Ilari Paananen

Kolmiulotteisen maaston generointi peleissä

Tietotekniikan kandidaatintutkielma

6. kesäkuuta 2015

Jyväskylän yliopisto

Tietotekniikan laitos

Tekijä: Ilari Paananen

Yhteystiedot: ilari.k.paananen@student.jyu.fi

Ohjaaja: Tytti Saksa

Työn nimi: Kolmiulotteisen maaston generointi peleissä

Title in English: Three-Dimensional Terrain Generation in Games

Työ: Kandidaatintutkielma

Sivumäärä: 24+0

Tiivistelmä: Monet tietokonepelit toteuttavat kolmiulotteisen maaston käyttäen korkeuskarttoja. Korkeuskartan luominen käsin on kuitenkin työlästä. Pelinkehityksen helpottamiseksi korkeuskarttoja voidaan generoida osittain tai täysin proseduraalisesti. Tässä tutkielmassa käsitellään erilaisia korkeuskartan generointimenetelmiä. Tutkielmassa kuvaillaan myös erilaisia eroosio-algoritmeja, joilla generoiduista maastoista saadaan realistisempia. Lopuksi pohditaan, kuinka esitettyjä menetelmiä voidaan hyödyntää peleissä.

Avainsanat: maasto, generointi, proseduraalinen, eroosio, pelit

Abstract: Many computer games implement three-dimensional terrain using heightmaps, but creating a heightmap by hand is tedious. To make game development easier heightmaps can be generated partially or completely with procedural methods. In this paper different kinds of heightmap generation methods are discussed. The paper also describes various erosion algorithms that add realism to the generated terrains. Finally, it is considered how the introduced methods can be utilized in games.

Keywords: terrain, generation, procedural, erosion, games

Kuviot

Sisältö

1 Johdanto

Interaktiivisissa kolmiulotteisissa sovelluksissa, erityisesti tietokonepeleissä, sisällön määrä ja yksityiskohtaisuus ovat lisääntyneet vuosien varrella paljon. Pelien sisällön luonti käsin vaatiikin runsaasti aikaa ja rahaa. Tämän vuoksi luomisprosessia pyritään nopeuttamaan ja automatisoimaan käyttämällä proseduraalisia menetelmiä. Jotkin pelit käyttävät proseduraalista sisällön luontia myös uudelleenpeluuarvon lisäämiseksi.

Proseduraalista generointia voidaan hyödyntää monenlaisen sisällön luontiin. Esimerkiksi tekstuurit, kolmiulotteiset mallit, ympäristön yksityiskohtien sijoittelu ja jopa kokonaiset pelikentät voivat olla proseduraalisesti generoituja. Usein myös pelin maasto generoidaan ainakin pääpiirteissään proseduraalisesti. Tässä tutkielmassa keskitytäänkin erilaisiin algoritmeihin, joita voidaan hyödyntää tietokonepelien kolmiulotteisten maastojen luonnissa.

Tutkielman luvussa kaksi selitetään, kuinka kolmiulotteinen maasto voidaan toteuttaa korkeuskartan avulla. Luvussa kolme käydään läpi usein käytettyjä menetelmiä kolmiulotteisen maaston korkeuskartan generoimiseksi sekä kuvaillaan menetelmiä, jotka antavat käyttäjälle enemmän valtaa maaston muotoihin. Neljännessä luvussa käsitellään korkeuskarttoihin sovellettavia eroosio-algoritmeja. Viidennessä luvussa pohditaan esitettyjen algoritmien ja menetelmien soveltuvuutta peleihin. Luku kuusi on tutkielman yhteenveto.

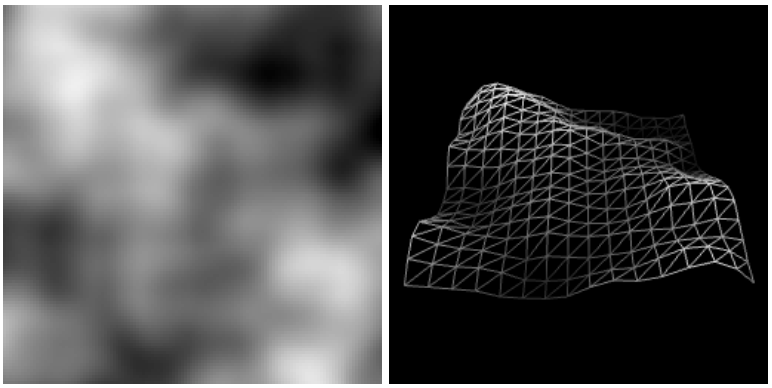
Kaikki tutkielmassa esiintyvät kuvat ovat kirjoittajan tekemiä. Kuvioissa 1, 3, 6, 8 ja 9 esitetyt kolmiulotteiset maastot on renderöity OpenGL-grafiikkarajapintaa käyttäen. Kuvion 1 korkeuskartta on luotu GIMP-kuvankäsittelyohjelmalla. Muiden kuvioiden korkeuskartat on luotu algoritmeilla, joita kyseiset kuviot havainnollistavat.

2 Kolmiulotteinen maasto ja korkeuskartta

Kolmiulotteiset maastot toteutetaan useimmiten käyttäen korkeuskarttaa (engl. *heightmap*) (Ong ym. 2005). Maasto voidaan toteuttaa myös vektorikenttänä (engl. *vector field*) tai vokseleilla (engl. *voxel*). Vektorikenttään tai vokseleihin perustuvassa maastossa voi olla ulkonemia ja luolia. Näiden esittäminen korkeuskartalla ei ole mahdollista (Ong ym. 2005). Tästä rajoituksesta huolimatta korkeuskartat ovat suosittuja, koska ne voidaan esittää tiiviisti ja niiden renderöintiin on useita tehokkaita algoritmeja (De Carpentier ja Bidarra 2009).

Maaston korkeuskartta voidaan ajatella kahden muuttujan funktioksi, joka kuvaa kaksiulotteisen tason pisteet maaston korkeusarvoiksi kyseisissä pisteissä. Korkeuskarttana käytetään yleensä kaksiulotteista taulukkoa, jonka alkiot sisältävät maaston korkeusarvoja. Korkeuskartta voidaan esittää harmaasävykuvana, jossa valkoinen väri tarkoittaa korkeinta ja musta matalinta mahdollista korkeusarvoa. (Ong ym. 2005)

Maasto voidaan visualisoida renderöimällä se kolmioverkkona (engl. *triangle mesh*). Olettaen, että visualisointiin käytetyssä koordinaatistossa maaston korkeusvaihtelu on z-akselin suuntaista, muodostaa kolmioverkko tasaisen ruudukon xy-tasossa. Kolmioverkon pisteiden z-komponenttien arvot puolestaan luetaan korkeuskartasta. Kuvio 1 havainnollistaa, kuinka korkeuskartta voidaan renderöidä kolmiulotteisena.



