Tomi Väisänen

Luolastomaisten pelikenttien proseduuralinen luominen hakuperusteisilla metodeilla

Tietotekniikan kandidaatintutkielma

30. huhtikuuta 2019

Jyväskylän yliopisto

Informaatioteknologian tiedekunta

Tekijä: Tomi Väisänen

Yhteystiedot: tomi.m.vaisanen@student.jyu.fi

Ohjaaja: Antti-Jussi Lakanen

Työn nimi: Luolastomaisten pelikenttien proseduuralinen luominen hakuperusteisilla metodeilla

Title in English: Procedural generation of dungeonlike levels in games using search-based methods

Työ: Kandidaatintutkielma

Sivumäärä: 33+0

Tiivistelmä: Proseduraalinen pelisisällön luominen tarkoittaa pelisisällön luomisen automatisointia proseduraalisten metodien avulla. Niitä käyttämällä voidaan esimerkiksi vähentää pelin kehityksen aiheuttamia kustannuksia sekä luoda monipuolisempia kenttiä. Tämän tutkielman tavoitteena on tutustua erilaisiin hakuperusteisiin metodeihin, joita voidaan käyttää luomaan luolastomaisia kenttiä peleihin. Tutkielman tuloksena huomattiin, että kenttien luomiseen joudutaan käyttämään useita eri metodeja. Lisäksi voidaan todeta, että proseduraalisesti luodut kentät eivät ainakaan vielä voi korvata huolella suunniteltuja kenttiä kaikissa tilanteissa, kuten pelitekniikoiden opettamisessa.

Avainsanat: Proseduuralinen sisällön luominen, pelit, luolasto, hakuperusteiset metodit

Abstract: Procedural game content creation is automating game content creation by using procedural methods. By using these methods developers can reduce the development costs and create diverse levels. The aim of this study is to explore search-based methods that can be used to create dungeonlike levels for videogames. In this study it was found that methods must be used with each other in order to create game levels. Also, it can be stated that procedural game levels cannot replace carefully crafted levels in games yet because they cannot for example teach the game mechanics as well to the player.

Keywords: Procedural content generation, PCG, games, dungeon, search-based methods

Kuviot

Sisältö

1 Johdanto

Peleiltä odotetaan päivä päivältä parempaa suorituskykyä ja graafista laatua, sekä vaaditaan, että jokaiselle löytyisi jotakin mieleistä pelattavaa. Tämä yleisen laadun sekä monipuolisuuden ylläpitäminen on kuitenkin usein hankalaa ja voi käydä kalliiksi, sillä siihen tarvittavan työvoiman löytäminen, kuten kaiken hyvän työvoiman löytäminen yleensäkin, voi joskus olla sekä kallista että vaikeaa. Tarvittavaa työvoimaa, ja siten kustannuksia, voitaisiin kuitenkin vähentää, jos osa kehityksestä automatisoitaisiin jollakin tavalla, jolloin kaikkea työtä ei tarvitsisi tehdä manuaalisesti (Togelius, Shaker ja Nelson 2016; Togelius ym. 2010). Yksi vastaus tähän ongelmaan on proseduraalinen sisällön luominen peleihin.

Proseduraalinen sisällön luominen peleissä tarkoittaa sitä, että pelisisällön luontia automatisoidaan erilaisia metodeja ja algoritmeja käyttäen, minkä jälkeen saadun sisällön laatu tarkastetaan tavalla tai toisella ennen sen sisällyttämistä peliin. Tällä tavoin peleihin saadaan luotua paljon kelvollista sisältöä vähällä työmäärällä, sillä kaikkea ei tarvitse tehdä manuaalisesti alusta alkaen. Erityisesti pienemmät kehittäjät ovat viime aikoina suosineet vahvasti proseduraalista sisällön luomista käyttäviä pelejä, mutta myös jotkin suuremmat yritykset käyttävät proseduraalisia tekniikoita esimerkiksi joidenkin suurien pelien osien luomiseen.

Proseduraalisesta pelisisällön luomisesta voi olla apua muussakin, kuten esimerkiksi tallennustilan säästämisessä. Pelien laadun nousu tarkoittaa myös usein sitä, että ne vaativat aiempaa enemmän muistitilaa, mikä voi olla ongelmallista, jos saatavilla oleva muistitila on vähäistä. Tähänkin voidaan yrittää vaikuttaa proseduraalisella sisällön luonnilla, sillä silloin valmiin sisällön sijaan voidaan tallentaa sisältöä tuottava metodi, jolloin sisältö voidaan tuottaa myöhemmin uudelleen. Näin uudelleen tehtävissä olevaksi tallennettu sisältö voi viedä huomattavasti vähemmän tilaa kuin valmiina tallennettu (Togelius, Stanley ja Browne 2011; Togelius ym. 2010). Tämän lisäksi proseduraalinen sisällön tuottaminen voi toimia inspiraationa ihmiskehittäjille tuoden samalla monipuolisuutta luotuun sisältöön, esimerkiksi erilaisten älykkäiden työkalujen kautta (Togelius, Shaker ja Nelson 2016). Näitä älykkäitä työkaluja ovat esimerkiksi Sentient Sketchbook (Liapis, Yannakakis ja Togelius 2013) sekä tässä tutkielmassa tarkemmin tarkasteltava Game Forge (Hartsook ym. 2011).

Tässä tutkimuksessa keskitytään luolastomaisten kenttien proseduraaliseen luomiseen hakuperusteisilla metodeilla peleissä, jotka koostuvat pääosin tämän tyyppisistä kentistä. Tutkimuksen tavoitteena on selvittää, ovatko näin luodut kentät hyviä ja miten niitä voitaisiin mahdollisesti parantaa, sekä selvittää mitä erilaisia metodeja tällaisten kenttien luomiseen voidaan hyödyntää. Lisäksi halutaan yrittää saada selville, onko tämä edes kannattavaa nykyisillä metodeilla vai pitääkö uusia metodeja luoda, jotta saataisiin tulokseksi tarpeeksi hyviä kenttiä.

Tämä tutkimus tehdään kirjallisuuskartoituksena, sillä aiheesta on jo tehty runsaasti erilaisia tutkimuksia, kuten monista kokoavista tutkimuksista voidaan nähdä (Hendrikx ym. 2013; Togelius, Stanley ja Browne 2011), ja niiden kokoaminen yhteen paikkaan olisi monin tavoin hyväksi. Ensinnäkin eri lähestymistapojen kokoaminen yhteen paikkaan selkeyttää sitä, mitä kaikkea on jo aiemmin tutkittu. Tämän ansiosta aiheesta on vaivattomampaa tehdä jatkotutkimuksia tai aivan uusia tutkimuksia myöhemmin. Toiseksi eri lähestymistapojen kerääminen yhteen paikkaan auttaa niiden vertailussa keskenään. Näin on helpompaa saada selville, mitkä lähestymistavat ovat toisia parempia tietyissä asioissa, jolloin voidaan suoraan alkaa käyttämään metodia, joka on haluttuun tarkoitukseen mahdollisimman sopiva. Näiden lisäksi metodien kerääminen yhteen paikkaan voi auttaa aiheesta kiinnostuneita tutustumaan aiheeseen huomattavan paljon helpommin.

Tämä tutkimus koostuu neljästä luvusta ja niiden aliluvuista. Luvussa 2 käsitellään sitä, mitä luolastomaiset kentät peleissä ovat ja millaisissa peleissä niitä käytetään, sekä esitellään muutama peli esimerkkinä tällaisten kenttien hyödyntämisestä. Luvussa 3 käsitellään proseduraalisten metodien ominaisuuksia ja muutamia hakuperusteisia metodeja, joilla luolastomaisia kenttiä voidaan luoda peleihin, sekä perustellaan tutkimuksen rajausta hakuperusteisiin metodeihin. Lisäksi yritetään tunnistaa parhaat käyttötarkoitukset kullekin metodille luolastomaisten kenttien luonnissa. Lopuksi luvussa 4 kootaan tutkimuksessa käydyt pääasiat yhteen, käsitellään tutkimuksessa saatuja johtopäätöksiä ja mahdollisia konkreettisia tuloksia sekä annetaan jatkotutkimusehdotuksia.

2 Luolastomaiset kentät peleissä

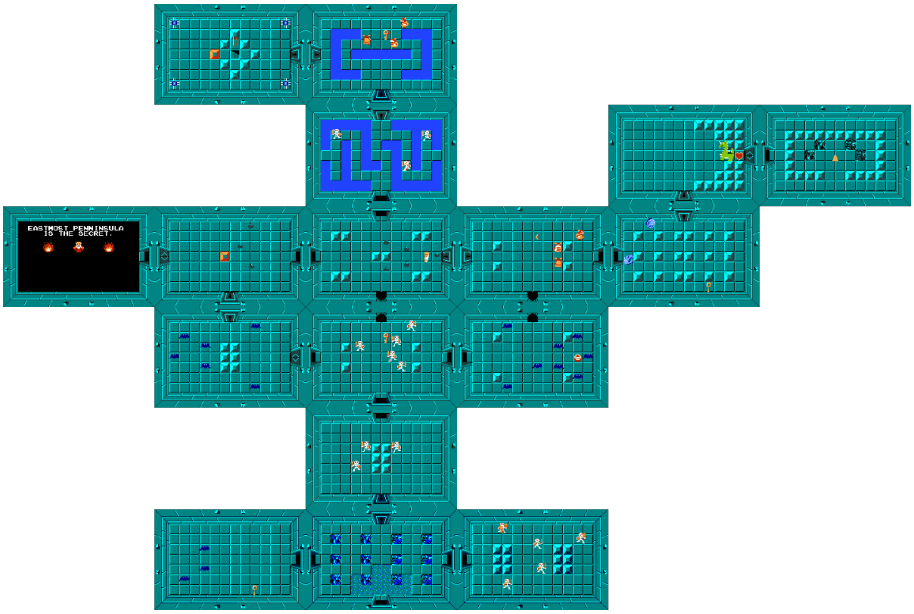
Luolastomaisia kenttiä voi esiintyä monissa erilaisissa peleissä. Tässä luvussa käydään läpi se, mitä luolastomaiset kentät ovat ja millaisista peleistä niitä löytyy. Tarkoituksena on toisin sanoen selkeyttää luolastomaisten kenttien luonnetta, jotta niiden luomiseen käytettyjen proseduraalisten metodien esittely on myöhemmin helpompaa. Lisäksi selvitetään, miksi juuri luolastomaisten kenttien proseduraalinen generointi ja sen tutkiminen on tärkeää.

Luolastomaiset kentät ovat nimensä mukaan luolamaisia ja koostuvat yleensä monista käytävillä yhdistetyistä huoneista, joista voi löytää sekä vihollisia että aarteita (Dahlskog, Björk ja Togelius 2015). Tällaisia erilaisilla palkinnoilla sekä vaaroilla täytettyjä luolastomaisia kenttiä voidaan myös käyttää hyväksi monissa erilaisissa peleissä esimerkiksi suuremman kentän osana eikä ainoastaan kokonaisena kenttänä. Esimerkiksi suurissa roolipeleissä yleensä tutkitaan tällaisia luolastoja, jotka ovat osana suurempaa aluetta, mutta jotkut pelit puolestaan käyttävät kenttinään ainoastaan luolastoja.

