Markku Velinen

2D-pelimaaston generointi

Metropolia Ammattikorkeakoulu
Insinööri (AMK)
Tietotekniikka
Insinöörityö
4.5.2012

Metropolia

Tiivistelmä

Tekijä Otsikko Sivumäärä Aika	Markku Velinen 2D-pelimaaston generointi 39 sivua 4.5.2012
Tutkinto	Insinööri (AMK)
Koulutusohjelma	Tietotekniikka
Suuntautumisvaihtoehto	Ohjelmistotekniikka
Ohjaajat	lehtori Miikka Mäki-Uuro lehtori Jorma Räty

Tässä työssä tutkitaan 2D-pelimaaston generointiin soveltuvia menetelmiä ja tutustutaan olemassaolevien pelien käyttämiin menetelmiin. Tarkoituksena on luoda maastogeneraattori Monster Conflict -peliin. Peli on sivultapäin kuvattu kaksiulotteinen Flash-peli. Tarkoituksena siinä on eliminoida vastustajan pelihahmot.

Monster Conflict -peli asettaa vaatimuksia generoidulle maastolle. Maaston tulee tarjota peliin taktista syvyyttä ja sen pitää olla ulkonäöllisesti miellyttävä. Maaston generoinnissa ei myöskään saisi nykykoneilla mennä sekuntia kauempaa.

Erilaisia menetelmiä testataan ja ne käydään läpi yksitellen. Testien perusteella havaitaan, että mikään menetelmä ei yksin tuota riittävän hyvää tulosta. Maastogeneraattorin implementaatio tehdään vaiheittain yhdistellen eri menetelmien parhaita puolia.

Tuloksena saatu maastogeneraattori täyttää kaikki sille asetetut vaatimukset. Maastot ovat tasapuolisia pelaajille ja sisältävät sopivan määrän yksityiskohtia. Maastot tarjoavat pelaajille myös huomattavan määrän vaihtoehtoisia varteenotettavia siirtoja. Maaston generointi tapahtuu nykykoneilla reilusti alle sekunnissa, ja maastot ovat miellyttävän näköisiä.

Avainsanat	maasto, generointi, peli, Monster Conflict, Perlin-kohina, fraktaalit, Flash

Metropolia

Abstract

Author Title Number of Pages Date	Markku Velinen Generation of 2D game terrain 39 pages 4 May 2012
Degree	Bachelor of Engineering
Degree Programme	Information Technology
Specialisation option	Software Engineering
Instructors	Miikka Mäki-Uuro, Senior Lecturer Jorma Räty, Senior Lecturer

This Bachelor's Thesis investigates different methods for generating 2D game terrain and examines the methods used in existing games. The goal was to create a terrain generator for a game called Monster Conflict. The game is a 2D side-view Flash game. The objective of the game is to eliminate the opponent's monsters.

Monster Conflict sets certain requirements for the generated terrain. The terrain should offer tactical depth as well as look pleasant for the viewer. Also the terrain generation process should not take longer than a second using a modern computer.

Different methods were tested and evaluated. The results indicated that no method by itself would create an adequate result. The implementation of the terrain generator was done by combining the best qualities of different methods.

The resulting terrain generator fulfills all the set requirements. The terrains are balanced for the players and contain just the right number of details. They also offer a considerable number of viable moves. The generation process using a modern computer took well under a second and the generated terrains are pleasant looking.

Keywords	terrain, generation, game, Monster Conflict, Perlin noise, fractals, Flash

Sisällys

1 Johdanto

Maaston generoinnilla tarkoitetaan maaston automaattista luomista. Samoja tekniikoita voidaan käyttää myös laajemmin esimerkiksi kokonaisten maailmojen ja olentojen luomiseen.

Tämän insinöörityön tavoitteena on tutkia erilaisia maaston generointimenetelmiä ja toteuttaa maastogeneraattori Monster Conflict -peliin. Maastogeneraattorin tulisi pystyä nopeasti luomaan miellyttävän näköisiä ja pelimekaanisesti sopivia maastoja.

Monster Conflict yhdistelee strategiaa ja toimintaa tarjoten huomattavan määrän erilaisia mahdollisia pelejä ja pelitilanteita. Silti jos maastoja olisi vain rajallinen määrä tai se olisi yksi ja sama aina, niin pelissä voisi oppia optimaalisen pelityylin ja sen jälkeen pelata pääasiassa vain muistin varassa. Tämänkaltainen pelaaminen olisi tylsää ja maastogeneraattorilla saadaan varmistettua se, että peliä ei voi ikinä oppia ulkoa. Näin pelaajan pitää jatkuvasti pystyä mukautumaan erilaisiin pelitilanteisiin. Haaste, jännitys ja mielenkiinto saadaan näin säilytettyä.

Työn luvussa kaksi selvitetään tunnettuja maaston generointimenetelmiä. Luvussa kolme perehdytään maaston ja maailmojen generointiin olemassa olevissa peleissä. Luvussa neljä tutustutaan Monster Conflict -peliin, jonka ehdoilla maasto tullaan luomaan. Luvussa viisi käydään läpi pelin vaatimukset maaston generoinnille. Luvussa kuusi tutkitaan erilaisten maaston generoinnin menetelmien sopivuutta Monster Conflict -peliin. Luvussa seitsemän käydään läpi toteutettua maastogeneraattoria ja eri vaiheita sen toiminnassa. Luvussa kahdeksan tarkistetaan vielä tulokset maaston generoinnin onnistumisesta. Luvussa yhdeksän on yhteenveto työstä.

2 Yleisiä maaston generointimenetelmiä

Pelimaaston generoinnissa eri menetelmät soveltuvat eri peleihin. Menetelmiä voi myös yhdistellä saadakseen mahdollisesti mielenkiintoisempia ja parempia tuloksia kuin yhdelläkään menetelmällä yksinään. Seuraavana esitellään joitain yleisiä maaston generointialgoritmeja.

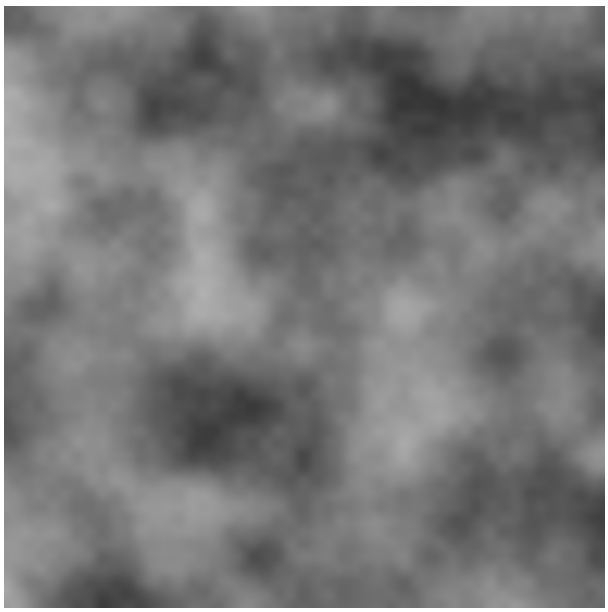
2.1 Perlin-kohina

On olemassa algoritmeja, joilla voi luoda erilaisia kohinoita simuloimaan luonnossa tapahtuvia asioita. Ken Perlinin kehittämä Perlin-kohina on näistä ehkä tunnetuin. Sitä käytetään simuloimaan esimerkiksi tulta, savua ja pilviä. [1.]

Perlin-kohinan avulla luodaan pseudosatunnaisia tekstuureita. Täysin satunnaisten tekstuurien luontiin verrattuna Perlin-kohinan ehkä suurin etu on se, että satunnaisten yksityiskohtien koko pysyy samana ja sen pystyy määrittämään. Kuvassa 1 on kaksiulotteista Perlin-kohinaa käyttäen luotu tekstuuri. [1.]

Kuva 1. Kaksiulotteista Perlin-kohinaa. [2.]

Hyödyntäen sitä, että Perlin-kohina pysyy samankokoisena, voidaan yhdistellä erikokoisia Perlin-kohinoita yhteen kuvaan ja saada aikaan fraktaalikohinaa kuvan 2 mukaisesti. [1.]



Kuva 2. Päällekkäisten Perlin-kohinoiden muodostamaa fraktaalikohinaa. [3.]

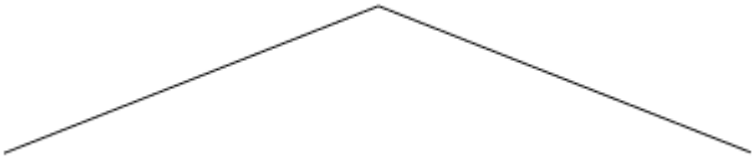
Perlin-kohinaa voi soveltaa maaston generointiin monella eri tavalla pelistä ja mielikuvituksesta riippuen. Sitä voi käyttää lähes sellaisenaan muodostaen korkeuskartan, jota voidaan käyttää ylhäältäpäin kuvatussa kaksiulotteisessa pelissä. Korkeuskarttojen avulla voidaan muodostaa kolmiulotteinen maasto käyttäen pikselin kirkkautta korkeutena ja sijaintia suoraan sijaintina.

Sivultapäin kuvattuun peliin voidaan haluta mieluummin päättää jokin tietty kirkkausaste, jota alempana tai ylempänä olevat pikselit ovat maata. Loppu voi olla ilmaa, avaruutta tai mitä pelissä halutaankin. Sitä voidaan myös yhdistää muihin menetelmiin esimerkiksi käyttäen sitä todennäköisyys- tai tiheysfunktiona.

2.2 Keskipisteen siirto

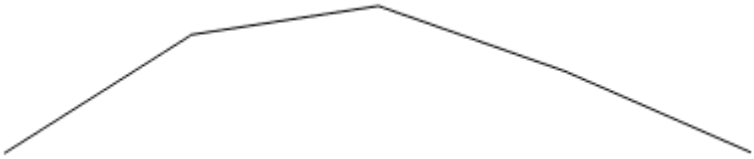
Keskipisteen siirto (midpoint displacement) on fraktaalimaaston luontiin käytetty menetelmä. Fraktaaleille tunnusomaista on se, miten läheltä tai kaukaa tahansa niitä katsoo, niin ne näyttävät toistavan samaa kuviota. [4.]

Keskipisteen siirrossa aloituspisteessä on yksi suora vaakaviiva. Viivasta etsitään keskipiste ja siirretään sitä satunnaisen määrän väliltä [-1, 1] ylös- tai alaspäin kuvan 3 mukaisesti. [4.]



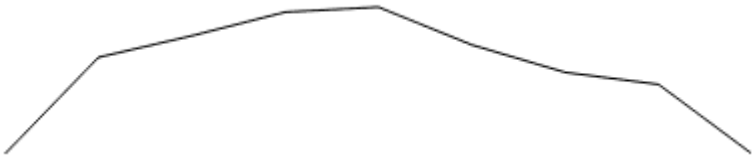
Kuva 3. Viiva, jonka keskipistettä on siirretty. [5.]

Tämän jälkeen saadaan kaksi viivaa. Sama operaatio toistetaan molemmille viivoille, mutta satunnaisväli puolitetaan, eli saadaan [-0,5, 0,5]. Molempien viivojen keskipisteitä siirretään sitten satunnaisesti kuvan 4 mukaisesti. [4.]



Kuva 4. Kaksi viivaa joiden keskipisteet on siirretty muodostaen neljä viivaa. [6.]

Nyt kuvassa on neljä viivaa. Sama operaatio toistetaan jälleen. Satunnaisvälin puolituksen jälkeen väli on [-0,25, 0,25]. Kuvio näyttää nyt kuvan 5 mukaiselta. [4.]



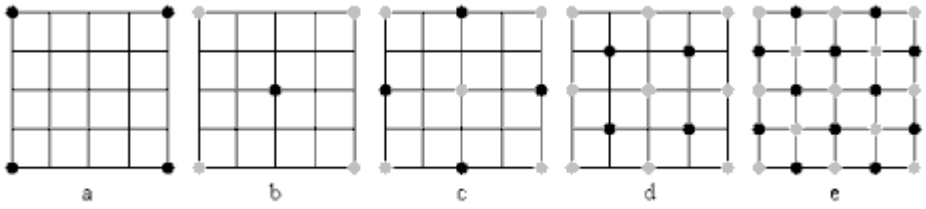
Kuva 5. Neljä viivaa joiden keskipisteet on siirretty muodostaen kahdeksan viivaa. [7.]

Tätä voi jatkaa niin pitkään kuin viivan pituus riittää siihen, että siinä on keskipiste. Mitä pidemmälle tätä jatketaan, sen tarkempi ja yksityiskohtaisempi siitä tulee. Kaksiulotteiseksi sivultapäin kuvatuksi maastoksi sen voi muuttaa yksinkertaisesti määrittämällä, että kaikki viivan alapuolella oleva osa on maastoa ja yläpuolella oleva on ilmaa. [4.]