Kuvio 1. Korkeuskartta (vas.) on renderöity kolmioverkkona (oik.).

3 Korkeuskartan generointi

Maaston korkeuskartan generointiin on kehitetty paljon erilaisia algoritmeja. Monet näistä algoritmeista on suunniteltu alunperin proseduraalisten tekstuurien generointiin. Koska luonnollisilla tekstuureilla on samankaltaisia ominaisuuksia kuin maastojen korkeuskartoilla, soveltuvat samat algoritmit usein myös maastojen korkeuskarttoihin. (Togelius, Shaker ja Nelson 2015)

Perinteisesti käyttäjä ei ole pystynyt määrittelemään tarkasti, millainen generoitavasta maastosta tulee. Monet algoritmit mahdollistavat vain muutamien parametrien, kuten korkeusvaihtelun ja karkeuden, säätämisen. Tämän ongelman ratkaisemiseksi on kehitetty menetelmiä, jotka antavat käyttäjälle enemmän mahdollisuuksia vaikuttaa korkeuskarttaan.

Seuraavaksi käsitellään eräitä maaston korkeuskartan generoinnissa yleisimmin käytettyjä algoritmeja. Näistä tarkemmin esitellään keskipisteen siirto ja timantti-neliö-algoritmit. Lisäksi kuvaillaan lyhyesti joitain korkeuskartan luomiseen käytettäviä menetelmiä ja algoritmeja, jotka antavat käyttäjälle enemmän valtaa maaston yleispiirteisiin.

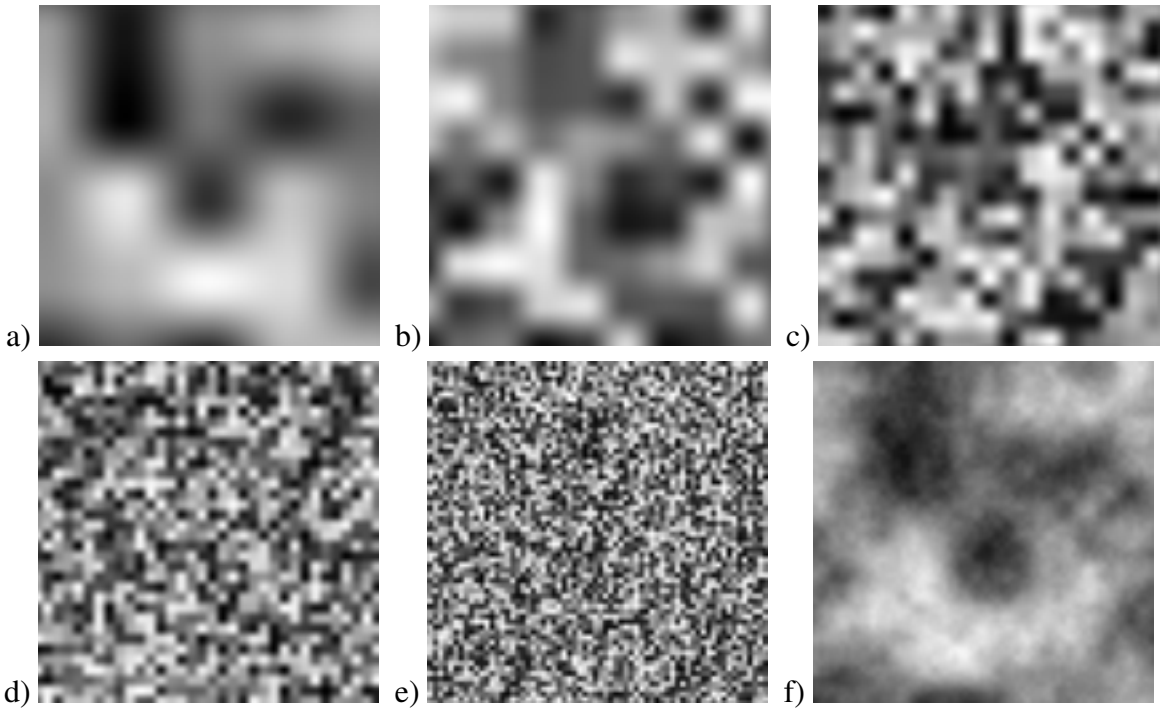
3.1 Fraktaali maasto ja erilaisia kohinoita

Oikeassa maailmassa maasto on luonteeltaan fraktaali. Tämä tarkoittaa sitä, että suurella mittakaavalla havaittavat maaston piirteet toistuvat uudelleen ja uudelleen samankaltaisina, mutta pienemmässä mittakaavassa, sitä mukaa kun maastoa siirrytään tarkastelemaan lähempää. (Togelius, Shaker ja Nelson 2015)

Maaston generoinnissa käytetään usein erilaisia kohinoita (engl. *noise*). Monet kohinat eivät ole itsessään fraktaaleja ja niiden käyttäminen suoraan korkeuskarttana ei tuota kovin uskottavaa maastoa. Kohinaan voidaan kuitenkin saada fraktaaleja ominaisuuksia soveltamalla fraktionaalista Brownin liikettä (engl. *fractional Brownian motion*). Fraktionaalisen Brownin liikkeen idea on summata monta tasoa kohinaa yhteen. Joka tasolla kohinan taajuutta kasvatetaan ja voimakkuutta pienennetään edelliseen tasoon nähden. Näin saatava kohina sisältää usean eri mittakaavan yksityiskohtia. Menetelmän heikkoutena on se, että kohinaan

vaadittava laskenta kertautuu summattavien tasojen määrällä. (Archer 2011)

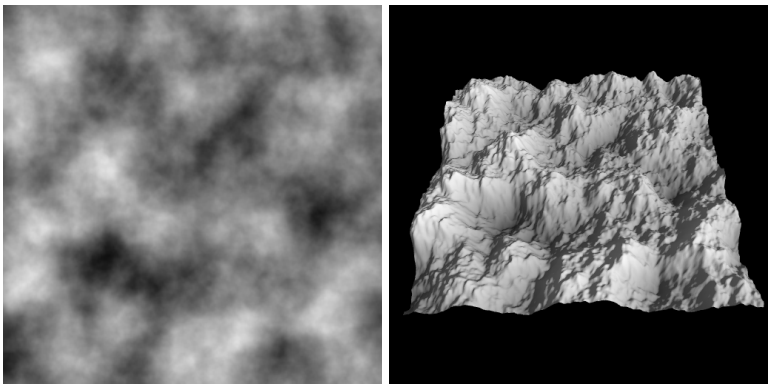
Kenties yksinkertaisin kohina on arvokohina (engl. *value noise*). Arvokohina saadaan aikaan luomalla tasaisin välein satunnaisia arvoja, joiden väliin sijoittuvien pisteiden arvot lasketaan interpoloimalla (Olsen 2004). Korkeuskarttaa varten arvokohina on helppo toteuttaa kaksiulotteisena taulukkona, jolloin interpoloinnissa käytetään haluttua pistettä lähinnä olevia neljää taulukon arvoa (Archer 2011). Kuutiollinen interpolointi tuottaa hyviä korkeuskarttoja, mutta on huomattavasti hitaampi kuin lineaarinen interpolointi (Archer 2011; Olsen 2004). Kuvio 2 havainnollistaa fraktionaalista Brownin liikettä arvokohinalla.



