Joonas Karttunen

Yleisten pelikenttien proseduraalisen generoinnin metodit

Tietotekniikan kandidaatintutkielma

13. toukokuuta 2016

Jyväskylän yliopisto

Tietotekniikan laitos

Tekijä: Joonas Karttunen

Yhteystiedot: joonas.v.a.karttunen@student.jyu.fi

Työn nimi: Yleisten pelikenttien proseduraalisen generoinnin metodit

Title in English: Procedural generation methods of general game levels

Työ: Kandidaatintutkielma

Sivumäärä: 22+0

Tiivistelmä: Proseduraalinen kenttien generointi on viime vuosina noussut indie-pelikehittäjien suosioon, sillä sen avulla voidaan peleihin luoda paljon sisältöä suhteellisen pienellä työmäärällä. Tässä tutkielmassa tarkastellaan yleisten kenttien proseduraalisen generoinnin metodeita. Tarkastelun pohjaksi tutkielmassa käsitellään myös eri kenttätyyppien vaatimuksia ja ominaisuuksia. Lopuksi eri metodityyppejä myös vertaillaan keskenään.

Avainsanat: proseduraalinen generointi, pelikenttä, pelit, proseduraalisen generoinnin metodit

Abstract: In recent years, procedural generation of game levels has risen to popularity among indie game developers, because it enables creation of game content with relatively small workload. In this paper, procedural generation methods of general game levels will be explored. As a basis for this exploration, the requirements and characteristics of different types of game levels will also be discussed. Finally, different types of generation methods will be compared with each other.

Keywords: procedural generation, game level, games, procedural generation methods

Sisältö

1 Johdanto

Proseduraalinen sisällön generointi (engl. procedural content generation, PCG) voidaan määritellä sisällön tuottamisena automaattisesti algoritmien avulla joko täysin tai ainakin lähes täysin ilman ihmisen osallistumista (Togelius, Kastbjerg ym. 2011). Pelikehityksessä sitä on pitkään hyödynnetty tehostamaan pelikehitysprosessia siirtämällä työmäärää ihmisiltä tietokoneille (Dormans 2010). Tässä tutkielmassa keskitytään pelikenttien proseduraaliseen generointiin, joka on noussut indie-pelikehittäjien suosioon mahdollistamalla kilpailun isojen peliyritysten kanssa ainakin sisällön määrän suhteen. Koska indie-pelikehittämisen suosio on viime vuosina ollut suuressa kasvussa (Ong 2014; Thongkham 2015), niin proseduraalista kenttägenerointia hyödynnetään aikaisempaa enemmän.

Kenttä tarkoittaa koko pelitilan (engl. game space) yksittäistä interaktiivista aluetta, jossa pelaajan ohjaama hahmo kulkee pelin asettamien sääntöjen rajoittamana (Smith, Cha ja Whitehead 2008). Kenttäsuunnittelun monimutkaisuuden vuoksi kenttien proseduraalisessa generoinnissa käytetään usein vain yksittäisen pelityypin kenttien tai jopa yksittäisen pelin kenttien generointiin soveltuvia algoritmeja (Diaz-Furlong ja Solis-Gonzalez Cosio 2013; Smith, Cha ja Whitehead 2008).

Tässä tutkielmassa tarkastellaan erityisesti sellaisia proseduraalisen kenttägeneroinnin metodeja, joilla voidaan luoda kenttiä useisiin eri pelityyppeihin. Yleisluontoiset metodit mahdollistavat sen, että jokaista peliä varten ei tarvitse luoda omaa generointimetodia, ja näin työmäärä vähenee huomattavasti (Sorenson ja Pasquier 2010b). Tähän liittyen pohditaan, mitä perusvaatimuksia jokaisen kentän tulisi noudattaa. Lisäksi tarkastellaan myös eri pelityyppien kenttien välisiä yhtäläisyyksiä, jotta voitaisiin määrittää, mitä yleisluontoisen kenttägeneroinnin metodin pitäisi pystyä ottamaan huomioon. Tutkielma on tehty kokonaan systemaattisen kirjallisuuskartoituksen keinoin.

Toisessa luvussa rajataan tutkielmassa käsiteltävät kenttätyypit annetun määritelmän pohjalta ja pohditaan, millainen on hyvä kenttä. Kolmannessa luvussa pääluokittain kuvaillaan erilaisia proseduraalisen kenttägeneroinnin metodeja, joita lopuksi vertaillaan neljännessä luvussa.

2 Pelikentän määritelmä ja hyvän kentän vaatimuksia

Tässä luvussa tarkastellaan hyvän kentän vaatimuksia. Tarkastelu rajataan erityisesti tasohyppely-, ammunta- ja luolastopelien kenttiin. Tasohyppelypeleissä (engl. platformer) tarkoituksena on – nimensä mukaisesti – edetä kentän läpi eri tasojen välillä hyppien ja samalla vältellen esteitä, kuten ansoja ja vihollisia (Baghdadi ym. 2015). Ammuntapeleillä (engl. first-person shooter, FPS) puolestaan viitataan peleihin, joissa pelaaja liikkuu kolmiulotteisessa ympäristössä pelihahmon perspektiivissä ja taistelee erilaisten aseiden avulla vihollisia vastaan (Cardamone ym. 2011). Luolastopeleillä viitataan kaikkiin sellaisiin peleihin, joissa kenttä on luolastomainen. Luolasto (engl. dungeon) on usein seikkailu- ja roolipeleissä käytetty kenttätyyppi, jonka ominaisuuksia ovat labyrinttimaisuus ja joka voi sisältää esimerkiksi vihollisia ja aarteita (Linden, Lopes ja Bidarra 2014).

Kentän käsitteen ulkopuolelle jäävät erityisesti strategiapelien kartat (engl. map) ja rallipelien radat (engl. track). Karttoja ei ajatella kentiksi, koska strategiapeleissä ohjataan useita yksikköjä samanaikaisesti, mikä on ristiriidassa johdannossa annetun kentän määritelmän kanssa. Rallipelit ja niiden radat puolestaan poikkeavat liiaksi muista peli- ja kenttätyypeistä, joten näiden vertailu keskenään ei olisi mielekästä. Kuitenkin esimerkiksi Togelius, Preuss ja Yannakakis (2010) ovat käsitelleet hyvän kartan vaatimuksia proseduraalisen generoinnin näkökulmasta. Proseduraalisesti generoituja ratoja ovat puolestaan tutkineet muun muassa Togelius, De Nardi ja Lucas (2007) ja Cardamone, Loiacono ja Lanzi (2011).