2.3 Salmiakki-neliö-algoritmi

Salmiakki-neliö-algoritmi (diamond-square algorithm) on keskipisteen siirrosta muunneltu algoritmi, millä voi tehdä kaksiulotteisia korkeuskarttoja. Algoritmia varten tarvitaan 2D-ruudukko pisteitä. Ruudukon tulisi olla neliö, ja sen sivun pituus tulisi olla kahden potenssi plus yksi. Eli esimerkiksi 33x33, 65x65 tai 129x129. Aluksi asetetaan neljälle reunapisteelle sama korkeusarvo. [4.]

Otetaan yksinkertaiseksi esimerkiksi kuvan 6 mukainen 5x5 ruudukko. Alkutilanteessa a on vain kulmapisteisiin asetettu korkeusarvo. [4.]

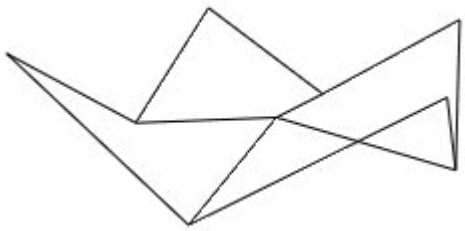


Kuva 6. Salmiakki-neliö-algoritmin vaiheet pisteruudukossa. [8.]

Ensin suoritetaan salmiakkiaskel. Otetaan kaikki neljästä pisteestä koostuvat neliöt ruudukossa ja asetetaan niiden keskipisteeseen neljän pisteen keskiarvo, johon on lisätty satunnaisarvo. Tämän seurauksena ruudukon tilanne on kuvan 6 b-kohdan mukainen. Uudet arvot on merkitty mustalla ja vanhat harmaalla. Tämä askel luo salmiakkeja, kunhan ruudukossa on useita neliöitä. [4.]

Seuraavaksi suoritetaan neliöaskel. Otetaan kaikki neljästä pisteestä koostuvat salmiakit ja asetetaan niiden keskipisteeseen neljän salmiakin pisteen keskiarvo, johon on jälleen lisätty satunnaisarvo. Saadaan kuvan 6 c-kohdan mukainen tilanne, jossa on neljä neliötä. [4.]

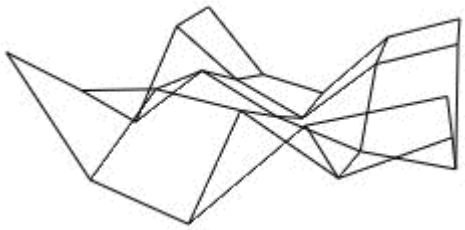
Salmiakkiaskelen ja neliöaskeleen jälkeen on suoritettu yksi iteraatio salmiakki-neliö-algoritmissa. Tilanne esitettynä eri kuvakulmasta pisteet yhdistettynä voi olla tämän jälkeen esimerkiksi kuvan 7 mukainen.



Kuva 7. 5x5-ruudukko yhden salmiakki-neliö-algoritmin iteraation jälkeen. [9.]

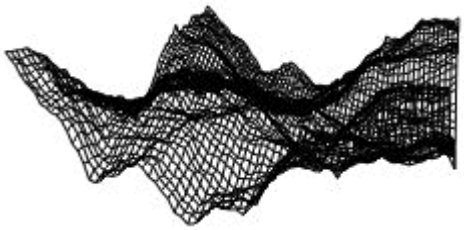
Toinen askel menee samaan tyyliin kuin ensimmäinen. Ensin salmiakkiaskeleessa lasketaan uudet pisteet kaikkien neljän neliön keskipisteisiin. Lisättävää satunnaislukua varten oleva satunnaislukuväli tulee puolittaa siitä mitä se oli aiemmin, jotta saadaan pienempiä yksityiskohtia. Salmiakkiaskeleen jälkeen tilanne on kuvan 6 d-kohdan

mukainen. Neliöaskeleen suorittamisen jälkeen tilanne on kuvan 6 e-kohdan mukainen ja eri kuvakulmasta esitettynä voi olla esimerkiksi kuvan 8 mukainen.



Kuva 8. 5x5 ruudukko kahden salmiakki-neliö-algoritmin iteraation jälkeen. [10.]

5x5 ruudukon kaikilla pisteillä on nyt arvo. Isommilla ruudukoilla algoritmia pystyisi jatkamaan pidempään lisäten yksityiskohtia. Esimerkiksi viiden iteraation jälkeen maasto voisi näyttää kuvan 9 mukaiselta.



Kuva 9. Ruudukko viiden salmiakki-neliö-algoritmin iteraation jälkeen. [11.]

3 Maaston ja maailmojen generointi

Videopeleissä normaalisti koko pelimaailma on pelintekijöiden valmiiksi suunnittelema ja tulee pelin mukana. Varhaisimmissa videopeleissä pelilaitteen muistin määrä rajoitti suuresti pelimaastojen ja maailmojen kokoa. Tämän takia otettiin käyttöön algoritmeja, joilla oli mahdollista luoda pelimaailma pelin ollessa käynnissä. Itse algoritmit käyttävät murto-osan siitä muistista, mitä valmiiksi luotu ja tallennettu sisältö käyttäisi. Näin saatiin erittäin suuria pelimaailmoja hyvin pienellä muistitarpeella. [12.]

Nykyään generointitekniikoita käytetään monenlaisiin tarkoituksiin. Yleisimmät hyödyt generoinnista ovat suurempi variaatio ja tekijöiden pienempi työmäärä maailman, maaston tai hahmojen suunnittelussa. [12.]

Useat pelit käyttävät maaston ja maailman generointia jossain määrin. Joihinkin pelityyppeihin sopii paremmin täysin synteettisen näköinen, mutta pelimekaanisesti toimiva maasto. Toisiin voidaan tarvita mahdollisimman realistisen näköistä maastoa, jotta saadaan säilytettyä illuusio siitä, että peli voisi sijoittua tosimaailmaan. Tässä on esiteltynä muutamia tunnettuja esimerkkejä.

3.1 Minecraft

Nykyään hyvin tunnetusta Minecraftista julkaistiin alpha-versio vuonna 2009 [13.]. Kuvassa 10 on Minecraftin classic-versio.

Kuva 10. Minecraft classic. [14.]

Maastongenerointi on Minecraftin monimutkaisimpia osia. Pelaaja voi lähteä tutkimaan maastoa mihin tahansa suuntaan ja kuinka pitkälle tahansa. Maaston täytyy olla siis ainakin lähes loputon, ja niin se Minecraftissa on. [15.]

Alun perin Minecraftin maasto oli kaksiulotteinen korkeuskartta, joka oli generoitu kaksiulotteisella Perlin-kohinalla. Minecraft on kolmiulotteinen peli ja kaksiulotteinen korkeuskartta ei mahdollista luolia tai muunlaisia päällekkäisiä maastoja. [15.]

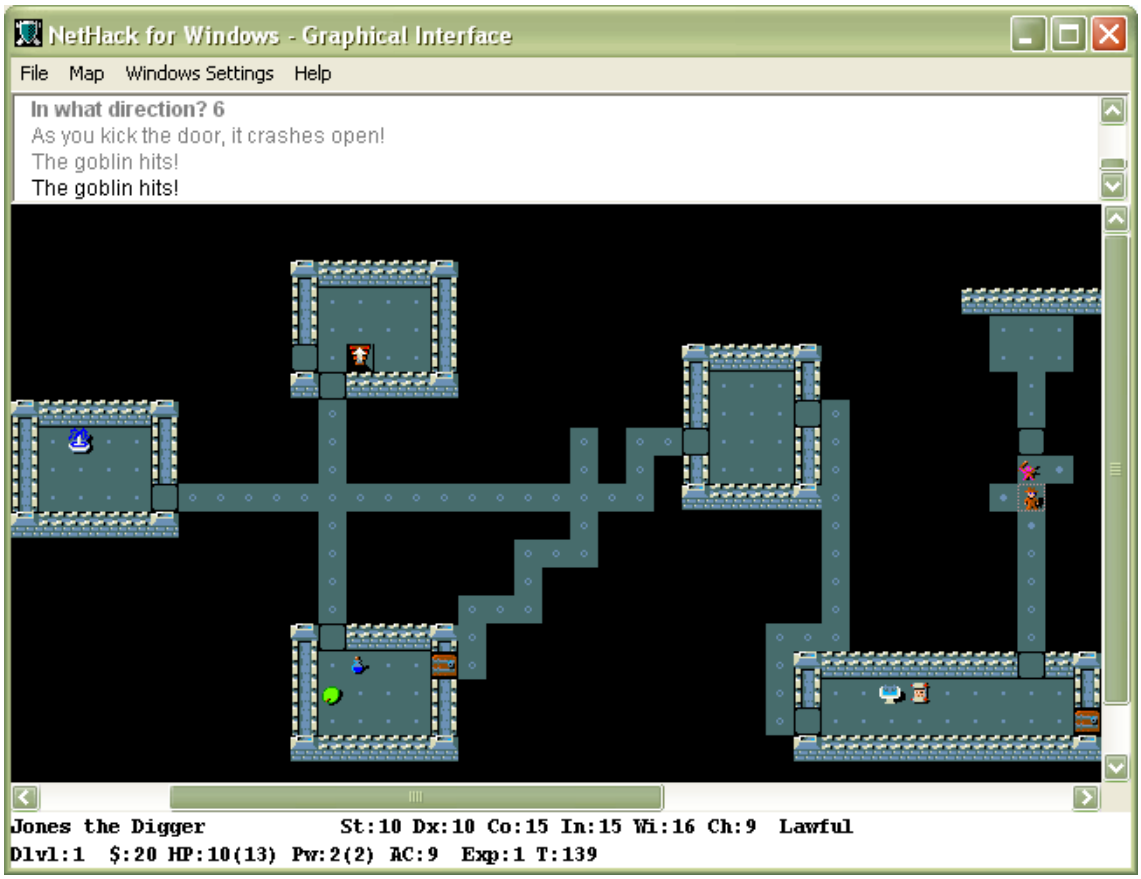
Tämän takia generointi vaihdettiin käyttämään kolmiulotteista Perlin-kohinaa. Aluksi siinä oli pelillisiä ja suorituskyvyllisiä ongelmia, mutta molemmat saatiin korjattua generoimalla maasto pienemmällä resoluutiolla ja skaalaamalla isommaksi siitä. Näin saatiin vähennettyä liian tarkkoja yksityiskohtia. [15.]

3.2 NetHack

Alun perin vuonna 1987 julkaistu NetHack on eräs tunnetuimmista Roguen kaltaisista peleistä. Roguen kaltaiset pelit sisältävät tyypillisesti satunnaisesti generoituja luolastoja. [16; 17.]

Nethackin luolastotason generointi tapahtuu kolmessa vaiheessa. Ensimmäiseksi generoidaan suorakulmaisia huoneita, joista ainakin kahden täytyy olla riittävän kokoisia rappusia varten. Toisessa vaiheessa yhdistetään huoneet käytävillä ja luodaan ovet. Kolmannessa vaiheessa luodaan hirviöt, aarteet, kaupat ja muut peliobjektit. [18.]

NetHackin luolasto voi näyttää esimerkiksi kuvan 11 mukaiselta.

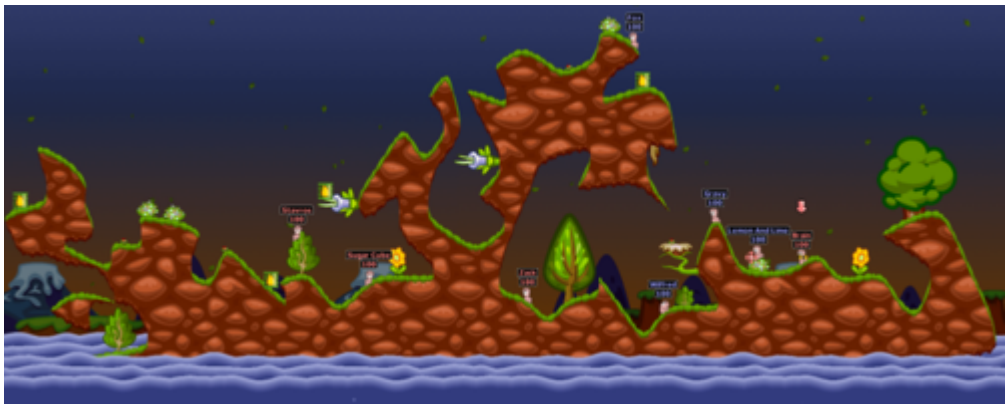


Kuva 11. Nethackin luolastoesimerkki. [19.]

