Elina Keränen

Hakuperustainen proseduraalinen sisällöntuotanto tietokonepeleissä

Tietotekniikan kandidaatintutkielma

28. huhtikuuta 2015

Jyväskylän yliopisto

Tietotekniikan laitos

Tekijä: Elina Keränen

Yhteystiedot: elinakera@gmail.com

Työn nimi: Hakuperustainen proseduraalinen sisällöntuotanto tietokonepeleissä

Title in English: Procedural Content Generation in Games

Työ: Kandidaatintutkielma

Sivumäärä: 26+0

Tiivistelmä: Proseduraaliset sisällöntuotantomenetelmät ovat viime aikoina kasvattaneet suosiotaan akateemisessa tutkimuksessa, erityisesti hakuperustaiset optimointiongelmia ratkaisevat menetelmät. Peliteollisuudessa näistä menetelmistä voi olla suurta hyötyä etenkin pienille yrityksille sekä yksittäisille ohjelmoijille, joilla ei ole resursseja tuottaa paljon sisältöä manuaalisesti. Aihetta käsittelevillä tutkimuksilla on tärkeä merkitys menetelmien kehittämisessä riittävän toimiviksi, että niitä voitaisiin käyttää pelituotannossa.

Avainsanat: pelit, proseduraalinen sisällöntuotanto, evolutiiviset algoritmit, hakuperustainen sisällöntuotanto

Abstract: Procedural content generation has recently been a surge in academic research. Especially search-based approaches have seen a rise in popularity. For the game industry these methods can be useful especially for small businesses and individual programmers, who do not have the resources to produce a lot of content manually. Studies on this theme play an important role in the developing methods to work adequately, so that they could be used for game production.

Keywords: games, procedural content generation, evolutionary algorithms, search-based content generation

Kuviot

Sisältö

1 Johdanto

Proseduraalinen sisällöntuotanto (engl. *Procedural Content Generation; PCG*) on pelisisällön luomista algoritmisesti. Sisällöntuotanto on kaikkein eniten kustannuksia vaativa osa pelituotannossa (Togelius, Shaker & Nelson, 2015), ja PCG:n tavoitteena on vastata tähän ongelmaan: se pyrkii automatisoimaan sisällöntuotantoa, mikä nopeuttaa tuotantoprosessia ja laskee kustannuksia. PCG-menetelmiä onkin jo pitkään käytetty pelituotannossa, ja tunnettuja ovat esimerkiksi Elite, Diablo, Dwarf Fortress sekä Rogue ja roguelike -genren pelit.

Yksi suuri ongelma PCG:ssä on, että sen toiminta voi olla ennakoimatonta tai ennakoiminen vaatii suurta panostusta algoritmiin (Frade, de Vega & Cotta, 2012), ja hakuperustaisista menetelmistä (engl. *search-based*, *SBPCG*) toivotaan ratkaisua tähän ongelmaan. Käsitteelle ei ole vakiintunutta suomenkielistä määritelmää, mutta tässä tutkielmassa käytetään käsitettä hakuperustaiset menetelmät. Se on yleisnimitys ratkaisumenetelmille jotka käsittelevät pelien sisällöntuotannon kombinatorisia optimointiongelmia. Menetelmien tarkoituksena on käydä ratkaisuvaihtoehtoja tehokkaasti läpi ja arvioida tulosten laatua.

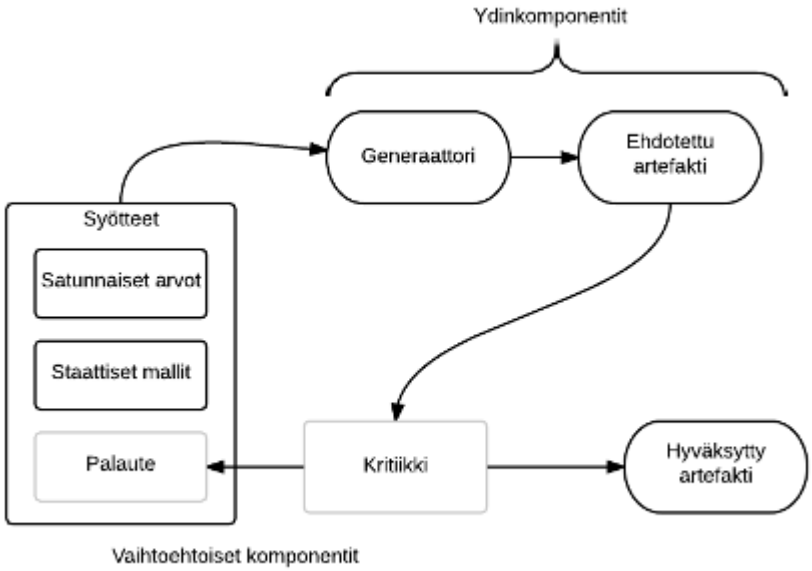
Tämän tutkimuksen tarkoituksena on luoda yleisnäkymä proseduraaliseen sisällöntuotantoon sekä selvittää, mikä on hakuperustainen sisällöntuotanto, mitkä ovat sen suurimmat ongelmat ja missä kehitysvaiheessa se on tutkimuksessa ja peliteollisuudessa. Tutkimusstrategiana on systemaattinen kirjallisuuskatsaus, joka perustuu pääasiassa kuluvan vuosikymmenen aikana julkaistuihin tieteellisiin artikkeleihin.

Toinen luku käsittelee proseduraalista sisällöntuotantoa yleisellä tasolla. Kolmas luku käsittelee hakuperustaista lähestymistapaa proseduraaliseen sisällöntuotantoon. Neljännessä luvussa esitellään tarkemmin mitä hakuperustaisilla menetelmillä on tehty käytännössä. Viidennessä luvussa tehdään yhteenveto ja hahmotellaan jatkotutkimusaiheita.

2 Proseduraalinen sisällöntuotanto

Proseduraalinen sisällöntuotanto on siis pelisisällön tuottamista algoritmisesti. Yleisesti proseduraalisessa järjestelmässä syötteet, jotka voivat olla esim. numerotaulukoita tai manuaalisesti piirrettyjä malleja, annetaan systeemin pääkomponentille, generaattorille, joka yhdistelee ja muokkaa näitä syötteitä ja luo niistä artefakteja (Compton, Osborn & Mateas, 2013, s. 2), eli yksittäisiä esiintymiä sisällöstä. Yksinkertaisimmillaan syöte voisi taulukko, joka sisältää esineen väri- ja kokovaihtoehdot, ja generaattori ottaa satunnaisesti jotkin arvot ja piirtää tämän esineen annetuilla parametreilla.

Kritiikki, eli tuloksena saadun artefaktin arvioiminen, taas on valinnainen komponentti, ja sitä käytetään testivetoisissa menetelmissä. Yleisgeneraattori on esitetty kaaviossa 1, mukaillen Compton ym. (2013, s. 2), ja tätä mallia voidaan käyttää yleisesti generaattorien analysoinnissa ja hahmottamisessa.