Seuraavaksi pohditaan hyvän pelikentän yleisiä vaatimuksia. Tämän jälkeen keskitytään tasohyppely-, FPS- ja luolastopelien kenttien vaatimusten yhtäläisyyksiin ja eroihin.

2.1 Hyvän kentän yleisiä vaatimuksia

Kaikkein tärkein vaatimus, jonka jokaisen kentän tulisi täyttää, on käyttökelpoisuus (engl. feasibility), ja siitä mainitaankin lähes jokaisessa proseduraalista kenttägenerointia käsittelevässä lähteessä (esim. Diaz-Furlong ja Solis-Gonzalez Cosio 2013; Togelius, Yannakakis ym. 2011; Preuss, Liapis ja Togelius 2014). Käyttökelpoisuudella tarkoitetaan ennen kaikkea sitä, että pelin asettamat tavoitteet olisivat saavutettavissa kyseisessä kentässä: pelaajan

etenemistä ei saa tehdä mahdottomaksi (Togelius, Yannakakis ym. 2011). Käyttökelvottomia kenttiä ovat siis erityisesti sellaiset kentät, joissa ulospääsy tai maali on pelaajan saavuttamattomissa, mutta myös muita rajoitteita on. Esimerkiksi kenttä, jossa pelaajan tarkoituksena on päihittää liian vaikea vihollinen, on käytännössä käyttökelvoton (Togelius, Yannakakis ym. 2011). Liian vaikealla vihollisella tarkoitetaan sellaista vihollista, jota valtaosa pelaajista ei kykene päihittämään kohtuullisessa ajassa.

Kun kentän käyttökelpoisuus on varmistettu, voidaan siirtää huomio itse pelikentän pelaajassa luomaan kokemukseen. Toivottavin kokemus on hauskuus, joka on kuitenkin monimuotoisuutensa vuoksi hankala määritellä, eikä sitä siksi tässä tutkielmassa käsitellä yksityiskohtaisesti. Useat tutkimukset kuitenkin mainitsevat kentän vaikeustason sopivuuden ja siitä seuraavan jännityksen määrän merkittävimmäksi hauskuuteen vaikuttavaksi tekijäksi (Hullett ja Whitehead 2010; Sorenson ja Pasquier 2010a; Sorenson, Pasquier ja DiPaola 2011). Liian vaikeiden kenttien on havaittu turhauttavan pelaajia, mutta liian helpot kentät puolestaan koetaan tylsiksi (Hullett ja Whitehead 2010; Sorenson, Pasquier ja DiPaola 2011). Lisäksi tutkijat Diaz-Furlong ja Solis-Gonzalez Cosio (2013) huomauttavat, että vaikeustason vaihtelu kentän eri vaiheissa on toivottavaa: esimerkiksi kentän loppua lähestyttäessä myös vaikeustaso kasvaisi. Vaikeustasoa voidaan suoraan nostaa lisäämällä esteiden, kuten vihollisten tai kuilujen, määrää, ja helpommaksi kentän tekee puolestaan kerättävien esineiden (engl. collectable items), kuten aseiden tai parantavien esineiden, suurempi määrä (Raffe ym. 2015).

2.2 Eri kenttätyyppien vaatimusten yhtäläisyyksiä ja eroja

Eri pelityyppien kentillä on yleisten vaatimusten lisäksi myös kyseisen pelityypin ominaisuuksien aiheuttamia vaatimuksia (Smith, Cha ja Whitehead 2008). Kenttätyypeille spesifien vaatimusten huomioiminen on tärkeää, sillä se auttaa yleisluontoisten kenttien generointimetodien hahmottamisessa. Monet näistä vaatimuksista ovat kuitenkin hyvin samanlaisia eri kenttätyyppien välillä. Smith, Cha ja Whitehead (2008) lukevat tasohyppelykenttien peruselementtien joukkoon muun muassa tasot, joilla pelaaja voi liikkua, esteet, kuten kuilut ja viholliset, sekä myös pelaajaa vahvistavat kerättävät esineet. Vastaavia peruselementtejä on myös luolastokentissä: Tasohyppelykentän tasoja vastaavat laatat (engl. tile), jotka ovat luo-

lastokentän tilan perusta ja joilla myös pelaaja liikkuu (Dahlskog, Togelius ja Björk 2015). Luolastokentät sisältävät myös vihollisia ja ansoja, eli esteitä, sekä aarteita ynnä muita pelaajaa vahvistavia kerättäviä esineitä (Dahlskog, Togelius ja Björk 2015). Viholliset, kerättävät esineet ja luonnollisesti myöskin taso, jolla liikutaan, ovat osa myös FPS-kenttiä (Hullett ja Whitehead 2010).

Eri pelityyppien kentillä on kuitenkin myös huomattavia eroja keskenään. Esimerkiksi FPS-kentissä olisi oltava sijainteja, joissa pelaaja voisi suojautua vihollisten ammuksilta (Hullett ja Whitehead 2010), mutta vastaavaa tarvetta ei ole useimmissa tasohyppely- tai luolastokentissä. Luolastokentissä puolestaan pitäisi olla muihin verrattuna enemmän mahdollisuuksia seikkailulle ja tutkimiselle monien vaihtoehtoisten reittien muodossa (Linden, Lopes ja Bidarra 2014; Dahlskog, Togelius ja Björk 2015), vaikkakin vaihtoehtoiset reitit voivat olla tärkeä osa myös FPS-kenttiä (Hullett ja Whitehead 2010). Compton ja Mateas (2006) mainitsevat vaihtoehtoiset reitit myös osana tasohyppelykenttiä. Luolasto- ja FPS-kentät usein sisältävät myös ystävällisiä pelihahmoja, jotka voivat toimia esimerkiksi neuvonantajina tai osana tehtävää (Linden, Lopes ja Bidarra 2014; Hullett ja Whitehead 2010). Kuitenkin tasohyppelykentissä ystävälliset hahmot lienevät harvinaisempia, sillä niistä ei ollut edes mainintaa useissa tasohyppelykenttiä käsittelevissä tutkimuksissa. Edellä mainitut ominaisuudet eivät kuitenkaan ole pelityyppiä täysin määrittäviä, sillä on täysin mahdollista luoda esimerkiksi tasohyppelypeli, jossa tarkoituksena on edetä kentän läpi suojautuen vihollisammuksilta.