3.3 Worms

Ensimmäinen Worms julkaistiin vuonna 1995. Se ei ollut ensimmäinen peli genressään, mutta sen iloinen ja humoristinen tyyli olivat uutta. [20.]

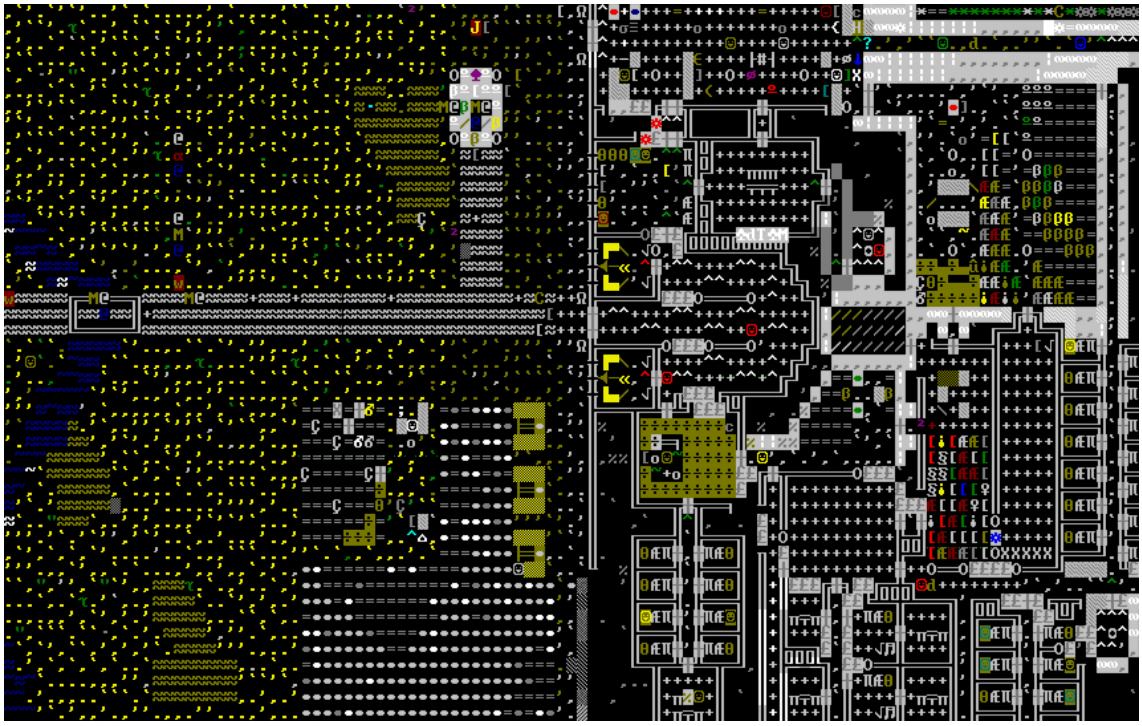
Worms on sivultapäin kuvattu kaksiulotteinen peli. Siinä kaikki maasto on tuhoutuvaa. Tässä työssä käytetty Monster Conflict -peli on ottanut paljon vaikutteita Wormsista ja maastongenerointi on myös samantyylistä kuin mitä Wormsissa on todennäköisesti käytetty. Kuvassa 12 on esimerkki Wormsista.



Kuva 12. Worms. [21.]

3.4 Dwarf Fortress

Dwarf Fortress julkaistiin vuonna 2006. Se yhdistää osia Roguen kaltaisista peleistä kaupunginrakennuksen kanssa monimutkaiseksi ja vaikeaksi kokonaisuudeksi. Peli voi näyttää esimerkiksi kuvan 13 mukaiselta. [22.]



Kuva 13. Dwarf Fortress. [23.]

Dwarf Fortressissa generoidaan laaja pelimaailma sisältäen rakennuksia, historian ja reaktiivisen ekosysteemin. Pelimaailma generoidaan ensimmäisellä käynnistyskerralla ja

se saattaa nykyisillä tietokoneillakin kestää jopa 30 minuuttia. Maailma tallennetaan sen jälkeen tekstitiedostoon, jonka koko voi nousta jopa yli 100 MB:n. [12.]

4 Monster Conflict -peli

Monster Conflict -peli luotiin asettamaan vaatimuksia maastongeneroinnille ja tarjoamaan mahdollisuus testata maaston soveltuvuutta peliin. Monster Conflict on kuvan 14 mukainen sivultapäin kuvattu kaksiulotteinen Flash-peli. Peliä pelataan toista ihmispelaajaa vastaan. Pelin tarkoitus on yksinkertaisesti eliminoida vastustajan pelihahmot.

Kuva 14. Monster Conflict -peli

Kaikilla pelihahmoilla on alussa kestävyys 100. Pelihahmo kuolee, jos sen kestävyys laskee nollaan. Pelissä on myös painovoima, ja pelihahmo kuolee, jos se putoaa pelialueelta.

Ennen pelin aloittamista valitaan pelivuoroon varattu aika ja pelimaasto. Oletuksena ovat 30 sekunnin vuorot ja sattumanvaraisella siemenluvulla generoitu maasto. Mikäli pelaajat eivät ole tyytyväisiä esillä olevaan maastoon, niin he voivat joko generoida uuden maaston uudella sattumanvaraisella siemenluvulla tai syöttää itse siemenluvun. Kun sitä käytetään, maastogeneraattori luo tietyn maaston.

Pelaajat voivat siis esimerkiksi generoida maastoja sattumanvaraisilla siemenluvuilla kunnes löytävät erityisen miellyttävän maaston. Kyseisen maaston generointiin käytetyn siemenluvun voi sen jälkeen ottaa ylös ja käyttää myöhemmin pelaajien halutessa pelata käyttäen samaa maastoa.

Pelin alkaessa ensimmäisen pelaajan hahmot asetetaan sattumanvaraisiin paikkoihin pelialueen vasempaan reunaan. Toisen pelaajan hahmot asetetaan vastaavasti pelialueen oikeaan reunaan.

4.1 Pelin kulku

Pelin aloittaa ensimmäinen pelaaja. Vuoro alkaa siten, että pelaaja valitsee omista pelihahmoistaan sen, jota haluaa käyttää kyseisellä vuorolla. Tämän jälkeen laskuri käynnistyy ilmoittaen pelivuorolla jäljellä olevan ajan.

Pelihahmoa ohjataan nuolinäppäimillä. Vasemmalle ja oikealle nuolet liikuttavat hahmot vastaavasti vasemmalle ja oikealla. Ylöspäin nuolesta hahmo hyppää. Hahmoa voi liikuttaa vasemmalle ja oikealle myös hypyn aikana. Alaspäin nuoli ei ole käytössä.

Pelihahmoa voi liikuttaa vuoron aikana vain rajallisen määrän. Pelihahmon liikkuessa ruudun yläreunassa oleva keltainen liikkumakykyä kuvaava palkki vähenee ja sen palkin kuluessa loppuun hahmo ei pysty enää liikkumaan.

Toinen ja tärkeämpi asia, mitä hahmot voivat tehdä, on käyttää erikoiskykyjään. Kyvyt toimivat eri tavoin ja tuovat peliin strategisia elementtejä. Vuoron aikana hahmo voi käyttää vain yhden kyvyn ja vuoro loppuu heti kyvyn käyttämisen jälkeen. Vuoro loppuu myös, jos pelivuoroon varattu aika laskee nollaan.

Kyvyn käyttäminen tapahtuu valitsemalla hiirellä yläreunasta kyky ja sen jälkeen painamalla jostain kohtaa pelialueelta. Tämän jälkeen on toisen pelaajan vuoro. Hän aloittaa myös valitsemalla hahmon, jota haluaa käyttää. Vuoron jälkeen on taas ensimmäisen pelaajan vuoro valita pelattava hahmonsa. Peli loppuu, kun toisella puolella ei ole enää yhtään elossa olevaa hahmoa.

4.2 Maasto ja sen tuhoaminen

Monster Conflict -pelissä maasto tekee pelistä mielenkiintoisen ja lisää huomattavasti strategisia mahdollisuuksia. Pelihahmot eivät voi liikkua normaalisti maaston läpi. Pelissä tulee helposti tilanne, että pelihahmo ei suoraan pysty liikkumaan haluamaansa paikkaan maaston takia. Tällöin pelihahmo voi tuhota maastoa ja päästä etenemään.

Maaston tuhoutuminen pelissä on toteutettu siten, että maastosta vain poistetaan tuhoutunut osa. Tuhoutuminen olisi voitu toteuttaa monimutkaisemmin ja esimerkiksi tuhota räjähdyksen alla olevasta maastosta vain osa ja antaa osan pudota painovoiman ja nestesimulaation avulla. Yksinkertaisempaa tuhoutumista päätettiin käyttää Flashin suorituskykyrajoitusten takia ja myös siksi, että esikuvana käytetyssä Wormsissa se toimi hyvin.

4.3 Pelihahmot

Pelissä on kolme erilaista pelihahmoa. Molempien pelaajien joukkueet koostuvat yhdestä jokaista pelihahmoa. Pelihahmoilla on eriävät kyvyt ja liikkumanopeudet.

Jokaisella pelihahmolla on aluksi neljä kykyä. Kahden vastustajan vuoron jälkeen hahmoille avautuu niiden viimeinen kyky. Viimeistä kykyä käytettäessä käyttäjän kestävyys vähenee 20:llä. Viimeinen kyky on sen takia tehokkaampi kuin muut kyvyt.

4.3.1 ”near”

”near” on hahmoista nopein ja sen kyvyt painottuvat lähitaisteluun. ”near” hahmo omaa kyvyt: Fireball, Tackle, Rift Walk, Vampiric Touch ja Evolve/Inferno.

Fireball ammutaan hiiren osoittamaan suuntaan. Se lähtee sen käyttäneestä hahmosta. Siihen vaikuttaa kuudesosa normaalista painovoimasta, joten sen voi ampua pitkälle melkein suoraan viivaa.

Fireball-ammus aiheuttaa pienen räjähdyksen osuessaan hahmoon tai maastoon. Räjähdyksen alueella olevat hahmot menettävät 10–30 pistettä kestävyyttä riippuen siitä, kuinka lähellä ne olivat räjähdyskeskipistettä. Lähempänä oleminen aiheuttaa enemmän vahinkoa. Fireball vahingoittaa myös käyttäjänsä kanssa samalla puolella olevia hahmoja.

Tackle-kykyä käyttänyt hahmo liikkuu lyhyen vakiopituisen matkan hiiren osoittamaan suuntaan. Samalla kaikki hahmon lähellä oleva maasto tuhoutuu, ja eteen osuneet viholliset menettävät 30 pistettä kestävyyttä.

Pelaaja ei pysty ohjaamaan hahmoa samalla kun se liikkuu Tackle-kyvyn avulla. Tällöin myöskään painovoima ei hetkellisesti vaikuta hahmoon.

Rift Walk -kyvyn avulla hahmo siirtyy välittömästi lyhyen matkan päähän hiiren osoittamaan kohtaan. Lisäksi kaikki siirtymäkohdan lähellä olevat vihollishahmot menettävät 40 pistettä kestävyyttä.

Vampiric Touch -kykyä käytettäessä kohteen pitää olla hyvin lähellä. Kyky käytetään yksittäiseen lähellä olevaan viholliseen, joka menettää välittömästi 40 pistettä kestävyyttä. Kykyä käyttänyt hahmo saa itse 20 pistettä kestävyyttä lisää.

Kahden vastustajan vuoron jälkeen ”near” saa viimeisen kykynsä. Tämä kyky toimii kahdessa osassa. Ensin hahmon pitää käyttää Evolve. Tämän jälkeen hahmo menettää 20 kestävyyttä, mutta kehittyy väliaikaisesti seuraavalle evoluutioasteelleen.

Hahmo pysyy toisella evoluutioasteella sen seuraavan vuoron loppuun asti. Siihen asti hahmolla on käytössään uusi kyky nimeltä Inferno. Eli seuraavan kerran kun pelaaja päättää pelata ”near”-hahmolla, niin hän voi käyttää Inferno-kykyä tai mitä tahansa muuta hahmolla olevaa, mutta tämän jälkeen hahmo palaa takaisin normaalimuotoonsa ja menettää Inferno-kyvyn, kunnes hahmo käyttää uudelleen Evolve-kykyä.

Inferno on pelin tehokkain hyökkäyskyky. Se räjäyttää ison alueen käyttäjänsä ympärillä ja kaikki alueella olevat kyvyn käyttäjään lukuun ottamatta menettävät 30–100 pistettä kestävyyttä riippuen, kuinka lähellä ne olivat kyvyn käyttäjää.

