärtustest. Igale võtmele vastab ülimalt üks väärtus. On efektiivne viis võtmete kaudu väärtuste leidmiseks.	Dictionary A data structure which consists of keys and values. Each key corresponds to a maximum of one value. It is an effective way to organize values and access them by keys.
Tekstuur Arvutigraafikas mudelitele rakendatud pilt, mis annab mudeli pinnale tekstureeritud pinna [27].	Texture In computer graphics an image applied to virtual models to give the impression of a real surface.
Tessellatsioon Tasandi tükkideks jaotamine väiksemate geomeetriliste kujundite abil nii, et terve tasand on kujundite poolt kaetud ning tükkide vahel ei ole kattuvusi ega auke.	Tessellation A tiling of a plane consisting of a collection of smaller shapes that fills the surface with no overlaps and no gaps between the tiles [17].

Voksel Kolmemõõtmelise regulaarse ruudustiku element. Analoogiliselt pikslile rastergraafikas on voksel kolmemõõtmelise ruumi minimaalne eristavate omadustega element.	Voxel An element of a regular three-dimensional grid. Like a pixel in raster graphics a voxel is the minimal three dimensional space element with distinguishable properties [10].
Ühikvektor Vektor, mille pikkus on 1 [20].	Unit vector A vector of length 1.

II. Litsents

Lihtlitsents lõputöö reprodutseerimiseks ja lõputöö üldsusele kättesaadavaks tegemiseks

Mina, **Andreas Sepp**,

1. annan Tartu Ülikoolile tasuta loa (lihtlitsentsi) enda loodud teose
   Protseduuriline lõpmatu maastiku 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, **12.05.2016**