Kaikki luolastomaiset kentät eivät kuitenkaan ole keskenään samanlaisia, vaan niiden sisällä on olemassa muutamia tunnistettavia kenttätyyppejä. Käytetyimpiä näistä ovat ehkä keskenään kytkeytyneet huoneet, jolloin yhdestä huoneesta siirrytään suoraan seuraavaan, sekä käytävillä yhdistetyt huoneet (Dahlskog, Björk ja Togelius 2015). Hyvä esimerkki keskenään kytkeytyneistä huoneista voidaan nähdä pelissä nimeltään The Legend of Zelda (Nintendo 1986), josta kuvio 1 on peräisin. Vastaavasti esimerkki käytävillä yhdistetyistä huoneista voidaan nähdä kuviossa 2, joka on peräisin pelistä Rogue (Wichman, Toy ja Arnold 1980). Näiden lisäksi tunnistettavissa ovat esimerkiksi labyrintit ja sokkelot, joista molemmat koostuvat lähes kokonaan pelkistä käytävistä, mutta niiden erona on se, että sokkeloissa loppuun voi päästä monia reittejä pitkin ja labyrinteissä ainoastaan yhtä (Dahlskog, Björk ja Togelius 2015). Jotkin luolastot ovat myös hieman avarampia eikä niissä juurikaan esiinny käytäviä, vaan pelkästään useita toisiinsa liittyneitä avaria huoneita (Dahlskog, Björk ja Togelius 2015).

Kuten monista erilaisista luolastomaisten kenttien kenttätyypeistä huomataan, voidaan niitä käyttää hyödyksi monissa erilaisissa tilanteissa, joten niiden luomiseen käytettävien prose-

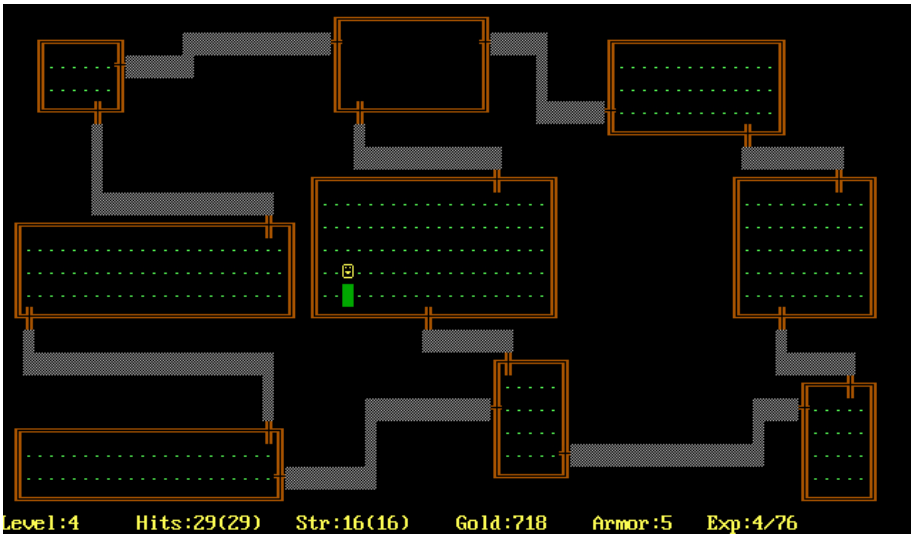
duraalisten metodien tutkiminen on tämän takia sekä tärkeää että erittäin hyödyllistä. Aihetta käsittelevästä kirjallisuudesta myös selvisi, että luolastot, ja siten myös luolastomaiset kentät ovat yksiä parhaiten proseduraalisten kenttien luomiseen soveltuvista kenttätyypeistä (Linden, Lopes ja Bidarra 2014). Ehkä juuri tämän takia monet proseduraalista sisällön luomista hyödyntävistä peleistä käyttävät kenttinään juurikin luolastomaisia kenttiä.



Kuvio 1. Toisiinsa kytkeytyneet huoneet pelissä The Legend of Zelda (1986)

2.1 Luolastomaisten kenttien esiintyminen peleissä

Yksi esimerkki luolastoja kenttinään käyttävistä peleistä ovat luolastoseikkailuihin (*engl. Dungeon Crawl*) kuuluvat pelit. Niillä tarkoitetaan pelejä, joissa pelaaja seikkailee erilaisissa luolastoissa tai luolastomaisissa kentissä taistellen vihollisia vastaan ja keräten aarteita. Luolastoseikkailujen ideana on tutkia luolastoja edetäkseen ja löytääkseen mahdollisia aarteita, joiden avulla luolaston erilaisten vihollisten voittaminen voi olla helpompaa. Tutkittuaan luolaston voi pelaaja yleensä edetä uuteen tutkimattomaan luolastoon, joka on edellistä haastavampi, mutta myös sisältää parempia aarteita. Sama voidaan toteuttaa myös yhtenä suurempana kenttänä, joka estää pelaajaa menemästä tiettyihin paikkoihin ennen kuin hän on edennyt tarpeeksi pitkälle ja löytänyt esimerkiksi uuden kyvyn tai esineen, joka mahdollistaa etenemisen aiemmin saavuttamattomissa oleviin paikkoihin.



Kuvio 2. Käytävillä yhdistetyt huoneet pelissä Rogue (1980)

Luolastoseikkailupelit on kuitenkin hyvin laaja käsite ja yleensä pelit kuuluvat moneen muuhunkin genreen tämän lisäksi. Lähes kaikki luolastoseikkailupelit kuitenkin hyötyisivät proseduraalisesta kenttien luomisesta hyvin paljon, sillä näin niistä on mahdollista tehdä periaatteessa loputtomia pelejä ja luolastoja pystytään parhaassa tapauksessa muokkaamaan pelaajakohtaisesti (Preuss, Liapis ja Togelius 2014), jolloin pelaajat eivät kyllästy niihin yhtä helposti. Tämän takia tutkitaan sitä, miten tällaisia kenttiä on mahdollista tehdä ja ennen kaikkea sitä, miten niitä kannattaa tehdä, jotta saataisiin paras lopputulos.

Eräänä luolastoseikkailujen osana voitaisiin pitää melko tunnettuja roguelike-pelejä, jotka käyttävät proseduraalista kenttien luontia hyväkseen. Voitaisiin myös sanoa, että tämä genre oli yksi ensimmäisistä, jotka käyttivät proseduraalista kenttien luontia hyväkseen (Ashlock, Lee ja McGuinness 2011a). Tämän genren pelit alkoivat Rogue (Wichman, Toy ja Arnold 1980) nimisestä pelistä, josta voidaan nähdä esimerkki kuviossa 2. Tämän genren ominaispiirteisiin kuuluu proseduraalisen generoinnin ja luolastomaisten kenttien lisäksi muun muassa se, että pelaaja ei voi jatkaa peliä häviön jälkeen, vaan hänen täytyy aloittaa peli alusta jokaisen häviön jälkeen. Koska pelaaja joutuu aloittamaan alusta, niin on erittäin tärkeää, että proseduraalisesti peliin luodut kentät ovat tarpeeksi hyviä ja kiinnostavia, jotta peliin ei kyllästy.

Viime aikoina proseduraalinen luolastomaisten kenttien luominen peleissä on kuitenkin nä-

kynyt parhaiten niin sanottuun rogue-lite genreen kuuluvissa peleissä, jotka ovat olleet hyvinkin suosittuja. Nämä pelit ottavat joitain roguelike pelien ominaisuuksia, mutta eivät yleensä esimerkiksi hävitä kaikkea edistymistä häviön jälkeen, vaan sen sijaan antavat pelaajan säilyttää joitain asioita seuraavalle pelikerralle. Toisin sanoen genre pohjautuu kevyesti roguelike genreen, mistä nimi rogue-lite on peräisin. Vaikka kaikki tähän genreen kuuluvat pelit eivät käytäkään luolastomaisia kenttiä, niin monet suosituimmista peleistä ovat kuitenkin puoliksi luolastoseikkailu pelejä, joten ne myös tapahtuvat luolastoissa. Seuraavaksi esitellään kaksi suosittua ja melko hyvin tunnettua peliä, jotka käyttävät proseduraalista sisällön luontia sekä luolastomaisia kenttiä hyväkseen.

2.2 Esimerkkejä peleistä

Nykyään on olemassa monia luolastomaisissa kentissä tapahtuvia pelejä, jotka ovat erittäin suosittuja ja käyttävät proseduraalista kenttien luomista apunaan. Tämän aliluvun tarkoituksena on tuoda esille joitain viime aikojen suosituimmista tai tunnetuimmista peleistä, jotka hyödyntävät proseduraalista kenttien luontia. Molemmat tässä esitellyt pelit kuuluvat aiemmin mainittuun rogue-lite genreen, sillä juuri tämä genren on ollut hyvin suosittu viime aikoina.

Eräs jo pitkään suosittu peli, joka käyttää proseduraalista kenttienluontia hyvin näkyvästi, on nimeltään The Binding of Isaac (McMillen ja Himsl 2011). Kyseinen peli käyttää kenttinään toisiinsa kytkeytyneitä samankokoisia ja -muotoisia huoneita, joiden ansiosta kenttien kokoaminen on helppoa. Tämä peli pitää pelaajan mielenkiinnon yllä antamalla pelaajalle mahdollisuuden avata uusia esineitä löydettäväksi seuraavalle pelikerralle tiettyjen ehtojen täyttymisen jälkeen, jonka ansiosta pelin lähes alusta aloittaminen kuolemisen jälkeen ei kyllästytä.

Toinen suosittu proseduraalista kenttienluontia käyttävä peli ilmestyi hiljattain, mutta sai heti monien huomion sen saamien hyvien arvostelujen perusteella. Kyseinen peli on nimeltään Dead Cells (MotionTwin 2017). Toisin kuin äsken mainittu peli, tämä antaa pelaajan avata monenlaisia vahvistuksia esimerkiksi aseisiin ja suojavarusteisiin, jotka säilyvät seuraavalle pelikerralle. Toisin sanoen, vaikka peli aloitetaan aina alusta, on pelaaja jokaisen pelikerran

jälkeen aiempaa vahvempi, minkä ansiosta alusta aloittaminen ei ole yhtä paha asia. Kuten The Binding of Isaac (McMillen ja Himsl 2011) myös Dead Cells (MotionTwin 2017) luo kenttänsä jokaisen kuoleman jälkeen proseduraalisesti ja käyttää siihen toisiinsa yhdistettyjä huoneita. Nämä huoneet ovat kuitenkin itsessään enemmän luolastomaisia kuin Binding of Isaacin (McMillen ja Himsl 2011) yksinkertaiset huoneet ja vaihtelevat kooltaan ja muodoltaan tuoden mieleen jotkin 90-luvun klassikkopelit.

Molemmat näistä peleistä saavat pidettyä pelaajan mielenkiinnon yllä juurikin vaihtuvien kenttiensä ansiosta, jotka pitävät pelaamisen aina jännittävänä siitä pitäville pelaajille. Tämän lisäksi molempien pelien ainutlaatuiset ilmapiirit ja pelimekaniikat antavat pelaajille syyn pelata lisää, mutta kumpikaan näistä peleistä ei kuitenkaan olisi mahdollinen ilman proseduraalista pelikenttien luomista, minkä takia siihen kannattaa perehtyä hieman tarkemmin. Seuraavaksi perehdytäänkin hieman enemmän proseduraaliseen kenttien luomiseen sekä muutamiin siihen käytettäviin metodeihin.