Kuvio 2. Arvokohina ja fraktionaalinen Brownin liike. Kuvissa (a) – (e) on kuutiollisesti interpoloitua arvokohinaa eri taajuuksilla ja kuvassa f nämä kohinatasot on yhdistetty eri voimakkuuksilla. Näin muodostetulla korkeuskartalla on fraktaaleja piirteitä.

Eräitä tärkeimpiä kohinoita ovat gradienttikohinat (engl. *gradient noise*). Gradienttikohinoiden toteutus on melko monimutkaista, mutta niillä on kuitenkin etunsa. Korkeusarvojen sijaan gradienttikohinoissa interpoloidaan korkeusarvojen muutosnopeutta, mikä lisää korkeuskartan pehmeyttä ja myös luonnollisuutta, kun huippukohdat ja syvänteet voivat sijoittua vapaammin (Togelius, Shaker ja Nelson 2015).

Perlin-kohina lienee tunnetuin ja eniten käytetty gradienttikohina. Sillä voidaan luoda moniulotteista kohinaa, mutta jos ulottuvuuksia halutaan enemmän kuin kolme, on Perlin-kohina melko tehoton (Archer 2011). Sen sijaan Simplex-gradienttikohinalle ulottuvuuksien määrä ei ole yhtä suuri ongelma. Archer (2011) kuvailee tarkemmin näiden kahden gradienttikohinan toteutusta. Kuviossa 3 esitetty korkeuskartta on generoitu Perlin-kohinalla ja fraktionaalisella Brownin liikkeellä.



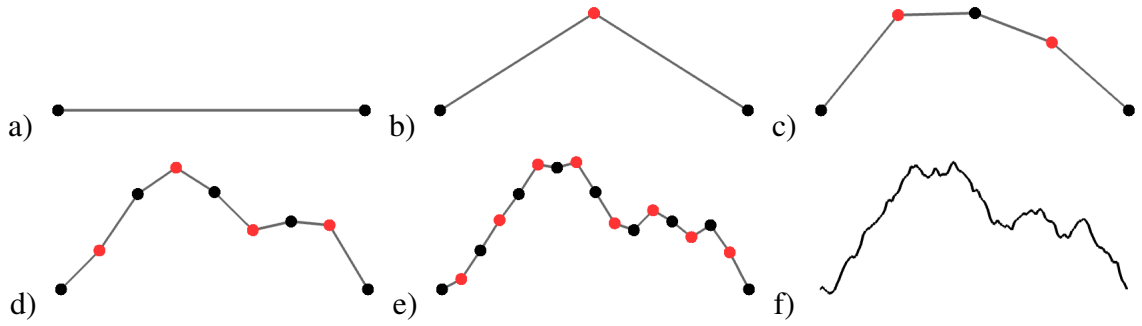
Kuvio 3. Perlin-kohinalla ja fraktionaalisella Brownin liikkeellä generoitu maasto.

Eräs mielenkiintoinen kohina on Worley-kohina. Siinä tasolle ripotellaan satunnaisia pisteitä ja valitaan $n \in \{1,2,3,...\}$. Jos $n = 1$, niin korkeuskartan yksittäisen pisteen arvo on pisteen etäisyys lähimpään satunnaispisteeseen. Jos $n = 2$, saadaan pisteen arvo etäisyytenä toiseksi lähimpään satunnaispisteeseen, jne. Valitsemalla eri n saadaan todella erilaisia korkeuskarttoja. Yhdistämällä samoilla satunnaispisteillä, mutta eri n :n arvoilla luotuja Worley-kohinoita saadaan myös erilaisia ja eri tilanteisiin sopivia korkeuskarttoja. Worley-kohina ei itsessään tuota kovin hyvää tai uskottavaa maastoa, mutta se saattaa olla erittäin hyödyllinen muihin kohinoihin yhdistettynä (Archer 2011).

3.2 Keskipisteen siirto ja timantti-neliö

Keskipisteen siirto -algoritmi (engl. *midpoint displacement algorithm*) on yksinkertainen menetelmä kohinan luomiseksi. Kuvio 4 esittää algoritmin toimintaa sovellettuna janaan. Algoritmi aloitetaan yhdellä janalla, jonka päätepisteiden arvot alustetaan satunnaisluvuilla tai haluttaessa määrittämällä arvot itse. Ensimmäisessä iteraatiossa jana jaetaan keskeltä kahdeksi janaksi. Tämä jakopiste on alkuperäisen janan keskipiste. Sen arvo saadaan pääte-

pisteiden arvojen keskiarvona, johon lisätään satunnainen siirto. Seuraavassa iteraatiossa sama toistetaan uusille janoille. Näin jatketaan, kunnes on suoritettu haluttu määrä iteraatioita. Jokaisella iteraatiolla keskipisteen satunnaisen siirron vaihteluväliä pienennetään. (Archer 2011)



Kuvio 4. Keskipisteen siirto -algoritmin soveltaminen janaan: (a) algoritmin alkutilanne, jossa janan päätepisteiden arvot on alustettu samalla arvolla, (b) – (e) algoritmin iteraatiot 1 – 4 sekä (f) algoritmin tulos seitsemän iteraation jälkeen.

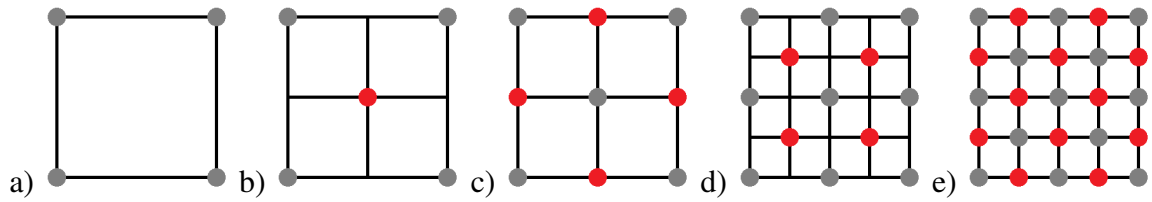
Keskipisteen siirto voidaan yleistää korkeuskartan luomiseen. Tällöin algoritmi aloitetaan janan sijasta neliöllä. Neliön sivujen keskipisteiden arvot määritetään kuten janan tapauksessa. Lisäksi neliön keskipisteelle määritetään arvo, jolloin keskiarvo, johon satunnainen siirto lisätään, lasketaan kaikista neliön kulmapisteistä. Näin muodostuu neljä uutta neliötä, joille voidaan toistaa sama. Iterointia jatketaan, kunnes haluttu tarkkuus on saavutettu. (Archer 2011)