Kuvio 1. PCG:n yleisrakenne.

Tärkeä osa määritelmää on, että käyttäjän tuottamat syötteet vaikuttavat PCG:hen joko vain epäsuorasti tai ei ollenkaan (Togelius ym., 2015, s. 6). Näin esimerkiksi

karttaeditori, jonka avulla pelaaja voi itse piirtää karttoja tai muuten muokata niitä, ei ole PCG:tä. Sen sijaan pelaajan mieltymyksiin mukautuva karttageneraattori, jossa algoritmi kerää parametrinsa pelaajan toiminnasta ja generoi niiden perusteella uutta sisältöä, on PCG:tä, koska siinä käyttäjän syötteet vaikuttavat vain epäsuorasti.

Compton ym. (2013, s. 2) tarkentavat, että generaattorin täytyy sisältää myös konteksti jossa sitä käytetään. Siten esimerkiksi ohjelmointikielten randomlukugeneraattoreita ei voida laskea PCG:hen, koska niiden tuloksilla ei ole itsestäänselvää käyttötarkoitusta, toisin kuin esim. puulla tai kartalla. Satunnaisluku voi kuitenkin olla osa syötettä, esimerkiksi esineiden paikat voidaan arpoa koordinaatteina, jolloin syöte olisi yhdistelmä satunnaislukua ja esinettä, ja tuloksena saataisiin esine jossain tietyssä kohdassa karttaa.

2.1 Mikä on sisältöä?

Keskeinen käsite PCG:ssä on "sisältö". Togelius ym. (2015, s. 1–2) määrittelevät sen hyvin laajasti: sisältöön lasketaan lähes kaikki mitä peli sisältää, siis esineiden, karttojen, maailman ja pelaajahahmojen lisäksi myös musiikki, tarina ja säännöt. Myös koko maailman voi luoda proseduraalisesti, kuten tehdään Dwarf Fortressissa (Dwarf Fortress, 2015). Kuitenkaan esimerkiksi tekoälyt tai pelimoottorit eivät ole sisältöä. Eivät myöskään pelaajan luomat sisällöt (Togelius, Kastbjerg, Schedl & Yannakakis, 2011a, s. 2), kuten tehdään esim. SimCityssä (SimCity, 2015). Generointi voi kuitenkin osittain pohjautua pelaajan tuottamaan sisältöön, esimerkiksi valikoida niistä parhaat ja generoida niiden perusteella uutta sisältöä.

Kuten Compton ym. (2013, s. 2–3) huomauttavat, on tärkeää huomioida että useilla proseduraalisen sisällöntuotannon osa-alueilla on juurensa jollain toisella alalla, esim. musiikkia ja grafiikkaa on muutenkin pitkään tuotettu proseduraalisesti. PCG:ssä on siis haasteena yhdistää eri alojen tietämystä sekä huomioida pelien omat erityispiirteet. Lisäksi jotkut osa-alueet, kuten sääntöjen tai pelin tasojen generointi ovat täysin pelispesifejä.

2.2 Taksonomia

Erityyppisiä PCG:itä on monia, ja Togelius, Yannakakis, Stanley & Browne (2011b, s. 173–174) määrittelevät niille seitsenosaisen taksonomian. Tutkimus käsittelee hakuperustaista PCG:tä, mutta Togelius ym. (2015, s. 7–9) täydentävät ja selventävät jaottelua siten että se koskee myös kaikkia PCG:itä. Taksonomia voi olla hyödyllinen analysoitaessa erilaisia ratkaisuja ja miettiessä mitä pitää ottaa huomioon generaattoria luotaessa.

Algoritmin ominaisuuksia ovat deterministisyys, kontrolloitavuus, adaptiivisuus ja automaattisuus. Sisällön ominaisuuksia ovat välttämättömyys sekä syötteen epäsuoruus, jonka määrittelivät Compton ym. (2013). Muita luokitteluja ovat generaattoreiden konstruktiivisuus sekä vaihe, jossa generointi tapahtuu.

Yksi PCG:n suurimmista ongelmista ovat *katastrofaaliset virheet*, eli peli lakkaa olemasta pelattava Togelius ym. (2011b, s. 12), (Zafar & Mujtaba, 2012, s. 62), (Togelius ym., 2015) - peli kaatuu, sitä ei ole mahdollista pelata läpi, tai se on liian helppo - lyhyesti, jotain mikä tuhoaa pelikokemuksen kokonaan. Näiden virheiden ennakoimisen ja estämisen voidaan ajatella olevan minimivaatimus mille tahansa generaattorille.

Konstruktiivisuus

Generaattori voi olla yksinkertaisesti konstruktiivinen niin, että algoritmi ajetaan kerran läpi ja se luo kerralla kaiken sisällön. Toinen lähestymistapa on testivetoinen generointi: generoitu sisältö testataan ja sille annetaan hyväksyntä tai hylkäys ja sitten muokataan tarpeen mukaan (Togelius ym., 2015). Testauksen palauttama tulos voi olla eksplisiittinen hyväksyntä/hylkäys, numeerinen hyvyysarvo eli "arvosana", tai korjattu artefakti (Compton ym., 2013, s. 2). Testivetoisen menetelmän erikoistapaus ovat hakuperustaiset menetelmät, joissa testaus on yksi tärkeä komponentti. Tätä lähestymistapaa esitellään tarkemmin luvussa 3.

Konstruktiivisen ratkaisun heikko puoli on sen epävarmuus, katastrofaaliset virheet eivät aina ole ennakoitavissa. Kuten Valtchanov & Brown (2012, s. 27) huomautta-

vat, konstruktiivinen ratkaisu pakottaa kirjoittamaan säännöt algoritmiin, mikä tekee tällaisten generaattorien arvioinnin ja kontrolloinnin vaikeaksi. Lisäksi se vaatii jokaiselle rakenteelle uuden ja usein hyvin erilaisen generaattorin, mikä vähentää generaattoreiden uudelleenkäyttöarvoa.

Testivetoisessa menetelmässä puolestaan jo generoitu sisältö testataan jonkin tietyn kriteerin perusteella, ja näin esimerkiksi ei olisi mahdollista luoda sokkeloa jossa päätepisteeseen ei ole reittiä, jos testausfunktio rakennetaan sellaiseksi että se testaa läpipelattavuuden. Testivetoisen systeemi hankalin puoli on, että se pitää kyetä kehittämään riittävän nopeaksi.

Sisältö

PCG voi generoida välttämätöntä tai valinnaista sisältöä (Togelius ym., 2011b, s. 173). Välttämätöntä sisältöä tuottava ratkaisu on esimerkiksi tasogeneraattori, jonka luotettavuuden pitää olla korkealla tasolla — mitä jos jostain tasosta ei voisi päästä läpi ollenkaan, koska generaattori loi seinän ulospääsyportin eteen? Valinnaisen sisällön generoinnissa taas voidaan usein joustaa laatuvaatimuksista: jos luodaan vaikkapa taustagrafiikkaa, pahinta mitä voi tapahtua on että puu näyttää omituiselta — se ei voi aiheuttaa pelaamisen kannalta kriittisiä katastrofaalisia virheitä.