3 Proseduraalinen kenttien luonti ja sen metodit peleissä

Hendrikxin ym. (2013) mukaan proseduraalinen sisällön luominen peleihin on tietokoneiden käyttämistä pelisisällön luomiseen ja kyseisen sisällön laadun varmistamista, jottei luotaisi pelaajalle huonoa pelisisältöä. Proseduraalisten metodien tulisi ideaalisesti olla helposti hallittavissa sekä tuottaa aina hyvää tai vähintäänkin hyväksyttävää sekä monipuolista sisältöä nopeasti, mutta todellisuudessa nämä ovat vain ideaalisia tavoitteita, sillä kaikkia näitä on erittäin vaikeaa saavuttaa samaan aikaan (Togelius, Shaker ja Nelson 2016). Tällaisia metodeja voidaan käyttää peleissä luomaan lähes mitä tahansa aina pienistä yksityiskohdista, kuten tekstuureista, suurempiin kokonaisuuksiin, kuten esimerkiksi kokonaisiin kenttiin tai niiden osiin (Hendrikx ym. 2013; Togelius, Stanley ja Browne 2011). Tässä tutkimuksessa kiinnostuksen kohteena ovat erityisesti näin luodut kentät.

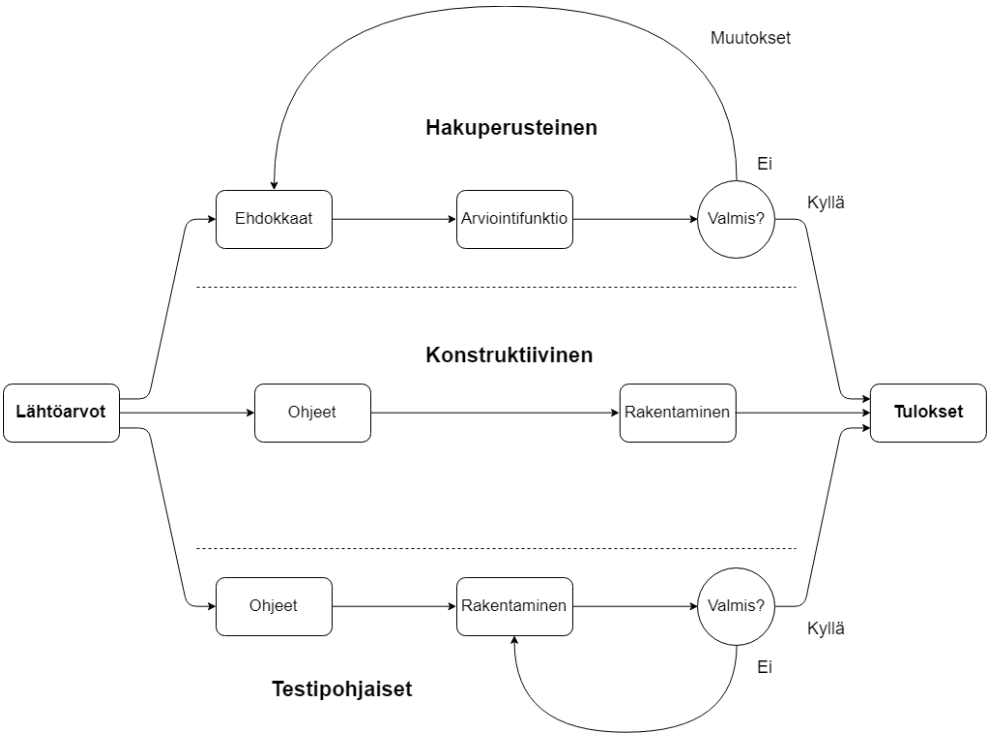
Peleissä esiintyy kuitenkin monia eri kenttätyyppejä ja näitä voidaan luoda monilla erilaisilla proseduraalisilla metodeilla, mutta tässä luvussa ja tutkimuksessa kuitenkin keskitytään erityisesti luolastomaisia kenttiä luoviin metodeihin. Suurinta osaa tässä tutkimuksessa esitellyistä metodeista voidaan käyttää myös muunlaisten kenttien tai muun sisällön luomiseen peleissä, mutta nämä käyttötarkoitukset sivuutetaan. Ennen metodien tarkastelua tulee kuitenkin perehtyä hieman siihen, millaisia ominaisuuksia proseduraalisilla metodeilla voi olla.

3.1 Metodien luokittelu ja ominaisuudet

Togelius, Shaker ja Nelson (2016, s. 7–10) esittelevät kirjassaan parantelemansa luokittelujärjestelmän proseduraalisille metodeille. Tämä järjestelmä koostuu seitsemästä vastakohtaparista, joiden jokaisen väliin voidaan kuvitella eräänlainen asteikko. Luokittelu tässä järjestelmässä toimii sijoittamalla metodit jokaisella asteikolla sopivaan kohtaan.

Tämän tutkimuksen kannalta kyseisestä luokittelujärjestelmästä (Togelius, Shaker ja Nelson 2016, s. 7–10) on tärkeintä kuitenkin ymmärtää se, mitä eri ominaisuuksia proseduraalisilla metodeilla voi olla. Ensinnäkin metodit voidaan jakaa sen mukaan millaista sisältöä ne luovat peleihin. Toiseksi jako voidaan tehdä sen suhteen, miten paljon parametreja niille voidaan antaa, toisin sanoen miten hallittavissa ne ovat. Metodit voidaan myös jakaa keskenään sen

mukaan mukautuvatko ne jotenkin esimerkiksi pelaajan pelitavan tai muun vastaavan asian mukaan vai eivät. Tämän lisäksi jotkin metodeista voivat luoda aina täysin saman sisällön samoista lähtöarvoista ja näitä metodeja kutsutaan deterministisiksi. Tälle vastakohtana olevat stokastiset metodit puolestaan eivät voi luoda samaa sisältöä uudestaan, sillä niillä halutaan saada aikaan enemmän tai vähemmän sattumanvaraisia tuloksia. Tärkein jako tämän tutkimuksen kannalta on jako konstruktiivisiin ja niin sanotusti testipohjaisiin (*engl. generate-and-test*) metodeihin, joiden ero on siinä, miten ne luovat sisältöä. Kyseinen jako myös havainnollistetaan kuviossa 3. Konstruktiiviset metodit luovat kaiken halutun sisällön yhdellä kerralla, samalla tarkastaen, että saatu sisältö on varmasti kelvollista. Vastakohtana näille ovat testipohjaiset metodit, jotka luovat kokonaisuuksia, jotka testataan vasta jälkikäteen ja tarpeen mukaan muutetaan saatua tulosta iteroiden. Viimeisenä metodien ominaisuuksista tulee tietää se, että jotkin metodit toimivat täysin itsenäisesti, kun taas toiset antavat kehittäjän tai jopa pelaajan vaikuttaa saatuun lopputulokseen luomisen aikana, jolloin lopputulos on myös huomattavasti paremmin hallittavissa, mikä tarkoittaa sitä, että niitä voidaan käyttää esimerkiksi älykkäissä kehitystyökaluissa.



Kuvio 3. Metodien jako konstruktiivisiin, testipohjaisiin sekä hakuperusteisiin (mukaillen Togelius ym. 2011)

3.2 Tutkimuksen rajaus

Tässä tutkimuksessa keskitytään hakuperusteisiin (*engl. search-based*) metodeihin, jotka ovat erikoistapaus testipohjaisista metodeista. Yleensä hakuperusteiset metodit ovat myös stokastisia, jolloin ne eivät pysty luomaan täysin samaa sisältöä aina uudestaan tarvittaessa. Tutkimuksen ulkopuolelle jäävät pääasiassa konstruktiiviset metodit, jotka luovat halutun sisällön kerralla valmiiksi, varmistaen samalla, että luotu sisältö on varmasti kelvollista, joten ne voivat käyttää vain tekniikoita, jotka tuottavat varmasti virheetöntä sisältöä (Togelius, Stanley ja Browne 2011). Rajaus tehtiin näin, sillä erityisesti hakuperusteisia metodeja on tutkittu erittäin paljon proseduraalisen sisällön luonnin yhteydessä viime aikoina (Togelius ja Shaker 2016). Kuviosta 3 voidaan havaita konstruktiivisten ja testipohjaisten metodien huomattavimpien erojen lisäksi se, miten hakuperusteiset metodit ovat erikoistuneet. Kyseinen kuvio perustuu Togeliuksen, Stanleyn ja Brownen (2011, s. 175) tutkimuksessaan esittämään jakoon sekä sitä kuvaavaan kuvioon. Seuraavaksi käsitellään hakuperusteisuutta hieman tarkemmin, jonka jälkeen esitellään muutamia esimerkkejä luolastomaisten kenttien luomiseen hyödynnettävistä hakuperusteisista metodeista.

3.3 Hakuperusteiset metodit

Hakuperusteiset (*engl. search-based*) metodit ovat metodeja, jotka ovat erikoistapaus testipohjaisista (*engl. generate-and-test*) metodeista. Testipohjaiset metodit luovat ehdokkaan halutulle sisällölle, jonka jälkeen sitä testataan ja arvioidaan halutulla tavalla, minkä jälkeen se joko hyväksytään tai hylätään. Hylkäyksen jälkeen osa tai kaikki hylätystä sisällöstä poistetaan ja tehdään uudestaan iteroiden, kunnes sisältö voidaan hyväksyä tai se hylätään lopullisesti liian monen yrityksen jälkeen (Togelius, Stanley ja Browne 2011; Togelius ym. 2010). Hakuperusteiset metodit erikoistavat tätä siten, että ne eivät ainoastaan hyväksy tai hylkää saatuja tuloksia täysin vaan ne arvioivat tuloksia ja antavat niille arvion esimerkiksi kokonaislukuvektorina, jonka jälkeen arviota voidaan käyttää uusien ehdotusten hakemiseen, jotta seuraavat ehdokkaat olisivat varmemmin edellisiä parempia (Togelius, Stanley ja Browne 2011; Togelius ym. 2010). Ero muodostuu siis siinä, että yksinkertaisemmat testipohjaiset metodit eivät huomioi hylättyä sisältöä lainkaan sen korvaamisen yhteydessä, sillä ne vain yksinkertaisesti päättävät, ettei saatu sisältö täytä haluttuja vaatimuksia tarpeeksi hyvin, jol-

loin hylätystä sisällöstä ei myöskään muodostu minkäänlaista arviota, jota voitaisiin käyttää apuna uuden sisällön luonnissa.