Keskipisteen siirrolla luodussa korkeuskartassa on havaittavissa ei-toivottuja säännönmukaisuuksia. Tämä johtuu siitä, että joidenkin pisteiden arvot lasketaan kahden pisteen perusteella ja toisten pisteiden arvot neljän pisteen perusteella. Ongelmasta päästään eroon käyttämällä timantti-neliö-algoritmia. Algoritmin vaiheita on havainnollistettu kuviossa 5. (Archer 2011)

Timantti-neliön iteraatiossa määritetään ensin neliöiden keskipisteiden arvot käyttäen neljää kulmapistettä, kuten keskipisteen siirrossa. Tämä on algoritmin neliö-vaihe. Seuraavaksi määritetään neliöiden sivujen keskipisteet käyttäen myös neljää pistettä, jotka ovat sivun yhdistämät kaksi kulmapistettä ja sivun kahden viereisen neliön keskipisteet. Voidaan ajatella, että nämä pisteet muodostavat timantti-kuvion, minkä vuoksi vaihetta kutsutaan timantti-

vaiheeksi. Tietenkään sivuilla, jotka muodostavat korkeuskartan reunat, ei ole kahta viereistä neliötä. Tällöin keskipisteen arvo voidaan määrittää kolmen pisteen avulla. Kuviossa 6 on esitetty timantti-neliö-algoritmilla luotu maasto, jonka reunat on määritetty kyseisellä tavalla. (Archer 2011)

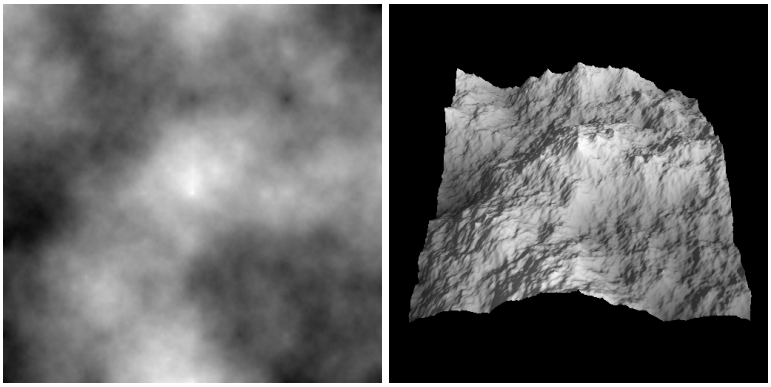
Toinen tapa on varmistaa, että jokaisella reunalla olevalla pisteellä on sama arvo kuin vastakkaisen reunan pisteellä, ja käyttää reunan pisteen arvoa määritettäessä neljäntenä pisteenä korkeuskartan vastakkaisella reunalla olevan neliön keskipistettä. Näin luodun korkeuskartan reunat sopivat yhteen saumattomasti, mikä on joissain tapauksissa haluttu ominaisuus. (Archer 2011)



Kuvio 5. Timantti-neliö-algoritmin kaksi ensimmäistä iteraatiota. Kuvassa (a) on esitetty algoritmin alkutilanne, eli neliö, jonka kulmapisteet on alustettu. Kuva (b) esittää ensimmäisen iteraation neliö-vaihetta, jossa määritetään neliön keskipisteen arvo, ja kuva (c) ensimmäisen iteraation timantti-vaihetta, jossa määritetään neliön sivujen keskipisteiden arvot. Kuvat (d) ja (e) ovat toisen iteraation neliö- ja timantti-vaiheet.

Keskipisteen siirrolla tai timantti-neliöllä luotaessa korkeuskartasta tulee neliö, jonka sivun pituus riippuu suoritettujen iteraatioiden määrästä. Kun suoritetaan n iteraatiota, tulee korkeuskartan sivun pituudeksi $2^n + 1$ (Archer 2011). Koska molemmissa algoritmeissa korkeuskartan pisteiden arvot ovat toisistaan riippuvaisia, tulee koko korkeuskartta luoda kerralla. Tästä syystä korkeuskartan koko ja muistin tarve on tiedettävä etukäteen.

Keskipisteen siirron ja timantti-neliön suurin hyöty on se, että niillä luoduilla korkeuskartoilla on suoraan fraktaalin maaston ominaisuuksia. Jokaiselle korkeuskartan pisteelle tarvitsee laskea arvo vain kerran, minkä vuoksi nämä algoritmit ovat nopeita verrattuna fraktionaalista Brownin liikettä hyödyntäviin menetelmiin. (Archer 2011)



Kuvio 6. Timantti-neliö-algoritmilla generoitu maasto.

3.3 Enemmän valtaa käyttäjälle

Edellä esitetyt maaston korkeuskartan generointiin käytettävät menetelmät luovat täysin satunnaisia maastoja. Usein kuitenkin käyttäjä, esimerkiksi pelin kenttäsuunnittelija, haluaa tietynlaisen maaston. Tämän vuoksi on kehitetty algoritmeja, jotka antavat käyttäjälle enemmän valtaa luotavan maaston piirteisiin.

De Carpentier ja Bidarra (2009) esittävät proseduraaliset siveltimet, joilla kenttäsuunnittelija voi interaktiivisesti maalata maastoon proseduraalisesti generoituja korkeusarvoja. Tällä tavoin suunnittelija pystyy määrittämään, millainen maastosta tulee, menettämättä algoritmisesti generoituja yksityiskohtia. De Carpentier ja Bidarra hyödyntävät toteutuksessaan näytönohjainta, jotta maaston maalaaminen voidaan tehdä reaaliaikaisesti.

Kamal ja Uddin (2007) esittävät algoritmin, joka luo vuoren siten, että sen korkeus, huipun sijainti ja juuren leveys ovat käyttäjän määriteltävissä. He myös havainnollistavat, että haluttaessa algoritmin lopputulokseen voidaan vaikuttaa enemmänkin. Oikein säädettynä algoritmi saadaan esimerkiksi luomaan kraattereita ja tulivuorimaisia muodostelmia.