Toisaalta valinnaista sisältöä voivat olla myös artefaktit joita pelaaja voi käyttää, mutta voi myös valita olla käyttämättä (Togelius ym., 2011b, s. 173). Generaattoria rakentaessa on siis tärkeää miettiä pahinta skenaariota mitä ratkaisun virheet voivat tuottaa ja nostaa luotettavuutta tarpeen mukaan riittävän korkeaksi. Välttämättömän sisällön generointi hyväksyttävästi onkin yksi PCG:n suurimmista ongelmista (Togelius ym., 2011b, s. 183). Koska hakuperustaiset menetelmät kykenevät arvioimaan tuottamaansa sisältöä, ne ovat hyvä valinta, jos täytyy tuottaa välttämätöntä sisältöä.

Toinen sisällölle huomioitava ominaisuus on rakeisuus (Compton ym., 2013, s. 4), jolla tarkoitetaan sitä, kuinka pieniä komponentteja syötetään. Tämä riippuu paljon algoritmista, esimerkiksi yksinkertainen L-systeemi jolla generoidaan kasveja,

tarvitsee syötteenä vain piirrettävän viivan pituuden sekä astemäärän joka käännytään kulmissa (Prusinkiewicz & Lindenmayer, 1991, s. 7). Numerot ovat siis kaikkein alimman tason syöte. Ylemmän tason syötteitä olisivat esimerkiksi valmiiksi piirretyt kasvit, joita generaattori kopioi ja muokkaa, tai labyrintin osat, joita generaattori yhdistelee säännöillään.

Vaihe

Sisältö voidaan generoida pelaamisen aikana, online, tai pelinkehityksen aikana, offline (Togelius ym., 2015, s. 7). Online-generaatiota on esimerkiksi se, että pelaajan avatessa oven rakennukseen, peli generoi kyseisen rakennuksen. Offline-generaatiossa taas algoritmi ehdottaa ulkoasuja ja ihmissuunnittelija muokkaa niitä ennenkuin peli menee myyntiin. Ratkaisu voi olla myös jotain näiden kahden puolivälistä, kuten tehtiin Infinite Mario Brosissa: seuraavaa karttaa luodaan samalla kun pelaaja pelaa edellistä, perustuen pelaajan painamiin näppäimiin (Togelius ym., 2011a, s. 4).

Erityisesti online-ratkaisuissa on tärkeää että generointi on riittävän nopeaa ja ennustettavaa. Online-ratkaisun hyöty on siinä että uutta sisältöä voidaan luoda loputtomasti, offline-ratkaisut taas ovat hyödyllisiä monimutkaisissa ympäristöissä. (Togelius ym., 2015, s. 7). Hakuperustaiset menetelmät ovat vahva ehdokas online-generaatioon, koska arvioinnin ansiosta vältetään katastrofaaliset virheet jotka johtaisivat pelattavuuden estymiseen kesken pelin.

Algoritmi

PCG-ratkaisu voi olla stokastinen tai deterministinen. Jos deterministiselle koneelle antaa samat lähtöparametrit, se tuottaa aina saman tuloksen, kun taas stokastiset, kuten myöhemmin esitellyt evolutiiviset algoritmit, voivat tuottaa erilaisia ratkaisuja joka kerralla. Deterministinen algoritmi on siis enemmänkin tiedon pakkaamista (Togelius ym., 2011b, s. 3). Deterministisiä ratkaisuja on käytetty esimerkiksi hyvin aikaisissa PCG-ratkaisuissa, esim. Elite (1980) (Hendrikx, Meijer, van der Velden & Iosup, 2013, s. 3). Stokastisessa ratkaisussa tulosta ei voi täysin kontrolloida jolloin

luotettavuutta voi olla vaikea varmentaa. Hakuperustaisissa menetelmissä luotettavuusongelma korjataan arvioimalla jokainen tuotettu artefakti.

Generaattoreita voidaan kontrolloida eri tavalla. Jos generaattori perustuu yhdelle siemenluvulle, sitä on vaikea kontrolloida. Toinen lähestymistapa on määritellä useita parametreja joita voidaan säädellä (Togelius ym., 2015, s. 8). Esimerkiksi sokkelo voidaan generoida laittamalla sen parametreiksi huoneiden määrä, käytävien haarautuvuus ja esineiden mahdolliset paikat.

Geneerinen systeemi ei huomioi pelaajaa lainkaan, kun taas adaptiivinen systeemi tarkkailee pelaajaa ja muokkaa sisältöä hänen pelitapansa huomioiden. Geneeristä systeemiä on käytetty paljon, ja adaptiivisuus on herättänyt viime aikoina paljon kiinnostusta (Togelius ym., 2015, s. 8). Adaptiivista systeemiä käyttää esim. Galactic Arms Race, joka luo seuraavan sukupolven aseet sen perusteella, mitä pelaaja käytti eniten (Hastings, Guha & Stanley, 2009). Adaptiivisessa systeemissä on tärkeää nopeus: pelaajan ei pidä joutua odottelemaan tuloksen valmistumista. Kuten online-generaatio, myös adaptiivisuus on turvallisempi toteuttaa hakuperustaisilla menetelmillä, jotta pystytään reagoimaan virheisiin.

Sisällön luonti voi olla täysin automaattista siinä mielessä että riittää pelkästään käynnistää generaattori, tai pelisuunnittelija tai pelaaja saattaa voida vaikuttaa siihen jotenkin (Togelius ym., 2015, s. 9): suunnittelija voi määritellä parametreja eksplisiittisesti tai arvioida lopputulosta karsimalla huonoimmat pois. Vaikka PCG:n tavoitteena onkin tehdä sisällönluonnista automaattista, myös ihmissuunnittelijan lisääminen voi monissa tapauksissa olla hyödyllistä, esimerkiksi jos tarkoitus on arvioida asioita joita kone ei kykene arvioimaan, kuten esteettisyyttä, kauneutta, uniikkiutta tai kiinnostavuutta (Compton ym., 2013, s. 2).

Näitä ominaisuuksia voidaan käyttää erilaisina kombinaatioina, ja näillä kombinaatioilla on erilaisia vaikutuksia ja hyviä ja huonoja puolia. Esimerkiksi online-generaatio yhdistettynä adaptiivisuuteen lakkasi Mario-pelissä olemasta PCG ja muuttui pelaajan työkaluksi Togelius ym. (2011a, s. 2), koska pelaaja pystyi käyttämään pelin mukautuvuutta hyväkseen ja luomaan kentistä mieleisiään.