4.3.2 ”mid”

”mid”-hahmon liikkumanopeus on ”near”:n ja ”far”:n välissä. Se erikoistuu tuhoisiin keskimatkan kykyihin. ”mid”-hahmolla on seuraavat kyvyt: Fireball, Tackle, Meteor, Cluster Grenade ja Wings. Fireball ja Tackle on selitetty yllä.

Meteor on samantyylinen hyökkäys kuin Fireball. Meteor-kyky eroaa Fireball-kyvystä siten, että siihen vaikuttaa painovoima normaalisti. Meteor räjähtää myös tuplasti isompana kuin Fireball, ja alueella olevat hahmot menettävät 15–50 pistettä kestävyyttä.

Cluster Grenade -kyky on toimii kuten Meteor, mutta ammus ei räjähdä heti osuessaan johonkin, vaan kimpoaa siitä. Ammus räjähtää joko pysähtyessään paikalleen tai viimeistään kolmen sekunnin kuluttua sen käytöstä.

Cluster Grenade -kyvyn ammuksen räjähdys on samanlainen kuin Meteor-kyvyssä, mutta heti räjähdyksen jälkeen räjähdyskeskipisteestä lentää ylöspäin eri kulmissa neljä pienempää ammusta, jotka myös kimpoilevat ja räjähtävät pysähtyessään tai viimeistään kolmen sekunnin kuluttua. Nämä pienemmät ammukset aiheuttavat samankokoisen räjähdyksen kuin Fireball-kyky ja alueella olevat hahmot menettävät 10–20 pistettä kestävyyttä.

Wings on ”mid”-hahmon viimeinen kyky. Sen käyttämällä hahmo menettää 20 kestävyyspistettä ja saa seuraavaksi vuorokseen siivet.

Siivet antavat hahmolle loputtoman liikkumakyvyn ja mahdollisuuden lentää hyppynappia käyttämällä. Siipien avulla hahmo pääsee pidemmälle ja esimerkiksi korkeisiin paikkoihin mihin se normaalisti ei pääsisi. ”mid”-hahmon on helppo vahingoittaa alapuolellaan olevia vihollishahmojaan käyttäen Meteor- ja Cluster Grenade -kykyjä.

4.3.3 ”far”

”far” on hahmoista hitain, mutta omaa pitkän kantaman kykyjä. ”far”-hahmon kyvyt ovat Fireball, Tackle, Dirtball, Poison Larpa ja Final Flash. Fireball ja Tackle on selitetty ”near”-hahmon kohdassa.

Dirtball-kyky ammutaan kuten Meteor, mutta räjähtäessään se ei aiheutakaan vahinkoa tai tuhoa maata, vaan luo sitä. Luodun alueen koko on sama kuin Meteor-kyvyn räjähdyksen.

Poison Larpa käyttäytyy samoin kuin Fireball, mutta ammuksen lentäessä ilmassa se pudottaa pieniä myrkkypalloja tihein väliajoin. Myrkkypallo eikä varsinainen ammus kumpikaan aiheuta maastoa tuhoavaa räjähdystä osuessaan, mutta aiheuttavat myrkkypilven, jonka koko on sama kuin Fireball-ammuksen räjähdyksen.

Myrkkypilveen osunut vihollishahmo menettää 10 pistettä kestävyyttä ja joutuu myrkyttyneeseen tilaan. Myrkyttyneessä tilassa hahmo menettää 10 pistettä kestävyyttä molempien vuorojen lopussa, ja myrkytys jatkuu, kunnes hahmo on kuollut.

Final Flash on ”far”-hahmon viimeinen kyky. Myös sen käyttäminen vähentää käyttäjänsä kestävyyspisteitä 20:llä. Final Flash kohdistetaan minne tahansa pelialueella. Tämän jälkeen hahmon vuoro loppuu, mutta ammus ei lähde heti.

Final Flash aktivoituu vasta heti vastustajan seuraavan vuoron jälkeen. Tämä tarkoittaa tilannetta, kun vastustajalta on juuri käyttänyt kykynsä tai vastustajalta on juuri loppunut vuoroaika. Jos kyvyn käyttänyt hahmo ehtii kuolla ennen vastustajan seuraavan vuoron loppumista, niin kyky ei aktivoidu.

Esimerkiksi jos vastustaja käyttää Fireball-kyvyn, niin Final Flash aktivoituu samalla, kun Fireball-ammus on luotu eikä ole vielä ehtinyt osumaan mihinkään. Rift Walk, Vampiric Touch ja Inferno kuitenkin tekevät vahinkoa suoraan eivätkä luo mitään erityistä ammusta. Näillä kolmella kyvyllä pystyy estämään Final Flash -kyvyn aktivoitumisen. Tämä tekee ”near”-hahmosta luonnollisen vihollisen ”far”-hahmolle.

Final Flash aktivoituessaan tuhoaa suoran viivan käyttäjästään kohteeseen. Kaikki viivalla olevat hahmot menettävät 60 pistettä kestävyyttä. Final Flash on erityisen kätevä tilanteessa, jossa kaksi vihollista on suoralla viivalla. Tämä lähes pakottaa toisen heistä ainakin liikkumaan, mutta molempia ei silti pysty pelastamaan ellei Final Flash -kykyä käyttävää hahmoa tapeta.

4.4 Apupaketit

Vuoron jälkeen pelialueelle ilmestyy sattumanvaraiseen paikkaan sattumanvarainen apupaketti. Apupaketit pystytään keräämään menemällä niiden lähelle ja ne tarjoavat jonkinlaisen parannuksen hahmolle. Pelissä on olemassa neljä erityyppistä apupakettia.

Parannuspaketti antaa hahmolle 20 kestävyyspistettä kerättäessä. Hahmon kestävyys voi nousta myös yli alussa olevan sadan pisteen.

Liikkumakykypaketti antaa hahmolle väliaikaisesti yhden kokonaisen liikkumakykypalkin verran lisää liikkumakykyä. Liikkumakyky voi nousta yli kokonaisen palkin.

Tehopaketti tekee hahmon seuraavasta vihollisia vahingoittavasta kyvystä 50 % tehokkaamman. Tehopaketteja voi kerätä monta, ja jokainen lisää tehoa 50 % alkuperäisestä tehosta.

Suojapaketti vähentää seuraavaa hahmoon kohdistuvaa vahinkoa 50 %. Suojapaketteja voi myös kerätä monta, jolloin hahmoon kohdistuva vahinko tullaan kertomaan kaavalla 1.0/(suojapakettien_määrä + 1).

5 Vaatimukset

Erilaiset maastot soveltuvat erilaisiin peleihin. Tässä tarkastellaan maastongenerointia Monster Conflict -pelin näkökulmasta ja asetetaan erilaisia vaatimuksia peliin generoidulle maastolle.

5.1 Pelimekaaniset vaatimukset

Monster Conflict -peli asettaa maastongeneroinnille monia pelimekaniikkaan liittyviä vaatimuksia. Osa näistä on pakollisia pelin toiminnan kannalta ja osa tarvitaan tuomaan peliin taktista syvyyttä.

Ensimmäiseksi koska pelissä on painovoima, niin pelissä on pakko olla ainakin jotain maastoa, etteivät hahmot putoaisi pelialueelta. Vähintään jonkinlainen pohja on siis pakko olla, mutta monesta kerroksesta koostuva maasto tarjoaa huomattavasti enemmän strategisia vaihtoehtoja.

Toiseksi pelialue ei myöskään saa olla täynnä maastoa, jotta hahmot pystyisivät aloittamaan tyhjästä paikasta. Olisi myös hyvä, jos pelialueella pystyisi liikkumaan jonkin verran.

Tärkeätä on myös, että hahmot eivät voi välittömästi tehdä huomattavaa vahinkoa toisilleen, vaan joutuvat vähän suunnittelemaan, mitä haluavat tehdä ja miten. Esimerkiksi jos ”far” pystyisi Poison Larpa -kykyä käyttäen myrkyttämään kaikki vastustajansa hahmot ensimmäisellä vuorolla, niin peli olisi ohi aika nopeasti. Myös pelitilanne olisi epäreilu toisena pelaavaa kohtaan.

Pelimaaston olisi myös hyvä olla tasapuolinen molemmille pelaajille. Tämän saa helpoiten ja varmimmin toteutettua tekemällä maastosta peilikuva.

5.2 Suorituskykyvaatimukset

Monster Conflict on kirjoitettu ActionScript 3:lla ja tarkoitettu ajettavaksi normaalisti selaimella Adobe Flash Player -pluginin avulla. ActionScript 3:lla kirjoitetun ohjelman suorituskyky on kymmeniä kertoja hitaampaa kuin C:llä kirjoitetun vastaavan ohjelman.

Maaston generointi olisi hyvä tapahtua normaalilla kuluttajakoneella mielellään alle sekunnissa. Kielen hitaudesta johtuen hyvin raskaita algoritmeja ei voi siis käyttää.

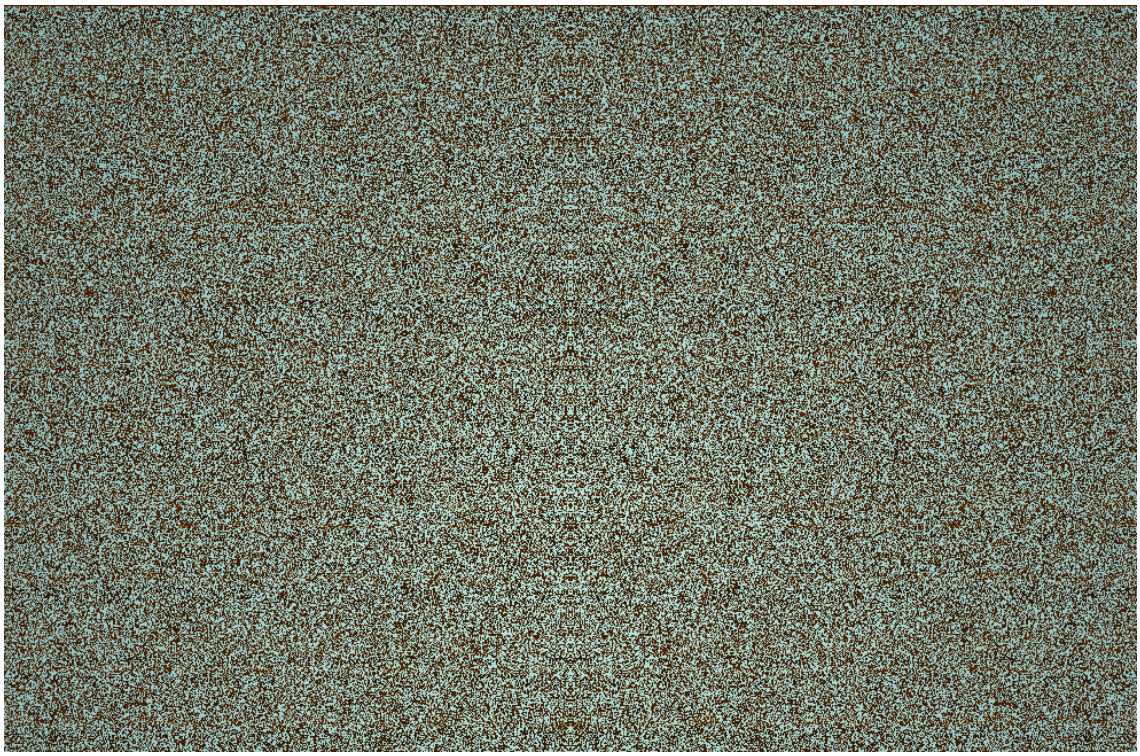
5.3 Ulkonäkövaatimukset

Koska pelin tarkoitus on viihdyttää ihmistä, niin maaston olisi myös hyvä olla miellyttävän näköinen. Tarkkoja ulkonäkövaatimuksia on hankala laatia, mutta mitä enemmän maasto näyttäisi siltä, että se voisi olla oikeasti olemassa tai ihminen olisi oikeasti suunnitellut sen, niin sen parempi.

6 Monster Conflict -pelissä testatut generointimenetelmät

6.1 for (pixel in map) { if (rand() % 2 == 0) { pixel.ground = true; } }

Generointimenetelmät voivat olla hyvinkin yksinkertaisia. Eräs erityisen naiivi menetelmä olisi erikseen asettaa jokainen pikseli pelialueella joko maaksi tai ilmaksi. Näin saadaan esimerkiksi kuvan 15 mukaista maastoa.



Kuva 15. Maastongenerointi asettamalla sattumanvaraiset pikselit maaksi.

Pelimekaanisesti tämän menetelmän generoimat maastot eivät toimisi ollenkaan. On hyvin epätodennäköistä, että tällä menetelmällä saisi generoitua maaston, jossa olisi

tilaa kaikille hahmoille aloittaa eikä yksikään niistä putoaisi pelialueelta alas. Maastossa ei myöskään pystyisi juuri ampumaan eteenpäin tai kulkemaan ja olisi siten melko tylsä. Silti jopa tämänkaltainen maasto toimisi, jos peli ja siten myös vaatimukset olisivat erilaiset.