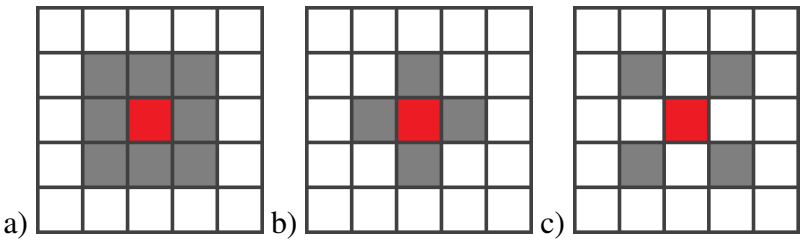
Artikkelissaan Hnaidi ym. (2010) kuvailevat menetelmän, joka antaa kättäjän määritellä parametrisoiduilla käyrillä generoitavan maaston yleispiirteet, kuten vuorten harjanteet, jokien uomat ja jyrkänteet. Belhadj (2007) puolestaan esittää keskipisteen siirtoon perustuvan algoritmin, joka luo maaston korkeuskartan mukaillen valmiiksi määritettyjä korkeusarvoja. Algoritmi on suunniteltu ensisijaisesti digitaalisten korkeuskarttojen (engl. *digital elevation map*) rekonstruointiin.

Smelik ym. (2010) esittävät maaston generointimenetelmän, jolla käyttäjä voi määritellä helposti ja nopeasti sekä yllättävän tarkasti, millaisen maaston haluaa. Menetelmässä maasto on jaettu pienempiin osiin ja käyttäjä määrittää, millaista maastoa kuhunkin osaan luodaan. Valmiissa paletissa on eri maastotyyppejä, joilla on yksilölliset värit. Jokin väri voi vastata vaikkapa korkeaa vuoristoa, toinen vuoren aluskukkuloita ja kolmas mäkimaastoa. Näillä väreillä käyttäjä piirtää haluamansa maaston. Piirroksen pohjalta erilaiset algoritmit generoivat yksityiskohtaisen maaston. Menetelmä hyödyntää esimerkiksi Perlin-kohinaa lopullista maastoa luodessaan.

4 Eroosion mallinnus

Proseduraalisesti generoitu maasto on usein melko karkea tai muuten luonnoton. Oikeassa maailmassa maasto on luonnonvoimien muovaama. Luonnollisten maastojen ulkonäköön merkittävästi vaikuttava tekijä on eroosio. Eroosion mallinnusta on tutkittu todenmukaisempien ja mielenkiintoisempien maastojen luomiseksi. Korkeuskarttojen jälkikäsittelyyn on kehitetty erilaisia eroosio-algoritmeja, jotka lisäävät maaston luonnollisuutta (Archer 2011).

Kaksi yleisimmin käytettyä eroosio-algoritmia ovat lämpöeroosio (engl. *thermal erosion*) ja vesieroosio (engl. *hydraulic erosion*). Lisäksi lämpöeroosiosta on vesieroosioon verrattavia tuloksia tuottava muunnelma. Nämä algoritmit ovat soluautomaatteja (engl. *cellular automata*), jotka käsittelevät iteratiivisesti korkeuskartan alkioita eli soluja (Olsen 2004). Solujen käsittelyssä käytetään kulloinkin vuorossa olevan solun lisäksi sen naapurisoluja. Mahdollisia naapurustoja ovat esimerkiksi Mooren naapurusto, von Neumannin naapurusto ja kierretty von Neumannin naapurusto (Archer 2011). Nämä naapurustot on esitetty kuviossa 7. Mooren naapurustoa käytettäessä eroosio-algoritmi on hitaampi kuin von Neumannin naapurustoilla, mutta tulos on parempi (Archer 2011).



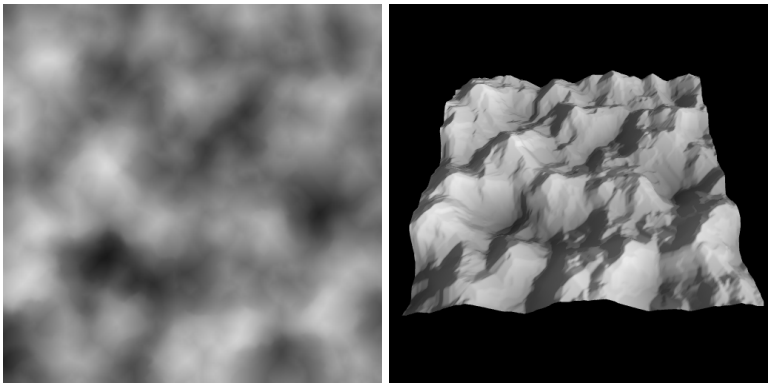
Kuvio 7. Mahdollisia soluautomaateissa käytettäviä naapurustoja: (a) Mooren naapurusto, (b) von Neumannin naapurusto ja (c) kierretty von Neumannin naapurusto.

4.1 Lämpöeroosio

Lämpöeroosio simuloi maastosta irtoavan aineksen kulkeutumista jyrkiltä alueilta alemmas (Musgrave, Kolb ja Mace 1989; Olsen 2004). Tämä toteutetaan niin, että jokaisen korkeuskartan solun korkeusarvoa verrataan sen naapurisolujen korkeusarvoihin. Mikäli solu on korkeammalla kuin naapurisolu ja näiden välinen korkeusero on enemmän kuin ennalta määri-

tetty suurin sallittu korkeusero, irrotetaan solusta maa-ainesta, eli vähennetään sen korkeusarvoa, ja lisätään saman verran naapurisoluun (Archer 2011). Algoritmin vaikutus alkaa näkyä vasta usean iteraation jälkeen.

Lämpöeroosio voidaan toteuttaa niin, että ylimääräistä muistia ei tarvita (Olsen 2004). Algoritmi on myös hyvin yksinkertainen ja nopea, mutta sen tulokset eivät ole kovin uskottavia (Archer 2011). Kuviossa 8 lämpöeroosiota on sovellettu kuvion 3 Perlin-maastoon.



Kuvio 8. Lämpöeroosion kuluttama maasto. Algoritmin vaikutusta on liioiteltu, jotta kuluminen näkyy paremmin.

4.2 Vesieroosio

Vesieroosion mallinnus perustuu liikkuvan veden aiheuttamaan maaperän kulumiseen (Musgrave, Kolb ja Mace 1989). Sadevesi kulkeutuu maastossa korkeammilta alueilta alemmas ja irrottaa samalla mukaansa maa-ainesta. Olsen (2004) jakaa vesieroosiota simuloivan algoritmin neljään vaiheeseen.

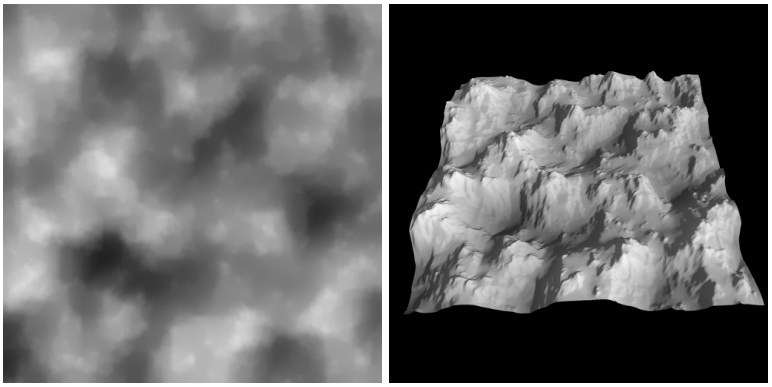
1. Jokaista korkeuskartan solua kohden lisätään vettä.
2. Vesi irrottaa solusta maa-ainesta.
3. Vettä ja irronnutta maa-ainesta siirretään matalammalla oleviin naapurisoluihin.
4. Osa vedestä haihtuu ja veden kuljettamaa maa-ainesta palautuu takaisin maastoon.