3 Proseduraalisen kenttägeneroinnin metodit

Tässä luvussa tarkastellaan erilaisia proseduraalisen kenttägeneroinnin metodeja. Erityisesti keskitytään sellaisiin metodeihin, joita voidaan käyttää vähintään kahden eri pelityypin kenttien generointiin. Ensimmäisenä lyhyesti pohditaan, mitä vaatimuksia jokaisen hyvän kenttägenerointimetodin tulisi toteuttaa. Tämän jälkeen itse metodien tarkastelu tehdään pääluokittain. Tarkastelu aloitetaan malliperustaisista metodeista, minkä jälkeen tarkastellaan kokemusperustaisia metodeja. Lopuksi tarkastellaan vielä vastauslähtöisiä ja kielioppiperustaisia metodeja.

3.1 Hyvän kenttägenerointimetodin yleisvaatimuksia

Hyvällä kenttägenerointimetodilla on useita vaatimuksia: luotettavuus, kontrolloitavuus, nopeus sekä kenttien monipuolisuus (Togelius, Shaker ja Nelson 2015b). Myös satunnaisuus on tärkeää, sillä muuten kyse olisi pikemminkin datan pakkauksesta kuin proseduraalisesta generoinnista (Togelius, Yannakakis ym. 2011). Käytännössä vaatimusten välillä joudutaan tekemään kompromisseja: esimerkiksi kenttien laatu usein riippuu kyseisen generointimetodin käyttämästä ajasta (Togelius, Shaker ja Nelson 2015b).

Generointimetodin luotettavuulla tarkoitetaan sen kykyä luoda riittävän hyvä kenttä aina, jolloin huonoimmatkin metodin luomat kentät ovat käyttökelpoisia ja mielenkiintoisia (Togelius, Yannakakis ym. 2011). Luvussa 2.1 tärkeimmän kentän vaatimuksen todettiin olevan sen käyttökelpoisuus. Tämän perusteella voidaan olettaa luotettavuuden olevan tärkein vaatimus useimmissa metodeissa. Myös metodin kontrolloitavuus vaikuttaa sen luotettavuuteen, sillä helposti kontrolloitavassa metodissa kenttäsuunnittelija voi varmistaa generoitujen kenttien käyttökelpoisuuden (Dahlskog ja Togelius 2014). Linden, Lopes ja Bidarra (2014) määrittelevät kontrolloitavuuden joukkona asetuksia, joiden avulla pelisuunnittelija voi vaikuttaa generointiprosessiin. Näiden asetusten vähäisyydestä tai epäselvyydestä seuraa huonosti kontrolloitava metodi (Linden, Lopes ja Bidarra 2014). Yannakakis ja Togelius (2011) myös mainitsevat, että hyvän kontrolloitavuuden takaamiseksi pienen muutoksen generoivan algoritmin asetuksissa pitäisi johtaa pieneen muutokseen lopputuloksessa. Huono luotettavuus

ja kontrolloitavuus johtavat suurempaan määrään tarvittavia testejä kenttien sopivuuden takaamiseksi, mikä puolestaan johtaa hitaampaan metodiin (Diaz-Furlong ja Solis-Gonzalez Cosio 2013). Metodin hyvä luotettavuus ja kontrolloitavuus voivat kuitenkin johtaa generoitujen kenttien alhaisempaan monipuolisuuteen (Dahlskog ja Togelius 2014; Togelius, Shaker ja Nelson 2015b).

Kenttägeneroinnin metodin monipuolisuudella tarkoitetaan sen kykyä luoda erityisesti rakenteeltaan erityyppisiä kenttiä (Dahlskog ja Togelius 2014). Monipuolisesti erilaisia kenttiä tuottavaa proseduraalisen generoinnin metodia on mahdollista käyttää useiden erityyppisten kenttien luomiseen (Smith ja Whitehead 2010). Myös yksittäisen pelin sisällä kenttien monipuolisuus on usein tärkeää, jotta pelaaja ei välittömästi kyllästyisi peliin. Joissakin metodeissa pelisuunnittelija voi valita generoiduista kentistä peliin parhaiten sopivat ja edelleen muokata niitä paremmiksi (Preuss, Liapis ja Togelius 2014). Tällöin metodin monipuolisuus voi olla jopa tärkeämpää kuin sen luotettavuus (Smith ja Whitehead 2010).

Generointimetodin eri vaatimusten tärkeys riippuu sen luonteesta ja käyttötarkoituksesta (Preuss, Liapis ja Togelius 2014), mikä jo osittain edellä huomattiin. Suurimman eron aiheuttaa proseduraalisen generoinnin ajankohta pelin pelaamiseen nähden. Mikäli kenttiä generoidaan pelin ajon aikana, niin metodin nopeuden, luotettavuuden ja huonoimman mahdollisen lopputuloksen laadun takaaminen on tärkeintä (Togelius, Yannakakis ym. 2011; Preuss, Liapis ja Togelius 2014). Vastaavasti mikäli generoituja kenttiä voidaan vielä muokata ennen kuin ne sisällytetään peliin, niin monipuolisuuden ja parhaimman mahdollisimman lopputuloksen arvo kasvaa (Preuss, Liapis ja Togelius 2014).

3.2 Malliperustaiset metodit

Suunnittelumallit (engl. design pattern) ovat suunnitteluratkaisujen formalisoituja luonnehdintoja (Dahlskog, Togelius ja Björk 2015). Tämä formaalisuus mahdollistaa mallien käytön prosaduraalisen generoinnin algoritmeissa (Dahlskog, Togelius ja Björk 2015). Malliperustaisen metodin avulla generoitujen kenttien tyyppi määräytyy käytetyistä malleista, joten tärkeitä vaiheita ovat kenttien mallien tunnistaminen, formalisoiminen ja niiden välisten suhteiden huomioiminen (Hullett ja Whitehead 2010). Dahlskog ja Togelius (2014) jaottelevat

mallit kahteen tyyppiin: mikro- ja makromalleihin.