Suorituskyvyllisesti arvioiden tämä menetelmä olisi lähes täydellinen. Juurikaan laskentatehoa ei kuluisi maaston luontiin eikä käyttäjä ehtisi tylsistymään. Ulkonäöllisesti maasto näyttäisi räikeän selkeästi satunnaisgeneroidulta.

6.2 Satunnaiset leikkaukset

Ensimmäinen testaamani menetelmä oli täyttää pelialue maalla ja sen jälkeen käyttäen satunnaisenkokoisia ellipsejä poistaa maastosta osia. Tarkoituksena on saada noin puolet pelialueesta maastoksi. Maaston alalaidasta ei poistettu mitään, etteivät hahmot putoaisi heti pelialueelta. Peilikuvauksen ja ruohon lisäämisen jälkeen tällä menetelmällä generoitu maasto voi näyttää esimerkiksi kuvan 16 mukaiselta.

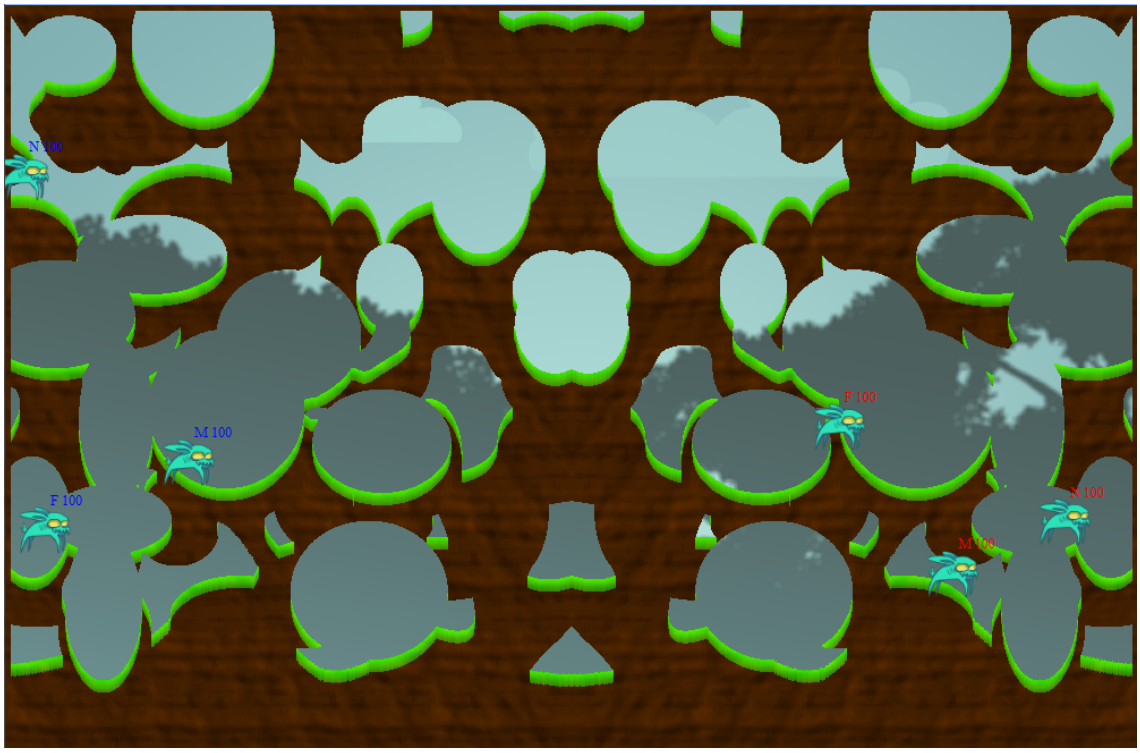
Kuva 16. Maastogenerointi satunnaisia leikkauksia käyttäen.

Pelimekaanisesti tällä menetelmällä generoidut maastot ovat hyvin vaihtelevia, ja osa on jopa hyvin mielenkiintoisia. Yksi tämän menetelmän ongelma tosin on se, että se jättää usein pelialueelle hyvin pieniä maakohtia. Maastoissa on myös usein liian suuria korkeuseroja. Tällä menetelmällä tehdyt maastot ovat jo silti pelikelpoisia. Suorituskyvyllisesti tämäkin menetelmä on riittävän kevyt.

Ulkonäöllisesti maasto näyttää aika paljon siltä mitä se on, eli siltä, että sitä on puhkottu ellipseillä. Maastojen päälle myöhemmin luotavalla ruoholla saadaan vähän kaunistettua maastoa ja saamaan se näyttämään vähemmän generoidulta, mutta ei se silti erityisen kaunis ole.

6.3 Täyttö ja leikkaus satunnaisilla muodoilla

Edelliseen algoritmiin voidaan tehdä muunnos siten, että aloitetaankin tyhjältä pohjalta ja täytetään maastoa muutamin satunnaisin ellipsein. Tämän jälkeen poistetaan maastosta satunnaisien ellipsien kokoisia palasia. Näin saadaan maastoja, jotka voivat olla esimerkiksi kuvan 17 mukaisia.



Kuva 17. Maastogenerointi satunnaisia täyttöjä ja leikkauksia käyttäen.

Pelimekaanisesti maastot ovat jälleen mielenkiintoisia. Niissä on enemmän yksityiskohtia kuin edellisissä, ja ne tarjoavat paljon strategisia mahdollisuuksia. Mutta myös irrallisten pikkuosien määrä on lisääntynyt. Pelimekaanisesti tällä ja edellisellä menetelmällä generoidut maastot ovat jonkin verran erilaisia, mutta mielestäni suhteellisen tasoissa. Se on hieman hitaampi kuin aikaisempi, mutta jälleen riittävän nopea, että se ei aiheuta mitään ongelmia.

Ulkonäöllisesti maasto näyttää karmivalta. Se on aivan liian täynnä tarpeettoman pieniä yksittäisiä ja omituisesti muodostuneita maapaloja tai leikkauksia.

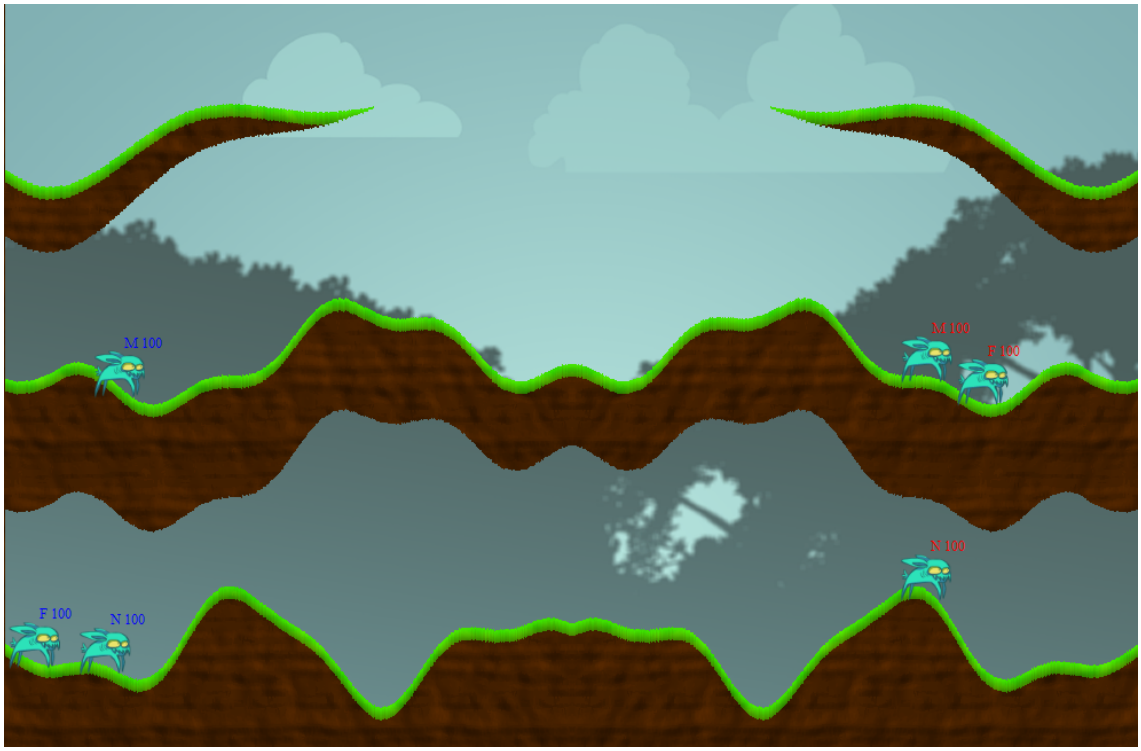
6.4 Päällekkäisiä eri taajuuksilla ja painoarvoilla olevia siniaaltoja

Pelimekaanisesti eräs haluttu elementti on maaston monikerroksisuus. Siitä tuli mieleeni kokeilla generoida maasto käyttäen kolmea päällekkäistä siniaaltoa.

Generointi aloitetaan ilman maastoa. Ensimmäiseksi luodaan maaston pohjasuikale. Tämä tapahtuu käyttämällä kolmea eri taajuuksilla ja painoarvoilla olevia siniaaltoja määrittämään maaston korkeuden.

Tämän jälkeen toinen kerros luodaan jokseenkin samaan tyyliin. Lisänä sen paksuus määritetään myös erillisellä sinifunktiolla ja sillä on tietty etäisyys alimmasta kerroksesta.

Viimeisenä on ylin kerros, joka luodaan samaan tyyliin kuin toinen kerros mutta ohuemmaksi. Tällä menetelmällä generoidusta maastosta voi tulla esimerkiksi kuvan 18 mukainen.



Kuva 18. Maastongenerointi käyttäen päällekkäisiä eri taajuuksilla olevia siniaaltoja.

Pelimekaanisesti monikerroksisuuden tavoite on jokseenkin saavutettu, mutta samalla maastossa on aivan liian vähän esteitä. Kuvan 3 maastossa pystyy ensimmäiselle vuorolla jo ampumaan ja osumaan vastustajan hahmoihin, eikä se tarjoa oikein mitään suojautumisen mahdollisuuksia. Tämä maasto tekee pelistä hyvin suoraviivaisen ja yksinkertaisen, mitä ei haluta.

Suorituskyvyllisesti tämä menetelmä on riittävän kevyt. Ulkonäöllisesti maasto on ehkä hieman liian säännöllisen näköinen, mutta jokseenkin luonnollisen oloinen silti.

6.5 Satunnaisista pisteistä koostuva käyrä

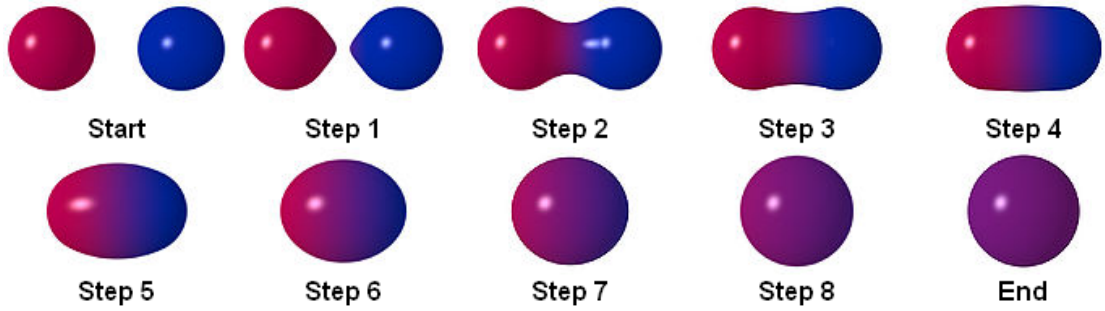
Hieman samantyylinen menetelmä edellisen kanssa on ensin generoida muutamia satunnaisella korkeudella olevia pisteitä. Tämän jälkeen pisteet yhdistetään jollakin miellyttävännäköistä jälkeä tekevällä käyräalgoritmilla. Sitten kaikki käyrän alla olevat pisteet voidaan laskea maaksi.

Pelimekaanisesti tässä on samat ongelmat kuin edellisessä. Varsinkin jos tehdään vain pohjamaasto, niin pelistä tulee liian yksinkertainen. Suorituskyvyllisesti mitään

ongelmia ei ole. Ulkonäöllisesti maasto sopisi varmaankin hyvin moniin peleihin. Tässäkään pelissä se ei näyttäisi huonolta, mutta saattaisi jättää paljon ylimäärästä tilaa ja voisi näyttää liian yksinkertaiselta.