3 Hakuperustaiset menetelmät

Luvussa 2.2 esiteltyyn taksonomiaan sijoittuen, hakuperustaiset menetelmät ovat testivetoisia ja stokastisia. Niiden generoima sisältö siis ei ole joka kerta samanlaista, ja ne testaavat tuotetun sisällön laadun. Hakuperustaisten menetelmien ideana on etsiä määritellystä hakuavaruudesta sisältöä ja arvioida täyttääkö se vaaditut ominaisuudet, minkä jälkeen muokataan hakuavaruutta (Togelius & Shaker, 2015, s. 17-18)

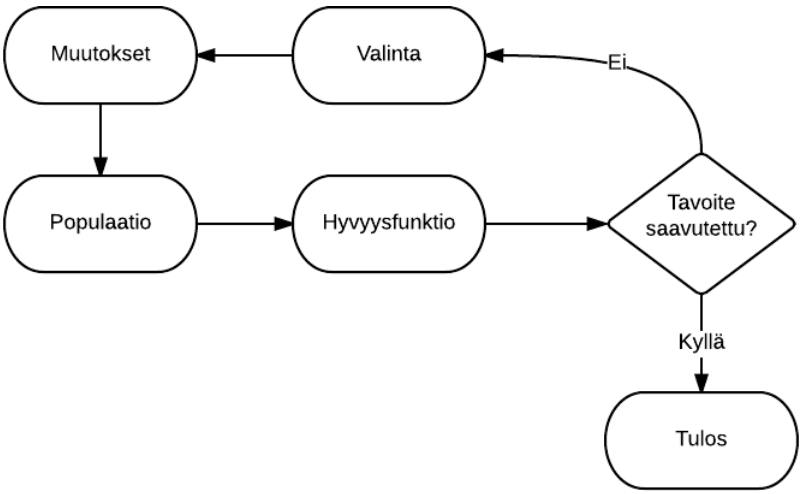
Tässä osiossa käsitellään hakuperustaisten menetelmien toiminta yleisesti, sen komponentit sekä esimerkkipelejä. Esittelyn ei ole tarkoitus olla kattava katsaus, vaan yleissilmäys. Hakuperustaisen sisällöntuotannon teoreettinen tutkimus peleissä on vielä hyvin aikaisessa vaiheessa, joten tämä osio nojaa teoreettisella tasolla suureksi osaksi Togeliuksen julkaisuihin.

Peleissä sisällöntuotannon kombinatorisia optimointiongelmia proseduraalisesti ratkaistaessa, näitä ratkaisumenetelmiä kutsutaan hakuperustaisiksi (engl. *search-based*). Kombinatoriset optimointongelmat ovat hyvin yleinen laskennan haaste. Niissä mahdollisia ratkaisuja ongelmaan on äärellinen, mutta eksponentiaalisesti kasvava määrä, joten parhaan ratkaisun löytämiseksi tarvittava aika kasvaa eksponentiaalisesti (Simula, 1997). Tätä varten laskennallisissa ongelmissa käytetään metaheuristisia ja muita hakuun ja optimointiin perustuvia algoritmeja, joilla etsitään parasta mahdollista tai riittävän hyvää ratkaisua.

Kuten metaheuristisissa menetelmissä yleisesti (Blum & Roli, 2003, s. 292), myös PCG:ssä hakuperustaisten menetelmien tavoitteena on käydä hakuavaruutta tehokkaasti läpi, löytää alueet joilla parhaimmat ratkaisut sijaitsevat sekä siirtyä tarpeen mukaan alueille joita ei ole vielä tutkittu. Hakuavaruutta voi siis ajatella konkreettisesti avaruutena: esimerkiksi labyrinttia luotaessa x-akseli voisi olla lyhimmän mahdollisen reitin pituus, y-akseli polkujen haarautuvuus ja z-akseli vihollisten määrä. Hakualgoritmin tehtävänä on siis etsiä tästä kolmiulotteisesta avaruudesta alueet, joilla nämä kolme parametria ovat toisiinsa sopivassa suhteessa siten että määritel-

lyt tavoitteet täyttyvät. Tavoitteeksi voi määritellä esimerkiksi että peli ei ole liian helppo tai se on läpipelattava. Kun algoritmi ehdottaa tiettyjä yksilöitä, arviointifunktio kertoo täyttyvätkö nämä tavoitteet.

Tärkeimmät osat ovat populaatio, arviointifunktio (hyvyysfunktio) sekä valinta. Komponentit esitetään kaaviossa 2, mukaillen Togelius ym. (2011b, s. 175). Populaatio on joukko esiintymiä sisällöstä jota tuotetaan. Kaikki hakuperustaiset PCG:t eivät käytä sisältökandidaattien joukkoa, mutta monet yleiset ratkaisut käyttävät. Hyvyysfunktion perusteella arvioidaan tätä populaatiota. Menetelmä vahvuus on iteroinnissa: arviointifunktion avulla säilytetään parhaat ratkaisut, poistetaan huonoimmat, muokataan hakuavaruutta ja tätä toistetaan kunnes tavoitteet on saavutettu (Togelius & Shaker, 2015, s. 17).



Kuvio 2. SB:n yleisrakenne.

Hakuperustaisia menetelmiä on useita, ja peleissä käytetään samoja algoritmeja kuin muissakin kombinatorisissa optimointiongelmissa. Yleisiä ovat evolutiiviset algoritmit, neuroverkot ja parviäly. Pelilliset erityispiirteet täytyy kuitenkin huomioida esim. sisältörepresentaation suunnittelussa. Ydinongelmat hakuperustaisissa menetelmissä ovat sisältörepresentaatio sekä arviointifunktio (Togelius & Shaker, 2015), ja ne esitellään seuraavissa alaluvuissa.

3.1 Sisältörepresentaatio

Verrattuna hakuperustaisiin optimointiongelmiin yleensä, pelien erityispiirre on sisältörepresentaatio, joka määrittelee sisällön jota luodaan. Koska menetelmän tehokkuus riippuu paljon siitä, kuinka tehokkaasti erilaisia ratkaisuvaihtoehtoja voidaan käydä läpi, on tärkeää koodata sisältö sellaiseen muotoon, että tarkistaminen on nopeaa: representaatio voi olla esimerkiksi lukutaulukko tai merkkijono. Representaatio onkin hakuperustaisen PCG:n toinen pääongelma: se määrittelee, onko tehokas haku ylipäätään mahdollista (Ashlock, Risi & Togelius, 2015, s. 153). Tutkimusten mukaan representaatiovalinnalla on myös tilastollisesti merkittävää vaikutusta haun tehokkuuteen (McGuinness, 2012, s. 312).

Yleistä on, että sisältö on n-ulotteinen vektori erilaisista parametreista, ja sisältökandidaatit esitetään näinä vektoreina (Togelius ym., 2011b, s. 174). Jos siis esineen parametrit olisivat väri, koko ja paikan koordinaatit, yksittäistä esinettä kuvaisi vektori ("sininen", 20, [1,5]). Parametrien määrää rajoittaa se, että liian lyhyet vektorit eivät voi representoida sisältöä riittävän hyvin, kun taas liian suurissa vektoreissa hakutehokkuus laskee vaihtoehtojen määrän kasvaessa.