Yksinkertaisimmat mallit eli mikromallit ovat kentän perusosasia, ja jokainen kenttä on pohjimmiltaan vain niiden sarja (Dahlskog ja Togelius 2014). Esimerkkejä mikromalleista ovat luvussa 2.2 mainitut peruselementit, kuten yksittäiset tasot ja viholliset. Pelkästään satunnaisten mikromallien sarjasta koostuva kenttä on kuitenkin erittäin todennäköisesti käyttökelvoton, joten tarvitaan malleja, jotka luovat kentälle toimivan ja mielenkiintoisen rakenteen (Dahlskog ja Togelius 2014).

Mesomallit ovat mikromallien ryhmiä, joiden avulla voidaan mallintaa monipuolisesti erilaisia kenttien rakenteita (Dahlskog, Togelius ja Björk 2015). Esimerkiksi pitkä kuilu, jonka ylle on asetettu taso ylittämisen mahdollistamiseksi, voisi olla yksi tasohyppelykentän mesomalli (Dahlskog ja Togelius 2014). Dahlskog, Togelius ja Björk (2015) ehdottavat lisäksi mikro- ja mesomalleista koostuvia makromalleja, joilla voidaan mallintaa koko kentän toimintaa ja vaikuttaa sen luomaan pelikokemukseen. Esimerkiksi pelaajan pakottaminen palaamaan jo kuljettua reittiä takaisin kohti kentän alkua voisi olla makromalli (Dahlskog, Togelius ja Björk 2015).

Mallipohjaisella metodilla voidaan luoda käyttökelpoisia ja monipuolisia kenttiä, kun mesomalleja käytetään mikromallien sijaintien päämäärinä (Dahlskog ja Togelius 2014). Dahlskog ja Togelius (2014) generoivat kenttiä evolutiivisellä algoritmilla. Ensin he luovat suuren määrän satunnaisista mikromalleista koostuvia kenttiä. Arviointialgoritmi pisteyttää näitä kenttiä mesomallien määrän mukaan: mitä enemmän mesomalleja kenttä sisältää, sitä todennäköisemmin se valitaan jatkokäsittelyä varten. Monipuolisempien kenttien takaamiseksi kenttä saa pisteitä myös sen koostavien mallien monipuolisuudesta (Dahlskog, Togelius ja Björk 2015).

Malliperustaisia metodeja voidaan hyödyntää monien erityyppisten kenttien generointiin, kun pohjatyönä on ensin määritelty tavoitekenttiä hyvin kuvaavat mallit (Dahlskog, Togelius ja Björk 2015). Yleisen kenttien generoinnin kannalta onkin tärkeää löytää eri kenttätyyppien kenttien välisiä yhtäläisyyksiä, joiden kautta malleja voidaan muodostaa.

3.3 Kokemusperustaiset metodit

Pelaaminen synnyttää pelaajassa monenlaisia tunnetiloja, jotka vaikuttavat pelaajan toimintaan pelissä (Yannakakis ja Togelius 2011). Yannakakis ja Togelius (2011) mainitsevat, että nämä tunnetilat sekä niistä riippuva pelaajan toiminta vaihtelevat eri pelaajien välillä sekä myös yksittäisen pelaajan kohdalla esimerkiksi oppimisen seurauksena, ja ne myös luovat yksilökohtaisen pelikokemuksen. Kokemusperustaisissa (engl. experience-driven) proseduraalisen generoinnin metodeissa tavoitteena on luoda kenttiä, jotka olisivat muokattu juuri peliä sillä hetkellä pelaavalle pelaajalle sopiviksi (Yannakakis ja Togelius 2011). Näin kenttien generointi pohjautuisi pelikokemukseen. Esimerkiksi pelaajan taitotaso, mieliala sekä mieltymykset voisivat vaikuttaa generoitavien kenttien ominaisuuksiin (Raffe ym. 2015).

Jotta kenttien generoinnissa voitaisiin hyödyntää yksittäisen pelaajan pelikokemusta, täytyy kokemusta pystyä mallintamaan. Mallinnuksen pohjana voidaan käyttää pelaajan pelinsisäistä toimintaa sekä muuta pelisession dataa (Yannakakis ja Togelius 2011). Voidaan esimerkiksi havainnoida, kuinka usein pelaaja hakeutuu taisteluihin, mitä aseita hän mieluiten käyttää tai kuinka nopeasti hän läpäisee kentän (Shaker, Yannakakis ja Togelius 2010). Togelius, Shaker ja Nelson (2015c) lisäksi huomauttavat, että pelaajan toiminnan tarkkailussa pitäisi huomioida myös pelikonteksti. Voi esimerkiksi olla, että pelaaja käyttää jotakin tiettyä asetta vain, koska hän ei koskaan löytänyt parempaa vaihtoehtoa. Kerättyä dataa voidaan sitten hyödyntää pelaajalle sopivien kenttien generoinnissa, usein evolutiivisten algoritmien avulla (Yannakakis ja Togelius 2011).

Myös pelaajilta itseltään voidaan kysyä heidän mieltymyksiään, ja saatuja tuloksia sitten hyödyntää generoinnissa (Yannakakis ja Togelius 2011). Raffe ym. (2015) esittävät metodin, jossa jokaisen kentän lopuksi pelaaja voi arvioida juuri pelaamansa kentän numeroasteikolla, ja näitä arvioita hyödyntäen pelaajalle automaattisesti rakennetaan hänen mieltymyksiään kuvaava profiili. Arvioinnin jälkeen pelaajalle esitetään monia erilaisia kenttämalleja, joista hän voi valita suosikkinsa. Valitun kenttämallin pohjalta generoidaan useita kenttäehdokkaita, joista seuraava pelattava kenttä valitaan pelaajan profiilin perusteella (Raffe ym. 2015). He kuitenkin mainitsevat, että kenttämallien esittäminen pelaajalle ennen kentän pelaamista vähentää mahdollisuuksia seikkailulle ja kentän tutkimiselle pelin aikana. Tästä syystä tämä metodi ei sellaisenaan ole sopiva luolastokenttien generoinnissa. Sen sijaan kyselyiden

kautta rakennettua pelaajaprofiilia voitaisiin käyttää yleisesti kaikkien kenttätyyppien generoinnissa (Raffe ym. 2015). Shaker, Yannakakis ja Togelius (2010) kuitenkin huomauttavat, että kyselyt voidaan kokea tungetteleviksi, ja ne saattavat häiritä pelikokemusta.