6.6 Satunnaiset metapalloleikkaukset

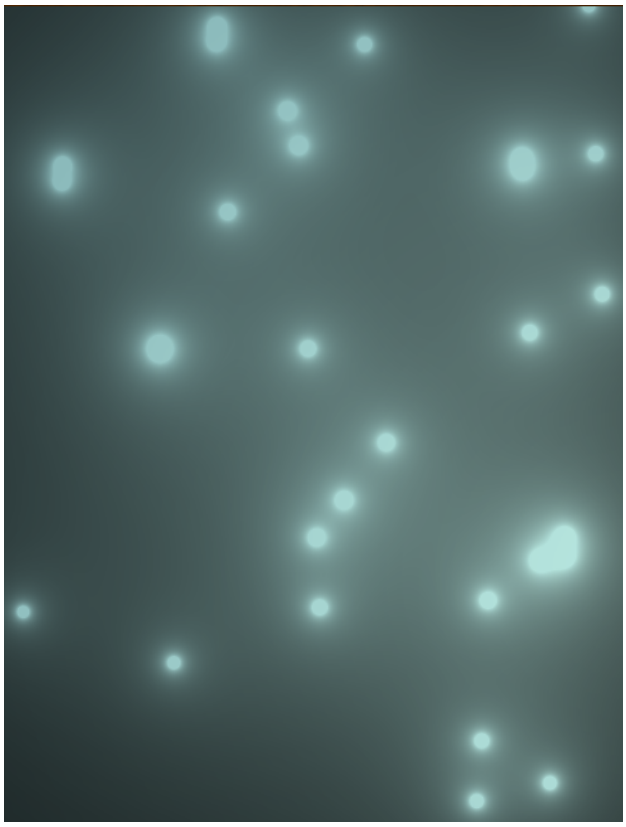
Metapallot ovat kuvassa 19 esitettyjä orgaanisen näköisiä kappaleita. Niitä käytetään yleensä kaksi- tai kolmiulotteisessa maailmassa, mutta algoritmin kannalta ne toimivat riippumatta ulottuvuuksien määrästä. Ne keksittiin 1980-luvun alussa Jim Blinnin toimesta. [24.]



Kuva 19. Kahden metapallon läheneminen ja yhteensulautuminen. [25.]

Tässä työssä käytämme kaksiulotteisia metapalloja. Algoritmi niihin on suhteellisen yksinkertainen.

Ensin asetellaan esimerkiksi täysin satunnaisesti haluttu määrä metapalloja ruudulle. Sitten käydään kaikki metapallot läpi ja lasketaan yhteen jokaiselle ruudun pikselille arvo kaavalla: 1 / etäisyys_metapallosta. Näin saadaan esimerkiksi kuvan 20 kaltainen kuva. [24.]



Kuva 20. Metapallot ilman suodatusta

Tämän jälkeen päätetään jokin vaadittu alaraja-arvo, jonka yläpuolella olevat pikselit ovat näkyviä [24.]. Tässä tapauksessa täydestä maastosta leikataan näkyvien metapallojen osoittama alue. Näin saadaan peilauksen kanssa kuvan 21 maasto.

Kuva 21. Metapalloilla generoitu maasto.

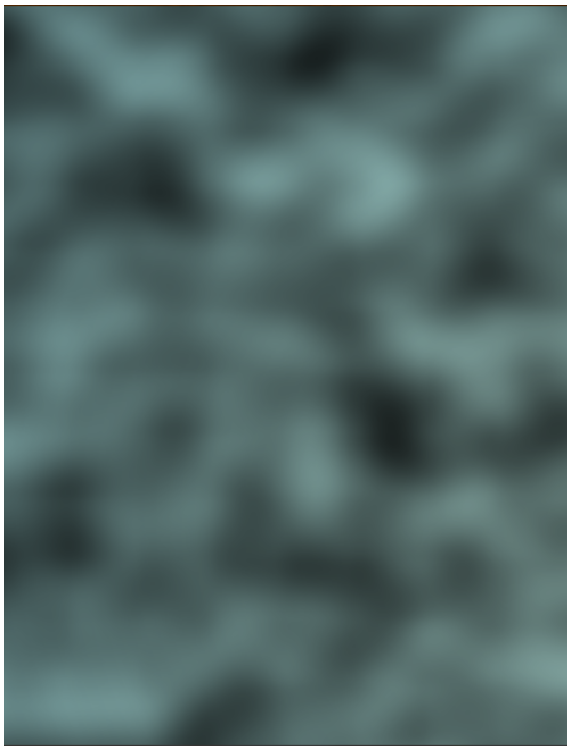
Pelimekaanisesti maaston isoin ongelma on isot tyhjät alueet. Ne eivät tarjoa suojaa tai juurikaan eri taktisia liikkumismahdollisuuksia. Yritin tehostaa menetelmää erilaisin tekniikoin. Pelimekaanisesti maasto parani silti vain hieman ja ulkonäöllisesti se kärsi pahasti. Mahdollisella lisätutkimuksella tästä menetelmästä voisi saada silti käyttökelpoisen.

Suorituskyvyllisesti tämä on raskain testatuista menetelmistä. Firefox-selaimella ajettuna 2,6 GHz Phenom II X4 -prosessorilla varustetulla Windows-koneella maaston generointi kesti noin 2 sekuntia, joka on jo sillä rajalla, että käyttäjä ei haluaisi odottaa niin kauan ja turhautuu.

Ulkonäöllisesti maastosta tulee oikein miellyttäviä pyöreitä muotoja. Peilikuvaa lukuun ottamatta maasto ei näytä edes välttämättä siltä, että se olisi generoitu.

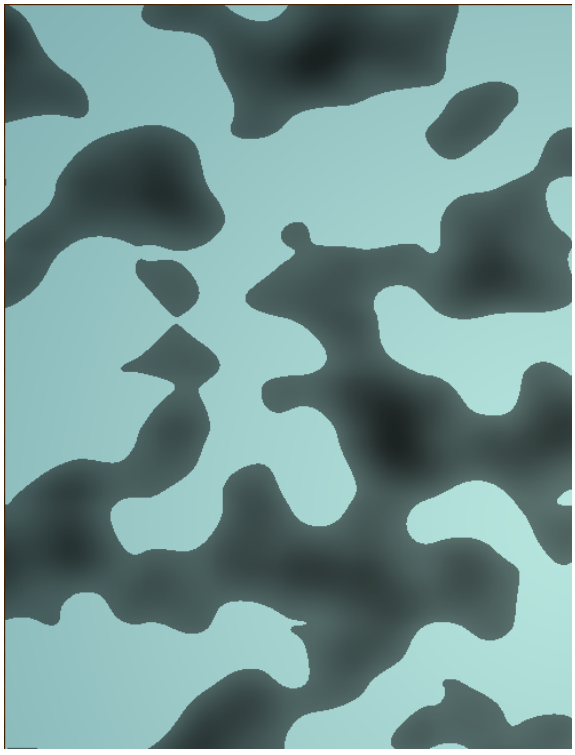
6.7 Perlin-kohina

Luvusta 2 tuttua Perlin-kohinaa käytetään tässä maastona siten, että ensin generoidaan kuvan 22 mukainen puolikkaan pelialueen kokoinen bittikartta.



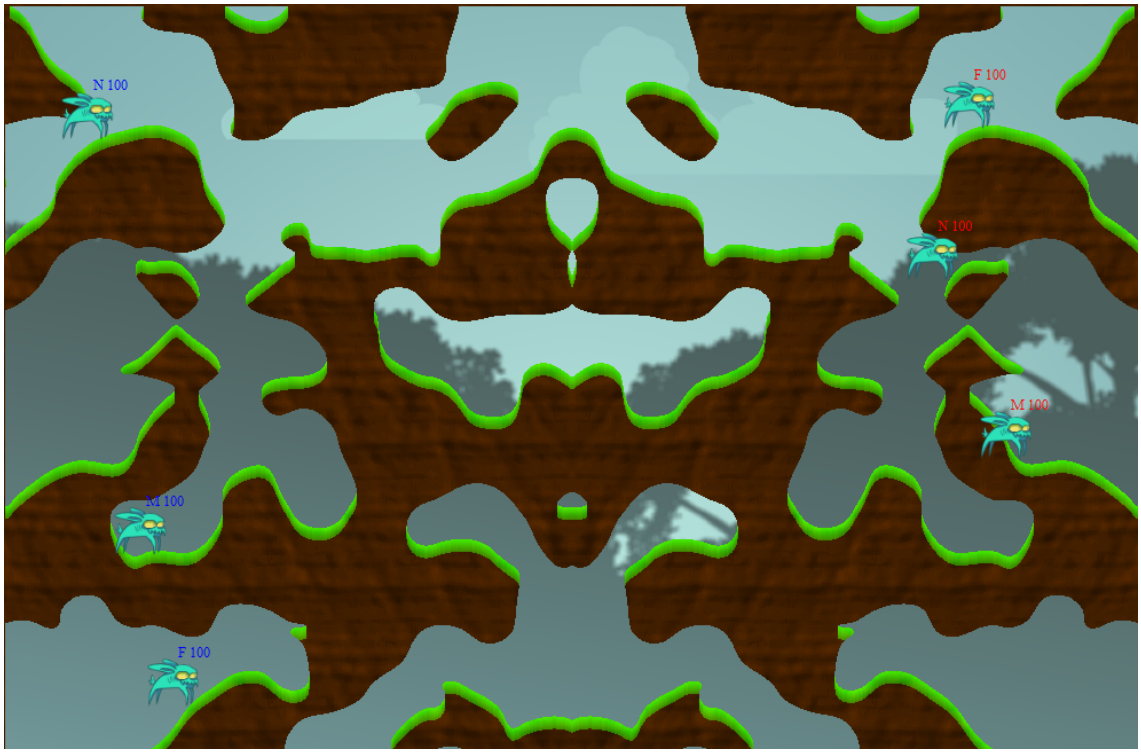
Kuva 22. Perlin-kohinaa ilman suodatusta.

Tämän jälkeen tyhjennetään kaikki väripaletin puoliväliä vaaleammat kohdat. Näin saadaan kuvan 23 mukainen bittikartta.



Kuva 23. Tummat osiot Perlin-kohinasta.

Sen jälkeen voidaan kaikkiin jäljellä oleviin kohtiin laittaa maastotekstuuria ja jälleen peilata pelialue tasapuolisuuden vuoksi. Näin saadaan kuvan 24 mukainen pelialue.



Kuva 24. Maastogenerointi Perlin-kohinaa käyttäen.

Pelimekaanisesti maastossa on riittävästi yksityiskohtia, mutta ei silti liikaa pieniä ja häiritseviä sellaisia. Joukkueitten välissä on myös riittävästi esteitä, että heti ei pysty vahingoittamaan vastustajia. Isoin puute maastossa on se, että alareunasta voi pudota helposti. Tämä voi olla myös mielenkiintoista ja saattaisi tarjota kokeneemmille pelaajille lisähaastetta, mutta pääosalle pelaajista luulisin sen aiheuttavan epäreilulta tuntuvia putoamiskuolemia.

Suorituskyvyllisesti tämä menetelmä tuottaisi normaalisti ongelman. Algoritmi on sen verran raskas, että tämänkokoisen kartan generoiminen ActionScript 3:lla olisi liian hidasta. Onneksi ActionScript 3 sisältää optimoidun toteutuksen Perlin-kohinasta ja näin ollen mahdollistaa sen käytön suorituskykyvaatimukset täyttäen.

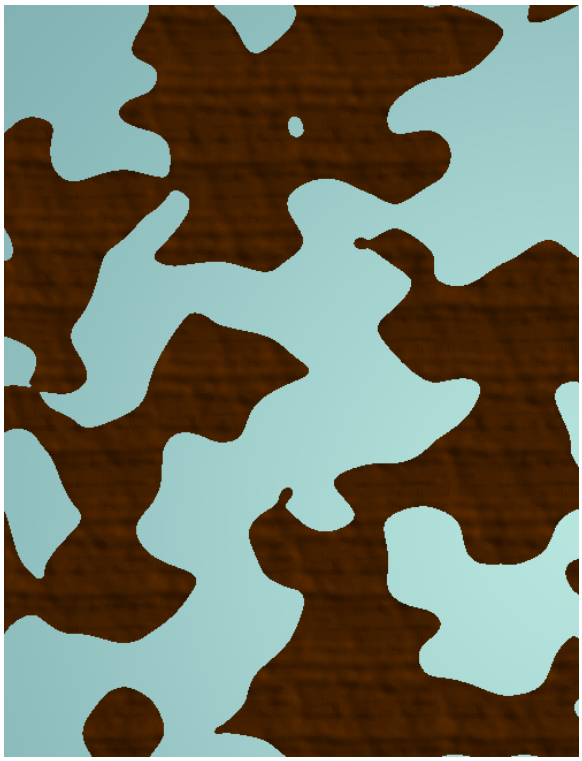
Ulkonäöllisesti maasto on hyvin orgaanisen näköinen ja sopivan monimuotoinen. Maasto ei välttämättä näytä siltä, että ihminen olisi sen tehnyt, mutta jossain määrin sellaiselta, mitä saattaisi luonnossa jossain olla (poislukien leijuvat maankaistaleet).