Hakuperusteiset metodit tarvitsevat jonkin algoritmin tai muun vastaavan menetelmän, jolla ne toteuttavat halutunlaisen sisällön hakemisen (Togelius ja Shaker 2016). Hyvin suosittu tapa toteuttaa tämä on evolutiivisten algoritmien (*engl. evolutionary algorithms*) käyttäminen uusien ehdokkaiden etsimiseen. Tällaiset algoritmit eivät poista huonoja osia parhaista ehdotuksista, vaan sen sijaan muuttavat niitä jollain satunnaisella tavalla (Togelius ym. 2010; Togelius ja Shaker 2016), jotta saataisiin uudenlainen ehdotus ratkaisuksi. Kyseiset algoritmit perustuvat evoluutioteoriaan, toisin sanoen aiemmin saadut ratkaisut kehittyvät pienten muutosten myötä uusiksi ratkaisuiksi (Togelius ja Shaker 2016). Kun tähän yhdistetään hakuperusteisten metodien periaate arvostelun käyttämisestä myös uuden ehdotuksen paranteluun, eikä ainoastaan huonojen erottamiseen, saadaan uudesta ehdotuksesta lähes varmasti tavalla tai toisella edellistä parempi. Hakuperusteisten metodien ei kuitenkaan tarvitse käyttää juuri evolutiivisia algoritmeja, vaan ne voivat aivan yhtä hyvin käyttää muitakin tarkoitukseen sopivia menetelmiä (Togelius ym. 2010), kuten esimerkiksi erilaisia optimointiin käytettyjä algoritmeja tai tapauksessa, jossa käytettävä hakuympäristö on hyvin pieni, voidaan se käydä vaikka kokonaan läpi satunnaishaun avulla (Togelius ja Shaker 2016).

Hakuperusteiset metodit tarvitsevat myös niin sanotun hakuympäristön, joka koostuu monista samanmuotoisista ratkaisuista, joiden joukossa kelvolliset ratkaisut ovat. Hakuperusteiset metodit käyttävät tätä hakuympäristöä hakeakseen sieltä sopivia ratkaisuja ja muuttavat näitä ratkaisuja päätyäkseen lopulta ratkaisuun, joka on tarpeeksi hyvä haluttuun tilanteeseen (Togelius ja Shaker 2016). Hakuperusteisia metodeja käytettäessä on näin ollen hyvin tärkeää valita käytettävä hakuympäristö hyvin, jotta metodin on mahdollista hakea ja esittää ratkaisu halutunlaisessa muodossa (Togelius ym. 2010). Etsittävän ratkaisun muoto tai esitystapa halutaan päättää mahdollisimman hyvin, sillä hakuympäristö riippuu hyvin vahvasti siitä, millaisessa muodossa ratkaisu halutaan esittää (Togelius ym. 2010). Toisin sanoen hakuympäristö on halutun muotoisten ratkaisujen joukko, josta juuri tietynlaisen ratkaisun etsintä aloitetaan, minkä takia ympäristö riippuu niin vahvasti etsittävän ratkaisun muodosta. Ratkaisun esitysmuoto voi olla esimerkiksi jonkinlainen taulukko tai vaikkapa sitten yksinkertainen merkkijono (Togelius ja Shaker 2016).

Tämän lisäksi hakuperusteisten metodien tärkeä osa on saatujen ehdotusten arviointiin käytettävä funktio tai funktiot, sillä juuri näiden perusteella metodit voivat parantaa saatuja tuloksia, mikä lopulta johtaa hyvään lopputulokseen (Togelius ym. 2010; Togelius ja Shaker 2016). Funktio kannattaa siis valita sen mukaan, mitä ominaisuutta tai ominaisuuksia sisällössä halutaan painottaa, kuten esimerkiksi monipuolisuutta tai yleistä pelattavuutta (Togelius ja Shaker 2016). Seuraavaksi esitellään muutamia luolastojen luontiin soveltuvia hakuperusteisia metodeja.

3.4 Luolastojen muodostaminen pieniä osia yhdistelemällä

Tässä aliluvussa esitellään kaksi eri metodia, joilla voidaan luoda luolastoja tai luolastomaisia kenttiä yhdistelemällä pienempiä kentän osia toisiinsa eri tavoin. Eräs yleinen tapa muodostaa luolastomaisia kenttiä on yhdistää jonkin algoritmin avulla valmiiksi tehtyjä tai valittuja huoneita tai muita mahdollisia luolaston osia toisiinsa enemmän tai vähemmän loogisella tavalla (Valtchanov ja Brown 2012). Valtchanov ja Brown (2012) tuovat kuitenkin esille kaksi ongelmaa, jotka tämä lähestymistapa saa aikaan. Yksi näistä ongelmista on lopputulokseen vaikuttamisen vaikeus, toisin sanoen tulokselta ei välttämättä voida vaatia kovin tarkkoja tai yksityiskohtaisia asioita. Toinen tähän lähestymistapaan liittyvä ongelma taas on tällaisten algoritmien huono yleistyvyys, joka pakottaa usein uuden algoritmin tekemiseen aina, kun halutaan tehdä erilaisia luolastoja (Valtchanov ja Brown 2012).

Valtchanov ja Brown (2012) yrittävät kuitenkin korjata nämä ongelmat käyttämällä evoluutioteoriaan perustuvaa niin sanottua geneettistä ohjelmaa (*engl. genetic program*) lähestymisessään. Tämä tarkoittaa sitä, että vain niin sanotusti parhaat yksilöt, jotka päätetään niiden saaman arvion perusteella, selviävät ja tässä tapauksessa kaikki huonot yksilöt korvataan hyvien yksilöiden muunnoksilla (Valtchanov ja Brown 2012). Näin ollen parhaat yksilöt vastaavat haluttuja ominaisuuksia mahdollisimman hyvin, jolloin lopputulokseen voidaan vaikuttaa huomattavan paljon enemmän, tarkemmin ja monipuolisemmin (Valtchanov ja Brown 2012). Arvioinnin perustana Valtchanov ja Brown (2012) käyttävät tutkimuksessaan mallia, jossa suositaan tiukasti toisiinsa pakattuja, tehokkaasti käytävillä yhdistettyjä ja monipuolisia huoneita, sekä kolmen erikoishuoneen läsnäoloa, joihin voidaan liittää esimerkiksi luolaston pomoja tai erilaisia tehtäviä. Tämä saavutetaan parantamalla arviota huomattavasti ai-

na kun tiettyä huonetyyppiä käytetään ensimmäisen kerran, käytävä yhdistää useita huoneita tai karttaan lisätään uusi erikoishuone (Valtchanov ja Brown 2012). Valtchanov ja Brown (2012) päättivät kuitenkin, että kolme erikoishuonetta on tarpeeksi, joten tämän rajan ylittäminen tiputtaa luolaston arvioinnin minimiin varmistaen, ettei tällaisia luolastoja hyväksytä.

Itse kartta ja huoneet kyseisessä tutkimuksessa esitetään alussa joustavasti puurakenteena, jossa jokainen solmu on huone ja solmujen väliset reitit vastaavat luolaston osia yhdistäviä ovia (Valtchanov ja Brown 2012). Tätä puurakennetta puolestaan parannetaan käyttämällä apuna evoluutiosta tuttuja yksilön mutaatiota sekä yksilöiden risteytystä, jotka tässä tapauksessa vastaavat puurakenteelle mahdollisia operaatioita (Valtchanov ja Brown 2012). Tässä tapauksessa risteytyksessä valitaan jokin puurakenteen alipuu ja vaihdetaan se toisen satunnaisesti luolastosta valitun alipuun kanssa, jolloin tiettyjä huonerykelmiä saadaan helposti siirrettyä parempaan paikkaan (Valtchanov ja Brown 2012). Mutaatiossa puolestaan tapahtuu yksi kolmesta mahdollisesta asiasta, joihin kuuluvat (1) satunnaisten huoneiden vapaana olevien ovien yhdistämisen uusiin huoneisiin, (2) satunnaisten huoneiden ovimäärien tai tyypin muuttaminen, sekä (3) lehtisolmussa sijaitsevien huoneiden poistaminen tai niiden vapaiden ovien yhdistämisen estäminen (Valtchanov ja Brown 2012).

Lopuksi tutkimuksen algoritmi muodostaa puurakenteesta kartan yhdistelemällä huoneet toisiinsa niiden ovien kohdalta, jolloin jotkin huoneet saattavat mennä päällekkäin, mutta tällaisissa tapauksissa viimeisimpänä sijoitettu huone ja kaikki muut siihen liittyneet huoneet hylätään ja poistetaan puusta (Valtchanov ja Brown 2012). Muodostamisen aikana saattaa myös muodostua uusia reittejä huoneiden välillä, jos niiden ovet sattuvat kohdakkain, ja vastaavasti jotkin ovista poistetaan, jos ne eivät yhdisty minnekään (Valtchanov ja Brown 2012). Lopuksi Valtchanov ja Brown (2012) päättivät poistaa kartasta sellaiset huoneistoryhmät, joihin pääseminen ylittää tietyn maksimimatkan.

Valtchanov ja Brown (2012) testasivat esittämäänsä metodia erilaisilla parametreilla, jonka tuloksena he saivat tehtyä kelvollisia sekä monipuolisia karttoja tehokkaasti. Tämä tarkoittaa sitä, että metodin avulla pystyttiin tuottamaan vaatimuksia vastaavia tuloksia luotettavasti, vaikka vaatimukset olisivatkin olleet hieman haastavampia. Lisäksi käytetty metodi on huomattavan yksinkertainen, jonka ansiosta useimmat voivat ymmärtää sen toiminnan hyvin. Näiden tulosten valossa Valtchanov ja Brown (2012) ovat sitä mieltä, että tätä metodia voitai-

siin käyttää esimerkiksi osana jotain yksityiskohtaisempaa kenttäeditoria, minkä takia myös valinta käyttää valmiita huoneita oli hyvä, sillä näin ollen valmiit huoneet voidaan määritellä erikseen käyttötarkoituksen mukaan. Tutkimuksessa metodilla myös kokeiltiin tuottaa karttoja, joissa tietyt aluekokonaisuudet on erotettu toisistaan jollain erikoishuoneella tai vastaavalla, jonka läpi pelaajan on kuljettava (Valtchanov ja Brown 2012). Tämä vastaa jo aiemmin mainittua kenttätyyppiä, jossa pelaajan on saatava esimerkiksi jokin erikoiskyky tai voitettava vahva vihollinen edetäkseen uuteen luolaston osaan.

Samankaltainen alueiden erottelu toisistaan voidaan toteuttaa myös lukittujen ovien avulla, joihin pelaajan täytyy löytää avain edetäkseen. Esimerkki tällaisesta lähestymistavasta voidaan löytää tutkimuksesta, jonka toteuttivat Pereira, Prado ja Toledo (2018). Pereira, Prado ja Toledo (2018) esittävät tutkimuksessaan metodin, joka käyttää hyväkseen evolutiivista algoritmia luodakseen luolastomaisen kentän kartan, joka tarjoaa pelaajalle haastetta lukittujen ovien avulla. Tässäkin tapauksessa luolastot, jotka luodaan aluksi erillisellä algoritmilla, esitetään puurakenteena (Pereira, Prado ja Toledo 2018). Tällä kertaa huoneet kuitenkin sisältävät tiedon siitä ovatko ne normaaleja, lukittuja vai avaimen sisältäviä huoneita, joka määräytyy muodostamisen aikana (Pereira, Prado ja Toledo 2018). Avaimet voivat myös valinnan mukaan avata ainoastaan tietyn oven tai kaikki ovet, mutta ne ovat aina kertakäyttöisiä ja luolaston tulee aina olla läpäistävissä, joten avaimet eivät voi olla sen oven takana, johon sitä tarvitaan (Pereira, Prado ja Toledo 2018). Tässäkin tapauksessa puurakenteesta muodostetaan lopulta oikea luolaston kartta, jolloin osa huoneista saatetaan joutua poistamaan niiden päällekkäisyyden takia (Valtchanov ja Brown 2012; Pereira, Prado ja Toledo 2018).