Pelaajan toimintaa voidaan rekisteröidä kenttätyypistä riippumatta, joten kokemusperustaisia metodeja voidaan käyttää yleisten pelikenttien generoinnissa. Kenttien mukauttaminen sopiviksi jokaista pelaajaa varten johtaisi potentiaalisesti juuri sellaisiin kenttiin, joiden pelaamisesta kukin pelaaja nauttii. Yannakakis ja Togelius (2011) kuitenkin mainitsevat pelaajasta kerätyn datan luotettavan tulkitsemisen olevan yksi kokemusperustaisten metodien suurimpia haasteita. Lisäksi kenttien mukautuminen pelaajan mieltymyksiin pitäisi olla nopeaa, mutta myös huomaamatonta (Yannakakis ja Togelius 2011). Yksi haaste on myös pelin ensimmäisten kenttien laadun takaaminen, sillä niiden generointia varten ei ole mahdollista kerätä riittävästi dataa. Esimerkiksi tutkijoiden Raffe ym. (2015) esittämässä metodissa ensimmäinen kenttä valitaan satunnaisesti.

3.4 Vastauslähtöiset metodit

Vastauslähtöisissä metodeissa keskitytään siihen, minkälaisia kenttiä halutaan generoida sen sijaan, että pohdittaisiin, kuinka niitä voidaan generoida (Neufeld, Mostaghim ja Perez-Liebana 2015). Pyrkimyksenä on haluttujen kenttien rakenteen, eli esimerkiksi seinien tai vihollisten sijainnin, formaali ilmaiseminen, jotta generointialgoritmi voisi tuottaa sen mukaisia kenttiä (Togelius, Shaker ja Nelson 2015a). Rakenteelle on myös pystyttävä asettamaan rajoitteita, joiden avulla voidaan esimerkiksi varmistaa, että kenttä on käyttökelpoinen tai että se on halutun kokoinen (Togelius, Shaker ja Nelson 2015a). Nämä vaatimukset voidaan ilmaista Answer Set Programming- eli ASP-ohjelmointikielillä (Togelius, Shaker ja Nelson 2015a).

ASP on ohjelmointiparadigma, jota noudatettaessa ei niinkään mietitä jonkin ongelman ratkaisukeinoja, vaan sen sijaan huomion kohteena on itse ongelma (Neufeld, Mostaghim ja Perez-Liebana 2015; Smith ja Mateas 2011). Ratkaistava ongelma ilmaistaan haluttua tulosta kuvaavilla säännöillä, jotka syötetään ASP-ratkaisijaohjelmalle, joka sitten tuottaa sääntöjen mukaisen tuloksen (Smith ja Mateas 2011). Proseduraalisen kenttägeneroinnin kohdalla

nämä säännöt koskevat kenttien rakennetta (Neufeld, Mostaghim ja Perez-Liebana 2015). Vastauslähtöisten metodien tärkein vaihe onkin kenttien haluttujen ominaisuuksien formalisoiminen sääntöinä, jotta ASP-ratkaisija voi niitä hyödyntää (Neufeld, Mostaghim ja Perez-Liebana 2015).

Kenttien rakenteeseen liittyvät säännöt voivat olla tosiasiaväitteitä, joiden avulla voidaan ilmaista esimerkiksi yksittäisen laatan koko luolastopelissä tai tason leveys tasohyppelypelissä (Togelius, Shaker ja Nelson 2015a). Tutkijoiden Togelius, Shaker ja Nelson (2015a) mukaan proseduraalisen generoinnin kannalta mielenkiintoisempia ovat kuitenkin niin sanotut valintasäännöt (engl. choice rules), joiden kohdalla ASP-ratkaisija satunnaisesti valitsee jonkin säännön asettamista vaihtoehdoista. Valintasääntö voi esimerkiksi määrätä, että kenttä sisältää satunnaisen määrän seinäobjekteja, jolloin generoidut kentät koostuisivat satunnaisesta määrästä seinäobjekteja satunnaisissa sijainneissa (Togelius, Shaker ja Nelson 2015a).

Valintasääntöjen satunnaisesta luonteesta johtuen ratkaisijalle on annettava myös rajoittavia sääntöjä, joiden avulla voidaan karsia generoitavien kenttien joukosta ne kentät, jotka hyvin todennäköisesti eivät olisi mielekkäitä (Neufeld, Mostaghim ja Perez-Liebana 2015; Togelius, Shaker ja Nelson 2015a). Rajoitteilla voidaan ilmaista, mitä ominaisuuksia generoiduilla kentillä ei tulisi missään tapauksessa olla (Smith ja Mateas 2011). Näin voidaan esimerkiksi varmistaa, että kentän jokainen huone tai taso on pelaajan saavutettavissa (Togelius, Shaker ja Nelson 2015a).

Vastauslähtöistä metodia noudattaessa kenttien laadun varmistukseen on useita tapoja. Smith ja Mateas (2011) ehdottavat metodia, jossa hyödynnetään jatkuvaa iterointia proseduraalisen generoinnin yhteydessä. Heidän metodissaan alkuun ASP-ratkaisijalle syötetään vain alustavia sääntöjä. Sitten näin generoituja kenttiä tarkastellaan manuaalisesti, ja epäonnistuneiden kenttien huonot ominaisuudet karsitaan rajoitteiden avulla. Lopulta saadut rajoitteet lisätään vanhojen sääntöjen joukkoon. Kenttien manuaalista tarkastelua ja uusien rajoitteiden lisäämistä toistetaan, kunnes generoidut kentät ovat tarpeeksi hyviä (Smith ja Mateas 2011).

Neufeld, Mostaghim ja Perez-Liebana (2015) puolestaan ehdottavat vastauslähtöistä metodia, jossa kenttien laatu varmistetaan eri tasoisia pelaajia mallintavien tekoälyagenttien avulla. Agenttien läpäisemät kentät ansaitsevat pisteitä, jolloin vain käyttökelpoiset kentät tu-

levat mukaan. Lisäksi he ehdottavat yksittäisen kentän vaikeustason arviointia vertaamalla huonosti pelaavan agentin kentälle kerryttämiä pisteitä hyvin pelaavan agentin kerryttämiin pisteisiin. He selittävät pienen piste-eron tarkoittavan, että kenttä on joko liian helppo tai liian vaikea, sillä eri tasoisten agenttien suoritus kentässä on yhtä hyvää. Kohtuullisen suuret piste-erot puolestaan viittaavat sopivan vaikeisiin kenttiin (Neufeld, Mostaghim ja Perez-Liebana 2015). Sopivan vaikeita kenttiä tuottavien ASP-sääntöjen joukko otetaan käyttöön lopullisessa kenttägeneroinnissa.