Ensimmäinen vaihe mallintaa veden satamista. Vettä voidaan lisätä tasaisesti jokaiseen soluun (Olsen 2004) tai lisättävän veden määrä voi riippua solun korkeusarvosta, kuten Musgrave, Kolb ja Mace (1989) esittävät. He perustelevat tätä sillä, että luonnossa sateilla on

taipumus esiintyä useammin vuoristossa kuin alavilla mailla.

Kolmannessa vaiheessa matalammalla olevat naapurisolut määritetään samaan tapaan kuin lämpöeroosiossa. Korkeuseroja laskettaessa on kuitenkin huomioitava soluissa olevan veden määrä. Solun kokonaiskorkeus on siis solun korkeusarvon ja sen sisältämän veden määrän summa. (Olsen 2004)

Näitä neljää vaihetta on toistettava useita kertoja ennen kuin eroosion vaikutus alkaa näkyä. Vesieroosio-algoritmi onkin todella hidas ja sen toteutus on hankalaa (Archer 2011). Algoritmi tarvitsee korkeuskartan lisäksi jokaista korkeuskartan solua varten veden määrän ja veden mukanaan kuljettaman maa-aineksen määrän (Archer 2011; Olsen 2004). Mallia voidaan yksinkertaistaa olettamalla, että jokaista vesiyksikköä kohden kulkeutuu saman verran maa-ainesta, jolloin muistin tarve pienenee (Olsen 2004). Silti algoritmi vaatii paljon muistia. Heikkouksistaan huolimatta vesieroosion mallinnuksella maastosta voidaan saada paljon luonnollisempi ja uskottavampi (Archer 2011). Kuvio 9 esittää vesieroosion vaikutusta kuvion 3 Perlin-maastoon.



Kuvio 9. Vesieroosion kuluttama maasto. Algoritmin vaikutusta on liioiteltu, jotta kuluminen näkyy paremmin.

Vesieroosio-algoritmin soveltaminen suureen ja yksityiskohtaiseen maastoon voi olla hyvin hidasta. Toisaalta nykyaikaiset näytönohjaimet kykenevät tehokkaaseen rinnakkaislaskentaan. Anh, Sourin ja Aswani (2007) esittävät menetelmän, joka hyödyntää näytönohjaimen laskentatehoa vesieroosion simuloinnin nopeuttamiseksi. Št'ava ym. (2008) puolestaan esittävät, että näytönohjaimella suoritettavaa vesieroosio-algoritmia voidaan käyttää maaston interaktiiviseen muokkaukseen.

4.3 Käänteinen lämpöeroosio

Vesieroosion mallinnuksella on taipumus tasoittaa jo ennestään tasaisia alueita ja tehdä rinteistä jyrkempiä (Olsen 2004). Sitä vastoin lämpöeroosio vaikuttaa juuri käänteisellä tavalla: tasaiset alueet muuttuvat rinteiksi ja jyrkänteet loivenevat. Olsen (2004) ehdottaa lämpöeroosiosta muunnettua algoritmia, jonka vaikutus maastoon on samankaltainen kuin vesieroosiolla. Tämän käänteisen lämpöeroosion ainoa ero tavalliseen lämpöeroosioon on ehto, jonka perusteella maa-ainesta siirretään solusta toiseen.

Tavallisessa lämpöeroosiossa maata siirretään matalammalla olevaan naapurisoluun vain, jos solujen korkeusero on tarpeeksi suuri, kun taas käänteisessä lämpöeroosiossa maa-ainesta siirretään, jos korkeusero on tarpeeksi pieni (Olsen 2004). Käänteinen lämpöeroosio on yhtä yksinkertainen ja tehokas kuin tavallinen lämpöeroosio ja sen vaikutus maastoon on verrattavissa vesieroosioon, mutta lopputulos ei ole yhtä vakuuttava (Archer 2011).

5 Soveltaminen tietokonepeleissä ja niiden kehityksessä

Korkeuskarttoihin perustuvia maastoja käytetään monenlaisissa peleissä, kuten useissa reaaliaikaisissa ja vuoropohjaisissa strategiapeleissä, räiskintä- ja toimintapeleissä sekä erilaisissa seikkailu- ja roolipeleissä. Kehitettävän pelin genre, visuaalinen tyyli ja pelimekaniikat asettavat maastolle erilaisia vaatimuksia, jotka olisi otettava huomioon valittaessa maaston luomiseen käytettäviä algoritmeja ja menetelmiä.

Enimmäkseen ylhäältäpäin kuvatuissa peleissä, kuten strategiapeleissä, maaston suuret korkeusvaihtelut saattavat olla hämääviä. Tällaisissa peleissä maaston äärimmäinen realistisuus ei yleensä ole kovin tärkeää. Sen sijaan maaston realistisuus ja suuret korkeusvaihtelut ovat usein toivottuja peleissä, joissa pelaaja voi katsella vapaasti eri suuntiin, kuten monissa ensimmäisestä persoonasta kuvatuissa peleissä. Esimerkiksi suuret vuoret voivat nopeuttaa peliä estämällä takseen jäävien kappaleiden näkymisen, jolloin ne voidaan jättää piirtämättä.

Korkeusvaihteluilla voidaan myös estää ja suunnata pelaajan liikkumista pelimaailmassa tai luoda mielenkiintoa herättäviä alueita tutkittavaksi. Joskus olennainen tekijä on pelimaailman esteettisyys, jolloin maastolla on tärkeä rooli. Suuret etäisyydet, mahtavat vuoret ja laajat lakeudet luovat vaikuttavia maisemia. Näitä ja muita vastaavia seikkoja voidaan hyödyntää pelin maastoa suunniteltaessa.

5.1 Kenttäeditorit

Yleensä tietokonepeleissä on kenttiä, jotka on suunniteltu ja luotu pelinkehityksen aikana. Tämä on lähes välttämätöntä esimerkiksi juonellisissa peleissä, joiden tapahtumat sijoittuvat tiettyihin paikkoihin. Myös Internet-yhteyden välityksellä samassa ympäristössä muiden pelaajien kanssa pelattavissa peleissä kentät ovat usein valmiiksi luotuja. Pelien kenttäeditorit ovat kenttien luontiin tarkoitettuja erillisiä ohjelmia, jotka yleensä mahdollistavat myös maaston muokkaamisen.