7 Implementaatio

Yksikään maaston generointitekniikka ei tarjonnut kaikkea mitä haluttiin. Tästä syystä Monster Conflict -pelissä tapahtuva maastongenerointi on yhdistelmä eri tekniikoita.

7.1 Maastonmuodot

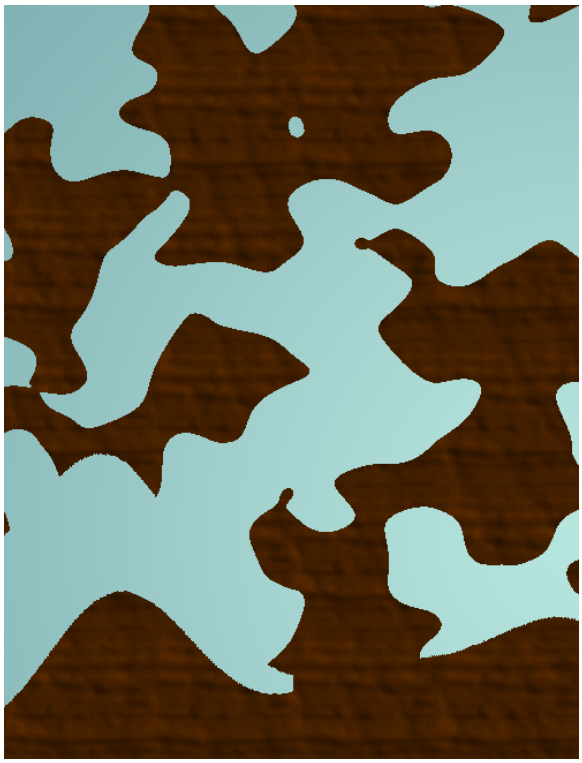
Ensimmäiseksi luodaan maaston perusmuodot käyttäen Perlin-kohinaa kuvan 25 mukaisesti. Näin meillä on mukavan orgaanisen näköinen perusmaasto valmiina.



Kuva 25. Maastongeneroinnin ensimmäinen vaihe.

7.2 Pohja

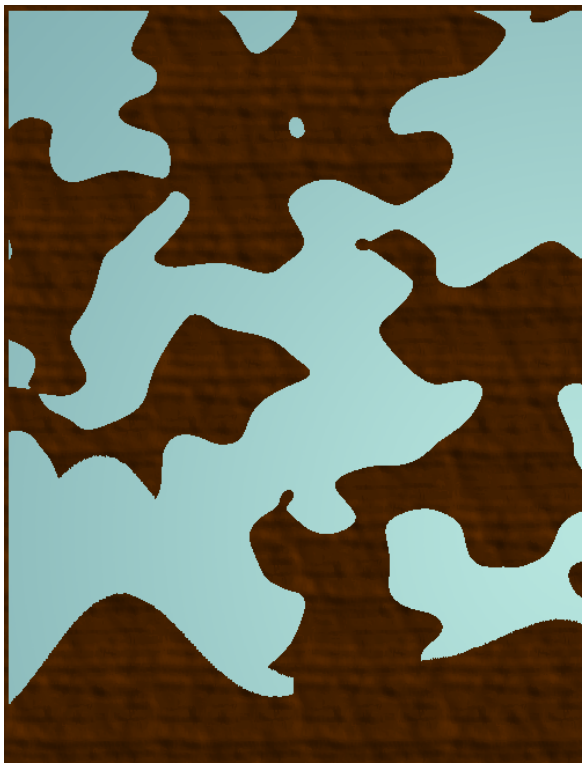
Maaston pohjaan käytetään siihen hyvin soveltuvaa siniaaltojen avulla luotua maastoa. Tässä tapauksessa tarvitaan vain yksi maastosuikale. Sama tyyli on käytössä kuin kohdassa 5.4, mutta tässä tilanteessa on yksi ylimääräinen vaatimus. Maaston pohjan päällä tulee olla riittävästi tyhjää tilaa hahmoille. Maata siis lisätään käyrän alle ja poistetaan sen päältä. Tulos on kuvan 26 mukainen.



Kuva 26. Maasto pohjan lisäyksen jälkeen.

7.3 Reuna ja katto

Nyt maastossa on paljon muotoja ja selkeä pohja. Jotta pelihahmoja ei pystyisi niin helposti työntämään ulos ruudulta ja, että pelialueen rajat olisivat selkeämmät, niin tämän jälkeen luodaan ohut reuna ja katto kuvan 27 mukaisesti.

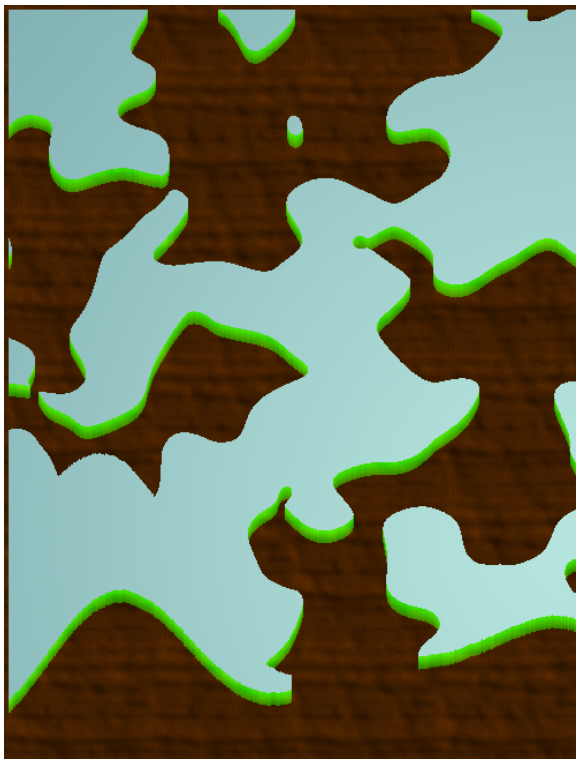


Kuva 27. Maasto reunan ja katon lisäyksen jälkeen.

7.4 Ruohon lisäys

Tässä vaiheessa maasto on pelimekaanisesti valmis. Siitä saadaan kuitenkin paljon mukavamman näköinen lisäämällä ruohoa kaikkiin ylöspäin osoittaviin pintoihin. Tämä tapahtuu seuraavasti.

Maaston jokainen pikselisarake tutkitaan ylhäältä alaspäin. Kun törmätään kohtaan, jossa tyhjyys vaihtuu maastoksi, niin tyhjyyden tilalle laitetaan yhden pikselin verran ruohoa. Sen jälkeen seuraavat kymmenen maastopikseliä väritetään vihreämmäksi. Lähempänä pintaa olevat pikselit väritetään voimakkaammin vihreäksi. Näin maastosta saadaan kuvan 28 mukainen.



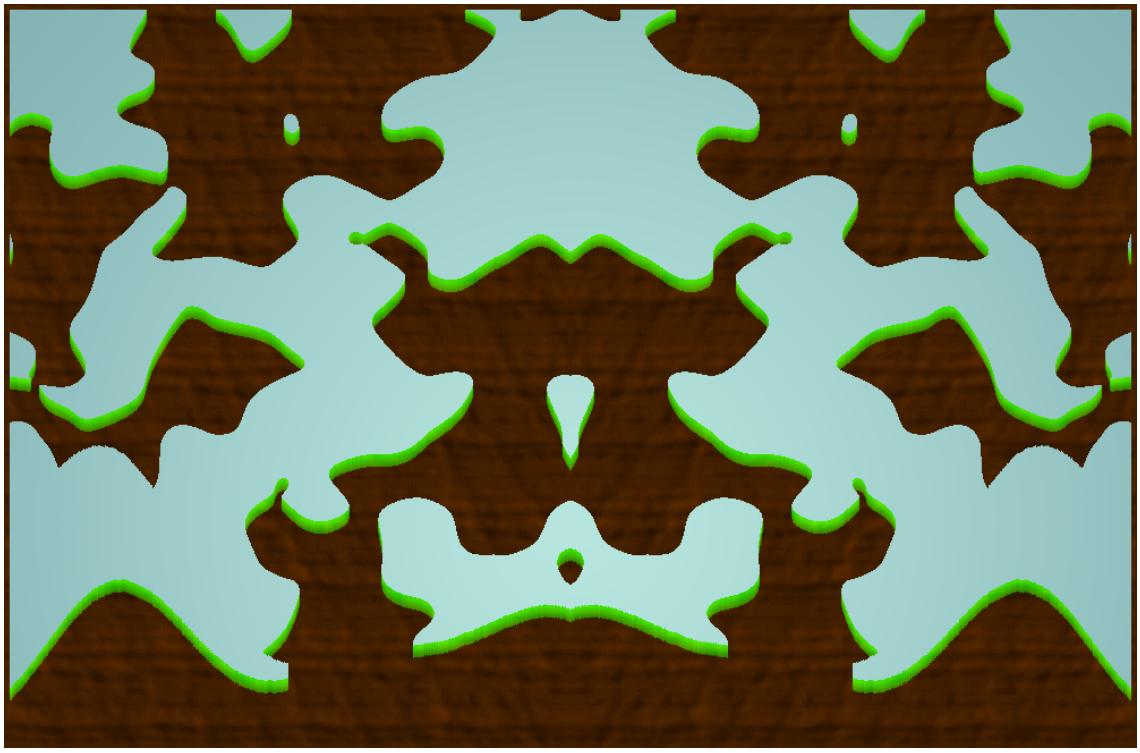
Kuva 28. Maasto ruohon lisäyksen jälkeen.

7.5 Kivien ja muiden asioiden asettelu

Tässä vaiheessa voitaisiin halutessa lisätä kiviä ja muita esteitä. Ne voisivat jopa olla täysin tuhoutumattomia. Toistaiseksi maasto on päätetty jättää nykytilaansa. Mahdollisissa tulevissa versioissa saatetaan lisätä nämä.

7.6 Peilikuvastaminen

Tähän mennessä maastosta on tehty vasta vasen puoli. Jotta saataisiin selvästi tasapuolinen maasto molemmille joukkueille, niin maaston toinen puoli tehdään ottamalla peilikuva tähän mennessä tehdystä puolesta. Näin saadaan kuvan 29 mukainen valmis maasto.



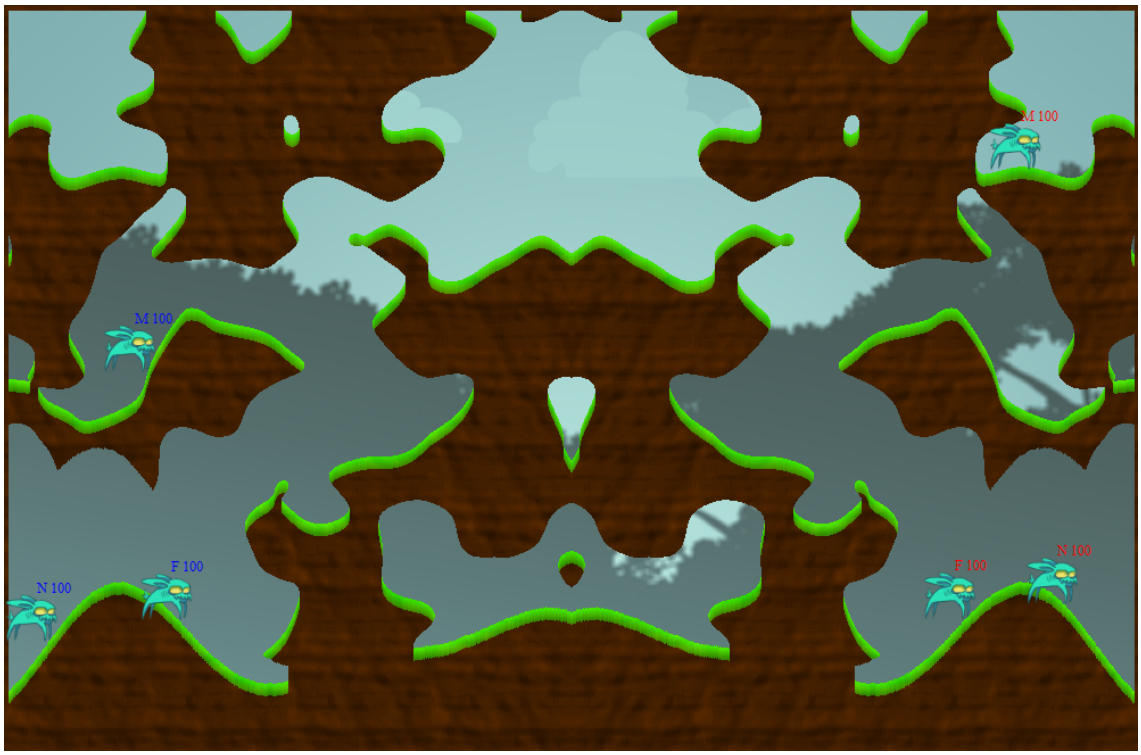
Kuva 29. Valmis maasto

7.7 Hahmojen asettelu

Ennen peliä on vielä jäljellä hahmojen asettelu. Hahmot asetellaan sattumanvaraisesti oman joukkueen puolelle. Sijoittelualgoritmi toimii seuraavasti.