Togelius & Shaker (2015, s. 20–21) esittävät erilaiset representaatiovaihtoehdot suora–epäsuora -jatkumossa. Luotaessa vaikkapa labyrinttia mahdollisimman suora esitystapa olisi tietty kartan osanen, jossa on käytäviä ja esineitä. Hieman epäsuorempi tapa olisi antaa tietyt parametrit, millä ehdoilla käytävät voi rakentaa ja mihin aarteet voi sijoittaa kartalle. Kaikkein epäsuorin tapa olisi siemenluku, josta kaikki mahdollinen generoitaisiin.

Suorin generointi antaa eniten kontrollia lopputulokseen, mikä lisää generaattoreilta yleensä vaadittua lokaliteettia, mutta vaatii panostusta ihmissuunnittelijalta (Ashlock ym., 2015, s. 153). Ongelmaksi Togelius & Shaker (2015, s. 21) arvioivat muodostuvan myös sen, että mitä suoremman generaation valitsee, sitä suuremmaksi hakuavaruus kasvaa. Tämä lisää laskentaa ja siten hidastaa prosessia ja myös ratkaisuja on vaikeampi löytää. Epäsuorin generointi taas vaatii vähiten ihmistyötä, mutta se täytyy pystyä kehittämään kyllin luotettavaksi. Lisäksi siemenlukutapauk-

sessa lokaliteettia ei ole ollenkaan ja siten sen tehokkuus ei juuri eroa satunnaisesta hausta (Togelius & Shaker, 2015, s. 21), eikä se siksi ole kovin hyvä menetelmä hakuperustaiseksi menetelmäksi.

Valitun representaation täytyy siis kyetä esittämään kaikki kiinnostavat ratkaisut (Togelius ym., 2011b, s. 175). Lisäksi generaattorin muut ominaisuudet voivat asettaa rajoituksia valitulle representaatiolle. Esimerkiksi online-ratkaisuissa nopeus on ensisijainen ominaisuus koska pelaajan ei voi antaa odottaa sisällön valmistumista kovin kauan, jolloin sisällön laatua voidaan joutua uhraamaan nopeuden vuoksi (Togelius ym., 2011b, s. 183). Offline-generaatiossa taas voidaan panostaa enemmän laatuominaisuuksiin nopeuden kustannuksella.

3.2 Arviointifunktio

Toinen tärkeä komponentti on arviointifunktio (engl. *evaluation function* tai *fitness function*). Arvioiminen toimii yleensä siten, että kustakin yksilöstä tuotetaan arviointifunktion avulla numero, joka kertoo sen hyvyyden, artefaktin laadusta kertovan arvosanan. Tämä on yleensä vaikein osa hakuperustaisesta ratkaisusta (Togelius & Shaker, 2015, s. 17–18).

Funktion tulisi olla suunniteltu mallintamaan jotain artefaktin laatuominaisuutta, esim. pelattavuutta tai viihdyttävyyttä (Togelius & Shaker, 2015, s. 21). Ominaisuudet täytyy kuitenkin olla hyvin määriteltyjä, esim. jos arvioitava ominaisuus on vaikeasti formalisoitavissa, kuten "hauskuus"tai "frustraatio", asioilla joita mitataan ei välttämättä olekaan niin suoraa korrelaatiota todelliseen hauskuuteen. Emotionaalisia tiloja on vaikea formalisoida (Togelius ym., 2011b, s. 176).

Hakuperustaisille menetelmille Togelius & Shaker (2015, s. 23–25) määrittelevät kolmen tyyppistä arviointifunktiota: suorat, simulaatioperusteiset ja interaktiiviset. Suorat evaluointifunktiot perustuvat fenotyypin hyvyysarvoon, ne ovat nopeita ja helppoja toteuttaa, mutta vaikea kehittää joillekin aspekteille (Togelius ym., 2011b, s. 176). Tämä ratkaisu on todennäköisimmin suunniteltu jollekin tietylle pelille tai sisältötyypille. Ne voivat perustua teoreettisiin tutkimuksiin pelaajakokemuksesta tai

kvantitatiivisiin tutkimuksiin kyselyistä tai fysiologisista mittauksista.

Simulaatioperusteisissa arviointifunktioissa tekoäly pelaa sisällön läpi ja samalla kerätään tietoa sen käyttäytymisestä (Togelius ym., 2011b, s. 176): voittiko se, kuinka nopeasti, mitä pelityylejä oli? Labyrinttiesimerkissä tekoälyn tarkoitus voisi siis olla selvitä labyrintin läpi tulematta tapetuksi. Tekoäly voi joko mukautua pelin aikana eli oppia miten peliä pelataan tai olla staattinen, täysin käsin koodattu. Tämä ratkaisu on kuitenkin laskennallisesti vaativa (Togelius ym., 2011b, s. 183).

Interaktiivisissa arviointifunktioissa sisältöä arvioidaan perustuen interaktioon ihmisten kanssa, joko implisiittisesti keräämällä tietoa mitä pelaaja tekee, tai eksplisiittisesti pelaajalta kysymällä. Eksplisiittisen ongelma on, että se keskeyttää pelaamisen, ellei sitä ole hyvin integroitu pelisuunnitteluun. Toisaalta implisiittisen keräyksen ongelmana on että perustuu väistämättä oletuksiin, jotka eivät välttämättä aina pidäkään paikkaansa niin itsestäänselvästi (Togelius & Shaker, 2015, s. 24).

Suoran arviointifunktion ongelma on, että voi olla hankalaa kuvata kaikkia aspekteja yhdellä luvulla (Togelius & Shaker, 2015, s. 19), jos on monta erilaista yhtä tärkeää arvioitavaa asiaa. Jos käytetään yksitavoitteista evolutiivista algoritmia kuten evolutiivinen strategia (ks. luku 4.1), voi käyttää painotettua summaa (Togelius & Shaker, 2015, s. 19). Tästäkin kuitenkin voi tulla ongelmia, jos joitakin funktioita painotetaan toisten kustannuksella: välttämättä ei voi tietää, mikä on oikea painotussuhde. Lisäksi yksittäinen numero ei aina kuvaa hyvin eri parametrien vuorovaikutusta (Togelius, Preuss, Beume, Wessing, Hagelback & Yannakakis, 2010b, s. 277).

Arviointifunktioiden suunnittelun vaikeutta kuvastaa seuraava tutkimus. Loiacono, Cardamone & Lanzi (2010) tekivät rallipelin geneettisillä algoritmeilla, ja arviointifunktion tavoitteena oli arvioida ratojen diversiteettiä. Toinen arviointifunktio perustui kaareutuvuusprofiiliin eli erilaisten mutkien tyyppiin ja määrään, mikä ehkä myös ihmissilmällä vaikuttaa loogisimmalta tavalta arvioida, näyttääkö rata yksitoikkoiselta vai vaihtelevalta. Toinen arviointifunktio puolestaan arvioi ratojen nopeusprofiilia, sitä, mikä on korkein nopeus joka radan eri osissa voidaan saavuttaa.