Tavallisesti pelien kenttäeditoreissa on erilaisia työkaluja, joilla käyttäjä voi esimerkiksi nostaa ja laskea maaston pisteitä. Näin suunnittelija voi luoda juuri sellaisen maaston, kuin ha-

luaa. Tällainen työskentely on kuitenkin erittäin hidasta ja yksityiskohtien lisääminen on työlästä. Kenttäsuunnittelijan työtä voidaan helpottaa proseduraalisilla maaston generointimenetelmillä. Eri menetelmät soveltuvat maaston luonnin eri vaiheisiin ja tarpeisiin.

Maaston luonti voitaisiin aloittaa generoimalla alustava maasto satunnaisesti käyttäen kohinoita. Muokkausta on helpompi jatkaa, kun maasto sisältää jo erilaisia muotoja ja yksityiskohtia. Alustava maasto voitaisiin luoda myös algoritmilla, jonka Belhadj (2007) esittää. Tällöin kenttäsuunnittelija saisi samalla hahmotella maaston tärkeimmät muodot.

Seuraavaksi kenttäsuunnittelija voisi luoda maastoon haluamiaan suurempia muotoja parametrisoiduilla käyrillä, kuten Hnaidi ym. (2010) esittävät, tai käyttäen algoritmia, jonka Kamal ja Uddin (2007) kuvailevat. Maaston tarkempi muokkaus onnistuu kenttäeditorin tavanomaisilla työkaluilla ja proseduraalisesti generoitujen yksityiskohtien lisääminen siveltimillä, jotka De Carpentier ja Bidarra (2009) esittävät.

Alustavan maaston luontiin voitaisiin käyttää myös menetelmää, jonka Smelik ym. (2010) esittävät. Menetelmä tuottaa pitkälti valmiin maaston, joten se saattaisi yksinäänkin riittää koko maaston luontiin. Luultavasti kenttäsuunnittelijoilla on kuitenkin tarve tehdä maastoon vielä joitain muokkauksia käsin.

Halutessaan suunnittelija voisi lopuksi soveltaa luomaansa maastoon eri eroosio-algoritmeja. Koska maasto luodaan kenttäeditorilla, ei eroosion mallinnuksen nopeudella ole hirveästi väliä, ja vesieroosiotakin voitaisiin käyttää huoletta. Uskottavuuden lisäyksen ohella eroosio tasottaisi maaston karkeita kohtia ja piilottaisi ilmeisimpiä muokkausten jälkiä.

5.2 Strategiapelit

Strategiapeleissä on usein mahdollisuus pelata satunnaisesti luotuja skenaarioita. Tällöin maaston generoinnin on tapahduttava melko nopeasti, jotta pelaaja ei turhaudu odotellessaan pelin alkamista. Erityisesti reaaliaikaisissa strategiapeleissä on tärkeää, että maasto mahdollistaa rakennusten sijoittelun ja joukkojen liikuttelun eri puolilla kenttää.

Olsen (2004) esittää menetelmän reaaliaikaiseen strategiapeliin soveltuvan maaston generoimiseksi. Hän käyttää nopeuden vuoksi timantti-neliö-algoritmia. Tähän hän lisää sopi-

vanlaista Worley-kohinaa, jotta maastoon muodostuisi luonnollisen oloisia kukkuloita. Varmistaakseen, että joukkoja voidaan liikutella ympäri kenttää, hän siirtää ja rajaa Worley-kohinaa siten, että maastoon syntyy eräänlaisia kulkureittejä. Lopuksi korkeuskarttaan sovelletaan eroosio-algoritmia, jotta maastoon saadaan enemmän tasaisia alueita rakennuksia varten, helpottaamaan joukkojen siirtelyä sekä lisäämään luonnonmukaisuutta. Tähän Olsen käyttää käänteistä lämpöeroosiota, koska se on riittävän nopea ja tuottaa haluttuja tuloksia.

Strategiapelien useista kentistä koostuvat kampanjat luodaan kenttäeditoreilla. Kampanjan kenttien luonnin helpottamiseksi ja nopeuttamiseksi editori voisi mahdollistaa satunnaisesti generoitavan skenaarion käyttämisen lopullisen kentän pohjana.

5.3 Loppumaton maasto

Joissain peleissä halutaan näennäisesti loputtomiin jatkuva maailma. Näissä peleissä pelaaja voi liikkua mihin suuntaan tahansa koskaan saavuttamatta pelimaailman reunaa. Näin suurta maastoa ei ole järkevää tallentaa tiedostoon. Maastoa pitää siis voida luoda pelin aikana lisää siten, että maasto jatkuu saumattomasti. Lisäksi maaston generoinnin pitäisi olla niin nopeaa, että pelaaminen pysyy sulavana.

Kaikki pelin aikana luotava maasto voi vaatia niin paljon muistia, ettei se mahdu kerralla keskusmuistiin. Joitain alueita voitaisiin vapauttaa, jos niitä ei tarvitse pitää muistissa. Esimerkiksi alueet, jotka ovat niin kaukana pelaajasta, että niitä ei voi nähdä, voitaisiin vapauttaa. Oletettavasti on toivottua, että pelaajan palatessa vanhoille, kertaalleen muistista poistetuille alueille loisi maaston generointiin käytetty menetelmä alueista samanlaisia, kuin ne aiemmin olivat. Keskusmuistista vapautettavat alueet voitaisiin toki kirjoittaa kiintolevylle ja tarvittaessa lukea sieltä takaisin muistiin, mutta tätä kannattanee mahdollisuuksien mukaan välttää kiintolevytilan säästämiseksi. Mikäli pelaaja pystyy muuttamaan maastoa jollain tavalla, ja muutosten halutaan säilyvän, on muutetut alueet kuitenkin tallennettava kiintolevylle.

Edellä mainitut vaatimukset karsivat tehokkaasti käytettäviä menetelmiä. Keskipisteen siirto ja timantti-neliö ovat nopeita korkeuskartan luonnissa, mutta niiden edellytyksenä on, että koko korkeuskartta luodaan kerralla. Toki niillä voitaisiin luoda tarvittaessa pienempiä

maaston palasia, jotka yhdessä muodostavat kokonaisen maaston, mutta näiden palojen yhtymäkohtia olisi hankalaa saada saumattomiksi. Lisäksi näitä palasia on vaikeaa luoda uudelleen samanlaisiksi, jos ne joudutaan poistamaan muistista. Eroosion mallintamisesta on myös luovuttava, koska eroosio-algoritmit tarvitsevat koko maaston korkeuskartan toimiakseen oikein.

Tällaisen maaston toteutus ei ole kuitenkaan mahdotonta. Maaston toteutuksessa voitaisiin käyttää esimerkiksi Perlin-kohinaa ja fraktionaalista Brownin liikettä, koska Perlin-kohinalla saadaan milloin tahansa ja mille pisteelle tahansa arvo siten, että saman pisteen arvo on aina sama. Uutta, saumatonta maastoa voidaan siis luoda koska tahansa, ja tietyn alueen maastosta tulee aina samanlainen. Perlin-kohina ja fraktionaalinen Brownin liike ovat myös riittävän nopeita, kunhan kohinatasoja ei luoda liikaa.