Ensin valitaan sattumanvarainen x-koordinaatti väliltä [20, 250[. Sen jälkeen valitaan sattumanvarainen y-koordinaatti väliltä [50, PELIALUEEN_KORKEUS – 50[. Sitten nostetaan hahmoa yksi pikseli kerralla niin kauan, kuin hahmo on törmänneenä maahan. Tämän jälkeen vuorostaan lasketaan hahmoa pikseli kerrallaan, kunnes ollaan yhden pikselin päässä törmäyksestä.

Jos hahmo noston takia nousi yli pelialueelta tai jos toinen hahmo on 30 pikselin säteellä sijoitettavasta hahmosta, niin yritetään uutta sijoitusta valitsemalla uudet x- ja y-koordinaatit. Jos 500 yrityksen jälkeenkään hahmolle ei ole löytynyt sopivaa paikkaa, niin sallitaan hahmon olla toisen hahmon vieressä. Tämä yhdessä pohjamaaston yläpuolella olevan tyhjän alueen kanssa varmistaa sen, että peli ei jää jumiin loputtomaan silmukkaan. Kuvassa 30 on valmis maasto hahmoineen.



Kuva 30. Hahmot aseteltuna maastoon.

8 Tulokset

Maaston generointi onnistui oikein hyvin. Perlin-kohinan avulla saatiin yllättävän yksinkertaisesti sopivia maastonmuotoja.

8.1 Tasapuolisuus

Maaston peilaamisen avulla saadaan maasto, josta on helppo nähdä, että se on tasapuolinen molemmille puolille. Hahmojen satunnainen asettelu saattaa joissain tilanteissa aiheuttaa epätasapuolisuuksia, mutta se tekee myös pelistä mielenkiintoisemman. Yleensä ei pysty suoraan sanomaan kummalla puolella on etu pelissä, joten tasapuolisuuden tavoite on saavutettu.

8.2 Pelimukavuus

Maastossa on riittävästi yksityiskohtia, mutta ei liikaa, että ne häiritsisivät pelaamista. Myös erilaisia suojat ja eri korkeuksilla olevat maastonosat mahdollistavat monia mielenkiintoisia pelitilanteita. Pelimukavuuden puolesta on tavoitteet myös saavutettu.

8.3 Strateginen syvyys

Maaston monipuolisuus ja eri reitit mahdollistavat suuren määrän siirtoja, joita ei ainakaan heti pysty sanomaan huonoiksi. Maaston kaivuun avulla voidaan myös tehdä omia reittejä.

Jos pelissä siirtää hahmonsa aukealle ja näkyvälle paikalle ilman, että pystyy itse aiheuttamaan vahinkoa hyökkäyksellään, niin todennäköisesti vastustaja pystyy tätä hyödyntämään ja saamaan edun ensimmäisellä iskulla. Useimmiten ensimmäinen isku on pelissä tärkeä, mutta sekin riippuu paljon muun oman joukkueen asemasta.

Pelin voi yleensä jakaa kahteen vaiheeseen. Alkupelissä pelaajat yrittävät parantaa hahmojensa strategista sijaintia yleensä aiheuttamatta vahinkoa vihollisiin. Alkupeli loppuu yleensä, kun hahmo tekee ensimmäisen kerran vahinkoa vastustajaansa. Tämän jälkeen vastuksella on neljä vaihtoehtoa.

Ensimmäinen vaihtoehto on hyökätä takaisin hahmolla, johon hyökättiin. Muiden hahmojen sijainneista ja taistelussa olevien hahmojen tyypeistä riippuen tämä on yleensä jommallekummalle edullinen vaihto. Jos hyökkäyksen kohteeksi joutunut hahmo kokee kaksintaistelun epäedulliseksi, niin on hyvä miettiä muita vaihtoehtoja.

Toinen vaihtoehto on yrittää päästä karkuun. ”near”-hahmo voi mahdollisesti tehdä tämän käyttäen Rift Walk tai vain juoksemalla karkuun. Toista ”near”-hahmoa karkuun se ei silti voi päästä. ”mid”-hahmo ei suoraan yleensä pääse karkuun, mutta käyttämällä Wings se pääsee karkuun melkein mitä tahansa. Tämä tosin vie yhden vuoron eikä yleensä ole kannattavaa. ”far”-hahmo ei pysty juoksemalla pääsemään karkuun, mutta voi Dirt Ball käyttäen estää vihollishahmoa pääsemästä hyökkäämään itseensä.

Kolmas vaihtoehto on toisella hahmolla hyökätä ensimmäiseksi hyökänneeseen hahmoon. Jos omat hahmot on hyvin asteltu, niin tämä voi olla hyvin tehokas vaihtoehto.

Neljäs vaihtoehto on olla välittämättä hyökkäyksestä ja sen sijaan toisella hahmolla hyökätä tai edetä toista vihollishahmoa kohti. Tämän valitseminen on usein ihmisille vaikeata ja arviointi sen kannattavuudesta myös. Mutta oikein käytettynä se on hyvin tehokas vaihtoehto.

Generoidut maastot mahdollistavat monenlaiset vaihtoehdot ja tekevät vastustajan lukemisesta myös vaikeaa. Strategisen syvyyden tavoite on saavutettu.

8.4 Suorituskyky

Maaston luonnin nopeus on riittävä siihen, että käyttäjän ei tarvitse turhautua odottelussa. Huomattavasti nykytasoa hitaammilla koneilla saattaa joutua odottamaan enemmän kuin tavoitteena ollut sekunti, mutta silloin itse pelikin saattaa toimia jo hitaammin. Generointi toimii peliin suhteutettuna riittävän nopeasti, eli suorituskyvyn tavoite on siis saavutettu.

8.5 Ulkonäkö

Ulkonäöllisesti maasto näyttää orgaaniselta ja ruohikko tekee siitä luonnollisemman näköisen. Hiemankin graafista taitoa omaava ihminen pystyisi kuitenkin tekemään helposti ja nopeasti selvästi hienomman maaston. Siinä mielessä ulkonäölliset vaatimukset eivät ehkä täysin toteutuneet, mutta tulos on silti riittävä.

9 Yhteenveto

Tämän insinöörityön tavoitteena oli toteuttaa Monster Conflict -peliin maaston generaattori. Maaston generaattorin pitää pystyä luoman peliin sopivaa maastoa riittävän nopeasti.

Monia erilaisia maaston generointiin käytettävää menetelmää tutkittiin ja osan niistä toimivuutta testattiin käytännössä. Yksikään testatuista menetelmistä yksin ei tuottanut tyydyttävää tulosta, mutta yhdistelemällä eri menetelmiä saatiin maaston generaattori tuottamaan vaatimukset täyttävää maastoa.

Menetelmää voi aina silti parantaa. Ulkonäöllisesti maastot näyttäisivät paremmilta ilman peilikuvastamista, mutta tässä työssä pelimekaaniset hyödyt ajoivat ulkonäöllisten edelle. Jos maaston generointi voitaisiin toteuttaa ilman peilikuvastamista taaten ainakin lähes tasapuolisen maaston, niin se olisi selvä parannus.

Toinen mahdollinen parannusidea olisi olla käyttämättä metapalloja satunnaisesti. Ne voitaisiin asetella joko käyttäen jonkinlaista jakaumaa tai esimerkiksi hyödyntää Perlin-kohinaa tiheys- tai todennäköisyysfunktiona. Näin saataisiin miellyttäviä pyöreitä muotoja, mutta toivottavasti enemmän keskisuuria yksityiskohtia myös.

Lähteet

[1] Perlin noise. 2012. Verkkodokumentti. Wikipedia. <http://en.wikipedia.org/wiki/Perlin_noise>. Luettu 14.4.2012.

[2] Perlin-kohina. <http://en.wikipedia.org/wiki/File:Perlin_noise.jpg>. Viitattu 14.4.2012.

[3] Fraktaali Perlin-kohina. <http://en.wikipedia.org/wiki/File:Perlin.png>. Viitattu 14.4.2012.

[4] Martz, Paul. 1996. Generating Random Fractal Terrain. Verkkodokumentti. <http://gameprogrammer.com/fractal.html>. Luettu 14.4.2012.

[5] Keskipisteen siirto yhdelle viivalle. <http://gameprogrammer.com/fractal/mpd1.gif>. Viitattu 7.4.2012.

[6] Keskipisteen siirto kahdelle viivalle. <http://gameprogrammer.com/fractal/mpd2.gif>. Viitattu 7.4.2012.

[7] Keskipisteen siirto neljälle viivalle. <http://gameprogrammer.com/fractal/mpd3.gif>. Viitattu 7.4.2012.

[8] Salmiakki-neliö-algoritmin iteraatiovaiheet. <http://gameprogrammer.com/fractal/dsa.gif>. Viitattu 14.4.2012.

[9] Maasto salmiakki-neliö-algoritmin yhden iteraation jälkeen. <http://gameprogrammer.com/fractal/dsap1.gif>. Viitattu 14.4.2012.

[10] Maasto salmiakki-neliö-algoritmin kahden iteraation jälkeen. <http://gameprogrammer.com/fractal/dsap2.gif>. Viitattu 14.4.2012.

[11] Maasto salmiakki-neliö-algoritmin viiden iteraation jälkeen. <http://gameprogrammer.com/fractal/dsap5.gif>. Viitattu 14.4.2012.

[12] Procedural generation. 2012. Verkkodokumentti. Wikipedia. <http://en.wikipedia.org/wiki/Procedural_generation>. Luettu 14.4.2012.

[13] Minecraft. 2012. Verkkodokumentti. Wikipedia <http://fi.wikipedia.org/wiki/Minecraft>. Luettu 14.4.2012.

[14] Minecraft classic. <http://fi.wikipedia.org/wiki/Tiedosto:Minecraft_classic.png>. Viitattu 14.4.2012.

[15] Persson, Markus. 2011. Terrain generation, Part 1. Verkkodokumentti. <http://notch.tumblr.com/post/3746989361/terrain-generation-part-1>. Luettu 14.4.2012.

[16] NetHack. 2012. Verkkodokumentti. Wikipedia. <http://en.wikipedia.org/wiki/NetHack>. Luettu 14.4.2012.

[17] Roguelike. 2012. Verkkodokumentti. Wikipedia. <http://fi.wikipedia.org/wiki/Roguelike>. Luettu 14.4.2012.

[18] Dungeon generation? 2008. Verkkodokumentti. TIGForums. <http://forums.tigsource.com/index.php?topic=3917.15> Luettu 14.4.2012

[19] NetHack. <http://fi.wikipedia.org/wiki/Tiedosto:NetHack_for_Windows_Screenshot.png>. Viitattu 14.4.2012.

[20] Worms (series) . 2012. Verkkodokumentti. Wikipedia. <http://en.wikipedia.org/wiki/Worms_%28series%29>. Luettu 14.4.2012.

[21] Worms Armageddon. <http://en.wikipedia.org/wiki/File:WormsArmageddon1.PNG>. Viitattu 14.4.2012.

[22] Dwarf Fortress. 2012. Verkkodokumentti. Wikipedia. <http://en.wikipedia.org/wiki/Slaves_to_Armok_II:_Dwarf_Fortress>. Luettu 14.4.2012.

[23] Dwarf Fortress. <http://2.bp.blogspot.com/-0yeT--9fCgY/TlpSXE-2qiI/AAAAAAAABng/XRj-rPNWhS8/s1600/Dwarf_Fortress_Ascii.png>. Viitattu 14.4.2012.

[24] Metaballs. 2012. Verkkodokumentti. Wikipedia. <http://en.wikipedia.org/wiki/Metaballs>. Luettu 14.4.2012.

[25] Metapallojen törmyäksen vaiheet. <http://en.wikipedia.org/wiki/File:Metaball_contact_sheet.png>. Viitattu 15.4.2012.