Arviointifunktioiden tuloksia vertailtaessa kävi selville, että ne eivät korreloineet keskenään paljon: korkeat pisteet kaareutuvuudessa saavuttaneella radalla saattoi olla vähän vaihtelua maksiminopeuksissa, ja vastaavasti vaihtelevanopeuksiset radat eivät välttämättä olleet monipuolisia kaareutuvuudessa. Epäintuitiivisempi ratkaisu siis johti vaihtelevampaan pelikokemukseen eli täytti tavoitteet paremmin, ja lisäksi näitä erikseen arvioitaessa ei kyetty saavuttamaan kumpaakin tavoitetta täyttävää ratkaisua.

Tämän ongelman vuoksi kannattaakin sen sijaan käyttää monitavoitteista evolutiivista algoritmia joka on tarkoituksella optimoitu useille tavoitteille (Togelius, Preuss & Yannakakis, 2010a). Tätä menetelmää käytettäessä edellämainittu Loiacono ym. (2010) havaitsivat, että tuloksena saadut radat saivat korkeat pisteet sekä nopeudessa että kaareutuvuudessa. Toinen esimerkki on Togelius ym. (2010a), jossa käytetty tekniikka oli yhden suositun monitavoitteisen evolutiivisen algoritmin, NSGA-II:n muunnelma SMS-EMOA.

4 Käytännön sovelluksia

Hakuperustaiset menetelmät ovat PCG-kontekstissa vielä kehitysvaiheessa. Kuitenkin monet algoritmit joihin hakuperustaiset menetelmät nojaavat, ovat tunnettuja muista yhteyksistä, joten teoriapohja on vahva, sitä täytyy vain soveltaa enemmän pelikontekstiin. Lisäksi on olemassa useita tapaustutkimuksia, joissa on tehty pelejä näillä menetelmillä. Tässä luvussa käsitellään evolutiiviset algoritmit sekä esimerkkejä peleistä, joissa on käytetty jotain hakuperustaista menetelmää.

Hakuperustaisissa menetelmissä pitää huomioida, että ne toimivat parhaiten raskaissa peleissä joissa on paljon sisältöä. Jos hakuavaruus on pieni tai sisällönluonnille on paljon aikaa, kannattaa evolutiivisen algoritmin sijasta käyttää kattavaa hakua (Togelius & Shaker, 2015, s. 18), ts. etsiä jokainen mahdollinen kombinaatio. Tämä löytää absoluuttisesti parhaat ratkaisut. Lisäksi, jos tavoitteena on ainoastaan luoda paljon erilaista sisältöä, kuten vaikka paljon erilaista kasvillisuutta, parempi ratkaisu on käyttää täysin satunnaista hakua.

4.1 Evolutiiviset algoritmit

PCG:ssä yleinen valinta hakuperustaiseksi menetelmäksi on evoluutiolaskenta (Togelius ym., 2011b, s. 174), jota on käytetty jo pitkään erilaisissa laskentaongelmissa. Sen ydinkomponentit ovat populaatio, muokkaus ja luonnonvalinta (Blum & Roli, 2003, s. 285). Hakuperustaisten menetelmien kontekstissa luonnonvalinnan tehtävää toimittaa arviointifunktio.

Evolutiivisten algoritmien ydinidea on pitää populaatiota yksilöistä, ratkaisuvaihtoehdoista, joita kehitetään eteenpäin (Simula, 1997). Jokaisessa sukupolvessa kandidaatit arvioidaan ja järjestetään parhausjärjestykseen, huonoimmat poistetaan ja korvataan hyvien kandidaattien kopioilla, joita on satunnaisesti mutatoitu tai joiden ominaisuuksia on yhdistelty eri tavalla (Togelius ym., 2011b, s. 174).

Algoritmin tuottama yksittäinen ratkaisu, artefakti, on genotyypin ja fenotyypin

yhdistelmä (Simula, 1997). Kuten hakuperustaisissa menetelmissä yleisestikin, hakuavaruuden ratkaisut koodataan haun helpottamiseksi. Evolutiivisten algoritmien kontekstissa tätä koodausta nimitetään genotyypiksi. Nämä genotyypit konvertoidaan fenotyypeiksi (Togelius & Shaker, 2015, s. 20), eli ilmiasuksi. Jos fenotyyppi on ase, genotyyppi aseelle on säännöt joilla se luodaan, vaikkapa numerotaulukko, joka määrittelee aseen kantaman ja tulitusnopeuden.

Jokainen sukupolvi jaetaan valinta- ja muokkausvaiheeseen (Togelius & Shaker, 2015, s. 18). Luonnonvalinnalla, toisinsanoen arviointifunktiolla, valitaan fenotyypeistä parhaat, ja uusi genotyyppien populaatio muodostuu niiden perusteella (Simula, 1997). Tämän jälkeen uutta populaatiota muokataan esimerkiksi risteyttämällä parhaimmiksi osoittautuneita ratkaisuja (rekombinaatio) tai kopioimalla parhaat pienin muutoksin (mutaatio).

Prosessi toimii myös jos alkuasetelma on täysin satunnainen, koska arviointifunktio pystyy valitsemaan huonoista vaihtoehdoista vähiten huonot (Togelius & Shaker, 2015, s. 18), tosin tällöin algoritmin tehokkuus vähenee kun sukupolvia täytyy generoida enemmän. Täytyy myös huomioida, että voidaan ajautua lokaaliin optimiin, mikä tarkoittaa, että populaatio on niin suuri ja yksilöt niin samankaltaisia, että mainittavaa kehitystä ei tapahdu (Simula, 1997). Näin voi tapahtua myös tilanteessa jossa muokkausstrategiana käytetään pelkästään risteyttämistä: tällöin populaatioon ei pääse missään vaiheessa uutta ainesta. Tähän voidaan evoluutiolaskennassa vastata esimerkiksi lisäämällä satunnaisuutta.

Kaksi yleisesti käytettyä evolutiivista algoritmia ovat evolutiivinen strategia sekä geneettinen algoritmi. Evolutiivinen strategia perustuu mutaatioihin, geneettinen algoritmi taas käyttää mutaation lisäksi risteyttämistä sekä erilaisia valintamekanismeja (Togelius & Shaker, 2015, s. 19).

Evolutiivinen strategia

Yksinkertainen, mutta paljon käytetty esimerkki evolutiivisista algoritmeista on evolutiivinen strategia, $\mu + \lambda$ ES (Togelius & Shaker, 2015, s. 18). Siinä μ on yksilöiden

määrä, jotka säilytetään samanlaisena sukupolvien välillä, niinkutsuttu eliitti, ja λ on yksilöiden määrä, jotka valitaan muokattavaksi jokaisesta sukupolvesta.