Vastauslähtöisten metodien suurin vahvuus on siinä, että hyviä ja valmiita ASP-ratkaisijoita on olemassa entuudestaan, joten sellaisen luomiseen ei tarvitse kuluttaa resursseja (Smith ja Mateas 2011). Lisäksi Smith ja Mateas (2011) mainitsevat ratkaisijoiden käytön mahdollistavan sen, että metodissa voidaan keskittyä kenttien yleisluontoisiin piirteisiin, sillä ratkaisija kykenee generoimaan toimivia kenttiä ilman yksityiskohtien määrittelyäkin. Toisaalta hyvien sääntöjen määritteleminen voi olla erittäin työlästä, mikä voidaan huomata esimerkiksi tutkijoiden Smith ja Mateas (2011) esittelemässä iteratiivisessa metodissa.

3.5 Kielioppiperustaiset metodit

Kielioppiperustaisissa metodeissa hyödynnetään jotakin generatiivista kielioppia (engl. generative grammar). Generatiivinen kielioppi koostuu jostakin aakkostosta ja joukosta sääntöjä. Säännöt määräävät, mitkä aakkoston merkit voidaan korvata joillakin toisilla merkeillä. Kielioppi on siis algoritmi, joka tuottaa merkkijonoja aakkoston ja sääntöjen pohjalta. (Dormans 2010)

Kielioppeja voidaan käyttää pelikenttien kuvaamiseen, kun sen aakkostona käytetään kenttien peruselementtejä kuvaavia symboleja (Dormans 2010). Dormans (2010) mainitsee, että säännöt on määriteltävä niin, että ne yhdisteleväisivät näitä peruselementtejä kenttien kannalta mielekkäällä tavalla. Hän myös antaa seuraavan yksinkertaisen esimerkin luolastokenttää kuvaavan kieliopin säännöistä:

1. Luolasto $\rightarrow$ Este + aarre
2. Este $\rightarrow$ avain + Este + lukko + Este
3. Este $\rightarrow$ vihollinen + Este

4. Este → huone

Tässä pienellä alkukirjaimella alkavat merkit ovat määritellyn aakkoston sisältämiä päätemerkkejä (engl. terminal symbols), ja isolla alkukirjaimella alkavat merkit ovat välikemerkkejä (engl. non-terminal symbols), jotka muutetaan sääntöjen pohjalta päätemerkeiksi. Sääntöjen välisten konfliktien ratkaisemiseksi säännöille voidaan asettaa tärkeysjärjestys, jolloin tärkeämmäksi asetettu sääntö suoritetaan vähemmän tärkeän säännön sijasta (Shaker ym. 2012). Dormans (2010) antaa lisäksi esimerkkejä merkkijonoista, joita tämä säännöstö voisi tuottaa:

1. avain + vihollinen + huone + lukko + vihollinen + huone + aarre
2. huone + aarre
3. vihollinen + vihollinen + vihollinen + vihollinen + huone + aarre

Hän myös huomauttaa, että generoidut merkkijonot eivät kuitenkaan välttämättä ole sellaisenaan sopivia kenttien pohjaksi. Tässä merkkijonon 2 kuvaama kenttä on liihan lyhyt jopa tämän yksinkertaisen esimerkin tapauksessa. Kenttien sopivuus voidaan kuitenkin taata luomalla parempia sääntöjä (Dormans 2010).

Kielioppiperustaisten metodien suurin vahvuus muihin metodeihen verrattuna on generoitujen kenttien ominaisuuksien helppo muokattavuus (Dormans 2010). Tämä mahdollistaa näiden metodien käytön yleisten kenttien generoinnissa, sillä jopa pienellä sääntöjen tai aakkoston muokkauksella voidaan saada aikaan toisenlaisia kenttiä. Dormans (2010) huomauttaa kuitenkin, että hyvien pohjasääntöjen luominen on erittäin haastavaa. Ne kuitenkin mahdollistavat kenttien generoinnin nopeasti ja luotettavasti (Dormans 2010).

4 Kenttägenerointimetodien yhtäläisyyksiä ja eroja

Tässä luvussa tarkastellaan eri generointimetodien pääluokkien ilmeisimpiä yhtäläisyyksiä ja eroja. Lisäksi pohditaan, kuinka erityyppisten metodien parhaita puolia voitaisiin yhdistellä.

Edellisen luvun pohjalta voidaan sanoa, että yhteistä kaikille metodeille on erityisesti varsinaista generointia edeltävän taustatyön tärkeys: Mallipohjaisia metodeja käytettäessä haluttujen kenttien mallien tunnistaminen toimii pohjana koko metodille. Kokemusperäisissä metodeissa oleellinen vaihe on erilaisten kokemusten ymmärtäminen ja mallintaminen. Vastauslähtöisten metodien tärkein vaihe puolestaan on haluttujen kenttien ominaisuuksien formalisoiminen, jotta ASP-ratkaisija voi niitä käyttää. Vastaavasti kielioppiperustaisissa metodeissa kenttien ominaisuudet on tunnistettava, jotta niiden pohjalta voitaisiin rakentaa kielioppi.

Käsiteltyjä metodien päätyyppejä voitaisiin erotella erityisesti sen perusteella, onko kenttiä arvioivien algoritmien käyttö pakollista metodin tuottamien kenttien laadun takaamiseksi. Arviointialgoritmit ovat algoritmeja, jotka automaattisesti arvioivat generoitujen kenttien eri ominaisuuksia (Linden, Lopes ja Bidarra 2014). Niitä voidaan käyttää muun muassa kenttien käyttökelpoisuuden tai sopivan vaikeustason varmistamisessa (Dahlskog ja Togelius 2014; Neufeld, Mostaghim ja Perez-Liebana 2015). Neufeld, Mostaghim ja Perez-Liebana (2015) kuitenkin huomauttavat, että yksittäinen arviointialgoritmi voi arvioida vain yhden kenttätyypin kenttiä, joten niiden käyttö heikentää metodin yleisluontoisuutta. Lisäksi sopivien arviointialgoritmien luominen sekä itse kenttien arvioiminen vaatii aikaa, mikä hidastaa metodia. Näistä syistä nopeaa ja monipuolista generointia tavoiteltaessa toivottavia olisivat sellaiset metodit, jotka eivät vaadi arviointialgoritmien käyttöä.