Alussa muodostettua luolastomaista kenttää parannetaan tämän jälkeen käyttäen evolutiivista algoritmia (Pereira, Prado ja Toledo 2018). Alussa muodostetut luolastot ottivat huomioon vain tärkeimmät asiat, kuten yhdistettyjen huoneiden määrän, puun korkeuden sekä erityyppisten huoneiden todennäköisyyden ja loi näiden pohjalta melko satunnaisen kartan (Pereira, Prado ja Toledo 2018). Käytetty evolutiivinen algoritmi voi taas puolestaan mutatoida tai risteyttää tämän kartan huoneita, joka tarkoittaa samaa, kuin aiemmin kerrotussa metodissa (Valtchanov ja Brown 2012), mutta tällä kertaa mutaatiossa voidaan muuttaa myös haluttujen avainten määrää siinä kohdassa luolastoa (Pereira, Prado ja Toledo 2018). Nämä saattavat rikkoa joitain luolaston osia tekemällä luolastosta läpäisemättömän, mutta evolutiivinen al-

goritmi korjaa nämä määrittelemällä lukittujen ovien ja avainten paikat uudelleen muutosten jälkeen (Pereira, Prado ja Toledo 2018).

Suurin muutos aiemmasta tutkimuksesta (Valtchanov ja Brown 2012) on se, että tällä kertaa karttoja arvioidaan niiden haarautuneisuuden sekä lukittujen ovien, lukkojen ja avainten määrän mukaan, joiden tulee vastata haluttuja arvoja mahdollisimman hyvin (Pereira, Prado ja Toledo 2018). Sekä Pereira, Prado ja Toledo (2018) että Valtchanov ja Brown (2012) käyttävät lopulta halutun muokkausmäärän jälkeen niin sanottua turnajaisvalintaa uusien luolastojen muodostamisessa. Tämä tarkoittaa sitä, että valitaan tietty määrä tuotettuja luolastoja ja asetetaan ne vastakkain, jonka jälkeen parhaiten arvioitu yksilö saa tuottaa jälkeläisen itsestään, jota sen jälkeen muokataan edelleen paremmaksi käyttämällä risteytystä ja mutaatiota apuna (Valtchanov ja Brown 2012; Pereira, Prado ja Toledo 2018).

Pereira, Prado ja Toledo (2018) esittävät vielä tutkimuksensa lopussa toteuttamansa testin, jonka perusteella heidän käyttämänsä metodin luomat luolastot vastasivat haluttuja parametreja erittäin hyvin ja olivat vaikeudeltaan ja miellyttävyydeltään samaa luokkaa manuaalisesti tehtyjen luolastojen kanssa. Toisin sanoen tämäkin metodi sai tuotettua erittäin käyttökelpoisia kenttiä peleihin, joissa on mukana myös jonkunlainen haluttu etenemisjärjestys lukittujen ovien ansiosta. Tämän perusteella näihin karttoihin voisi olla mahdollista myös sisällyttää tarina, jos niin haluttaisiin.

3.5 Tarinaan pohjautuva luolastojen muodostaminen

Luolastomaisia kenttiä voidaan kuitenkin muodostaa myös niin, että tarinaa ajatellaan alusta alkaen, kuten Hartsook ym. (2011) tekevät tutkimuksessaan. Hartsook ym. (2011) käyttävät toteutuksessaan Game Forge nimistä alustaa, joka käyttää tekoälyä tarinan ja pelimaailman integroimiseen keskenään. Kyseisessä tutkimuksessa Game Forge rakentaa maailman, joka tukee sille juoniviitteinä syötettyä tarinaa. Nämä viitteet ovat yksinkertaisia ja tietyssä järjestyksessä tapahtuvia tarinan osia, kuten vihollisten kukistaminen tietyssä paikassa, jonkin esineen hakeminen toisesta paikasta tai vaikka vain tietylle hahmolle puhuminen pelin sisässä. Näiden lisäksi Game Forge tarvitsee pelimaailman luomisen avuksi listan hahmoista, esineistä ja paikoista, jotka tulee sisällyttää maailmaan (Hartsook ym. 2011). Hartsook

ym. (2011) käyttivät tätä roolipelin maailman rakentamiseen, mutta lopputuloksen maailmaa voisi yhtä hyvin ajatella pienten muutosten kanssa luolastona, sillä se on haarautuva ja käytävämäinen luolastojen tapaan.

Algoritmi, jota Game Forge käyttää, luo siis edellä mainittuja asioita hyväksikäyttäen maailman, jossa sille annettu tarina voi tapahtua (Hartsook ym. 2011). Tämä tapahtuu erittelemällä tärkeät tapahtumapaikat ja yhdistämällä nämä keskenään erilaisilla käytäville, jotka voivat haarautua moneen eri paikkaan (Hartsook ym. 2011). Samalla algoritmi täyttää luomansa käytävät muilla tapahtumilla, kuten vihollisilla ja aarteilla, jolloin rakennetusta maailmasta tulisi mielenkiintoisempi. Kaiken tämän lisäksi Hartsook ym. (2011) käyttävät maailman luomisessa myös pelaajan mieltymyksiä, jotka saadaan selville kysymällä pelaajalta pelin alussa kysymyksiä, jolloin luotu maailma tulee olemaan pelaajalle mieleinen. Tutkimuksen algoritmi ottaa myös huomioon muita seikkoja, jotka ovat tyypillisiä roolipeleille, mutta joita ei tarvittaisi luolastoja luotaessa, kuten esimerkiksi tiettyjen ympäristöjen vierekkäisyyden todennäköisyys.

Tarinaan pohjautuvan kartan luominen itsessään tapahtuu käyttämällä geneettistä algoritmia sekä joukkoa kenttiä, jotka esitetään puurakenteen avulla (Hartsook ym. 2011). Nämä puurakenteet tulee olla siirrettävissä ruudukkomaiseen rakenteeseen, joka koostuu huoneen kaltaisista kokonaisuuksista, sillä se on ainut tapa, jolla Hartsook ym. (2011) tutkimuksessaan toteuttavat kentät. Tämä tarkoittaa toisin sanoen sitä, että yhdellä tapahtumapaikalla voi olla enintään neljä viereistä tapahtumapaikkaa tai huonetta, joka yhdistää tapahtumapaikan kentän muihin osiin. Itse algoritmi valitsee aivan alussa yhden näistä kentistä, jonka jälkeen se muokkaa tätä mutatoimalla (Hartsook ym. 2011). Tässä tapauksessa puun solmuja eli huoneita voidaan joko poistaa, lisätä tai muokata (Hartsook ym. 2011). Hartsook ym. (2011) käyttävät muokkauksena huoneen ympäristötyypin muuttamista, mutta luolastomaisessa kentässä tämä voisi tarkoittaa huonetyypin tai sen sisältämien vihollisten tai aarteiden määrien ja tyyppien muuttamista. Näitä muokattuja puurakenteita tarkastellaan tämän jälkeen arviointifunktion avulla, joka ottaa huomioon pelaajan mieltymysten mukaisesti muun muassa luodun puurakenteen koon, haarautuneisuuden sekä sisällön tiheyden.

Sopivan ehdokkaan löydyttyä puurakenteiden joukosta, aloitetaan sen perusteella kentän luominen, joka tapahtuu siirtämällä puu aiemmin mainittuun ruudukkorakenteeseen, jolloin jo-

kainen ruutu luodaan puun solmun tietojen perusteella (Hartsook ym. 2011). Näin saatu maailma on luolastomainen, ruudukkoon perustuva ja tarinaa tukeva, sillä se on sijoittanut halutun tarinan tapahtumapaikat pelaajalle mieluisiin paikkoihin (Hartsook ym. 2011). Tällaisen metodin avulla voitaisiin mahdollisesti luoda ainakin osa kentistä proseduraalisesti myös tarinapohjaisiin peleihin. Kuten aiemmin mainittiin, Hartsook ym. (2011) saivat tämän metodin avulla tulokseksi haarautuvia käytävämäisiä kenttiä, joita voitaisiin aiemmin mainituin pienin muutoksin käyttää luolastoina.

3.6 Malliperustainen luolaston huoneiden luominen

Eräs lähestymistapa luolastojen luomiseen on malliperustainen luolastojen luominen. Malliperustaisen luolastojen luomisen perustana ovat suunnittelumallit (*engl. design patterns*), joiden ideana on abstraktoida tiettyjä ratkaisuja ongelmiin siten, että niitä voidaan käyttää uudelleen samankaltaisten ongelmien kanssa (Dahlskog, Björk ja Togelius 2015). Dahlskog, Björk ja Togelius (2015) analysoivat tutkimuksessaan suunnittelumalleja, joita luolastomaisissa kentissä yleensä esiintyy. Kyseisessä tutkimuksessa Dahlskog, Björk ja Togelius (2015) etsivät suunnittelumalleja ruutupohjaisista luolastoista, toisin sanoen luolastomaisista kentistä, jotka voidaan jakaa pieniin ruutuihin. Nämä ruudut voivat olla esimerkiksi osa seinää tai lattiaa, mutta ruudut voivat myös olla paikkoja, joihin vihollinen tai aarre on sijoitettu. Suunnittelumallit voidaan myös eritellä niiden koon mukaan mikro-, meso- ja makromalleihin (Dahlskog, Björk ja Togelius 2015). Mikromallit voisivat yksinkertaisuudessaan olla pieniä luolaston osia, kuten esimerkiksi yksittäinen vihollinen, risteys luolastossa tai sisäänkäynti luolastoon (Dahlskog, Björk ja Togelius 2015). Mesomallit ovat vastaavasti mikromallien yhdistelmiä, kuten joukko vihollisia tai risteyksiä vierekkäin, kun taas makromallit ovat näin ollen erilaisten mesomallien muodostamia suurempia kokonaisuuksia (Dahlskog, Björk ja Togelius 2015).

Suunnittelumallien etsinnän tarkoituksena on löytää rakennuspalikoita, joita voidaan käyttää hyödyksi hakuperusteisen luolastogeneraattorin toteuttamisessa (Dahlskog, Björk ja Togelius 2015). Mallien käyttämistä kenttägeneraattorin apuna voidaan perustella sillä, että mallit ovat itsessään melko yksinkertaisia ja itsenäisiä kokonaisuuksia, joten niitä on helppoa yhdistellä suuremmiksi kokonaisuuksiksi (Dahlskog, Björk ja Togelius 2015), kuten eriko-

koisista malleista ja niiden yhdistelmistä jo huomattiin. Näin ollen käyttämällä löydettyjä mikromalleja ja yhdistelemällä niitä luolastogeneraattorilla, saadaan aikaan luolastoja, joista voidaan mahdollisesti havaita myös meso- ja makromalleja.