Ensin alustetaan populaatio joko tekemällä artefaktit käsin tai generoimalla ne satunnaisesti annetuilla parametreilla. Labyrinttitapauksessa populaatio on erilaisten labyrinttien joukko. Seuraavaksi arvioidaan labyrintti arviointifunktiolla: voiko labyrintin päästä läpi? Kuinka haastava labyrintti on, huomioiden sekä reitin pituus että käytävien haarautuvuus? Onko mahdolliset viholliset mahdollista voittaa? Arviointifunktion tuloksena tulee numeroarvo, joka kertoo sen hyvyyden.

Tämän jälkeen populaatio järjestetään parhausjärjestykseen, ja λ huonointa yksilöä poistetaan. Sitten poistetut yksilöt korvataan kopioimalla niiden tilalle μ yksilöä jäljellä jääneestä populaatiosta. Näitä kutsutaan jälkeläisiksi. Jos $\mu = \lambda$, jokainen yksilö eliitistä saa yhden kopion itsestään. Seuraavaksi mutatoidaan jälkeläiset, eli muutetaan niiden ominaisuuksia satunnaisesti, esimerkiksi numeroparametreja voi muokata lisäämällä niihin satunnaisluku.

Tämän jälkeen yksi vaihe on valmis, ja jos populaatiossa on yksilö jolla on riittävä laatu tai sukupolvien maksimimäärä on saavutettu, lopetetaan. Muuten aloitetaan uusi sukupolvi arvioimalla tämä uusi populaatio arviointifunktiolla. Koska jokaisesta sukupolvesta parhaat yksilöt säilytetään sellaisinaan, jokainen sukupolvi on edellistä parempi. Tämä menetelmä on arvioitu merkittävän tehokkaaksi ja myös 1+1 ES voi joissakin tapauksissa toimia (Togelius & Shaker, 2015, s. 19).

4.2 Esimerkkipelejä

Hakualgoritmissa yksinkertaiset evolutiiviset ratkaisut toimivat yleensä hyvin. Muita hakuperustaisia menetelmiä, jotka eivät ole evolutiivisia, PCG-kontekstissa on tutkittu mm. parviälyä (Togelius & Shaker, 2015, s. 19) ja neuroverkkoja (Ashlock ym., 2015, s. 162). Evolutiivisista algoritmeista ovat hyviä esimerkkejä aiemmassa luvuissa mainitut Togelius ym. (2010b) jossa kehitettiin StarCraftiin karttoja sekä McGuinness & Ashlock (2011), jossa kehitettiin sokkeloita.

Geneettisiä algoritmeja käyttivät Valtchanov & Brown (2012), jotka kehittivät sokkeloita. Tässä tutkimuksessa havaittiin satunnaisen muokkauksen ja risteytyksen yhdistämisen kasvattavan olemassaolevien rakenteiden tuhoutumistodennäköisyyttä, joten todennäköisyyksiä erilaisten mutaatioiden ilmaantumiseen laskettiin.

Frade ym. (2012) esittelivät hakuperusteisen tekniikan nimeltä GTP, Genetic Terrain Programming, joka käyttää hyväkseen geneettistä ohjelmointia muokattuna erikseen maaston generointia varten. Tässä parametrien syöttö nojaa ainoastaan geomorfologisiin metriikkoihin, mikä tarkoittaa, että ihmistä ei tarvita ollenkaan prosessissa edes parametrien syöttöön, vaan toiminta on kokonaan automaattista. Tuloksena saatiin että GTP kykenee löytämään paljon erilaista sisältöä joka sopi tavoitteisiin.

GTP:tä käyttivät myös Ferreira & Toledo (2014), jotka kehittivät sillä tasoja Angry Birds -klooniin. Algoritmissa jokaisella elementillä on todennäköisyystaulukko, mihin kohtaan se tulee milläkin todennäköisyydellä sijoittumaan. Yksilöt ovat xml-tiedostona, ja kun se on valmis, peli ajetaan ottamalla tämä xml-tiedosto syötteeksi. Tuloksena saatiin, että algoritmi pystyy tuottamaan mielenkiintoisia ja vakaita tasoja.

Neuroverkkoihin perustuvaa tekniikkaa nimeltä CPPN (compositional pattern producing networks) käytettiin Petalzissa, joka on sosiaalinen, ei kilpailuun perustuva facebook-peli (Risi, Lehman, D'Ambrosio, Hall & Stanley, 2015). Siinä tutkittiin mahdollisuutta käyttää PCG:tä epätyypillisissä pelimekaniikoissa, ja tultiin siihen tulokseen että käytetyn menetelmän uudelleenpeluuarvon vuoksi sillä oli potentiaalia tukemaan mm. pelaajien välistä virtuaalitaloutta.

CPPN:ää käytettiin myös Hastings ym. (2009) tutkimuksessa, joka kehitti aseita Galactic Arms Race -peliin. Lisäksi siihen yhdistettiin myös adaptiivisuutta: aseita kehitettiin sen perusteella mitkä pelaaja valitsi eniten. Tutkimuksen päätulos oli, että adaptiivisuus on ylivoimainen sisällönluontitekniikka verrattuna satunnaiseen sisällönluontiin.

5 Yhteenveto

Tutkimuksen keskeiset johtopäätökset ovat, että proseduraalinen sisällöntuotanto nopeuttaa pelintuotantoprosessia ja että hakuperustainen menetelmä on tehokas suurissa ja monimutkaisissa peleissä. PCG:itä on olemassa paljon erilaisia, ja hakuperustainen lähestymistapa on vielä kehittymisvaiheessa.

Suurin etu hakuperustaisessa menetelmässä on helpompi virheiden ehkäisy, ja lisäksi se tukee erilaisia generaattoreiden ominaisuuksia paremmin kuin konstruktiiviset ratkaisut, mm. online-generaatiota sekä adaptiivisuutta. Kaksi suurinta ongelmaa hakuperustaisissa menetelmissä ovat arviointifunktion kehittäminen uskottavaksi sekä sisältörepresentaation kehittäminen siten, että hakeminen on riittävän tehokasta mutta sisältö pysyy silti kontrolloitavana.

Proseduraalisella sisällöntuotannolla on tärkeä käytännön merkitys pelituotannossa, koska se vähentää ihmistyötä ja siten alentaa kustannuksia, ja riittävällä osaamisella se myös antaa pienillekin yrityksille ja yksittäisille ohjelmointiharrastajille mahdollisuuden luoda paljon sisältöä peleihinsä. Lisäksi Togelius ym. (2015, s. 2–4) näkevät, että tulevaisuudessa PCG:tä voidaan hyödyntää reaaliaikaisessa pelaajaan mukautumisessa, sekä se mahdollistaa että pelin ei tarvitse loppua lainkaan.