Erityisesti vastauslähtöisissä ja kielioppiperustaisissa metodeissa arviointialgoritmien käyttö ei välttämättä ole pakollista, sillä kenttien käyttökelpoisuus ja mielenkiintoisuus voidaan varmistaa luomalla paljon hyviä sääntöjä generoinnin pohjaksi (Neufeld, Mostaghim ja Perez-Liebana 2015; Dormans 2010). Toisaalta Dormans (2010) mainitsee, että hyvien pohjasääntöjen luominen on erittäin haastavaa, jolloin itse metodin toteutukseen kuluu enemmän aikaa. Smith ym. (2009) tukevat tätä sanomalla, että sääntöjen lisääminen huonojen kent-

tien karsimiseksi usein karsii myös hyviä kenttiä. Siksipä arviointialgoritmeja hyödynnetään usein myös näiden metodien yhteydessä, kuten esimerkiksi Shaker ym. (2012) ovat tehneet kielioppiperustaisen metodin kohdalla ja Neufeld, Mostaghim ja Perez-Liebana (2015) vastauslähtöisen metodin kohdalla.

Eri kenttägeneroinnin metodeita voidaan myös yhdistää ja näin koota metodien hyviä puolia yhteen. Esimerkiksi Compton ja Mateas (2006) ehdottavat kielioppien käyttöä malliperustaisen metodin kanssa, jolloin voidaan välttää arviointialgoritmien tarve myös malliperustaisen generoinnin yhteydessä. Heidän metodissaan päätemerkit symboloivat kenttien peruselementtejä eli mikromalleja. Välikemerkkien ja mielekkäiden sääntöjen avulla puolestaan kuvataan monimutkaisemmat mallit eli mesomallit (Compton ja Mateas 2006). Dormans (2010) sen sijaan ehdottaa kielioppiperustaisten ja kokemusperustaisten metodien yhteistyötä. Hän mainitsee, että kieliopin sääntöjä tai välikemerkkejä voidaan muokata pelaajan pelin aikaisen toiminnan pohjalta. Esimerkiksi monipuolisemman pelikokemuksen aikaansaamiseksi voitaisiin tarkkailla, minkä tyyppisiä vihollisia pelaaja kohtaa. Kohtaamisen seurauksena kieliopin säännöt muuttuisivat niin, että muunlaisten vihollisten määrä kasvaisi, kun taas jo kohdattujen vihollisten määrä vähenisi. Vastaavasti vastauslähtöisen metodin sääntöjen pelin aikaisen muokkauksen voidaan olettaa olevan mahdollista.

5 Yhteenveto

Tässä tutkielmassa käsiteltiin proseduraalista kenttägenerointia tavoitteena löytää generointimetodeja, joilla voitaisiin luoda useita erityyppisiä kenttiä. Tätä varten kenttien vaatimuksia tarkasteltiin yleisesti, ja lisäksi eri kenttätyyppien ominaisuuksia käsiteltiin tarkemmin. Kenttien vaadittujen ominaisuuksien pohjalta esiteltiin kenttägenerointimetodien yleisvaatimuksia. Tutkielmassa löydettiin neljänlaisia generointimetodeja, jotka täyttivät asetetut vaatimukset ja joiden avulla on mahdollista luoda useita erityyppisiä kenttiä. Näiden generointimetodien luokkien pääpirteet esiteltiin, ja lopuksi eri metodityyppejä lyhyesti vertailtiin keskenään.

Eri metodityyppejä vertailtaessa huomattiin hyvän taustatyön olevan yksi tärkeimpiä vaiheita metodista riippumatta. Metodin riippuvuuden kenttien automaattisesta arvioinnista havaittiin olevan suurimpia sen yleisluontoisuutta ja nopeutta heikentäviä tekijöitä. Toisaalta arviointialgoritmien avulla metodin luotettavuuden ja generoitujen kenttien laadun takaamisen havaittiin olevan helpompaa.

Tämän tutkielman yhteydessä tehdyssä kirjallisuuskatsauksessa ei löytynyt sellaisia tutkimuksia, joissa metodien kaikkia oleellisia tietoja olisi kvantitatiivisesti mitattu metodien käytön yhteydessä. Eräs jatkotutkimusmahdollisuus voisi siis olla tässä tutkielmassa esiteltyjen metodien vertailu käytännön kokeilujen kautta. Tämä mahdollistaisi muun muassa metodien kontrolloitavuuden, nopeuden ja generoitujen kenttien laadun tarkemman vertailun. Myös eri generointi- ja arviointialgoritmien vaikutusta eri metodien kohdalla voitaisiin yksityiskohtaisemmin vertailla käytännön testien avulla.

Lähteet

Baghdadi, Walaa, Fawzya Shams Eddin, Rawan Al-Omari, Zeina Alhalawani, Mohammad Shaker ja Noor Shaker. 2015. “A Procedural Method for Automatic Generation of Spelunky Levels”. Teoksessa *Applications of Evolutionary Computation,* 305–317. Springer.

Cardamone, Luigi, Daniele Loiacono ja Pier Luca Lanzi. 2011. “Interactive Evolution for the Procedural Generation of Tracks in a High-End Racing Game”. Teoksessa *Proceedings of the 13th annual conference on Genetic and evolutionary computation,* 395–402. ACM.

Cardamone, Luigi, Georgios N Yannakakis, Julian Togelius ja Pier Luca Lanzi. 2011. “Evolving Interesting Maps for a First Person Shooter”. Teoksessa *Applications of Evolutionary Computation,* 63–72. Springer.

Compton, Kate, ja Michael Mateas. 2006. “Procedural Level Design for Platform Games”. Teoksessa *AIIDE,* 109–111.

Dahlskog, Steve, J Togelius ja S Björk. 2015. “Patterns, Dungeons and Generators”. Teoksessa *Proceedings of the 10th Conference on the Foundations of Digital Games.*

Dahlskog, Steve, ja Julian Togelius. 2014. “Procedural Content Generation Using Patterns as Objectives”. Teoksessa *Applications of Evolutionary Computation,* 325–336. Springer.

Diaz-Furlong, H.A., ja A.L. Solis-Gonzalez Cosio. 2013. “An Approach to Level Design Using Procedural Content Generation and Difficulty Curves”. Teoksessa *Computational Intelligence in Games (CIG), 2013 IEEE Conference,* 1–8. Elokuu.