Suunnittelumalleja voidaan kuitenkin käyttää proseduraalisessa kenttien luonnissa muutenkin kuin vain rakennuspalikoina. Dahlskog ja Togelius (2013) esittävät tutkimuksessaan, että suunnittelumalleja voitaisiin paremmin käyttää nimenomaan arvioimaan hakuperusteisten metodien käyttämien algoritmien tuottamien tulosehdotusten tasoa niiden rakentamisen sijaan. Toisin sanoen käytetään valittuja tai löydettyjä suunnittelumalleja apuna ehdotettujen ratkaisujen arvioimisessa, jolloin ehdotus on parempi sen perusteella, että siitä löytyy sopivasti halutunlaisia suunnittelumalleja vastaavia kohtia (Dahlskog ja Togelius 2013). Dahlskog ja Togelius (2013) huomasivat tutkimuksessaan myös sen, että näin toteutetut kentät olivat paljon lähempänä ilman proseduraalista luontia toteutettuja kenttiä, kuin suunnittelumalleja vain rakennuspalikoina käyttävät kentät.

Näiden tutkimuksien jälkeen Baldwin ym. (2017) esittävät tutkimuksessaan hakuperusteisen geneettisen algoritmin nimeltään FI-2Pop (*Feasible-Infeasible Two-Population Genetic Algorithm*), joka käyttää juuri suunnittelumalleja apunaan. Kyseinen tutkimus käyttää suunnittelumalleja sekä rakentamisen että arvioinnin apuna huoneita tehdessään (Baldwin ym. 2017) ottaen näin ollen huomioon kaksi aiemmin mainittua tutkimusta (Dahlskog, Björk ja Togelius 2015; Dahlskog ja Togelius 2013). Algoritmia käytetään osana Evolutionary Dungeon Designer nimistä, luolastomaisten kenttien luontiin käytettävää kenttägeneraattoria, jossa kehittäjä ohjaa luomista yhdessä algoritmin kanssa. Algoritmin tarkoituksena on tuottaa sisältö huoneisiin, jotka ovat jo valmiina Evolutionary World Designer nimisen generaattorin luomassa kartassa (Font ym. 2016). Kartan täytyy siis olla jo muuten valmiina, jotta algoritmi osaa luoda kartan huoneisiin optimaalisen sisällön.

Tutkimuksessa esitelty algoritmi tuottaa huoneisiin seinien ja ovien lisäksi myös viholliset sekä aarteet (Baldwin ym. 2017), jotta ne olisivat mahdollisimman mielenkiintoisia. Huoneiden sisältö itsessään tuotetaan työkalulle annettujen parametrien mukaan, joilla voidaan esittää toiveena kyseisen huoneen vaikeustason sekä vihollisten ja aarteiden määrän lisäksi käytettyjen suunnittelumallien määrä ja tyyppi, joiden perusteella algoritmi lopulta luo huoneen sisällön (Baldwin ym. 2017). Algoritmi tunnistaa ja käyttää kuitenkin vain aivan

pienimpiä suunnittelumalleista eli niin sanottuja mikromalleja, eikä suurempia, kuten meso- tai makromalleja. Tunnistettuja ja näin ollen luomisessa apuna käytettäviä malleja kyseisessä tutkimuksessa ovat esimerkiksi kammiot, käytävät sekä erilaiset mutkat ja risteykset luolastoissa (Baldwin ym. 2017).

Tutkimuksessa käytetty FI-2Pop -algoritmi asettelee huoneen viholliset ja aarteet valitun vaikeusasteen mukaan, jolloin esimerkiksi helpommalla asteella turvalliset alueet ovat suurempia ja aarteet ovat helpommin saatavissa (Baldwin ym. 2017), kun vastaavasti vaikeampi taso tarkoittaa päinvastaista. Itse luonnin aikana algoritmilla on käytettävissä nimensä mukaan kaksi eri populaatiota, hyväksyttävät huoneet ja hyväksymättömät huoneet, jotka ovat toisin sanoen mahdottomia läpäistä tai jollain muulla tavalla käyttökelvottomia (Baldwin ym. 2017). Luonnin aikana algoritmi joko muuttaa yhden osan, tai tässä tapauksessa ruudun, satunnaisesti valitusta satunnaisen populaation kartasta erilaiseksi osaksi, kuten esimerkiksi seinän aarteeksi, tai pienellä todennäköisyydellä algoritmi voi myös kääntää koko karttaa ympäri puoli kierrosta (Baldwin ym. 2017). Näitä muutoksia voidaan siis myös tehdä hyväksymättömien huoneiden populaation yksilöille siinä toivossa, että ne voitaisiin muutoksen jälkeen hyväksyä ja näin ollen siirtää hyväksyttyjen populaatioon. Baldwin ym. (2017) päättivät myös lopettaa huoneen muuntamisen tietyn muutosmäärän jälkeen, jotta tulos saataisiin ulos aina vakioajassa, eikä algoritmi jäisi jumiin. Luodut huoneet arvioidaan sen mukaan vastaako huoneen asettelu hyvin valittua vaikeutta sekä löytyykö huoneesta sopiva määrä erilaisia suunnittelumalleja, kun sitä verrataan annettujen parametrien arvoihin (Baldwin ym. 2017). Luotujen huoneiden tulee tietenkin olla myös läpäistävissä, jotta niitä voitaisiin käyttää, mikä tulee hoidettua pitämällä kelvolliset ehdotukset erillään epäkelvollisista (Baldwin ym. 2017).

Kaiken kaikkiaan Baldwin ym. (2017) arvioivat heidän käyttämänsä algoritmin tuottavan toiveiden mukaisia sekä monipuolisia huoneita, mutta he myös ottavat huomioon sen, että tulokset voisivat olla parempia, jos käytettäisiin aiemmissa tutkimuksissa (Dahlskog, Björk ja Togelius 2015) mainittuja meso- ja makromalleja sekä luomisen että arvioinnin apuna. Hyvien tulosten perusteella tekniikka voisi kuitenkin kehittää pidemmälle niin, että se pystyisi tuottamaan huoneiden sisältöä myös muiden kuin Evolutionary World Designerin (Font ym. 2016) muodostamiin luolastokarttoihin, jolloin sen yhdistäminen muiden luolastoja luo-

vien metodien kanssa olisi mahdollista.

3.7 Sokkeloiden luominen erilaisilla esitystavoilla

Luolastomaisten kenttien monipuolisuutta voidaan lisätä myös muuttamalla niissä käytettyä esitystapaa. Tässä aliluvussa käsitellään muutamaa erilaista esitystapaa sokkeloille, joita käytetään seuraavassa aliluvussa esiteltävässä metodissa suurempien sokkeloiden muodostukseen. Ashlock, Lee ja McGuinness (2011a) esittävät tutkimuksessaan evolutiivisiin algoritmeihin perustuvan järjestelmän itse luolaston, tarkemmin sokkelon, generointiin ilman sen täyttämistä vihollisilla, aarteilla tai muilla vastaavilla asioilla. Tutkimuksessa esitetään neljä erilaista esitystapaa sokkeloille, jotka kaikki tuottavat keskenään hyvin erinäköisiä lopputuloksia, vaikka kaikkien kehittämiseen käytetään keskenään samoja arviointifunktioita (Ashlock, Lee ja McGuinness 2011a). Käytetyt arviointifunktiot pohjautuvat niin sanottuun dynaamiseen ohjelmointiin, joka tarkoittaa yksinkertaistettuna jonkinlaisen verkoston läpi kulkemista, jonka aikana jokaiseen pysähdyspisteeseen kirjataan kyseisen polun minimipituus ja mahdollisesti joitain muita siinä tilanteessa tärkeitä asioita (Ashlock, Lee ja McGuinness 2011a; McGuinness ja Ashlock 2011).

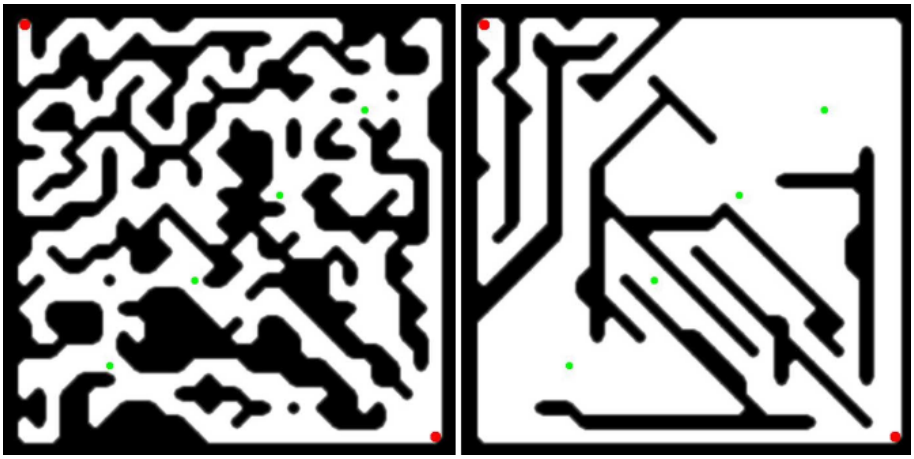
Käytetyistä neljästä sokkelon esitystavasta, joita Ashlock, Lee ja McGuinness (2011a) käyttivät, tutustutaan tässä tutkimuksessa paremmin näistä kahteen ja loput jätetään maininnan tasolle. Kaikkien tutkimuksen esitystapojen pohjana on jonkin kokoinen suorakulmio, jonka oikeassa alakulmassa on sisäänkäynti ja sitä vastakkaisessa kulmassa on ulospääsy. Ensimmäinen maininnan tasolle jätettävistä esitystavoista käyttää kuutta eri väriä sokkelon esittämiseen, jolloin tietyn värisestä ruudusta voidaan liikkua viereiseen vain, jos se on samanvärinen tai viereinen väri erikseen määrätyssä järjestyksessä (Ashlock, Lee ja McGuinness 2011a). Tätä voidaan kutsua kromaattiseksi esitystavaksi. Toinen vähemmälle tutustumiselle jätettävä esitystapa on eräänlainen vastakohta toiselle tarkemmin käytävälle esitystavalle, joten ne esitellään peräkkäin myöhemmin.

Ensimmäinen tarkemmin käytävä esitystapa esittää sokkelon bittijonona, jonka pituus riippuu luotavan sokkelon koosta (Ashlock, Lee ja McGuinness 2011a). Esimerkiksi sokkelo, jonka koko on X x Y, voidaan esittää näiden lukujen tulon suuruisena eli XY bitin pituisena

jonona, jonka jokainen bitti vastaa joko avointa tai tukittua kohtaa sokkelossa (Ashlock, Lee ja McGuinness 2011a). Näin bittijonona esitettyjä sokkeloita voidaan kehittää käyttämällä kahta eri operaatiota, joista ensimmäinen vaihtaa satunnaisten bittien paikkoja keskenään ja toinen muuttaa bittejä vastakkaisiksi (Ashlock, Lee ja McGuinness 2011a). Ashlock, Lee ja McGuinness (2011a) määrittelevät näille molemmille omat todennäköisyytensä, joilla ne muokkaamisen aikana tapahtuvat.