Luonnollisia jatkotutkimusaiheita on neljänlaisia. Itsestäänselvä on tietysti hakuperustaisten menetelmien tehostaminen niin sisältörepresentaation kuin arviointifunktionkin osalta. Toinen on yleisesti metaheurististen tekniikoiden soveltaminen pelien sisällöntuotantoon: evolutiivisilla algoritmeilla on vahvat puolensa, mutta monet muilla aloilla käytetyt menetelmät odottavat vielä soveltamistaan. Millaista sisällöntuotantopotentiaalia olisi esimerkiksi tabuhaulla, käkihaulla, meemialgoritmilla tai tulikärpäsalgoritmilla?

Kolmas kiinnostava aihe on uudenlaiset pelityypit joita proseduraaliset ja hakuperustaiset menetelmät mahdollistavat, sillä kuten (Risi ym., 2015) havaitsivat Petalzpelin yhteydessä, hakuperustaiset menetelmät voivat tukea myös epätyypillisiä pelimekaniikkoja. Ei pidä väheksyä myöskään puhtaasti ludologisia tutkimuksia PCG:n

tiimoilta. Kuten Hendrikx ym. (2013, s. 2) huomauttavat, pitäisi voida arvioida myös generaattoreiden kulttuurisia arvoja, tai millaista vaikutusta generoidulla sisällöllä on pelikokemukseen ja koettuun laatuun, tai onko pelaajan pelityyleillä tai tunteilla vaikutusta siihen miten generoitu sisältö koetaan.

Kirjallisuutta

Ashlock, D., Lee, C. & McGuinness C. 2011. *Search-Based Procedural Generation of Maze-Like Levels*. IEEE Transactions on Computational Intelligence and AI in Games, Vol. 3, No. 3, s. 260–273

Ashlock, D., Risi, S. & Togelius, J. 2015. *Representations for search-based methods*. Teoksessa Procedural Content Generation in Games: A Textbook and an Overview of Current Research. s. 153–173, Springer 2015

Blum, C. & Roli, A. 2003. *Metaheuristics in Combinatorial Optimization: Overview and Conceptual Comparison*. ACM Computing Surveys, Vol. 35, No. 3, s. 268–308

Compton, K., Osborn, J. & Mateas M. 2013. *Generative Methods*. PCG Workshop 2013

Dwarf Fortress. Saatavilla WWW-muodossa `http://www.bay12games.com/dwarves/`. Viitattu 24.4.2015

Ferreira, L. & Toledo C. 2014. *A Search-based Approach for Generating Angry Birds Levels*. IEEE Conference on Computational Intelligence and Games

Frade, M., de Vega, F. & Cotta C. 2012. *Aesthetic Terrain Programs Database for Creativity Assessment*. IEEE Conference on Computational Intelligence and Games (CIG) s. 350–354

Hastings, J., Guha, R. & Stanley, K. 2009. *Evolving Content in the Galactic Arms Race Video Game*. IEEE Transactions on Computational Intelligence and AI in Games, Vol. 1. No. 4. s. 241–248

Henrikx, M., Meijer, S., van der Velden, J. & Iosup A. 2013. *Procedural Content Generation for Games: A Survey*. ACM Transactions on Multimedia Computing, Communications and Applications, Vol. 9, s. 1–22

Loiacono, D., Cardamone, L. & Lanzi, P. 2010. *Automatic Track Generation for High-End Racing Games Using Evolutionary Computation*. IEEE Transactions on Computational Intelligence and AI in Games, Vol. 3. No. 3. s. 245–259

McGuinness, C. 2012. *Statistical Analyses of Representation Choice in Level Generation*. IEEE Conference on Computational Intelligence and Games, s. 312–319

McGuinness, C. & Ashlock, D. 2011. *Decomposing the Level Generation Problem with Tiles*. IEEE, s. 849–856

Nelson, M. & Smith, A. 2015. *ASP with applications to mazes and levels*. Teoksessa Procedural Content Generation in Games: A Textbook and an Overview of Current Research. s. 139–152, Springer 2015

Prusinkiewicz, P. & Lindenmayer, A. 1991. *The Algorithmic Beauty of Plants*. Springer Verlag

Risi, S., Lehman, J, D'Ambrosio, D., Hall, R. & Stanley O. 2015. *Petalz: Search-based Procedural Content Generation for the Casual Gamer*. IEEE Transactions on Computational Intelligence and AI in Games, hyväksytty julkaistavaksi. Saatavilla WWW-muodossa `http://doi.org/10.1109/TCIAIG.2015.2416206`.

Shaker, N., Smith, G. & Yannakakis N. 2015a. *Evaluating content generators*. Teoksessa Procedural Content Generation in Games: A Textbook and an Overview of Current Research. s. 211–218, Springer 2015

Shaker, N., Togelius, J. & Nelson M. 2015b. *Procedural Content Generation in Games: A Textbook and an Overview of Current Research*. Springer 2015

SimCity. Saatavilla WWW-muodossa `http://www.simcity.com/`. Viitattu 18.4.2015

Simula, P. 1997. *Johdatus evoluutiolaskentaan ja geneettisiin algoritmeihin*. Esitelmä Laskennallisen tekniikan seminaariin, HUT. Saatavilla WWW-muodossa `http://www.lce.hut.fi/teaching/S-114.240/k97/ga/gasis.html`. Viitattu 2.3.2015

Togelius, J., Shaker, N. & Nelson, M. 2015. *Introduction*. Teoksessa Procedural Content Generation in Games: A Textbook and an Overview of Current Research. s. 1–15, Springer 2015

Togelius, J. & Shaker, N. 2015. *The search-based approach*. Teoksessa Procedural Content Generation in Games: A Textbook and an Overview of Current Research. s. 17–30, Springer 2015

Togelius, J., Kastbjerg, E., Schedl, D. & Yannakakis G. 2011a. *What is Procedural Content Generation? Mario on the borderline*. PCGames '11 Proceedings of the 2nd International Workshop on Procedural Content Generation in Games, ACM

Togelius, J., Yannakakis, G., Stanley, K. & Browne C. 2011b. *Search-based Procedural Content Generation: A Taxonomy and Survey*. IEEE Transactions on Computational

Intelligence and AI in Games, Vol. 3, No. 3, s. 172–186

Togelius, J., Preuss, M. & Yannakakis G. 2010a. *Towards multiobjective procedural map generation*. Proceedings of the 2010 Workshop on Procedural Content Generation in Games, ACM

Togelius, J., Preuss, M., Beume, N., Wessing, S., Hagelback, J. & Yannakakis G. 2010b. *Multiobjective Exploration of the StarCraft Map Space*. Proceedings of the IEEE Conference on Computational Intelligence and Games, 2010, s. 265–272.

Valtchanov, V. & Brown, J. 2012. *Evolving Dungeon Crawler Levels With Relative Placement*. Proceedings of the Fifth International C* Conference on Computer Science and Software Engineering s. 27–35

Zafar, A. & Mujtaba H. 2012. *Identifying Catastrophic Failures in Offline Level Generation for Mario*. 10th International Conference on Frontiers of Information Technology, s. 62–67