6 Yhteenveto

Tässä tutkielmassa käsiteltiin kolmiulotteisen maaston toteuttamista korkeuskartoilla, niiden generointiin käytettäviä proseduraalisia menetelmiä sekä erilaisia eroosio-algoritmeja. Lisäksi pohdittiin, kuinka näitä algoritmeja voidaan hyödyntää tietokonepeleissä.

Korkeuskartan generointiin on olemassa useita menetelmiä, joista monet perustuvat erilaisiin kohinoihin. Oikea maasto on luonteeltaan fraktaali, mutta tätä ominaisuutta ei kaikilla kohinoilla ole. Fraktionaalisella Brownin liikkeellä kohinoihin voidaan lisätä fraktaaleja piirteitä. Keskipisteen siirto ja timantti-neliö ovat algoritmeja, jotka luovat suoraan fraktaaleja korkeuskarttoja ja ovat varsin tehokkaita. Rajoitteidensa vuoksi ne eivät kuitenkaan sovellu kaikkiin tilanteisiin ja tarpeisiin. Lisäksi maaston generointiin on olemassa algoritmeja, joilla käyttäjä voi luoda helposti haluamansa laisen maaston.

Kun maasto generoidaan proseduraalisesti, maastosta saadaan pienemmällä vaivalla paljon yksityiskohtaisempi kuin täysin käsin luotaessa. Proseduraalisesti generoitu maasto saadaan myös näyttämään luonnolliselta, kun generoinnissa käytetään sopivia algoritmeja, kuten eroosion mallinnusta. Yleisimmät eroosio-algoritmit mallintavat joko lämpö- tai vesieroosiota. Erityisesti vesieroosio lisää paljon maaston realistisuutta, mutta algoritmi on todella raskas. Vesieroosion mallinnuksen tehostamiseksi simulaatio voidaan suorittaa näytönohjaimella.

Erilaisia maaston generointimenetelmiä ja eroosio-algoritmeja voidaan hyödyntää monin tavoin tietokonepeleissä ja niiden kehityksessä. Esimerkiksi pelin kenttäeditoriin yhdistettynä eri menetelmistä voidaan saada tehokkaita työkaluja kenttäsuunnittelijan käyttöön. Algoritmit soveltuvat myös maastojen luontiin dynaamisesti vaikkapa strategiapelin satunnaisskenaarioita varten. Joka tapauksessa maastoa suunniteltaessa on huomioitava pelin tarpeet ja vaatimukset, sillä ne määrittävät, mitkä menetelmät ja algoritmit soveltuvat parhaiten maaston luontiin.

Lähteet

Anh, Nguyen Hoang, Alexei Sourin ja Parimal Aswani. 2007. “Physically Based Hydraulic Erosion Simulation on Graphics Processing Unit”. Teoksessa *Proceedings of the 5th International Conference on Computer Graphics and Interactive Techniques in Australia and Southeast Asia,* 257–264. GRAPHITE ’07. Perth, Australia: ACM. ISBN: 978-1-59593-912-8. doi:10.1145/1321261.1321308.

Archer, Travis. 2011. *Procedurally Generating Terrain.* Tekninen raportti. Morningside College, Iowa.

Belhadj, Farès. 2007. “Terrain Modeling: A Constrained Fractal Model”. Teoksessa *Proceedings of the 5th International Conference on Computer Graphics, Virtual Reality, Visualisation and Interaction in Africa,* 197–204. AFRIGRAPH ’07. Grahamstown, South Africa: ACM. ISBN: 978-1-59593-906-7. doi:10.1145/1294685.1294717.

De Carpentier, Giliam J. P., ja Rafael Bidarra. 2009. “Interactive GPU-based Procedural Heightfield Brushes”. Teoksessa *Proceedings of the 4th International Conference on Foundations of Digital Games,* 55–62. FDG ’09. Orlando, Florida: ACM. ISBN: 978-1-60558-437-9. doi:10.1145/1536513.1536532.

Hnaidi, Houssam, Eric Gu rin, Samir Akkouche, Adrien Peytavie ja Eric Galin. 2010. “Feature based terrain generation using diffusion equation”. *Computer Graphics Forum* 29 (7): 2179–2186. ISSN: 1467-8659. doi:10.1111/j.1467-8659.2010.01806.x.

Kamal, K. Raiyan, ja Yusuf Sarwar Uddin. 2007. “Parametrically Controlled Terrain Generation”. Teoksessa *Proceedings of the 5th International Conference on Computer Graphics and Interactive Techniques in Australia and Southeast Asia,* 17–23. GRAPHITE ’07. Perth, Australia: ACM. ISBN: 978-1-59593-912-8. doi:10.1145/1321261.1321264.

Musgrave, F Kenton, Craig E Kolb ja Robert S Mace. 1989. “The synthesis and rendering of eroded fractal terrains”. Teoksessa *ACM SIGGRAPH Computer Graphics,* 23:41–50. 3. ACM.

Olsen, Jacob. 2004. *Realtime Procedural Terrain Generation.* Tekninen raportti. Department of Mathematics And Computer Science, University of Southern Denmark.

Ong, Teong Joo, Ryan Saunders, John Keyser ja John J. Leggett. 2005. "Terrain Generation Using Genetic Algorithms". Teoksessa *Proceedings of the 7th Annual Conference on Genetic and Evolutionary Computation,* 1463–1470. GECCO '05. Washington DC, USA: ACM. ISBN: 1-59593-010-8. doi:10.1145/1068009.1068241.

Smelik, Ruben M., Tim Tutenel, Klaas Jan de Kraker ja Rafael Bidarra. 2010. "Declarative Terrain Modeling for Military Training Games". *Int. J. Comput. Games Technol.* (New York, NY, United States) 2010 (tammikuu): 2:1–2:11. ISSN: 1687-7047. doi:10.1155/2010/360458.

Šťava, Ondřej, Bedřich Beneš, Matthew Brisbin ja Jaroslav Křivánek. 2008. "Interactive Terrain Modeling Using Hydraulic Erosion". Teoksessa *Proceedings of the 2008 ACM SIGGRAPH/Eurographics Symposium on Computer Animation,* 201–210. SCA '08. Dublin, Ireland: Eurographics Association. ISBN: 978-3-905674-10-1.

Togelius, Julian, Noor Shaker ja Mark J. Nelson. 2015. "Fractals, noise and agents with applications to landscapes and textures". Teoksessa *Procedural Content Generation in Games: A Textbook and an Overview of Current Research,* toimittanut Noor Shaker, Julian Togelius ja Mark J. Nelson. Springer.