Toinen tarkemmin esiteltävä sokkeloiden esitystapa kuvaa sokkelot epäsuorasti, toisin kuin bittijonona kuvattu sokkelo, joka käyttää niin sanottua suoraa esitystapaa (Ashlock, Lee ja McGuinness 2011a). Tämä tarkoittaa sitä, että aiemmin käyty esitystapa kuvaa luodun sokkelon ruutuja biteillä, kun taas tämä esitystapa kuvaa sokkeloa hieman kiertäen. Esimerkit näiden molempien esitystapojen lopputuloksista voidaan nähdä kuviossa 4 (Ashlock, Lee ja McGuinness 2011a), josta nähdään myös se, kuinka suuren eron ainoastaan esitystavan muuttaminen voi saada aikaan. Tässä epäsuorassa esitystavassa sokkelo kuvataan listana, joka sisältää 80 kokonaislukua lukujen 0 ja 9999 väliltä, joista jokainen lukupari kuvaa tyhjään X x Y sokkelopohjaan luotavaa estettä (Ashlock, Lee ja McGuinness 2011a). Molemmat näiden lukuparien luvuista pilkotaan pienemmiksi luvuiksi, jolloin parin ensimmäinen luku kuvaa esteen ulottuvuuksia ja toinen sen sijaintia (Ashlock, Lee ja McGuinness 2011a). Kuten aiemmassakin esitystavassa, voidaan tämän esitystavan lukuja muuttaa tai niiden paikkoja voidaan vaihtaa jonkin muun listan luvun kanssa, jotta sokkeloa saadaan muokattua (Ashlock, Lee ja McGuinness 2011a). Lisäksi Ashlock, Lee ja McGuinness (2011a) esittävät vaihtoehtoisen epäsuoran esitystavan, jossa tyhjään pohjaan esteiden luomisen sijaan luodaankin käytäviä täyteen pohjaan samankaltaisella periaatteella.

Molempien tarkemmin esiteltyjen esitystapojen ongelmana voi olla muodostetun sokkelon kelvollisuus, toisin sanoen voidaanko luotua sokkeloa läpäistä lainkaan (Ashlock, Lee ja McGuinness 2011a). Ashlock, Lee ja McGuinness (2011a) ottavat tämän huomioon luomalla metodien käyttämien alkupopulaatioiden ratkaisuehdokkaat siten, että niissä on suhteellisesti vähän esteitä, jolloin niiden läpäistävyys säilyy hyvänä. Esteiden vähentäminen voi toimia esimerkiksi rajoittamalla luotujen esteiden suuruutta tai niiden määrää, joka voi käytännössä toimia sivuuttamalla loput parametrit tietyn täyttymisprosentin jälkeen tai asettamalla käytetyille arvoille jonkin maksimin (Ashlock, Lee ja McGuinness 2011a). Tämä ei ole ongelma



Kuvio 4. Oikealla puolella suoralla esitystavalla luotu sokkelo ja vasemmalla puolella epäsuoralla esitystavalla (Ashlock, Lee ja McGuinness 2011a).

sokkeloiden monimutkaisuutta ajatellen, sillä alkupopulaatioiden ehdokkaita ruvetaan myöhemmin muokkaamaan monimutkaisemmiksi (Ashlock, Lee ja McGuinness 2011a).

Ashlock, Lee ja McGuinness (2011a) käyttävät tutkimuksensa arviointifunktioissa dynaamisen ohjelmoinnin lisäksi eräänlaisia tarkistuspisteitä, jotka ovat vain tietyissä kohtaa sokkeloa sijaitsevia pisteitä. Kyseiset tarkistuspisteet voidaan nähdä kuviossa 4 pieninä vihreinä pisteinä kenttien käytävien väleissä. Näitä tarkistuspisteitä käytetään hyödyksi antamalla dynaamisen ohjelmointialgoritmin tallentaa lyhyimpien reittien lisäksi tarkistuspisteet, jotka ohitetaan käyttäen lyhintä reittiä tiettyyn pisteeseen pääsemiseen (Ashlock, Lee ja McGuinness 2011a). Tutkimuksessa käytetyillä arviointifunktioilla saatiin tarkistuspisteiden ansiosta luotua hyvin erilaisia sokkeloita. Eräs arviointifunktio sai aikaan pitkiä haarautumattomia polkuja, kun taas jokin toinen sai aikaan haarautuvia polkuja, jotka tarkistuspisteiden ylittämisen jälkeen yhdistyivät uudestaan yhdeksi poluksi (Ashlock, Lee ja McGuinness 2011a). Ashlock, Lee ja McGuinness (2011a) saivat arviointifunktioillaan painotettua myös monien tarkistuspisteiden sisällyttämistä samalle reitille tai umpikujien luomista sokkeloon.

Ashlock, Lee ja McGuinness (2011a) käyttivät samankaltaista evolutiivista algoritmia kaikkien eri esitystapojen sokkeloiden luomiseen. Tämä algoritmi käytti jo aiemmin mainittua turnajaisvalintaa kenttäehdotusten kehittämiseen siten, että se valitsi aina seitsemän eri ehdotusta yhden turnajaisen pitämiseen (Ashlock, Lee ja McGuinness 2011a). Näissä turnajai-

sissa algoritmi korvasi kaksi arviointifunktion mielestä huonointa ehdotusta kahden sen mielestä parhaan ehdotuksen kopioilla, jonka jälkeen suoritettiin evoluutio ensin näiden kahden kopion välillä ja lopuksi jokaiselle kenttäehdotukselle suoritettiin yksittäinen mutaatio (Ashlock, Lee ja McGuinness 2011a). Ashlock, Lee ja McGuinness (2011a) testasivat kaikkia esitystapoja monilla arviointifunktioilla, joiden ominaisuuksista mainittiin edellisessä kappaleessa.

Tutkimuksessaan Ashlock, Lee ja McGuinness (2011a) huomasivat kuitenkin, että pitkän polun painottaminen hyvänä luo vain haarautumattomia ja pitkiä polkuja sisäänkäynniltä ulospääsylle, mikä ei ollut toivottua luolastomaista kenttää luotaessa. Muilla funktioilla Ashlock, Lee ja McGuinness (2011a) saivat kuitenkin aikaan haarautuvia ja jopa useita polkuja, jotka johtivat loppujen lopuksi ulospääsylle. Kaiken kaikkiaan suoraan bittijonona esitetyt sokkelot alkoivat muistuttamaan luonnollisia luolastoja, kun taas epäsuorasti lukupareilla kuvatut sokkelot muistuttivat enemmänkin suunniteltuja rakennelmia (Ashlock, Lee ja McGuinness 2011a), mikä voidaan havaita myös hyvin kuviosta 4. Näitä eri esitystavoilla muodostettuja sokkeloita voitaisiin siis käyttää eri tarkoituksiin niiden piirteidensä ansiosta. Ashlock, Lee ja McGuinness (2011a) mainitsevat myös, että seuraava vaihe tutkimuksessa olisi heidän esitystapojensa ja arviointifunktioidensa hyödyntäminen kenttien kehityksessä käytettävän työkalun muodostamiseen.

3.8 Sokkeloiden muodostaminen useita esitystapoja yhdistelemällä

McGuinness ja Ashlock (2011) puolestaan laajentavat tutkimusta, jonka Ashlock, Lee ja McGuinness (2011a) tekivät, käyttämällä lähes samoja metodeja niin sanottujen tiilien luontiin, joiden avulla he saavat muodostettua suurempia karttoja helposti. Nämä tiilet ovat keskenään samankokoisia ja -muotoisia kartan osia, esimerkiksi määrätyn kokoisia neliöitä tai muita säännöllisiä muotoja, jolloin niiden yhdistäminen keskenään on vaivatonta, minkä ansiosta suurien alueiden muodostaminen näiden avulla on nopeaa (McGuinness ja Ashlock 2011). Kyseinen kartan osien yhdistäminen toteutetaan käyttämällä evolutiivisia algoritmeja eräänlaisten rakennusohjeiden muodostamiseen, joiden avulla osat on helppo yhdistellä (McGuinness ja Ashlock 2011).

McGuinness ja Ashlock (2011) käyttivät siis kartan osien luonnissa juuri niitä kahta eri esitystapaa, jotka esiteltiin tässä tutkimuksessa tarkemmin Ashlockin, Leen ja McGuinnessin (2011a) tekemän tutkimuksen neljästä eri esitystavasta. Nämä olivat sokkelon esittäminen suoraan bittijonona sekä vaihtoehtoisesti sen esittäminen epäsuorasti lukuparien avulla. Näiden esitystapojen konkreettiset tulokset voidaan nähdä kuviossa 4. McGuinness ja Ashlock (2011) muuttavat kuitenkin näiden sokkeloiden rakennetta antamalla niille mahdollisuuden useampaan sisäänpääsypisteeseen, minkä ansiosta jokaisella sokkelon sivulla voi olla joko yksi tällainen piste keskellä tai kaksi pistettä symmetrisesti yhtä kaukana toisistaan (McGuinness ja Ashlock 2011).

Tutkimuksessaan McGuinness ja Ashlock (2011) laskevat muodostettujen sokkeloiden osien jokaisen sisäänpääsypisteparin välille muodostuneen käytävän pituuden, jonka jälkeen kaikkien näiden reittien pituutta käytetään arvioimaan muodostunutta sokkeloa. Tämän arviointi tarkoittaa sitä, että McGuinness ja Ashlock (2011) käyttävät sokkeloiden muokkaamisen apuna arviointifunktiota, joka yrittää maksimoida kaikkien muodostuneiden reittien pituuksien summaa. Tämä ei tietenkään toimi tilanteessa, jossa syntyneellä sokkelon osalla on olemassa vain yksi sisäänpääsypiste, joten tällaisessa tilanteessa McGuinness ja Ashlock (2011) käyttivät sokkeloon erikseen määriteltyjä tarkistuspisteitä apunaan, jolloin pituus laskettiinkin näiden pisteiden välillä sisäänpääsypisteiden sijasta.

Itse sokkelon niin sanottu kokoamisohje määrittelee näiden sisäänkäyntien paikat sekä määrät käyttäen 8 x 8 kokoista taulukkoa, joka on täytetty reaalisilla kokonaisluvuilla suhteellisen pieneltä väliltä käyttäen normaalijakaumaa hyödyksi (McGuinness ja Ashlock 2011). Nämä luvut kuvaavat kunkin ruudun, jotka taas kuvaavat sokkeloon sijoitettavia aiemmin mainittuja tiiliä, korkeutta (McGuinness ja Ashlock 2011). Ashlock, Lee ja McGuinness (2011b) käyttivät kyseistä lähestymistapaa eräänlaisen kaksoissokkelon luomiseen, jossa korkeuden avulla pystyttiin määrittämään kaksi eri sokkeloa saman esitystavan sisään. Tämä korkeus on vaihteleva ja kokoamisen yhteydessä kahden ruudun välillä ei luoda yhtään kulkureittiä, jos niiden korkeuksien ero on enemmän kuin 2,0 (McGuinness ja Ashlock 2011). Jos tämä ero on kuitenkin vähemmän kuin 2,0, mutta enemmän kuin 1,0, päättivät McGuinness ja Ashlock (2011) luoda ruutujen välille yhden kulkureitin. Vastaavasti eron ollessa alle 1,0 muodostetaan ruutujen välillä kaksi kulkureittiä (McGuinness ja Ashlock 2011). Näin saadaan

lopulta aikaan ruudukko, jonka ruudut ovat yhdistyneet toisiinsa erilaisilla määrillä kulkureittejä, jonka jälkeen näiden määrien mukaan ruutuihin voidaan sijoittaa aiemmin mainittuja tiiliä suuremman kartan muodostamiseksi (McGuinness ja Ashlock 2011).