Dormans, Joris. 2010. “Adventures in Level Design: Generating Missions and Spaces for Action Adventure Games”. Teoksessa *Proceedings of the 2010 workshop on procedural content generation in games,* 1. ACM.

Hullett, Kenneth, ja Jim Whitehead. 2010. “Design Patterns in FPS Levels”. Teoksessa *proceedings of the Fifth International Conference on the Foundations of Digital Games,* 78–85. ACM.

Linden, Ruud van der, Roseli Lopes ja Rafael Bidarra. 2014. “Procedural generation of dungeons”. *Computational Intelligence and AI in Games, IEEE Transactions on* 6 (1): 78–89.

Neufeld, Xenija, Sanaz Mostaghim ja Diego Perez-Liebana. 2015. “Procedural Level Generation with Answer Set Programming for General Video Game Playing”. Teoksessa *Computer Science and Electronic Engineering Conference (CEEC), 2015 7th,* 207–212. IEEE.

Ong, Justin. 2014. *The Rise of Indie Game Development.* Saatavilla WWW-muodossa, `http://www.acmi.net.au/acmi-channel/2014/the-rise-of-indie-game-development/`, viitattu 10.2.2016.

Preuss, Mike, Antonios Liapis ja Julian Togelius. 2014. “Searching for Good and Diverse Game Levels”. Teoksessa *Computational Intelligence and Games (CIG), 2014 IEEE Conference on,* 1–8. IEEE.

Raffe, William L, Fabio Zambetta, Xiaodong Li ja Kenneth O Stanley. 2015. “Integrated Approach to Personalized Procedural Map Generation Using Evolutionary Algorithms”. *Computational Intelligence and AI in Games, IEEE Transactions on* 7 (2): 139–155.

Shaker, Noor, Miguel Nicolau, Georgios N Yannakakis, Julian Togelius ja Michael O Neill. 2012. “Evolving Levels for Super Mario Bros Using Grammatical Evolution”. Teoksessa *Computational Intelligence and Games (CIG), 2012 IEEE Conference on,* 304–311. IEEE.

Shaker, Noor, Georgios N Yannakakis ja Julian Togelius. 2010. “Towards Automatic Personalized Content Generation for Platform Games.” Teoksessa *AIIDE.*

Smith, Adam M, ja Michael Mateas. 2011. “Answer Set Programming for Procedural Content Generation: A Design Space Approach”. *Computational Intelligence and AI in Games, IEEE Transactions on* 3 (3): 187–200.

Smith, Gillian, ja Jim Whitehead. 2010. “Analyzing the Expressive Range of a Level Generator”. Teoksessa *Proceedings of the 2010 Workshop on Procedural Content Generation in Games,* 4. ACM.

Smith, Gillian, Mee Cha ja Jim Whitehead. 2008. “A Framework for Analysis of 2D Platformer Levels”. Teoksessa *Proceedings of the 2008 ACM SIGGRAPH symposium on Video games,* 75–80. ACM.

Smith, Gillian, Mike Treanor, Jim Whitehead ja Michael Mateas. 2009. “Rhythm-Based Level Generation for 2D Platformers”. Teoksessa *Proceedings of the 4th International Conference on Foundations of Digital Games,* 175–182. ACM.

Sorenson, Nathan, ja Philippe Pasquier. 2010a. “The Evolution of Fun: Automatic Level Design through Challenge Modeling”. Teoksessa *Proceedings of the First International Conference on Computational Creativity (ICCCX). Lissabon, Portugali: ACM,* 258–267.

———. 2010b. “Towards a Generic Framework for Automated Video Game Level Creation”. Teoksessa *Applications of Evolutionary Computation,* 131–140. Springer.

Sorenson, Nathan, Philippe Pasquier ja Steve DiPaola. 2011. “A Generic Approach to Challenge Modeling for the Procedural Creation of Video Game Levels”. *Computational Intelligence and AI in Games, IEEE Transactions on* 3 (3): 229–244.

Thongkham, Cheylene. 2015. *The Rise of Indie Game Developers.* Saatavilla WWW-muodossa, `https://leantesting.com/resources/rise-indie-game-developers/`, viitattu 10.2.2016.

Togelius, Julian, Renzo De Nardi ja Simon M Lucas. 2007. “Towards automatic personalised content creation for racing games”. Teoksessa *Computational Intelligence and Games, 2007. CIG 2007. IEEE Symposium on,* 252–259. IEEE.

Togelius, Julian, Emil Kastbjerg, David Schedl ja Georgios N. Yannakakis. 2011. “What is Procedural Content Generation?: Mario on the Borderline”. Teoksessa *Proceedings of the 2nd International Workshop on Procedural Content Generation in Games,* 3:1–3:6. PCGames ’11. Bordeaux, Ranska: ACM.

Togelius, Julian, Mike Preuss ja Georgios N Yannakakis. 2010. “Towards Multiobjective Procedural Map Generation”. Teoksessa *Proceedings of the 2010 workshop on procedural content generation in games,* 3. ACM.

Togelius, Julian, Noor Shaker ja Mark J. Nelson. 2015a. “ASP with applications to mazes and levels (DRAFT)”. Teoksessa *Procedural Content Generation in Games: A Textbook and an Overview of Current Research,* toimittanut Noor Shaker, Julian Togelius ja Mark J. Nelson. Springer.

Togelius, Julian, Noor Shaker ja Mark J. Nelson. 2015b. “Introduction”. Teoksessa *Procedural Content Generation in Games: A Textbook and an Overview of Current Research,* toimittanut Noor Shaker, Julian Togelius ja Mark J. Nelson. Springer.

———. 2015c. “The experience-driven perspective (DRAFT)”. Teoksessa *Procedural Content Generation in Games: A Textbook and an Overview of Current Research,* toimittanut Noor Shaker, Julian Togelius ja Mark J. Nelson. Springer.

Togelius, Julian, Georgios N Yannakakis, Kenneth O Stanley ja Cameron Browne. 2011. “Search-Based Procedural Content Generation: A Taxonomy and Survey”. *Computational Intelligence and AI in Games, IEEE Transactions on* 3 (3): 172–186.

Yannakakis, Georgios N, ja Julian Togelius. 2011. “Experience-Driven Procedural Content Generation”. *Affective Computing, IEEE Transactions on* 2 (3): 147–161.