Tarvittavat tiilet voidaan luoda ja tallentaa jo etukäteen, sillä McGuinness ja Ashlock (2011) osoittavat, että tarvittavia tiilityyppejä on ainoastaan 80, jos sokkelon osan kullekin sivulle sallitaan 0, 1 tai 2 sisäänpääsypistettä sekä ei huomioida sisäänpääsypisteetöntä osaa. Tätä määrä voitaisiin kuitenkin pienentää entisestään esimerkiksi laskemalla jokaisen yhden sisäänkäynnin sokkelon osan samaksi, mutta näin luotujen sokkeloiden monipuolisuus saattaisi kärsiä ja McGuinness ja Ashlock (2011) olivat sitä mieltä, että se ei tässä tapauksessa tuottaisi tuloksia huomattavasti nopeammin. Näin ollen tällaisen rakennusohjeen avulla suurempienkaan sokkeloiden tuottaminen ei pitäisi olla vaikeaa (McGuinness ja Ashlock 2011). Itse rakennusohjeen, joka on tässä tapauksessa 8 x 8 kokoinen reaaliluvuilla täytetty ruudukko, muokkaaminen tapahtuu käyttämällä vaihtelevaa reaalilukua, joka valitaan käyttäen normaalijakaumaa, jonka keskipiste on nollassa ja keskihajonta on 0.5 (McGuinness ja Ashlock 2011). Tämän luvun arvo voi muuttua jokaisen muokkauksen jälkeen.

Suurien tasojen suuren luomisnopeuden lisäksi McGuinness ja Ashlock (2011) osoittivat tutkimuksessaan tämän luomistavan tukevan monilla eri tavoilla luotuja tiiliä. Kuten jo aiemmin mainittiin McGuinness ja Ashlock (2011) käyttivät kahta eri esitystapaa sokkeloiden osien luomiseen ja he pystyivät yhdistämään nämä eri esitystavat yhteen sokkeloon käyttämällä tiiliä molemmin eri tavoin esitetyistä sokkelon osista. Näin ollen kyseinen tapa luoda sokkeloita on sekä nopea että sallii luotavan sokkelon olevan monipuolinen (McGuinness ja Ashlock 2011). McGuinness ja Ashlock (2011) kuitenkin myöntävät tämän tavan sisältävän ongelmia. Luodut sokkelot voivat nimittäin käydä yksipuolisiksi yhden arviointifunktion takia, eivätkä ne sisällä mitään mielenkiintoisia esteitä tai palkintoja, kuten ansoja, vihollisia tai aarteita (McGuinness ja Ashlock 2011). McGuinness ja Ashlock (2011) esittävätkin tutkimusten voivan jatkua juuri näihin mainittuihin suuntiin tai esimerkiksi huoneiden lisäämiseen käytävien sekaan.

4 Yhteenveto

Proseduraalisen pelisisällön luonnin metodit voivat luoda hyvinkin uskottavia sekä mielenkiintoisia luolastomaisia kenttiä peleihin automatisoidusti käyttäen hyväksi monia erilaisia algoritmeja, jolloin luominen on myös vaivattomampaa. Kuitenkaan ne eivät varmaankaan yllä vielä hyvin pitkään aikaan, jos ikinä, samalle tasolle kuin alusta asti suunnittelijoiden luomat kentät. Jos pelissä on esimerkiksi hyvin monia erilaisia vaikeita mekaniikkoja, on parasta suunnitella nämä mekaniikat opettava aloituskenttä tarkasti, jotta pelaaja varmasti oppii tarvittavat asiat kunnolla. Tässä tutkimuksessa esiteltyjä sekä monia muita proseduraalisia metodeja voidaan kuitenkin jo käyttää luomaan ainutlaatuisia pelejä, joiden toteuttaminen ilman proseduraalisia metodeja olisi joko täysin mahdotonta tai vähintään erittäin hankalaa.

Vaikka proseduraalisesti luodut kentät eivät vielä yltäisikään samalle tasolla suunniteltujen kenttien kanssa, on niillä kuitenkin omat hyvät puolensa. Niitä voidaan käyttää ainutlaatuisten pelien tekemiseen, jotka eivät muuten olisi mahdollisia. Esimerkiksi proseduraalisten menetelmien avulla voidaan muokata kenttää pelaajan mieltymysten tai pelitapojen mukaan (Hartsook ym. 2011), mikä myös tarkoittaa sitä, että kenttiä on mahdollista luoda pelin aikana. Monet pelit myös käyttävät tätä kesken pelin tapahtuvaa kenttien luomista hyvin hyödykseen, jolloin niiden ei tarvitse luoda kaikkea pelikerran aikana tarvittavaa sisältöä ennen sen aloittamista. Tämän lisäksi monet esitellyt luomistavat voivat tuottaa suuriakin luolastoja huomattavan nopeasti (McGuinness ja Ashlock 2011), jolloin kehittäjien ei tarvitse rajoittaa kenttiensä kokoa. Huomattiin myös luolastomaisten kenttien olevan mahdollisia luoda proseduraalisesti alkeellisen tarinan pohjalta (Hartsook ym. 2011).

Proseduraalisten metodien käytöstä voi siis seurata suuriakin hyötyjä ja vaikka yhden metodin avulla ei voidakaan luoda kaikkea (Linden, Lopes ja Bidarra 2014), voidaan niitä silti periaatteessa yhdistellä keskenään. Esimerkiksi pienien muutosten avulla voitaisiin käyttää jotakin aliluvussa 3.4 mainittua metodia kartan yleiskuvan luomiseen, jonka jälkeen voisimme käyttää aliluvussa 3.6 mainittuja metodeja tarkemman huoneiden tai kartan osien luomiseen. Näin voitaisiin saada aikaiseksi jo melko hyvä luolastomainen kenttä, jonka jälkeen sitä voitaisiin vielä hioa käyttämällä muita metodeja, jos se koettaisiin tarpeelliseksi.

Samankaltainen osien erillinen luonti ja yhdistely tuli myös esille aliluvussa 3.8, vaikkakin tässä tapauksessa kenttään tulisi vielä tuoda mahdolliset aarteet sekä viholliset käyttäen erillistä menetelmää.

Kaiken kaikkiaan vaikuttaa siltä, että proseduraaliset metodit saavat aikaan jopa erittäin hyvää pelisisältöä, jos metodeihin vain panostetaan tarpeeksi. Tämä kuitenkin vaatii sitä, että metodien kehittäjät oikeasti tietävät mitä tekevät sekä hiovat tekemäänsä metodia tarpeeksi, mikä puolestaan voi viedä paljonkin aikaa. Lisäksi teknologian kehittyessä saadaan luultavammin kehitettyä entistä tehokkaampia ja tarkempia proseduraalisia metodeja, minkä takia on edelleen tärkeää jatkaa proseduraalisen sisällön luonnin kehittämistä ja sen tutkimista monista eri näkökulmista.

Lähteet

Wichman, Glenn, Michael Toy ja Ken Arnold. 1980. *Rogue [videopeli].*

Ashlock, Daniel, Colin Lee ja Cameron McGuinness. 2011a. “Search-Based Procedural Generation of Maze-Like Levels”. *IEEE Transactions on Computational Intelligence and AI in Games* 3 (3): 260–273.

———. 2011b. “Simultaneous dual level creation for games”. *IEEE Computational Intelligence Magazine* 6 (2): 26–37.

Baldwin, A., S. Dahlskog, J. M. Font ja J. Holmberg. 2017. “Mixed-initiative procedural generation of dungeons using game design patterns”. Teoksessa *2017 IEEE Conference on Computational Intelligence and Games (CIG),* 25–32.

Dahlskog, Steve, Staffan Björk ja Julian Togelius. 2015. “Patterns, dungeons and generators”. Teoksessa *Foundations of Digital Games Conference, FDG.* Foundations of Digital Games.

Dahlskog, Steve, ja Julian Togelius. 2013. “Patterns as objectives for level generation”. Teoksessa *Proceedings of the Second Workshop on Design Patterns in Games;* ACM.

Font, Jose M, Roberto Izquierdo, Daniel Manrique ja Julian Togelius. 2016. “Constrained level generation through grammar-based evolutionary algorithms”. Teoksessa *European Conference on the Applications of Evolutionary Computation,* 558–573. Springer.

Hartsook, Ken, Alexander Zook, Sauvik Das ja Mark O. Riedl. 2011. “Toward supporting stories with procedurally generated game worlds”. Teoksessa *2011 IEEE Conference on Computational Intelligence and Games (CIG’11),* 297–304. IEEE.

Hendrikx, Mark, Sebastiaan Meijer, Joeri Van Der Velden ja Alexandru Iosup. 2013. “Procedural content generation for games: A survey”. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)* 9 (1): 1.

Liapis, Antonios, Georgios N. Yannakakis ja Julian Togelius. 2013. “Sentient Sketchbook: Computer-aided game level authoring.” Teoksessa *FDG,* 213–220.

Linden, R. van der, R. Lopes ja R. Bidarra. 2014. “Procedural Generation of Dungeons”. *IEEE Transactions on Computational Intelligence and AI in Games* 6 (1): 78–89.

McGuinness, Cameron, ja Daniel Ashlock. 2011. “Decomposing the level generation problem with tiles”. Teoksessa *2011 IEEE Congress of Evolutionary Computation (CEC),* 849–856. IEEE.

McMillen, Edmund, ja Florian Himsl. 2011. *The Binding of Isaac [videopeli].*

MotionTwin. 2017. *Dead Cells [videopeli].*

Nintendo. 1986. *The Legend of Zelda [videopeli].*

Pereira, L. T., P. V. S. Prado ja C. Toledo. 2018. “Evolving Dungeon Maps With Locked Door Missions”. Teoksessa *2018 IEEE Congress on Evolutionary Computation (CEC),* 1–8. ID: 1.

Preuss, M., A. Liapis ja J. Togelius. 2014. “Searching for good and diverse game levels”. Teoksessa *2014 IEEE Conference on Computational Intelligence and Games,* 1–8.

Togelius, Georgios N., Julian a nd Yannakakis, Kenneth O. Stanley ja Cameron Browne. 2011. “Search-based procedural content generation: A taxonomy and survey”. *IEEE Transactions on Computational Intelligence and AI in Games* 3 (3): 172–186.

Togelius, Julian, ja Noor Shaker. 2016. “The search-based approach”, 17–30. Procedural Content Generation in Games. Springer.

Togelius, Julian, Noor Shaker ja Mark J. Nelson. 2016. “Introduction”, 1–15. Procedural Content Generation in Games. Springer.

Togelius, Julian, Georgios N. Yannakakis, Kenneth O. Stanley ja Cameron Browne. 2010. “Search-based procedural content generation”. Teoksessa *European Conference on the Applications of Evolutionary Computation,* 141–150. Springer.

Valtchanov, Valtchan, ja Joseph A. Brown. 2012. “Evolving dungeon crawler levels with relative placement”. Teoksessa *Proceedings of the Fifth International C* Conference on Computer Science and Software Engineering,* 27–35. ACM.